

## 6. Anhang: Eigenschaften und Methoden des Internet Explorer

Hinweise

für ELEMENT das HTML-Tag des Objektes einsetzen

für object eine Referenz einsetzen z.B. laut ID-Attribut des Objektes (logischer Objektname), dessen Stringwert einen Zeigerbezeichner darstellt

### Microsoft ändert fortlaufend die Active-X-Eigenschaften von Windows und somit auch des Internet Explorers

Diese fortlaufenden Änderungen muss der Programmierer in Erfahrung bringen.

Der Programmierer kann sich definitiv nicht auf Verfügbarkeit von Active-X-Controls verlassen und muss damit rechnen, dass seine Webseiten schlagartig nicht mehr komplett laufen weil u.a. Programmcode noch nicht angepasst ist. Ebenfalls muss der Programmierer Varianten von Windows und Patchzustände beachten, die prinzipiell Kostenprobleme verursachen können.

Mit anderen Worten: Wer Microsoft-Komponenten nutzt, muss wissen, was ihm blüht ... siehe nachfolgende Beispiel für Risiken.

### Prinzipielle Lizenzprobleme für den Programmierer

Microsoft verlangt Lizenzierung von Windows. Bezüglich Windows-Versionen gibt es die Updatestufen z.B. per Servicepacks

Ein Windows mit Servicepack fällt unter die Lizenz des geupdateten Windows.

Ein Windows mit Vorversion zum Servicepack bedarf einer anderen Lizenz.

Will man z.B. den Internet Explorer 7 und 6 parallel testen, benötigt man 2 Windowslizenzen, da beide Versionen nicht parallel installierbar. Dazu kommt, dass es den IE 6 in 2 Versionen gibt: Win SP1 und SP2 (IE 7 nur ab Win SP2).

Für 3 Browserversionen benötigt man 3 Windowslizenzen, will man parallel testen.

Ein Blick auf Browser-Konkurrenzprodukte klärt die Sachlage unschlagbar: Opera ist z.B. parallel installierbar.

Hinweis: Man suche doch mal im Internet nach einem kostenlosen HTTP-Server vom Microsoft, um IE-Seite testen zu können, die JScript nutzen (inklusive Debugger). Denn sollte kein kostenloses Angebot findbar sein, kommen die Kosten von Entwicklungssoftware zum IE hinzu. Ein Blick auf Konkurrenz-HTTP-Server klärt die Sachlage: Apache-HTTP-Server ist kostenlos, allerdings nicht einfach einzurichten (Hinweis: Der HTTP-Server sollte virtuelle Hosts einrichten können und korrekt mit der Firewall des Users zusammenarbeiten können).

### Abänderungen wegen Sicherheitspatches der jeweiligen Windows-Versionen

Abschaltungen von Active-X-Controls erfolgen auch im Rahmen der Sicherheitspatches zu Windows-Versionen.

Es ist auch möglich, dass wegen Sicherheitslücken abgeschaltet wird und somit Komponenten einer Webseite je nach Windowsversion nicht mehr laufen.

Im Rahmen der Sicherheitspatches ist es Microsoft sogar gelungen, Webseiten, die den MS-Encoder zur Komprimierung von

HTML- und JScript-Code nutzen, schlagartig unnutzbar zu machen: Ein Bug in einem Patch zu Windows XP - Q918899

Das Patch verursacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit

Kompression genutzt wird z.B. bei

onclick-Handler auf IMG

klick ins Fenster per aktivem Popup

Der Absturz ist "read" -Fehler von immer ein und derselben Speicherstelle.

User, die dieses Patch installiert haben, können ab sofort keine IE-Seiten mit codiertem Script mehr ansehen.

Microsoft stellt Abhilfe nach geraumer Zeit zur Verfügung, jedoch spezifisch nach Windows XP-Version:

Patch Q918899 für

Windows XP SP1Download für jedermann bereitgestellt

SP2 nur auf kostenpflichtige telefonische Anfrage des Users per Downloadlink bereitgestellt, da

Microsoft explizit die User registriert haben will, bei denen das

Patchproblem auftritt (User muss sich Telefonnummer besorgen)

Solange also das Patch zum fehlerhaften Patch vom User nicht installiert wird,

z.B. weil der User keine Ahnung hat, dass und wo er sich die Telefonnummer

von Microsoft besorgen muss bzw. zu besorgen hat, wird der User

IE-Seiten mit komprimierten Code dauerhaft nicht nutzen können.

(Microsoft-Support ist z.T. nur in Englisch).

### Abänderungen wegen Browser-Inkompatibilität

#### *Popupblocker-Fehler*

Die Microsoft Browser-Version IE 7 ist nicht abwärtskompatibel bezüglich Popup per window.createPopup()

Popup per window-Objekt ist ein Markenzeichen des IE, das im IE 7 nicht mehr fehlerfrei nutzbar ist.

Der Fehler liegt in der Popup-Blockerverwaltung des IE und wurde mit dem IE 7 implementiert.

Der Fehler tritt nicht auf, wenn ein Fenster per window.open() erzeugt wurde.

Bedingung:

Scriptfehleranzeige ist erlaubt im IE 7

Popupblocker ist im IE abgeschaltet

ein aktives Fenster (Register) mit Dokument, dass fortlaufend (rekursiv) genau 1 window.popup per .show()erzeugt.

ein weiteres Fenster (Register) z.B. leere Seite (about:blank)

beide (Register) liegen in einer gemeinsamen IE-Instanz

Ablauf: Wird Focus auf Register der leeren Seite gehalten und wird parallel das Popup per .show() erzeugt,



bricht der Browser das Dokument mit .show() ab (Scriptfehler).

Der Popupblocker für die leere Seite verursacht den Programmfehler im Dokument mit .show(). Es wird folgende Meldung angezeigt (in der Informationsleiste):

'Ein Popup wurde geblockt. Klicken Sie hier, um das Popup bzw. weitere Optionen anzuzeigen.'

Die Bedeutung der Meldung laut Microsoft-Hilfe im IE 7:

Der Popupblocker hat ein Pop-up-Fenster geblockt. Sie können den Popupblocker deaktivieren oder Pop-ups temporär zulassen, indem Sie auf die Informationsleiste klicken.

Die Realität zur obigen Meldung ist völlig anders:

Linke oder rechte Maus auf die Meldung liefert z.B. Einstellungen darunter

- Popupblocker einschalten
- weitere Informationen
- jedoch keine Möglichkeit wie laut Bedeutung

Damit gilt: Der abgeschaltete Popupblocker ist in Wirklichkeit aktiv.

Pikant: Ein Popup erscheint normalerweise auch über fremde Fenster, die nicht das Popup erzeugt haben (z.B. Fenster einer Windowsanwendung z.B. einer anderen IE-Instanz)

Der Popupblocker des IE bemerkt aber NUR Webseite, die das Popup erzeugt.

Durch das Abwürgen von Popup wird das Popup natürlich auf und für anderen Seiten nicht relevant; im Falle einer anderen IE-Instanz also auch für diese nicht relevant, obwohl diese Instanz per Popupblocker verwaltet wird.

Der Popupblocker beschneidet die Popup-Reichweite an der Wurzel, ist aber nicht objektorientiert zu den anderen Webseiten (die nicht das Popup erzeugt haben).

Der Popupblocker ist nicht als Filter aufgesetzt sondern reingestrickt worden.

Der Popupblockerfehler verändert die Eventverwaltung:

Es werden u.a. ignoriert

- onfocus
- onblur
- onfocusin
- onfocusout

und viele andere, so dass trotz Events z.B. des Body der Popupblockerfehler entsteht.

```
// nachfolgender Code setzt focus nicht neu: Fenstereintrag in Taskleiste blinkt eventuell
window.focus();
window.document.focus();
if(document.body!=null)
{if(document.body.style!="hidden")      // wenn hidden so focus() nicht möglich (Scriptfehler erzeugt)
{document.body.focus();}
}
// wenn paralleles Fenster offen (on oder offline), so Scriptfehler erzeugt
popupzeiger.show(...);
```

Hinweis: Der Popupfehler ist so elementar, dass die vielen Beta-Testphasen des IE mehr als fragwürdig erscheinen, wie die Angabe von Microsoft, dass Code neu programmiert wurde, um den IE sicherer zu machen.

#### *focus-Methode beim IE 7*

windows.focus() document.focus() und body.focus() funktionieren NICHT  
zwischen Register in einem IE-Fenster  
zwischen Fenstern z.B. in Taskleiste

Hinweis:

.focus() setzt Element aktiv, gibt dem Element den Focus und feuert dann onfocus  
.setActive() ist Teilmenge von .focus(): nur das aktiv setzen  
funktioniert nicht mit allen Elementen, mit denen .focus() funktioniert

#### *animierte Gif (mit Timer)*

Animierte Gifs (mit Timer), die unter IE 6 korrekt laufen, müssen unter IE 7 im Timer nicht mehr laufen:  
z.B. garnicht mehr sichtbar, oder Timer nicht verwendet.  
Dann müssen animierte Gif-Bilder nach IE-Version bereitgestellt werden.

#### Abänderungen wegen Rechtstreitigkeiten von Microsoft mit Fremdanbietern

Ein sehr bekanntes Beispiel ist die nachträglich eingeführte Einschränkung von Active-X-Controls wegen Patentwahrung durch Microsoft, wobei für den JScript-Programmierer massive Änderungen eintreten.

Wegen Patentwahrung hat Microsoft ein zunächst freiwilliges Patch herausgegeben, dass bei ActiveX-Control per APPLET, EMBED oder OBJECT, die auf dem Bildschirm rendern (mit oder ohne Userschnittstelle), dafür sorgt, dass bei mouseover über das Control eine Sprechblase erscheint, die darauf hinweist, dass das Objekt als ActiveX-Control klickbar ist. Diese Sprechblase erscheint auch, wenn das Control keine Userschnittstelle hat, also diese gar nicht klickbar ist.

Es wurde das Eventmodell gleichzeitig geändert:

Es werden alle Events solange unterdrückt, bis der User die Sprechblase geklickt hat.  
Das Klicken muss auf das Objekt im Sprechblasenrahmen erfolgen, der so groß ist, wie die Dimension, in der gerendert wurde.



Es muss also ERST per Mausklick das Control aktiviert werden, ehe das Control klickbar und damit die Eventsteuerung aktiviert ist.  
 Ein Control, dass programmtechnisch zwar was rendert, aber ansonsten ohne sichtbare programmtechnisch startet, muss ebenfalls geklickt werden, obwohl es bereits läuft und es nichts zu klicken gäbe (wenn keine Eventsteuerung eingebaut wurde).  
 Wegen blockierter Eventsteuerung ist also die Sprechblase z.B. nicht automatisch klickbar.  
 Die Eventauslösung per nicht-objekteigenen Eventhandler, der für das Objekt per fireEvent() ein Event auslöst, ist solange blockiert, bis der User die Sprechblase geklickt hat.

style.visibility='hidden' wird ignoriert

Die Sprechblase erscheint auch dann, wenn das Control mit style.visibility='hidden' belegt ist, also sich unsichtbar rendert:

Der Sprechblasenrahmen hat genau die Dimension wie die des unsichtbaren Controls. Der Sprechblasenrahmen erscheint also Zusammenhangslos, und der User weiß nicht, warum er klicken soll, wenn er nichts sieht. Vor allem weiß er nicht, WAS er klickt ... ideale Basis für Schadsoftware per Script.

Diese Sprechblase erscheint nur DANN NICHT, wenn die Userschnittstelle mit Breite == Höhe == 0 gerendert wird. Sollte die Userschnittstelle in einem Container liegen, z.B. DIV, dann wird der Container, wenn er in der Dimension kleiner ist, also die Userschnittstelle, angepasst. Daher muss der Container ebenfalls mit Breite == Höhe == 0 gerendert werden. Wegen Dimensionierung auf 0 sollte style.visibility="hidden" sein. Im Falle eines Containers reicht es, den style des Containers zu ändern, da visibility normalerweise vererbt wird an Kinder, also auch an das Control.

#### Abänderung wegen Abschaltungen

DirectX ist wegen Abschaltung von Active-X--Controls nicht mehr abwärtskompatibel:

Z.B. wurde bei Win XP SP2 Direct Animation aus DirectX schlagartig durch Abschaltung von Bibliotheken dezimiert, die es bei Win XP SP1 aber noch gibt.

Hier ein Beispiel aus dem Jahr 2004: Abschaltungen von Active-X-Controls

ActiveX-Controls und Unterstützung/Verbot 20041215

#### erlaubt sind noch

Tabular Data-Steuerelement {333C7BC4-460F-11D0-BC04-0080C7055A83} Das TDC (Tabular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformat vorliegen, beispielsweise durch Darstellung in einer Tabelle oder Sortierung. Weitere Informationen:•

[http://msdn.microsoft.com/workshop/database/tdc/tabular\\_data\\_control\\_node\\_entry.asp](http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp)([http://msdn.microsoft.com/workshop/database/tdc/tabular\\_data\\_control\\_node\\_entry.asp](http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp))

Microsoft Agent Control - Version 2.0 {D45FD31B-5C6E-11D1-9EC1-00C04FD7081F} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Weitere Informationen:•

<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>(<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

Microsoft MSChat-Steuerelement-Objekt 2.0 - 2.5 {D6526FE0-E651-11CF-99CB-00C04FD64497}

Dieses Steuerelement wird von Webautoren verwendet, um text- und graphisch basierte Chatgemeinden für Echtzeitkonversationen im Web zu erstellen.

Microsoft ActiveX Upload-Steuerelement, Version 1.5 {886e7bf0-c867-11cf-b1ae-00aa00a3f2c3} Dieses Steuerelement kann auf vielerlei Art genutzt werden, um auf einfache Weise Webinhalte via Drag and Drop zu veröffentlichen. Weitere Informationen:• 230298 (<http://support.microsoft.com/kb/230298/DE/>) - Posting Acceptor Release Notes

• [http://msdn.microsoft.com/workshop/management/tools/reference/file\\_upload\\_control.asp](http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp) ([http://msdn.microsoft.com/workshop/management/tools/reference/file\\_upload\\_control.asp](http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp))

#### verboten sind



Datenbindung RDS {BD96C556-65A3-11D0-983A-00C04FC29E36} {BD96C556-65A3-11D0-983A-00C04FC29E33} Die RDS (Remote Data Service) Steuerelemente ermöglichen dem Browser, client-basierte SQL Abfragen an einen Webserver zu stellen. Inzwischen wurde RDS jedoch durch neuere Standards wie SOAP abgelöst, von einer weiteren Verwendung von RDS wird daher abgeraten. Weitere Informationen:• 184375 (<http://support.microsoft.com/kb/184375/DE/>) - Sicherheitsaspekte bei RDS 1.5, IIS 3.0 oder 4.0 und ODBC

<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>  
(<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>)  
[http://msdn.microsoft.com/library/en-us/dnmdac/html/data\\_mdacroadmap.asp](http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp)  
([http://msdn.microsoft.com/library/en-us/dnmdac/html/data\\_mdacroadmap.asp](http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp))

XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer {550dda30-0541-11d2-9ca9-0060b0ec3d39} {CFC399AF-D876-11d0-9C10-00C04FC99C8E} {e54941b2-7756-11d1-bc2a-00c04fb925f3} {7108ECB4-AFDC-11D1-ADC1-00805FC752D8} XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer ermöglichen die Verarbeitung von XML Daten, etwa die Bindung von HTML Elementen an einen XML Datensatz, oder das Einlesen, Manipulieren, und Zurückschreiben von XML Daten.

Die Steuerelemente DOMDocument und XMLIslandPeer bzw. die dazugehörigen ClassIDs sind nicht mehr aktuell, so dass von einer generellen Freigabe dieser Steuerelementgruppe abgeraten wird. Weitere Informationen:• [http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml\\_concepts2\\_7ook.asp](http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp)([http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml\\_concepts2\\_7ook.asp](http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp))

#### Internet Explorer

Active Setup / IE Active Setup-Steuerelement {F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1} Dieses Steuerelement enthält die in Microsoft Security Bulletin MS99-037 beschriebene Sicherheitsanfälligkeit. Um eine weitere Ausführung zu verhindern wurde im Rahmen dieses Security Bulletins ein Kill-Bit gesetzt, so dass selbst bei einer Freigabe dieses Controls eine Ausführung blockiert wird. Weitere Informationen:• <http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>(<http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>)  
<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>  
(<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>)  
240797 (<http://support.microsoft.com/kb/240797/DE/>) - So verhindern Sie die Ausführung von ActiveX-Steuerelementen in Internet Explorer

Media Player / Active Movie Runtime {A4001DE0-7075-11d0-89AB-00A0C9054129} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Runtime Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / ActiveMovie-Steuerelement {05589FA1-C356-11CE-BF01-00AA0055595A} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / Microsoft NetShow Player {2179C5D3-EBFF-11CF-B6FD-00AA00B4E220} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das NetShow Player Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / Windows Media Player {22D6F312-B0F6-11D0-94AB-0080C74C7E95} Dies ist das Steuerelement für Windows Media Player version 6.4 und war Installationsbestandteil bis einschließlich Windows Media Player Version 8. Ab Windows Media Player 9 wurde diese ClassID durch die neue ClassID {6BF52A52-394A-11D3-B153-00C04F79FAA6} abgelöst, deren Verwendung stattdessen empfohlen wird. Ab Windows Media Player Version 9 wird ferner die alte ClassID anhand eines Wrappers automatisch auf die neue



ClassID umgeleitet. Die ClassID für Windows Media Player Version 9 ist jedoch nicht in der Liste der vom Administrator genehmigten Steuerelemente enthalten, und muss bei Bedarf manuell hinzugefügt werden.

Animierte Schaltflächen {0482B100-739C-11CF-A3A9-00A0C9034920} Dieses Steuerelement erlaubte in frühen Versionen des Internet Explorer die Verwendung animierter Schaltflächen auf Webseiten. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von der Freigabe des Steuerelements wird daher abgeraten.

#### IE Label-Steuerelement

{99B42120-6EC7-11CF-A6C7-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 auch kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

IE Menu-Steuerelement {74701400-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ermöglicht die Handhabung von Menüstrukturen in Webseiten, wird jedoch nicht mehr unterstützt und dürfte nur noch selten Verwendung finden. Von einer Freigabe des Steuerelements wird daher abgeraten.

IE Preloader-Steuerelement {16E349E0-702C-11CF-A3A9-00A0C9034920} Dieses Steuerelement ermöglichte das Vorladen von Webseiten, ist jedoch inzwischen nicht mehr aktuell, wird nicht mehr unterstützt und dürfte nicht mehr im Einsatz sein. Aufgrund einer potentiellen Sicherheitsanfälligkeit in diesem Steuerelement wird von einer Freigabe abgeraten. Weitere Informationen: • 231452 (<http://support.microsoft.com/kb/231452/DE/>) - Update Available for "Legacy ActiveX Control" Issue

IE Timer-Steuerelement {59CCB4A0-727D-11CF-AC36-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

MCSiMenü {275E2FE0-7486-11D0-89D6-00A0C90C9B67} Dieses Steuerelement dient der Anpassung von Pop-upmenüs, ist jedoch nicht mehr aktuell und wurde nach Windows 98 nicht mehr ausgeliefert. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten.

Pop-upmenüobjekt {7823A620-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

Microsoft Agent Control - Version 1.5 {F5BE8BD2-7DE6-11D0-91FE-00C04FD701A5} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Diese Version des Steuerelements ist jedoch nicht mehr aktuell und wird nicht mehr unterstützt. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: •



<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>  
 (http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm)

**Eigenschaften****@page**

Regel für Dimensionen, Orientierungen und Margins in einer Seite per styleSheet Objekt

Regel ist eine Pseudoklasse von Style

ab IE 5.5

Seite wird als Rechteckbereich aufgefasst, der eingeteilt ist in:

Seitenbereich: Inhalt der Seite z.B. Text, Grafik

Marginbereich: Rand um den Seitenbereich

siehe auch Objekt page

":first" Regel gehört der 1. Seite

":footer" Regel gehört der Fußnote

":header" Regel gehört der Kopfnote

":left" Regel gehört der linken Seite

":left : header" Regel gehört der Kopfnote Seite links

":left : footer" Regel gehört der Fußnote Seite links

":right" Regel gehört der rechten Seite

":right:header" Regel gehört der Kopfnote Seite rechts

":right:footer" Regel gehört der Fußnote Seite rechts

Beispiel: @page : left {font-weight:bold;font\_style:italic;}

.0 bis .n

Wert des Argumentes mit Index 0 ... bzw. n im Script-Objekt arguments als Eigenschaft eines Funktionsobjektes (siehe Script-Objekt Function und Anweisung function)

.0 entspricht auch funktions\_bezeichner.arguments[0]

.n entspricht auch funktions\_bezeichner.arguments[n]

n von 0 bis beliebig ganzzahlig-numerische Ziffernfolge ohne Vornull

ist die Position ab 0 innerhalb der

Argumentenliste von links nach rechts

Parameterliste von links nach rechts

Hinweis: Argumentenliste und Parameterliste mit gleicher Elementanzahl, denn

**jedes Argument** muss mit Wert versorgt werden

Beispiel für Einlesen der Werte der Argumente aus dem Funktionsaufruf:

var GlobalesFeld = new Array();

function FunktionsBezeichner()  
{// Argumentenliste der Funktion **lokal** zur Funktion referenzieren  
var ArgumentenListeAlsFeld = FunktionsBezeichner.arguments;// Anzahl der Argumente  
var ArgumenteAnzahl = ArgumentenListeAlsFeld.length;if ( ArgumenteAnzahl > 0 )  
{

// Arrgumentenliste auslesen

GlobalesFeld[0] = ArgumentenListeAlsFeld.0;

GlobalesFeld[1] = ArgumentenListeAlsFeld.1;

}

// hier die weiteren Anweisungen der Funktion

}

.abstract

Beschreibung einer Media-Datei auf der Timeline

bei Advanced Stream Redirector (ASX)-Datei wird der Text unter ABSTRACT

vom aktuellen Eintrag geliefert und nicht der Datei selbst

siehe Objekt currTimeState und Behavior .style.time2

Beispiel 1:

&lt;HTML XMLNS:t="urn:schemas-microsoft-com:time"&gt;

&lt;HEAD&gt;

&lt;?IMPORT namespace="t" implementation="#default#time2"&gt;

&lt;STYLE&gt;

.time\_line\_klasse { behavior: url(#default#time2) }

&lt;/STYLE&gt;

&lt;/HEAD&gt;

&lt;BODY&gt;

&lt;t:VIDEO

ID="ID\_Video"

SRC="test.wmv"

STYLE="position:absolute;top:50px;height:100px"

&gt;





```

</t:VIDEO>
<SPAN ID="ID_Span"
      STYLE="position:absolute;top:165px;"
>
</SPAN>
<BUTTON ID="ID_Button"
        onclick="ID_Span.innerText= ID_Video.abstract"
>
      Klick
</BUTTON>
</BODY>
</HTML>

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
function ButtonUpdate()
{
    if (ID_Media.currTimeState.isActive)
    {
        ID_Button1.disabled=true;
        ID_Button2.disabled=false;
        ID_Button3.disabled=false;
        ID_Button4.disabled=false;
    }
    else
    {
        ID_Button1.disabled=false;
        ID_Button2.disabled=true;
        ID_Button3.disabled=true;
        ID_Button4.disabled=true;
    }
}

function AnzeigeUpdate()
{
    ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
    ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
    ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
    ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
    ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
    ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
    ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
    ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
}

function AnzeigeLoeschen()
{
    ID_Span1.innerText = "Titel: ";
    ID_Span2.innerText = "Autor: ";
    ID_Span3.innerText = "Abstract: ";
    ID_Span4.innerText = "Copyright: ";
    ID_Span5.innerText = "Filename: ";
    ID_Span6.innerText = "Banner: ";
    ID_Span7.innerText = "Banner Abstract: ";
    ID_Span8.innerText = "Banner MoreInfo: ";
}
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

<t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"
        TIMEACTION="visibility"
        onend="ButtonUpdate();"
        ontrackchange="AnzeigeUpdate();"
        onmediacomplete="ButtonUpdate();AnzeigeUpdate();"

```



```

>
</t:MEDIA>

<SPAN ID="ID_Span1">Titel:</SPAN>
<BR>
<SPAN ID="ID_Span2">Autor:</SPAN>
<BR>
<SPAN ID="ID_Span3">Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span4">Copyright:</SPAN>
<BR>
<SPAN ID="ID_Span5">Filename:</SPAN>
<BR>
<SPAN ID="ID_Span6">Banner:</SPAN>
<BR>
<SPAN ID="ID_Span7">Banner Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

<BR>
<BUTTON ID="ID_Button1"
        onclick="ID_Media.beginElement(); ButtonUpdate();"
>
    Start
</BUTTON>
<BUTTON ID="ID_Button2"
        onclick="ID_Media.playlist.nextTrack();"
>
    naechster Track
</BUTTON>
<BUTTON ID="ID_Button3"
        onclick="ID_Media.playlist.prevTrack();"
>
    vorhergehender Track
</BUTTON>
<BUTTON ID="ID_Button4"
        onclick="ID_Media.endElement(); AnzeigeLoeschen();"
>
    Stop
</BUTTON>
</BODY>
</HTML>

```

**.accelerate** Beschleunigung des Elementes auf der Timeline  
 hat keinen Einfluss auf die Dauer der Timeline  
 auch wirksam bei Wiederholungen per Attribute .repeatCount oder .repeatDur  
 Summe der Werte der Attribute .accelerate und .decelerate darf nicht 1 überschreiten  
 wenn ja, so werden beide Attribute ignoriert, also nicht verwendet  
 siehe Objekt currTimeState und Behavior .style.time2  
 siehe .decelerate

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:ANIMATE TARGETELEMENT="ID_Div"
              ATTRIBUTENAME="left"
              TO="400"
              DUR="3"
              ACCELERATE="1"
              REPEATCOUNT="3"
    >
    </t:ANIMATE>
    <DIV ID="ID_Div" CLASS="time_line_klasse" STYLE="position: absolute;left:10px">
    </DIV>
</BODY>
</HTML>

```

**.accept** Liste von Multipurpose Internet Mail Extensions (MIME)-Typen  
 z.B. "text/html"





```
"image/png"
"image/gif"
"video/mpeg"
"audio/basic"
"text/tcl"
"text/javascript"
"text/vbscript"
```

Listenelemente mit Komma trennen

Beispiel:

```
<INPUT TYPE=file ACCEPT="text/html, image/png">
```

es können nur laut ACCEPT-Wert angegebene Dateitypen selektiert werden

.acceptCharset

Liste von UTF-8 Zeichen für Encoden der Eingabedaten durch den Server  
Liste deklariert alle Zeichen, die nicht im Charset des Dokumentes erfasst werden  
Liste wird vom Server benötigt  
wenn nicht kodiert, so nur der Charset des Dokumentes verwendet  
UTF-8 Zeichen: Teil des Unicode (0 bis 255)

.accessKey

Tastaturzugriff auf ein Objekt per Alt + Taste  
bei Ausführung der Tastenkombination wird  
das Sprungziel wird focussiert  
die Sprungquelle defocussiert  
das Focus-Ereignis ausgelöst  
vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt

Beispiel:

```
<LABEL FOR="ID_Input" ACCESSKEY="I">
  #<SPAN>I</SPAN>:
  Alt+I fuer Sprung zur Textbox
</LABEL>
<INPUT TYPE="text"
  ID="ID_Input "
  NAME="T1"
  VALUE=text1
  SIZE="20"
  TABINDEX="1"
>
```

.accumulate

Kumulative Animation eines Elementes auf der Timeline  
Element darf nicht aktiv sein, wenn .accumulate definiert werden soll:  
Element erst danach starten.  
Es kann jede numerische Style-Eigenschaft kumulativ verändert werden und damit  
die kumulative Animation des Elementes in der Style-Eigenschaft erzeugen.  
Wert der Eigenschaft: numerisch aber Einheiten wie px, in, cm, mm, pt, pc können Teil des  
Wertes sein (mathematische Berechnungen erst nach Entfernung der Einheit  
per Stringoperationen möglich)  
Name der Eigenschaft laut .attributName  
Änderung des Wertes pro Durchlauf bzw. kumulativ um den Maximalwert laut .by  
Anzahl der Durchläufe bzw. der Wert-Kumulationen laut .repeatCount  
Kumulationsschrittweite laut .calcMode  
Ereignis des Startes eines Durchlaufes bzw. Kumulation um Maximalwert ist onrepeat.  
Wert der Eigenschaft .repeatCount muss > 1 sein  
Kumulation nicht aktiv:  
Nach jedem Durchlauf laut .repeatCount erfolgt Rücksetzen der Style-Eigenschaft des  
Elementes auf den Zustand vor dem Durchlaufbeginn.  
Es wird also die Style-Eigenschaft nicht wertmäßig kumuliert.  
Das Element wird also laut .repeatCount mehrmals animiert.  
Pro Durchlauf wird für die Eigenschaft laut .attributName der Wert  
ab Startwert  
in Schrittweite laut .calcMode  
um den maximalen Wert laut .by  
erhöht.  
Kumulation aktiv:  
Die Style-Eigenschaft des Elementes wird pro Durchlauf wertmäßig laut  
.repeatCount um den Wert laut .by kumuliert.  
Kumulationsschrittweite laut .calcMode  
Das Element wird also genau 1 mal animiert.  
Mit Beginn des Startes des Elementes wird für die Eigenschaft laut .attributName  
der Wert ab Startwert  
in Schrittweite laut .calcMode  
um den maximalen Wert aus dem Produkt aus  
.by mal Anzahl der Wiederholungen  
z.B. laut .repeatCount  
erhöht.



Es wird also über **alle** Wiederholungen der Wert kumuliert.  
 siehe .additive .calcMode  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft .width) nach rechts:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function SpanInhaltInit(Kette)
{
    // Zustand
    ID_Span1.innerText = "";

    // Kumulationsart
    ID_Span2.innerText =Kette;

    // Zaehler auf 1
    ID_Span3.innerText ="1";

    // DIV-Text festlegen
    ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
    if (Kette == "sum")
    {
        // Wertkumulation von .width
        ID_Div.innerText += " genau 1x kumulativ um ";
        ID_Div.innerText += ID_Animate.by
        ID_Div.innerText += " mal "
        ID_Div.innerText += ID_Animate.repeatCount;
    }
    else
    {
        // keine Wertkumulation von .width
        ID_Div.innerText += ID_Animate.repeatCount;
        ID_Div.innerText += " mal um ";
        ID_Div.innerText += ID_Animate.by
    }

    ID_Div.innerText += " aus";
}

function KumulationAus()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("none");
    ID_Animate.accumulate="none";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function KumulationEin()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("sum");
    ID_Animate.accumulate="sum";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function Anzeige()
{
    ID_Span1.innerText = "Animation ist beendet ";
}
```



```

</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    <t:ANIMATE>

animierter DIV
<DIV ID="ID_Div"
    CLASS="time_line_klasse"
    STYLE= "position:absolute;
        top:125px;
        left:25px;
        height:100px;
        width:125px;
        border:solid black 2px;
        "
    >
</DIV>
<BR>

Zustand der Animation:
<SPAN ID="ID_Span1"></SPAN>
<BR>

Kumulationsart:
<SPAN ID="ID_Span2"></SPAN>
<BR>

Durchlaufzaehler:
<SPAN ID="ID_Span3"></SPAN>
<BR>

<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
</BODY>
</HTML>

```

.action                      Url des Servers und des dortigen Dokumentes bei Formular

Beispiel 1:

```

<HTML>
    <FORM ACTION="http://www.test.de/sample.asp" METHOD="POST">
        <SELECT NAME="Flavor">
            <OPTION VALUE="Test1"> Test1
            <OPTION VALUE="Test2"> Test2
            <OPTION VALUE="Test3" SELECTED> Test3
        </SELECT>
        <INPUT TYPE=SUBMIT>
    </FORM>
</HTML>

```

Beispiel 2:

```

<FORM ACTION="mailto:test@test.de" METHOD=GET>
    <INPUT NAME=subject TYPE=hidden
        VALUE="Testt%20Product%20Information%20Anforderung"
    >
        volle Mailadresse eingeben
    <BR>
    <TEXTAREA NAME=body COLS=40></TEXTAREA>
    <INPUT TYPE=submit VALUE="absenden "
</FORM>

```

Beispiel 3:

```

<FORM ACTION="test.htm">
    <INPUT TYPE="submit" VALUE="Gehe zu test.htm">

```



&lt;/FORM&gt;

Beispiele für mailto-Formen:

Standard	mailto:aaa@bbb
carbon copy	mailto:aaa@bbb?cc=mailto:ccc@ddd (mit Kopie an anderen Empfänger)
blind copy	mailto:aaa@bbb?bcc=mailto:ccc@ddd (mit Blind-Kopie an anderen Empfänger)
carbon copy und Betreff	mailto:aaa@bbb?cc=mailto:ccc@ddd&subject=betreff_text
carbon copy und Betreff und Mailtext	mailto:aaa@bbb?cc=mailto:ccc@ddd&subject=betreff_text&body=mail_text

Betreff/Mailtext analog für Standard und blind copy

Beispiel für Spam-Schutz einer Email-Adresse (Schutz vor Missbrauch nach dem Scannen der Email-Adresse durch Suchmaschine):

```
// test@test.de wird per Script und nicht als HTML im Quellcode kodiert

var user_name="test";
var host_name="test.de";

document.write(      '<A HREF="mailto:'
                    + user_name
                    + '@'
                    + host_name
                    + '">'
                    + user_name
                    + '@'
                    + host_name
                    + '</A>'
                    );
```

Beispiel für Festplattenverzeichnis unter Windows auslesen:

```
<BODY>
<FORM ACTION="file:///C:/"
      METHOD="GET"
>
      <INPUT TYPE="submit" VALUE="Root von LW C anzeigen!">
</FORM>
</BODY>
```

.activeDur                    totale Dauer der Timeline in Sekunden  
                              inklusive aller Wiederholungen und Autoreverse  
                              siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
      .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
      function DauerAnzeigen()
      {alert('Totale Dauer = ' + ID_Timeline.currTimeState.activeDur + ' Sekunden');}
</SCRIPT>
</HEAD>
<BODY>
      <!-- 3 * 3 * 2 = 18 Sekunden-->
      <t:EXCL ID="ID_Timeline"
            DUR="3"
            REPEATCOUNT="3"
            AUTOREVERSE="true"
      >
            <DIV ID="ID_Div"
                  CLASS="time_line_klasse"
                  BEGIN="0"
                  DUR="1"
            >
                  1
            </DIV>
      </t:EXCL>
```



```

<BR>
<BUTTON          ID="ID_Button"
                 onclick=" DauerAnzeigen()"
>
        Klick mich
</BUTTON>
</BODY>
</HTML>

```

**.activeElement**                      Referenz auf dasjenige Objekt im Dokument , das Focus hat  
    IE unter 5.x                      Zeiger auf BODY-Objekt  
    ab IE 5.x                      null

Beispiel:

```
var Zeiger = document.activeElement
```

**.activeTime**                              aktueller Zeitpunkt in Sekunden in der Timeline ab Start  
    inklusive aller Repeats, autoReverse  
    siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
        function WindowOnLoad()
        {
                // Timer mit Intervall von 100 Millisekunden
                window.setInterval(Anzeigen, 100);
        }

        function Anzeigen()
        {
                ID_Div2.innerHTML = "simpleTime:&nbsp;"
                                + (ID_Animation.currTimeState.simpleTime);

                ID_Div3.innerHTML = "segmentTime:&nbsp;"
                                + (ID_Animation.currTimeState.segmentTime);

                ID_Div4.innerHTML = "activeTime:&nbsp;"
                                + (ID_Animation.currTimeState.activeTime);
        }

        function Wiederholen()
        {
                ID_Div1.innerHTML = "repeatCount:&nbsp;"
                                + (ID_Animation.currTimeState.repeatCount + 1 );
                                // repeatCount ab 0, aber Anzeige ab 1
        }
        window.onload = WindowOnLoad;           // mit () so sofort ausgeführt
                                                // ohne () so reiner Zeiger

</SCRIPT>
</HEAD>
<BODY>
        <DIV>
                <DIV ID="ID_Div1">repeatCount:&nbsp;1</DIV>
                <DIV ID="ID_Div2">simpleTime:&nbsp;0</DIV>
                <DIV ID="ID_Div3">segmentTime:&nbsp;0</DIV>
                <DIV ID="ID_Div4">activeTime:&nbsp;0</DIV>
        </DIV>

        <DIV ID="ID_DivTimeLine"
                CLASS="time"
                STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
        >
        </DIV>
        <t:ANIMATEMOTION          ID="ID_Animation"
                                TARGETELEMENT="ID_DivTimeLine"
                                TO="340,40"
                                DUR="2"
                                AUTOREVERSE="true"
                                REPEATCOUNT="5"

```



```

        >
        </t:ANIMATEMOTION>
    </BODY>
</HTML>

```

**.activeTrack** Referenz auf aktives Objekt playItem, das gerade in der Playliste aktiv ist  
 Achtung: Playliste muss aktiv sein, sonst wird Fehler erzeugt  
 Aktivstatus prüfen per Eigenschaft .isActive  
 Track entspricht playItem Objekt  
 Trackliste liegt in der Behavior-Collection .style.time2.playList als Playliste:  
 Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
 Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei  
 aktiver Track in der Liste: wird abgespielt  
 Track in der Liste auf aktiv setzen: Abspielen des Tracks sofort starten

Beispiel 1:

```

object.playList.activeTrack = object.playList.item(index);
var Zeiger = object.playList.aktiveTrack;

index Integer, ab 0

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: " + ID_Media.playList.activeTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playList.activeTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playList.activeTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playList.activeTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playList.activeTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playList.activeTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playList.activeTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playList.activeTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()
    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }
    }
</SCRIPT>

```





```

</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA          ID="ID_Media"
                      SRC="test.asx"
                      BEGIN="indefinite"
                      TIMEACTION="visibility"
                      onend="ButtonUpdate();"
                      ontrackchange="AnzeigeUpdate();"
                      onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON          ID="ID_Button1"
                    onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON          ID="ID_Button2"
                    onclick="ID_Media.playlist.nextTrack();"
    >
        naechster Track
    </BUTTON>
    <BUTTON          ID="ID_Button3"
                    onclick="ID_Media.playlist.prevTrack();"
    >
        vorhergehender Track
    </BUTTON>
    <BUTTON          ID="ID_Button4"
                    onclick="ID_Media.endElement(); AnzeigeLoeschen();"
    >
        Stop
    </BUTTON>
</BODY>
</HTML>

```

**.additive** numerische Werte von Eigenschaften eines Elementes ersetzen oder zu den numerischen Werten der gleichnamigen Eigenschaften anderer Elemente addieren  
während der Elemente-Animation auf der Timeline  
Wert der Eigenschaft: numerisch aber Einheiten wie px, in, cm, mm, pt, pc können Teil des Wertes sein  
gültige Eigenschaften sind z.B. `.values` oder `.by`  
sinnvoll für Kombinationen von Elementen (auch bei Wiederholung eines Elementes)  
Element darf nicht aktiv sein, wenn `.additive` definiert werden soll:  
Element erst danach starten.  
Achtung: Im Objekt `animatecolor` (`t:ANIMATECOLOR`) ist `.values` auch als Farbliste definiert !  
siehe `.accumulate` `.by` `.attributeName`  
siehe Objekt `currentTimeState` und Behavior `.style.time2`

**.align** Ausrichtung  
"center" default nur bei TH-Tag  
"justify"  
"left" default außer bei TH-Tag  
"right"

Beispiel für Erzeugung einer Tabelle in Script per HTML, TOM und DOM:

```
<HTML>
```



```

<HEAD>
<SCRIPT LANGUAGE="JScript">
    function TabelleFuellen()
    {
        var Zeile;
        var Zelle;

        //+++++ Tabellenrahmen und Rahmenfarbe ++++++
        ID_Tabelle.border = 1;
        ID_Tabelle.bgColor = "magenta";

        //+++++ TabellenKopf füllen ++++++
        //----- erzeugen
        var TabellenKopf = ID_Tabelle.createTHead();

        //----- Hintergrundfarbe
        TabellenKopf.bgColor = "blue";

        //----- mit 1 Zeile
        Zeile = TabellenKopf.insertRow();

        // Zeile mit 1 Zellenfüllen
        Zelle = Zeile.insertCell();
        Zelle.align = "center";
        Zelle.style.fontWeight = "bold";
        Zelle.innerText = "Tabellenkopf Zelle";

        //+++++ Tabellenbody 0 füllen ++++++
        //----- wurde per HTML-Kodierung im BODY erzeugt
        //----- Hintergrundfarbe
        ID_TBody0.bgColor = "green";

        //----- 2 Zeilen mit je 1 Zelle erzeugen
        for ( var ZeilenZahler = 0 ; ZeilenZahler < 2; ZeilenZahler ++ )
        {
            //----- Zeile erzeugen
            Zeile = ID_TBody0.insertRow();

            //----- Zelle in der Zeile erzeugen
            Zelle = Zeile.insertCell();

            // und füllen
            Zelle.innerText = "TBODY 0 Zeile " + ZeilenZahler + " Zelle"
        }

        //+++++ Tabellenbody 1 füllen ++++++
        //----- wurde per HTML-Kodierung im BODY erzeugt
        //----- Hintergrundfarbe
        ID_TBody1.bgColor = "yellow";

        //----- 2 Zeilen mit je 1 Zelle erzeugen
        for ( var ZeilenZahler = 0 ; ZeilenZahler < 2; ZeilenZahler ++ )
        {
            //----- Zeile erzeugen
            Zeile = ID_TBody1.insertRow();

            //----- Zelle in der Zeile erzeugen
            Zelle = Zeile.insertCell();

            // und füllen
            Zelle.innerText = "TBODY 1 Zeile " + ZeilenZahler + " Zelle"
        }

        //+++++ TabellenFuss füllen ++++++
        //----- erzeugen
        var TabellenFuss = ID_Tabelle.createTFoot();

        //----- mit Hintergrundfarbe
        TabellenFuss.bgColor = "brown";

        //----- mit 1 Zeile
        Zeile = TabellenFuss.insertRow();

        // Zeile mit 1 Zellenfüllen

```



```

        Zelle                = Zeile.insertCell();
        Zelle.align          = "center";
        Zelle.style.fontWeight = "bold";
        Zelle.innerText      = "Tabellenfuss Zelle";
        Zelle.colSpan        = "1";
        Zelle.id              = "ZelleImTabellenFuss";

        // +++++ TabellenCaption füllen +++++
        // ----- erzeugen
        var TabellenCaption = ID_Tabelle.createCaption();

        // ----- und füllen
        TabellenCaption.align = "bottom";
        TabellenCaption.style.fontSize = "10";
        TabellenCaption.innerText = "TabelleCaption"
    }

    function ZelleImTabellenFussAendern()
    {ZelleImTabellenFuss.innerText = "Tabellenfuss Zelle neu"}
</SCRIPT>
</HEAD>
<BODY onload="TabelleFuellen();">
    <! --- Tabelle leer mit allen TBODY erzeugen, aber komplett ohne Daten, da sonst nicht manipulierbar !!! --->
    <TABLE ID="ID_Tabelle"
        BORDER
        BGCOLOR="gray"
    >
        <TBODY ID="ID_TBody0"></TBODY>
        <TBODY ID="ID_TBody1"></TBODY>
    </TABLE>
    <BR>
    Tabelle mit HTML und TOM erzeugt
    <BR>
    <BUTTON onclick="ZelleImTabellenFussAendern();">Zelle im Tabellenfuss aendern</BUTTON>
</BODY>
</HTML>

```

**.aLink**                      Farbe eines ALinks im Dokument  
 document.body.aLink  
 "#rrggbb" oder vordefinierte Farbname

**.alinkColor**              Farbe aller aktiven Links im Dokument  
 document.body.alinkColor  
 "#rrggbb" oder vordefinierte Farbname

**.allowTransparency**        Transparenz ein/aus  
 Transparenz: z.B. Hintergrundfarbe des Elternobjektes wird zur Hintergrundfarbe des Frames  
                                 Achtung: Wenn Frame-eigenes STYLE-Attribut mit Wert für background-color  
     also    STYLE="background-color: farb\_bezeichner"  
     kodiert wurde, so wird die Transparenz ignoriert und die  
     Hintergrundfarbe laut STYLE verwendet

Beispiel:

```

<BODY STYLE="background-color: red">

    // roter Hintergrund im Frame laut Eltern = Body
    <IFRAME ID="Frame1" SRC="transparentBody.htm" allowTransparency="true"></IFRAME>

    // roter Hintergrund im Frame laut Eltern wird durch STYLE-Wert überschrieben
    //                                  also grün angezeigt
    <IFRAME ID="Frame2" SRC="transparentBody.htm" allowTransparency="true"
        STYLE="background-color: green">
    </IFRAME>

    // roter Hintergrund im Frame laut Eltern nicht angezeigt, sondern Standard = weiss
    <IFRAME ID="Frame3" SRC="transparentBody.htm"></IFRAME>

    // roter Hintergrund im Frame laut Eltern nicht angezeigt, auch nicht Standard = weiss
    //                                  sondern laut STYLE-Wert, also grün
    <IFRAME ID="Frame4" SRC="transparentBody.htm" STYLE="background-color: green">
    </IFRAME>
</BODY>

```

**.alt**                              Alternativer Text als Tooltip  
 Beispiel:



```
<IMG SRC="http://www.test.de/test.gif" ALT="Ein Testbild ">
```

**.altHTML** HTML-Script, der ausgeführt wird, wenn Objekt nicht geladen werden kann

Beispiel:

```
zeiger_auf_applet.altHTML="<B>Applet nicht geladen !<B>";
```

**.altKey**

ALT-Tasten-Status

Objekt event

false ALT-Taste ist nicht gedrückt

true ALT-Taste ist gedrückt

Beispiel:

```
<HEAD>
<SCRIPT>
function InnerTextSetzen(Zeiger, Kette)
{Zeiger.innerText=Kette;}

function init()
{
    InnerTextSetzen(ID_Span1,'false');
    InnerTextSetzen(ID_Span2,'false');
}
function AltDown()
{
    if (event.altLeft)
    { InnerTextSetzen(ID_Span1,'true'); }
    else
    {
        if (event.altKey)
        { InnerTextSetzen(ID_Span2,'true'); }
    }
}

function AltUp()
{
    if (!event.altKey)
    {
        InnerTextSetzen(ID_Span1,'false');
        InnerTextSetzen(ID_Span2,'false');
    }
}
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="AltDown();" onkeyup="AltUp();">
<TABLE>
<TR>
<TD><I>Linke ALT-Taste gedrueckt</I></TD>
<TD><I>Rechte ALT-Taste gedrueckt</I></TD>
</TR>
<TR>
<TD ALIGN="center">
<SPAN ID="ID_Span1"></SPAN>
</TD>
<TD ALIGN="center">
<SPAN ID="ID_Span2"></SPAN>
</TD>
</TR>
</TABLE>
</BODY>
```

**.altLeft**

linke ALT-Taste-Status

Objekt event

nicht für Win9x

nur für Dokument, das den Fokus hat

false linkeALT-Taste ist nicht gedrückt

true linke ALT-Taste ist gedrückt

Beispiel:

```
<HEAD>
<SCRIPT>
function InnerTextSetzen(Zeiger, Kette)
{Zeiger.innerText=Kette;}

function init()
{
```



```

        InnerTextSetzen(ID_Span1,'false');
        InnerTextSetzen(ID_Span2,'false');
    }
    function AltDown()
    {
        if (event.altLeft)
        { InnerTextSetzen(ID_Span1,'true'); }
        else
        {
            if (event.altKey)
            { InnerTextSetzen(ID_Span2,'true'); }

        }
    }

    function AltUp()
    {
        if (!event.altKey)
        {
            InnerTextSetzen(ID_Span1,'false');
            InnerTextSetzen(ID_Span2,'false');
        }
    }
}
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="AltDown();" onkeyup="AltUp();">
    <TABLE>
        <TR>
            <TD><I>Linke ALT-Taste gedrueckt</I></TD>
            <TD><I>Recchte ALT-Taste gedrueckt</I></TD>
        </TR>
        <TR>
            <TD ALIGN="center">
                <SPAN ID="ID_Span1"></SPAN>
            </TD>
            <TD ALIGN="center">
                <SPAN ID="ID_Span2"></SPAN>
            </TD>
        </TR>
    </TABLE>
</BODY>

```

.appName	<p>Codename des Browsers per navigator Objekt  navigator.appCodeName  "Mozilla" Default  geliefert von IE und NS  sonst browserspezifisch  Achtung Opera-Browser gibt sich als IE aus !</p>
.appCodeName	<p>Codename des Browsers  siehe Objekt window.clientInformation  "Mozilla" Default  geliefert von IE und NS  sonst browserspezifisch  Achtung Opera-Browser gibt sich als IE aus !</p>
APPLICATION	<p>Umgebung des Frames mit Vertrauen-Status im Rahmen des Browser-Sicherheitsmodells  Attribut wird nicht vererbt an Kinder-Frame  "no" Default, keine Vertrautheit  "yes" Vertrautheit</p>
.appMinorVersion	<p>Update der Browserversion per navigator Objekt  z.B. 1 des IE 4.1  navigator.appMinorVersion</p>
.appMinorVersion	<p>Browser-Versionsnummer: Unternummer der Hauptnummer  z.B. 1 des IE 4.1  siehe Objekt window.clientInformation</p>
.appName	<p>Name des Browsers per navigator Objekt  navigator.appName  "Microsoft Internet Explorer" Default  "Netscape" ist der Netscape Navigator</p>

Beispiel:

```
<SCRIPT LANGUAGE="JavaScript">
```



```

<!--      var ie4=0;
           if (      (navigator.appName.indexOf("Explorer")>=0)
           &&      (navigator.appVersion.indexOf('4.0')    >=0)
           )
           ie4=1;
// -->
</SCRIPT>

```

für Netscape bitte "Netscape" verwenden und entsprechende Version

**.appName**                      Browsername  
siehe Objekt window.clientInformation  
"Microsoft Internet Explorer"    Default  
"Netscape"                      ist der Netscape Navigator

**.appVersion**                      Betriebssystem-Plattform und Browserversion per navigator Objekt  
navigator.appVersion  
Aufbau  
                 clientVersion (platform; information; extraInformation)  
**oder**    5.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)  
Hinweis: Haupt-Versionsnummer ermittelbar per parseFloat(navigator.appVersion)  
                 z.B. 4 des IE 4.1

Beispiel:

```

<SCRIPT LANGUAGE="JavaScript">
<!--      var ie4=0;
           if (      (navigator.appName.indexOf("Explorer")>=0)
           &&      (navigator.appVersion.indexOf('4.0')    >=0)
           )
           ie4=1;
// -->
</SCRIPT>

```

für Netscape bitte "Netscape" verwenden und entsprechende Version

**.appVersion**                      Plattform und Browserversion  
Aufbau  
                 clientVersion (platform; information; extraInformation)  
**oder**    5.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)  
Hinweis: Haupt-Versionsnummer ermittelbar per parseFloat(navigator.appVersion)  
                 z.B. 4 des IE 4.1  
siehe Objekt window.clientInformation

**.archive**                      Zeichenkette für Archive-Funktionalität eines Objektes

**.arguments**                      Zeiger auf das Script-Objekt arguments  
alle Eigenschaften des Script-Objektes arguments können referenziert werden  
arguments nur verfügbar während der Ausführung der Funktion  
siehe Script-Objekt Funktion und Script-Objekt arguments und Anweisung function

Beispiel:

```

function Test1()
{
    var AnzahleArgumente = arguments.length;
    var Kette = "Anzahl Argumente = " + AnzahleArgumente + "\n";

    for ( var i = 0; i < AnzahleArgumente; i++)
    {Kette = "Argument[" + i + "] = " + arguments[i] + "\n"; }

    alert(Kette);
}

function Test2(Wert1,Wert2)
{
    var AnzahleArgumente = arguments.length;
    var Kette = "Anzahl Argumente = " + AnzahleArgumente + "\n";

    for ( var i = 0; i < AnzahleArgumente; i++)
    {Kette = "Argument[" + i + "] = " + arguments[i] + "\n"; }

    alert(Kette);
}

Test1();
Test2(10,20);

```





**.AtEndOfLine** Satzzeiger bezüglich Zeilenende (End Of Line (EOL)) per newline-Zeichen  
 wird mit jedem Lesen oder Schreiben verändert  
 Hinweis: newline-Zeichen per `.WriteLine` oder `.WriteBlankLines()` schreibbar  
 immer bei satzweisem Schreiben/Lesen  
 möglich beim zeichenweisem Schreiben/Lesen

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
var Kette = "";
while (!DateiOffen.AtEndOfLine)
{Kette = Kette + DateiOffen.ReadLine() + "\n";}
DateiOffen.Close();
alert(Kette);
```

**.AtEndOfStream** Satzzeiger bezüglich Dateiende, also Textstream-Ende (End Of Stream (EOS))  
 wird mit jedem Lesen oder Schreiben verändert

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
var Kette = "";
while (!DateiOffen.AtEndOfStream)
{Kette = Kette + DateiOffen.ReadLine() + "\n";}
DateiOffen.Close();
alert(Kette);
```

**ATOMICSELECTION** Selektierbarkeit des Objektes einstellen  
 false für nicht selektierbar, Default  
 true für selektierbar, aber Dokumentdarstellung ist langsamer als bei false

**.attributeCount** Anzahl der mit dem MediaItem Objekt verbundenen Attribute liefern  
 Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende Methoden nutzbar:  
`.getItemInfo()`  
`.getAttributeName()`  
 Eigenschaften nutzbar:  
`.attributeCount`  
 Es ist also vorher die Eigenschaft `.playState` auf Werte > 10 zu prüfen.  
 siehe Behavior `.style.mediaBar`

Beispiel:

```
<DIV ID="ID_Div"
STYLE="behavior:url(#default#mediaBar)"
>
</DIV>
<INPUT TYPE=button
VALUE='abspielen von test.asx'
onclick="ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
ID_Div.disabledUI = true;
ID_Div.enabled = false;
"
>
```

**.attributeName** Bezeichner einer Style-Eigenschaft, also Name eines Attributes, für die Animation anhand dieses Attributes auf der Timeline  
 siehe `.accumulate` `.by` `.additive` `.calcMode`  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft `.width`) nach rechts:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function SpanInhaltInit(Kette)
{
// Zustand
ID_Span1.innerText = "";

// Kumulationsart
ID_Span2.innerText =Kette;
```



```

// Zaehler auf 1
ID_Span3.innerText ="1";

// DIV-Text festlegen
ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
if (Kette == "sum")
{
    // Wertkumulation von .width
    ID_Div.innerText += " genau 1x kumulativ um ";
    ID_Div.innerText += ID_Animate.by
    ID_Div.innerText += " mal "
    ID_Div.innerText += ID_Animate.repeatCount;
}
else
{
    // keine Wertkumulation von .width
    ID_Div.innerText += ID_Animate.repeatCount;
    ID_Div.innerText += " mal um ";
    ID_Div.innerText += ID_Animate.by
}

ID_Div.innerText += " aus";
}

function KumulationAus()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("none");
    ID_Animate.accumulate="none";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function KumulationEin()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("sum");
    ID_Animate.accumulate="sum";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function Anzeige()
{
    ID_Span1.innerText = "Animation ist beendet ";
}
</SCRIPT>
</HEAD>
<BODY>
<t:ANIMATE      ID="ID_Animate"
                TARGETELEMENT="ID_Div"
                ATTRIBUTENAME="width"
                BY="150px"
                DUR="3"
                REPEATCOUNT="3"
                BEGIN="indefinite"
                FILL="freeze"
                onend="Anzeige();"
                onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
                >
<t:ANIMATE>

animierter DIV
<DIV      ID="ID_Div"
          CLASS="time_line_klasse"
          STYLE= "position:absolute;
                top:125px;

```



```

        left:25px;
        height:100px;
        width:125px;
        border:solid black 2px;
        "
    >
</DIV>
<BR>

Zustand der Animation:
<SPAN ID="ID_Span1"></SPAN>
<BR>

Kumulationsart:
<SPAN ID="ID_Span2"></SPAN>
<BR>

Durchlaufzaehler:
<SPAN ID="ID_Span3"></SPAN>
<BR>

<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
</BODY>
</HTML>

```

.Attributes                      Attribute des Ordner

Beispiel:

```

var OrdnerName                      = "c:\\windows\\desktop\\";
var DateiSystem                    = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                          = DateiSystem.GetFolder(OrdnerName);
var Ordner_Attribute               = Ordner.attributes;

// auf gesetztes Archivattribut prüfen
if (Ordner_Attribute && 32)
{
    // löschen
    Ordner_Attribute -= 32;
}
else
{
    // setzen
    Ordner_Attribute += 32;
}

```

.Attributes                      Attribute der Datei

Beispiel:

```

var DateinameMitPfad               = "c:\\test.txt";
var DateiSystem                    = new ActiveXObject("Scripting.FileSystemObject");
var Datei                          = DateiSystem.GetFile(DateinameMitPfad);
var Datei_Attribute               = Datei.attributes;

// auf gesetztes Archivattribut prüfen
if (Datei_Attribute && 32)
{
    // löschen
    Datei_Attribute -= 32;
}
else
{
    // setzen
    Datei_Attribute += 32;
}

```

.author

Name des Autor der Media-Datei auf der Timeline

bei Advanced Stream Redirector (ASX)-Datei: Es wird der Text von AUTHOR des aktiven Eintrages geliefert und nicht den der Datei.

Track entspricht playItem Objekt laut object.playlist.item() bzw. object.playlist.aktiveTrack  
Trackliste liegt in der Behavior-Collection .style.time2.playlist als Playliste:

Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.

Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:

```
var Kette = object.playlist.item(index).author;
```



index Integer, ab 0

```
var Kette = object.playList.aktiveTrack.author;
```

Beispiel 2:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO          ID="ID_Video"
                     SRC="test.wmv"
                     STYLE="position:absolute;top:50px;height:100px"
    >
    </t:VIDEO>
    <SPAN ID="ID_Span"
        STYLE="position:absolute;top:165px;"
    >
    </SPAN>
    <BUTTON          ID="ID_Button"
                    onclick="ID_Span.innerText= ID_Video.author"
    >
        Klick
    </BUTTON>
</BODY>
</HTML>
```

Beispiel 3:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: " + ID_Media.playList.aktiveTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playList.aktiveTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playList.aktiveTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playList.aktiveTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playList.aktiveTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playList.aktiveTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playList.aktiveTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playList.aktiveTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()
    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
    }
</SCRIPT>
</HTML>
```



```

        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA          ID="ID_Media"
                     SRC="test.asx"
                     BEGIN="indefinite"
                     TIMEACTION="visibility"
                     onend="ButtonUpdate();"
                     ontrackchange="AnzeigeUpdate();"
                     onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN  ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN  ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN  ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN  ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN  ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN  ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN  ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN  ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON          ID="ID_Button1"
                     onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON          ID="ID_Button2"
                     onclick="ID_Media.playlist.nextTrack();"
    >
        naechster Track
    </BUTTON>
    <BUTTON          ID="ID_Button3"
                     onclick="ID_Media.playlist.prevTrack();"
    >
        vorhergehender Track
    </BUTTON>
    <BUTTON          ID="ID_Button4"
                     onclick="ID_Media.endElement(); AnzeigeLoeschen();"
    >
        Stop
    </BUTTON>
</BODY>
</HTML>

```

.autocomplete      Status des Autocomplete (Autovervollständigung) zum Formular  
 Einstellung der Autovervollständigung im Browser per  
 Extras-Internetoptionen-Inhalte  
**Achtung: Missbrauch bei input password Objekt möglich !**  
 "on" oder "off"

Beispiel:

```
<INPUT TYPE="password" AUTOCOMPLETE="off">
```

Hinweise: AutoComplete betrifft folgende Werte:

laut VALUE-Attribut von Textfeldern im Formular, aber NUR, wenn diese  
 Felder auch das NAME-Attribut besitzen, wobei VALUE-Werte  
 automatisch für die automatische Wiederverwendung im Browser  
 gespeichert werden (auch bei Password-Feldern !!!)



Werte aus dem vCard-Schema laut dem Objekt navigator.userProfile verwendet  
(siehe dort)

**vCard.XXX**

vCard.Business.City  
vCard.Business.Country  
vCard.Business.Fax  
vCard.Business.Phone  
vCard.Business.State  
vCard.Business.StreetAddress  
vCard.Business.URL  
vCard.Business.Zipcode  
vCard.Cellular  
vCard.Company  
vCard.Department  
vCard.DisplayName  
vCard.Email  
vCard.FirstName  
vCard.Gender  
vCard.Home.City  
vCard.Home.Country  
vCard.Home.Fax  
vCard.Home.Phone  
vCard.Home.State  
vCard.Home.StreetAddress  
vCard.Home.Zipcode  
vCard.Homepage  
vCard.JobTitle  
vCard.LastName  
vCard.MiddleName  
vCard.Notes  
vCard.Office  
vCard.Pager

Die Nutzung von AutoComplete sollte reiflich überlegt sein.

Das nachträgliche Löschen von per AutoComplete automatisch gespeicherte Werte  
ist in den Internet-Optionen des IE vollziehbar.

.autoReverse automatische Rückwärtsanimation eines Elementes nach dem kompletten Ende der Vorwärtsanimation  
des Elementes auf der Timeline  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
    function HochZaehlen()
    {
        ID_Div1.innerHTML = "Zahler: "
                        + (ID_Animation.currTimeState.repeatCount + 1);
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV ID="ID_Div1"></DIV>
    <DIV ID="ID_Div2"
        CLASS="time_line_klasse"
        STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
    >
    </DIV>
    <t:ANIMATEMOTION
        ID="ID_Animation"
        TARGETELEMENT="ID_Div2"
        TO="150,0"
        DUR="1"
        AUTOREVERSE="true"
        REPEATCOUNT="5"
        onrepeat="HochZaehlen()"
    >
    </t:ANIMATEMOTION>
</BODY>
</HTML>
```





.AvailableSpace	Freien Speicher in Bytes auf Laufwerk ermitteln
Beispiel:	<pre> var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject"); var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:"); var Laufwerk          = DateiSystem.GetDrive(Laufwerk_Bezeichnung); var Laufwerk_VolumeName = Laufwerk.VolumeName; var Laufwerk_FreierPlatz = Laufwerk.AvailableSpace; alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_FreierPlatz); </pre>
.availHeight	verfügbare Höhe des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
.availHeight	verfügbare Höhe des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar Behavior .style.clientCaps
.availWidth	verfügbare Breite des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
.availWidth	verfügbare Breite des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar Behavior .style.clientCaps
.background	Hintergrundbild im Dokument document.body.background
.background	Hintergrundbild eines Objektes
.balance	Balance im Objekt BGSOUND -10 bis +10 -10 ganz links +10 ganz rechts 0 genau mittig, default
Beispiel:	<pre> &lt;HEAD&gt; &lt;SCRIPT&gt; function KanalVerteilung(Wert) { ID_bgsound.balance = Wert;}  function GenauEinmal() {     ID_bgsound.loop = 1;     ID_bgsound.src = ID_bgsound.src; // restart }  function Endlos() {     ID_bgsound.loop = -1;     ID_bgsound.src = ID_bgsound.src; // restart } &lt;/SCRIPT&gt; &lt;BGSOUND ID="ID_bgsound" SRC="sound.wav"&gt; &lt;/HEAD&gt; &lt;BODY&gt; &lt;BUTTON onclick="GenauEinmal()"&gt;&lt;/BUTTON&gt; &lt;BUTTON onclick="Endlos()"&gt;&lt;/BUTTON&gt; &lt;BUTTON onclick="KanalVerteilung(-10)"&gt;&lt;/BUTTON&gt; &lt;BUTTON onclick="KanalVerteilung(10)"&gt;&lt;/BUTTON&gt; &lt;BUTTON onclick="KanalVerteilung (0)"&gt;&lt;/BUTTON&gt; &lt;B&gt;Volume control:&lt;/B&gt;&amp;nbsp; &lt;INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" onpropertychange="ID_bgsound.volume=-10;" &gt;Mute  &lt;INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" onpropertychange="ID_bgsound.volume=-7;" &gt;25% Volume  &lt;INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED onpropertychange="ID_bgsound.volume=-5;" &gt;50% Volume  &lt;INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" onpropertychange="ID_bgsound.volume=-2;" &gt;75% Volume </pre>



```

< INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=0;"
        >100% Volume
</BODY>

```

.Banner  
Text laut BANNER des aktiven Eintrages in einer Advanced Stream Redirector (ASX) –Datei  
anhand Behavior-Collection .style.time2.playList  
ab IE 6.x  
Track entspricht playItem Objekt  
Trackliste liegt in der Behavior-Collection .style.time2.playList als Playliste:  
Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline  
aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:  
var Kette = object.playList.aktiveTrack.Banner;

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: " + ID_Media.playList.aktiveTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playList.aktiveTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playList.aktiveTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playList.aktiveTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playList.aktiveTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playList.aktiveTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playList.aktiveTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playList.aktiveTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()
    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

<t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"

```



```

TIMEACTION="visibility"
onend="ButtonUpdate();"
ontrackchange="AnzeigeUpdate();"
onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
>
</t:MEDIA>

<SPAN ID="ID_Span1">Titel:</SPAN>
<BR>
<SPAN ID="ID_Span2">Autor:</SPAN>
<BR>
<SPAN ID="ID_Span3">Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span4">Copyright:</SPAN>
<BR>
<SPAN ID="ID_Span5">Filename:</SPAN>
<BR>
<SPAN ID="ID_Span6">Banner:</SPAN>
<BR>
<SPAN ID="ID_Span7">Banner Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

<BR>
<BUTTON ID="ID_Button1"
onclick="ID_Media.beginElement(); ButtonUpdate();"
>
    Start
</BUTTON>
<BUTTON ID="ID_Button2"
onclick="ID_Media.playlist.nextTrack();"
>
    naechster Track
</BUTTON>
<BUTTON ID="ID_Button3"
onclick="ID_Media.playlist.prevTrack();"
>
    vorhergehender Track
</BUTTON>
<BUTTON ID="ID_Button4"
onclick="ID_Media.endElement(); AnzeigeLoeschen();"
>
    Stop
</BUTTON>
</BODY>
</HTML>

```

**.BannerAbstract**      Text laut BANNER und dort ABSTRACT des aktiven Eintrages in einer Advanced Stream Redirector (ASX) –Datei anhand Behavior-Collection `.style.time2.playlist` ab IE 6.x  
Track entspricht `playItem` Objekt  
Trackliste liegt in der Behavior-Collection `.style.time2.playlist` als Playliste:  
Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:  

```
var Kette = object.playlist.aktiveTrack.BannerAbstract;
```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;

```



```

        ID_Button4.disabled=false;
    }
    else
    {
        ID_Button1.disabled=false;
        ID_Button2.disabled=true;
        ID_Button3.disabled=true;
        ID_Button4.disabled=true;
    }
}

function AnzeigeUpdate()
{
    ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
    ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
    ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
    ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
    ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
    ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
    ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
    ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
}

function AnzeigeLoeschen()
{
    ID_Span1.innerText = "Titel: ";
    ID_Span2.innerText = "Autor: ";
    ID_Span3.innerText = "Abstract: ";
    ID_Span4.innerText = "Copyright: ";
    ID_Span5.innerText = "Filename: ";
    ID_Span6.innerText = "Banner: ";
    ID_Span7.innerText = "Banner Abstract: ";
    ID_Span8.innerText = "Banner MoreInfo: ";
}
}
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"
        TIMEACTION="visibility"
        onend="ButtonUpdate();"
        ontrackchange="AnzeigeUpdate();"
        onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON ID="ID_Button1"
        onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Media.playlist.nextTrack();"
    >
        naechster Track

```



```

</BUTTON>
<BUTTON          ID="ID_Button3"
                  onclick="ID_Media.playlist.prevTrack();"
>
    vorhergehender Track
</BUTTON>
<BUTTON          ID="ID_Button4"
                  onclick="ID_Media.endElement(); AnzeigeLoeschen();"
>
    Stop
</BUTTON>
</BODY>
</HTML>

```

.BannerMoreInfo Text laut BANNER und dort MOREINFO des aktiven Eintrages in einer Advanced Stream Redirector (ASX) –Datei anhand Behavior-Collection .style.time2.playlist  
 ab IE 6.x  
 Track entspricht playItem Objekt  
 Trackliste liegt in der Behavior-Collection .style.time2.playlist als Playliste:  
 Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
 Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:  
 var Kette = object.playlist.aktiveTrack.BannerMoreInfo;

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: "          + ID_Media.playlist.activeTrack.title;
        ID_Span3.innerText = "Autor: "          + ID_Media.playlist.activeTrack.author;
        ID_Span3.innerText = "Abstract: "        + ID_Media.playlist.activeTrack.abstract;
        ID_Span4.innerText = "Copyright: "       + ID_Media.playlist.activeTrack.copyright;
        ID_Span5.innerText = "Filename: "       + ID_Media.playlist.activeTrack.src;
        ID_Span6.innerText = "Banner: "         + ID_Media.playlist.activeTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()
    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }

```



```

    }
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA          ID="ID_Media"
                     SRC="test.aspx"
                     BEGIN="indefinite"
                     TIMEACTION="visibility"
                     onend="ButtonUpdate();"
                     ontrackchange="AnzeigeUpdate();"
                     onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
</t:MEDIA>

<SPAN ID="ID_Span1">Titel:</SPAN>
<BR>
<SPAN ID="ID_Span2">Autor:</SPAN>
<BR>
<SPAN ID="ID_Span3">Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span4">Copyright:</SPAN>
<BR>
<SPAN ID="ID_Span5">Filename:</SPAN>
<BR>
<SPAN ID="ID_Span6">Banner:</SPAN>
<BR>
<SPAN ID="ID_Span7">Banner Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

<BR>
<BUTTON          ID="ID_Button1"
                  onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
</BUTTON>
<BUTTON          ID="ID_Button2"
                  onclick="ID_Media.playList.nextTrack();"
    >
        naechster Track
</BUTTON>
<BUTTON          ID="ID_Button3"
                  onclick="ID_Media.playList.prevTrack();"
    >
        vorhergehender Track
</BUTTON>
<BUTTON          ID="ID_Button4"
                  onclick="ID_Media.endElement(); AnzeigeLoeschen();"
    >
        Stop
</BUTTON>
</BODY>
</HTML>

```

**.BaseHref**                      Url des Dokumentes, das <OBJECT> .... </OBJECT> enthält (auch frame und iframe)  
ist meistens die Url des Dokumentes selbst

**.begin**                              Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction  
siehe Objekt currTimeState und Behavior .style.time2  
Wert im Time-Format  
z.B.                      "h:min:s.f"  
                                 id.event  
                                 id.event+zeit\_wert\_als\_string

Beispiele für Kodierung von Time in der Eigenschaft .begin:

```

object.begin="objekt_zeiger.begin+10s"           // warten bis Event "onbegin" zum Objekt laut
                                                    //          objekt_zeiger (z.B. laut ID-Attribut)
                                                    //          eintritt,
                                                    //          dann 10 Sekunden warten
                                                    //          dann Objekt laut object starten
object.begin="objekt_zeiger.focus+10s"           // warten bis Event "onfocus" zum Objekt laut
                                                    //          objekt_zeiger (z.B. laut ID-Attribut)

```





```

//      eintritt,
//      dann 10 Sekunden warten
//      dann Objekt laut object starten

object.begin="2; objekt_zeiger.click+1" // 2 Sekunden nach dem Laden des Elternobjektes von
//      object warten
//      dann auf das Ereignis click zum Objekt laut
//      objekt_zeiger warten
//      dann 1 Sekunde warten
//      dann Objekt laut object starten

```

Hinweis: object laut ID-Attribut des Behavior-Objektes von .style.time2.

```

"25:45:10" 25 Stunden, 45 Minuten, 10 Sekunden
"45:35"    45 Minuten, 35 Sekunden
"45:00.275" 45 Minuten, 0,275 Sekunde
"10.5"     10,5 Sekunden

```

Beispiele für Kodierung von .begin:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY BGCOLOR="white">
  <SPAN ID="ID_Span1"
        CLASS=time_line_klasse
        STYLE="COLOR:Red;"
        BEGIN="0" DUR="3" TIMEACTION="display"
  >
    <H3>Test2</H3>
  </SPAN>
  <SPAN ID="ID_Span2"
        CLASS=time_line_klasse
        STYLE="COLOR:Blue;" BEGIN="4" DUR="4"
        TIMEACTION="display"
  >
    <H3>Test2</H3>
  </SPAN>
  <H1 ID="ID_H1"
      CLASS=time_line_klasse
      BEGIN="8"
      DUR="indefinite"
      TIMEACTION="display"
  >
    Ende
  </H1>
</BODY>
</HTML>

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
  <P>Klick Button: 2 Sekunden nach Klick wird Span fuer 5 Sekunden angezeigt</P>
  <BR>
  <BUTTON ID="ID_Button">Klick mich</BUTTON>
  <BR>
  <SPAN ID="ID_Span"
        CLASS=time_line_klasse
        BEGIN="ID_Button.click+2"
        DUR="5"
        TIMEACTION="display"
        STYLE="COLOR:Red;"
  >

```



```

        </H3>Test</H3>
    </SPAN>
</BODY>
</HTML>

```

**.behavior** Verhalten des Scrollens des marquee Objektes

- "scroll" Default  
Scrollrichtung laut Eigenschaft .direction  
wenn Grenze des Elterncontainer erreicht wurde, wird an der entgegengesetzten Grenze wieder in den Elterncontainer reingescrollt
- "alternate" Scrollrichtung umkehren sobald Grenze des Elterncontainer erreicht wurde
- "slide" Scrollrichtung laut Eigenschaft .direction  
wenn Grenze des Elterncontainer erreicht wurde, wird rausgescrollt und dann Scrollen gestoppt

Beispiel:

```

<MARQUEE LOOP=1
  HEIGHT=200
  WIDTH=740
  SCROLLAMOUNT=10
  SCROLLDELAY=20
  BEHAVIOR="SLIDE"
  DIRECTION="DOWN"
  STYLE="position:absolute; top:0; left:10"
>
  Dieser Text scrollt
</MARQUEE>

```

**.bgColor** deprecated und durch STYLE-Attribut zu ersetzen  
Hintergrundfarbe des Objektes bzw. Dokumentes (document.body)  
#rrggbb Standard ist #0000FF  
vordefinierter Farbname (browserspezifisch)

**.bgProperties** Scroll-Eigenschaften des Hintergrundbildes im Dokument  
"" für scrollen  
"fixed" für nichtscrollen

Beispiel:

```

<BODY BACKGROUND="/images/test.gif" BGPROPERTIES="fixed">

```

**.blockDirection** Umfluss um ein Objekt  
"ltr" Umfluss von links nach rechts  
"rtl" Umfluss von rechts nach links

**.border** Rahmendicke in Pixel

**.borderColor** Borderfarbe (Rahmenfarbe)  
wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"  
Eigenschaft .border mit Wert 0  
#rrggbb  
vordefinierter Farbname (browserspezifisch)

**.borderColorDark** deprecated und durch Eigenschaft .borderColor zu ersetzen  
dunkle Farbe des 3D-Rahmens eines Objektes  
zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border)

**.borderColorLight** deprecated und durch Eigenschaft .borderColor zu ersetzen  
helle Farbe des 3D-Rahmens eines Objektes  
zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border)

**.bottom** untere Pixelposition des Rechteckes um ein Objekt  
auch textrectangle Objekt

Beispiel für Nicht-TextRectangle-Objekt:

```

<SCRIPT>
function Anzeigen(ObjektZeiger)
{
    var Rechteck = ObjektZeiger.getBoundingClientRect();

    alert(
        "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
        + "\n"
        + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
    );
}
</SCRIPT>
<P onclick="Anzeigen(this)">

```



Beispiel für TextRectangleObjekt:

```
<SCRIPT>
function Anzeigen(ObjektZeiger)
{
    var TextRectangleCollection = document.body.ObjektZeiger.TextRange.getClientRects();

    // .getClientRects() liefert immer Collection der textrectangle Objekte !!!

    var Rechteck = TextRectangleCollection[0];

    alert(    "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
            + "\n"
            + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
            );
}
</SCRIPT>
<BODY>
<DIV onclick="Anzeigen(this)">
    Test
</DIV>
</BODY>
```

.bottomMargin      Bottom Margin des Dokumentes in Pixel  
Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
Padding: Abstand des Objekthinhaltes zum Aussenrand

BOUNDARY      wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software

.boundingHeight      Höhe des Rechteckes in Pixel um den Textbereich des TextRange Objektes  
per textrange Objekt

Beispiel:

```
<SCRIPT>
function Anzeige(ZeigerAufTextarea)
{
    var FeldAllterTextArea = document.all.tags("TEXTAREA");

    // prüfen ob mindestens 1 TEXTAREA-Element im Body vorliegt
    if (FeldAllterTextArea != null)
    {
        var TextBereich = ZeigerAufTextarea.createTextRange();

        // prüfen ob Textbereich erzeugt wurde
        if (TextBereich != null)
        {alert(".boundingHeight = " + TextBereich.boundingHeight); }
    }
}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS=100 ROWS=2 onclick="Anzeige(this)">
.....
</TEXTAREA>
```

.boundingLeft      Abstand linker Rand des Rechteckes in Pixel um den Textbereich zur linkem Rand des Container-Objektes,  
in dem der Textbereich liegt  
per textrange Objekt

Beispiel:

```
<SCRIPT>
function Anzeige(ZeigerAufTextarea)
{
    var FeldAllterTextArea = document.all.tags("TEXTAREA");

    // prüfen ob mindestens 1 TEXTAREA-Element im Body vorliegt
    if (FeldAllterTextArea != null)
    {
        var TextBereich = ZeigerAufTextarea.createTextRange();

        // prüfen ob Textbereich erzeugt wurde
        if (TextBereich != null)
        {alert(".boundingLeft = " + TextBereich.boundingLeft); }
    }
}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS=100 ROWS=2 onclick="Anzeige(this)">
```



.....  
</TEXTAREA>

.boundingTop Abstand oberer Rand des Rechteckes in Pixel um den Textbereich zum oberen Rand des Container-Objektes, in dem der Textbereich liegt  
per textrange Objekt

Beispiel:

```
<SCRIPT>
function Anzeige(ZeigerAufTextarea)
{
    var FeldAllterTextArea = document.all.tags("TEXTAREA");

    // prüfen ob mindestens 1 TEXTAREA-Element im Body vorliegt
    if (FeldAllterTextArea != null)
    {
        var TextBereich = ZeigerAufTextarea.createTextRange();

        // prüfen ob Textbereich erzeugt wurde
        if (TextBereich != null)
        {alert(".boundingTop = " + TextBereich.boundingTop); }
    }
}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS=100 ROWS=2 onclick="Anzeige(this)">
.....
</TEXTAREA>
```

.boundingWidth Breite des Rechteckes in Pixel um den Textbereich des TextRange Objektes  
per textrange Objekt

Beispiel:

```
<SCRIPT>
function Anzeige(ZeigerAufTextarea)
{
    var FeldAllterTextArea = document.all.tags("TEXTAREA");

    // prüfen ob mindestens 1 TEXTAREA-Element im Body vorliegt
    if (FeldAllterTextArea != null)
    {
        var TextBereich = ZeigerAufTextarea.createTextRange();

        // prüfen ob Textbereich erzeugt wurde
        if (TextBereich != null)
        {alert(".boundingWidth = " + TextBereich.boundingWidth); }
    }
}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS=100 ROWS=2 onclick="Anzeige(this)">
.....
</TEXTAREA>
```

.browserLanguage Sprache des Betriebssystems und **nicht** des Browsers (Browsersprache kann andere sein als die des Betriebssystems z.B. französischer Browser auf deutschem Windows)  
per navigator Objekt  
Achtung: ab IE 5.x Sprache **immer** laut **regionalen** Einstellungen des Users im **Betriebssystems**  
z.B. "en" oder "de"

.browserLanguage Browsersprache  
Achtung: ab IE 5.x Sprache **immer** laut **regionalen** Einstellungen des Users im **Betriebssystems**  
z.B. "en" oder "de"  
siehe Objekt window.clientInformation

.bufferDepth Anzahl der Bits pro Pixel für eine Farbe UND Verwendung des off-screen bitmap buffer  
per screen Objekt

.bufferDepth Anzahl der Bits pro Pixel für eine Farbe UND Verwendung des off-screen bitmap buffer  
Behavior .style.clientCaps

.bufferingProgress aktueller prozentualer Status des Pufferns eines Datenflusses (Data Stream) einer Media-Datei auf der Timeline  
Prozentwert des bisher erfolgten Pufferns  
nur für Media-Datei mit Datenfluss  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
```



```

<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:MEDIA          ID="ID_Media"
                  SRC="test.asx"
                  SYNCBEHAVIOR="locked"
                  TIMEACTION="display"
>
</t:MEDIA>
<SPAN ID="ID_Span"
      CLASS="time_line_klasse"
      DUR="0.1"
      REPEATCOUNT="indefinite"
      onrepeat="ID_Span.innerText= ID_Media.bufferingProgress;"
>
0
</SPAN>
</BODY>
</HTML>

```

**.button** Maustaste die das Event auslöst NUR per onmousedown, onmouseup, und onmousemove events  
Objekt event  
Integer

0	Default. keine Taste gedrückt oder obiges Event nicht erzeugt
1	linke Maustaste ist gedrückt
2	rechte Maustaste ist gedrückt
3	linke UND rechte Maustaste sind gleichzeitig gedrückt
4	mittlere Maustaste ist gedrückt
5	linke UND mittlere Maustaste sind gleichzeitig gedrückt
6	rechte UND mittlere Maustaste sind gleichzeitig gedrückt
7	rechte UND mittlere UND linke Maustaste sind gleichzeitig gedrückt

**.by** Wert um den die Werterhöhung bei Elemente-Animation(en) auf der Timeline  
per .additive oder .accumulate  
erfolgen soll  
Wert der Eigenschaft:  
numerisch aber Einheiten wie px, in, cm, mm, pt, pc können Teil des Wertes sein  
wird für Animation in Einheiten zerlegt laut Schrittweite laut .calcMode  
für die Objekte animate, animateMotion und animateColor gilt:  
.by wird von .path, .to und .values überschrieben  
Achtung: Im Objekt animatecolor (t:ANIMATECOLOR) ist .values auch als Farbliste definiert !  
siehe .attributeName .accumulate .additive .calcMode  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft .width) nach rechts:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function SpanInhaltInit(Kette)
{
    // Zustand
    ID_Span1.innerText = "";

    // Kumulationsart
    ID_Span2.innerText =Kette;

    // Zaehler auf 1
    ID_Span3.innerText = "1";

    // DIV-Text festlegen
    ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
    if (Kette == "sum")
    {
        // Wertkumulation von .width
        ID_Div.innerText += " genau 1x kumulativ um ";
    }
}

```



```

        ID_Div.innerText += ID_Animate.by
        ID_Div.innerText += " mal "
        ID_Div.innerText += ID_Animate.repeatCount;
    }
    else
    {
        // keine Wertkumulation von .width
        ID_Div.innerText += ID_Animate.repeatCount;
        ID_Div.innerText += " mal um ";
        ID_Div.innerText += ID_Animate.by
    }

    ID_Div.innerText += " aus";
}

function KumulationAus()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("none");
    ID_Animate.accumulate="none";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function KumulationEin()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("sum");
    ID_Animate.accumulate="sum";

    // DANACH Animation starten
    ID_Animate.beginElement();
}

function Anzeige()
{
    ID_Span1.innerText = "Animation ist beendet ";
}
</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE
        ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    <t:ANIMATE>

animierter DIV
<DIV
    ID="ID_Div"
    CLASS="time_line_klasse"
    STYLE= "position:absolute;
            top:125px;
            left:25px;
            height:100px;
            width:125px;
            border:solid black 2px;
            "
    >
</DIV>
<BR>

```



Zustand der Animation:

```
<SPAN ID="ID_Span1"></SPAN>
<BR>
```

Kumulationsart:

```
<SPAN ID="ID_Span2"></SPAN>
<BR>
```

Durchlaufzaehler:

```
<SPAN ID="ID_Span3" ></SPAN>
<BR>
```

```
<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
```

```
</BODY>
</HTML>
```

.by

Schrittweite des Übergangsfilters per .style.time2.transitionFilter Behavior-Objekt  
als Offset zum Wert der kodierten Eigenschaft .from des Behavior  
Schrittweite als Prozentsatz des Überganges nach dessen Vollendung  
nicht zusammen mit Eigenschaft .to oder .value kodieren, sonst wird .by ignoriert  
siehe Objekt currTimeState und Behavior .style.time2  
Ziffernfolge Floating point  
Wert numerisch von 0 bis 1  
0 entspricht 0 % des kompletten Überganges  
1 entspricht 100% des kompletten Überganges  
Bsp.: 0.3 entspricht Schrittweite ist 30 % des kompletten Überganges

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter"
FROM="0.3"
BY="0.4"
TYPE="barWipe"
DUR="3"
TARGETELEMENT="ID_Div"
>
</t:TRANSITIONFILTER>
<DIV ID="ID_Div"
CLASS="time_line_Klasse"
DUR="indefinite"
STYLE="position:relative; left:20px; width:420px; height:100px;
background-image:url(test.gif); background-repeat: no-repeat;
"
>
</DIV>
</BODY>
</HTML>
```

.calcMode

Schrittweite der Werterhöhung bzw. Art der Animation in der Ebene (1D oder 2D)  
bei Elemente-Animation(en) auf der Timeline  
per .additive oder .accumulate

Achtung: Im Objekt animatecolor (t:ANIMATECOLOR) ist .values auch als Farbliste definiert !  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft .width) nach rechts:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function SpanInhaltInit(Kette)
{
// Zustand
```



```

        ID_Span1.innerText = "";

        // Kumulationsart
        ID_Span2.innerText =Kette;

        // Zaehler auf 1
        ID_Span3.innerText ="1";

        // DIV-Text festlegen
        ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
        if (Kette == "sum")
        {
            // Wertkumulation von .width
            ID_Div.innerText += " genau 1x kumulativ um ";
            ID_Div.innerText += ID_Animate.by
            ID_Div.innerText += " mal "
            ID_Div.innerText += ID_Animate.repeatCount;
        }
        else
        {
            // keine Wertkumulation von .width
            ID_Div.innerText += ID_Animate.repeatCount;
            ID_Div.innerText += " mal um ";
            ID_Div.innerText += ID_Animate.by
        }

        ID_Div.innerText += " aus";
    }

    function KumulationAus()
    {
        // Element darf nicht aktiv sein !!

        // ERST .accumulate definieren
        SpanInhaltInit("none");
        ID_Animate.accumulate="none";

        // DANACH Animation starten
        ID_Animate.beginElement();

    }

    function KumulationEin()
    {
        // Element darf nicht aktiv sein !!

        // ERST .accumulate definieren
        SpanInhaltInit("sum");
        ID_Animate.accumulate="sum";

        // DANACH Animation starten
        ID_Animate.beginElement();

    }

    function Anzeige()
    {
        ID_Span1.innerText = "Animation ist beendet ";
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE
        ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        CALCMODE="linear"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    <t:ANIMATE>

```





```

animierter DIV
<DIV ID="ID_Div"
      CLASS="time_line_klasse"
      STYLE= "position:absolute;
              top:125px;
              left:25px;
              height:100px;
              width:125px;
              border:solid black 2px;
              "
>
</DIV>
<BR>

Zustand der Animation:
<SPAN ID="ID_Span1"></SPAN>
<BR>

Kumulationsart:
<SPAN ID="ID_Span2"></SPAN>
<BR>

Durchlaufzaehler:
<SPAN ID="ID_Span3"></SPAN>
<BR>

<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
</BODY>
</HTML>

```

.calcMode

Interpolation der Aufteilung der internen Animationsschritte auf

Schrittweite      laut .by  
 Schritt            laut.from bei optionalen .to  
 Gesamtschritt    bei fehlendem .by bzw. .from .to  
 Animationsschritte laut .values

per Übergangfilter per .style.time2.transitionFilter Behavior-Objekt

Bsp.: Lineare Aufteilung durch Interpolation: Interne Schritte zu gleichen Teilen verteilt

keine Interpolation, so Animation laut Schrittweite, also eventuell nicht fließende Animation

Größe des internen Animationsschrittes: Je nach der Gesamtdauer der Animation laut .dur des Behavior

siehe Objekt currTimeState und Behavior .style.time2

"discrete" keine Interpolation

"linear" Default

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

  <t:TRANSITIONFILTER ID="ID_Transfilter1"
    BEGIN="ID_Div1.begin"
    TYPE="barWipe"
    DUR="5"
    TARGETELEMENT="ID_Div1"
    VALUES=".1;.17;.27;.37;.47;.56;.65;.71;.82;.92;1.0"
    CALCMODE="discrete"
  >
  </t:TRANSITIONFILTER>

  <t:TRANSITIONFILTER ID="ID_Transfilter2"
    BEGIN="ID_Div1.begin"
    TYPE="barWipe"
    DUR="5"
    TARGETELEMENT="ID_Div2"
    VALUES=".1;.17;.27;.37;.47;.56;.65;.71;.82;.92;1.0"
    CALCMODE="linear"
  >
  </t:TRANSITIONFILTER>
<BR>

```





```

        ZeigerAufSPANObjekt.appendChild(ZeigerAufTextObjekt);

        for(var Index=0; Index <document.all.length; Index ++)
        {
            if(document.all[Index].canHaveChildren==true)
            {
                document.all[Index].appendChild(ZeigerAufSPANObjekt);
                break;
            }
        }
    }
</SCRIPT>
<INPUT TYPE=button VALUE="Kind hinzufügen " onclick="KindHinzufuegen()">
<DIV>
    Test<BR>
</DIV>

```

**.canHaveHTML** prüfen ob Objekt HTML-Tags enthalten darf

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Antwort(BooleanWert)
    { BooleanWert ? alert("Ja") : alert("Nein");}
</SCRIPT>
</HEAD>
<BODY>
    <P>
        INPUT:
        <INPUT type="text" ID="ID_Input" VALUE="Test" >
    </P>

    <P>
        SPAN:
        <SPAN ID="ID_Span">Test</SPAN>
    </P>

    <BUTTON onclick="Antwort(ID_Input.canHaveHTML);">
        Kann INPUT-Element HTML besitzen ?
    </BUTTON>

    &nbsp;

    <BUTTON onclick="Antwort(ID_Span.canHaveHTML);">
        Kann SPAN-Element HTML besitzen ?
    </BUTTON>
</BODY>
</HTML>

```

**.canPause** generelle Pausierungsmöglichkeit einer Media-Datei auf der Timeline  
siehe Objekt `currTimeState` und Behavior `.style.time2`

**.canSeek** generelle Möglichkeit der Auswahl eines Zeitpunktes einer Media-Datei auf der Timeline  
Auswahl per Seek-Methoden  
`.seekActiveTime()`  
`.seekSegmentTime()`  
`.seekTo()`  
`.seekToFrame()`  
 nicht jeder Media-Typ unterstützt Seek  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

**.caption** Zeiger auf das Objekt `table.caption`  
es darf nur 1 CAPTION zur Tabelle existieren

**.cellIndex** Index der Zelle in der Collection `table.rows.cells`  
siehe Objekt `table.tr.td`  
siehe Objekt `table.tr.th`

**.cellPadding** Abstand zwischen Zellrahmen und dem Inhalt der Zelle  
siehe Objekt `table`

Beispiel:

```

<TABLE ID="ID_Tabelle" BORDER CELLPADDING=10>
<TR>

```



```

        <TD>Zelle 1</TD>
        <TD>Zelle 2</TD>
    </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.cellPadding=20">20 </BUTTON>
<BUTTON onclick="ID_Tabelle.cellPadding=5">5 </BUTTON>

```

**.cellSpacing** Abstand zwischen den Zellen  
siehe Objekt table

Beispiel:

```

<TABLE ID="ID_Tabelle" BORDER CELSPACING=10>
    <TR>
        <TD>Zelle 1</TD>
        <TD>Zelle 2</TD>
    </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.cellSpacing=20">20 </BUTTON>
<BUTTON onclick="ID_Tabelle.cellSpacing=5">5 </BUTTON>

```

**.charset** Zeichensatz zum Encoden eines Objektes im Dokument

**.checked** Selektionsstatus des Checkbox-Control bzw. Radio Button-Control  
bezüglich Selektion durch User  
(Objekt input checkbox bzw. Objekt input radio)  
wird bei Ceckbox-Control auch verändert durch Eigenschaft .indeterminate  
false Default  
keine Selektion des Control-Elementes  
true Selektion des Control-Elementes  
Hinweise: Der Wert kann nur gesendet werden, wenn das Control selektiert ist.  
Werte von unselektierten Controls werden nicht gesendet.  
Radio Button-Control nur selektierbar, wenn NAME-Attribut kodiert wurde !

Beispiel:

```

<HEAD>
<SCRIPT>
    function Anzeige()
    {alert("Checkbox-Status = " + ID_Checkbox.checked);}
</SCRIPT>
</HEAD>
<BODY>
    selektiere !
    <INPUT TYPE=checkbox
        ID="ID_Checkbox"
        NAME="checkbox1"
        onclick=Anzeige()
    >
</BODY>

```

**.classid** Klassenbezeichner (ID) eines damit dem Objekt zugeordneten ActiveX-Controls von  
Microsoft Windows (Microsoft eigene oder Fremdhersteller)  
dient als Ersatz für die browserspezifischen MIME-Typen  
Gross-Klein egal

Beispiel:

```
"CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
```

**.className** Klassenreferenz, Klassenname

Beispiel 1:

```

<HEAD>
    <STYLE TYPE="text/css">
        P {font-size: 24pt;}
        .RoterText {color: red;}
        .BlauerText {color: blue;}
        .KursiverText {font-style: italic;}
    </STYLE>
</HEAD>
<BODY>
    <P>fontsize = 24</P>
    <P CLASS="RoterText"> rot UND fontsize = 24</P>
    <P CLASS="BlauerText KursiverText ">blau kursiv UND fontsize = 24</P>
</BODY>

```

Beispiel 2 für globalen Style per HEAD:

```

<HEAD>
<STYLE>

```



```

        .an {text-decoration: underline overline; color:blue;}
        .aus {text-decoration: none; color:black;}
    </STYLE>
</HEAD>
<BODY>
    <A      HREF="test.htm"
           CLASS="aus"
           onmouseover="this.className='an';"
           onmouseout="this.className='aus';"
    >
    </A>
</BODY>

```

**.clientHeight**                      Objekthöhe in Pixel    ohne Abstand zum Umgebungsobjekt  
    ohne Rahmen  
    ohne Scrollbalken  
    Bezüglich des Internet Explorer 6.0 unter Windows 98 und Windows XP bei maximiertem Browserfenster mit Standard-Browser-Menü und einer Bildschirmauflösung von 1024x 768 liefert .clientHeight **unterschiedliche** Werte: Die Höhe unter Windows XP beträgt **10 Pixel mehr** !

Beispiel:

```

<DIV ID="ID_Div" STYLE="overflow:scroll; width:200; height:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientHeight)">client height</BUTTON>

```

**.clientInformation**                      Zeiger auf das Objekt window.clientInformation, welches vom Objekt navigator erbt  
    siehe Objekt window

**.clientLeft**                                      Abstand in Pixel zum linken Rand des Fensters

Beispiel:

```

<DIV ID="ID_Div" STYLE="overflow:scroll; top:200; left:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientLeft)">client left</BUTTON>

```

**.clientTop**                                      Abstand in Pixel zum oberen Rand des Fensters

Beispiel:

```

<DIV ID="ID_Div" STYLE="overflow:scroll; top:200; left:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientTop)">client top</BUTTON>

```

**.clientWidth**                                      Objekt-Breite in Pixel                      ohne Abstand zum Umgebungsobjekt  
    ohne Rahmen  
    ohne Scrollbalken

Beispiel:

```

<DIV ID="ID_Div" STYLE="overflow:scroll; width:200; height:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientWidth)">client width</BUTTON>

```

**.clientX**    X-Koordinate Maus relativ zum Fenster aber ohne Rahmen, Scrollbar etc.  
    Objekt event

Beispiel:

```

<HEAD>
<SCRIPT>
    function AlertAnzeige()
    {
        var Kette  = "X= " + window.event.clientX + "\r";
        Kette    += "Y= " + window.event.clientY + "\r";
        alert(Kette);
    }

    function StatusZeileAnzeigen()
    {
        var Kette  = "X= " + window.event.clientX + " ";
        Kette    += "Y= " + window.event.clientY;
        window.status = Kette;
    }
</SCRIPT>
</HEAD>
<BODY onmousemove="StatusZeileAnzeigen()" ondblclick="AlertAnzeige()">
</BODY>

```

**.clientY**    Y-Koordinate Maus relativ zum Fenster aber ohne Rahmen, Scrollbar etc.  
    Objekt event

Beispiel:

```

<HEAD>
<SCRIPT>
    function AlertAnzeige()
    {
        var Kette  = "X= " + window.event.clientX + "\r";

```



```

        Kette += "Y= " + window.event.clientY + "\r";
        alert(Kette);
    }

    function StatusZeileAnzeigen()
    {
        var Kette = "X= " + window.event.clientX + " ";
        Kette += "Y= " + window.event.clientY;
        window.status = Kette;
    }
</SCRIPT>
</HEAD>
<BODY onmousemove="StatusZeileAnzeigen()" ondblclick="AlertAnzeige()">
</BODY>

```

**.clipBegin** Startzeitpunkt eines Unterbereiches (Clip) innerhalb des Media-Elementes auf der Timeline  
Eigenschaft `.canSeek` muss true liefern  
nicht jeder Media-Typ unterstützt Clipping  
siehe Objekt `currTimeState` und Behavior `.style.time2`

**.clipboardData** Zeiger auf Objekt `clipboardData` als Windows-Zwischenablage  
zu verwenden mit Eventhandlern `onbeforecut` und `onbeforepaste`  
siehe auch `event.dataTransfer` Objekt  
siehe Objekt `window`

**.clipBottom** untere Koordinate des Ausschnittes (Clipping Region) per `currentStyle` Objekt  
prüfen ob ein Ausschnitt vorliegt

Beispiel:

```

<SCRIPT>
function AusschnittSetzen()
{
    ID_Image.style.clip= "rect(0,100, "
                        + ID_select.options(ID_select.selectedIndex).value
                        + ",100)";

    if (ID_Image.currentStyle.clipBottom == "60px")
    { alert("60 Pixel"); }
}
</SCRIPT>
<IMG ID="ID_Image" SRC="test.jpg">
<SELECT ID="ID_Select" onchange="AusschnittSetzen()">
    <OPTION VALUE=100>100 Pixel </OPTION>
    <OPTION VALUE=40>40 Pixel </OPTION>
    <OPTION VALUE=50>50 Pixel </OPTION>
    <OPTION VALUE=60>60 Pixel </OPTION>
</SELECT>

```

**.clipEnd** Endezeitpunkt eines Unterbereiches (Clip) innerhalb des Media-Elementes auf der Timeline  
Eigenschaft `.canSeek` muss true liefern  
nicht jeder Media-Typ unterstützt Clipping  
siehe Objekt `currTimeState` und Behavior `.style.time2`

**.clipLeft** linke Koordinate des Ausschnittes (Clipping Region) per `currentStyle` Objekt  
prüfen ob ein Ausschnitt vorliegt

Beispiel:

```

<SCRIPT>
function AusschnittSetzen()
{
    ID_Image.style.clip= "rect(0,100,100,"
                        + ID_select.options(ID_select.selectedIndex).value
                        + ")";

    if (ID_Image.currentStyle.clipLeft == "60px")
    { alert("60 Pixel"); }
}
</SCRIPT>
<IMG ID="ID_Image" SRC="test.jpg">
<SELECT ID="ID_Select" onchange="AusschnittSetzen()">
    <OPTION VALUE=100>100 Pixel </OPTION>
    <OPTION VALUE=40>40 Pixel </OPTION>
    <OPTION VALUE=50>50 Pixel </OPTION>
    <OPTION VALUE=60>60 Pixel </OPTION>
</SELECT>

```



**.clipRight** rechte Koordinate des Ausschnittes (Clipping Region) per currentStyle Objekt prüfen ob ein Ausschnitt vorliegt

Beispiel:

```
<SCRIPT>
function AusschnittSetzen()
{
    ID_Image.style.clip= "rect(0,"
                        + ID_select.options(ID_select.selectedIndex).value
                        + ",100,0)";

    if (ID_Image.currentStyle.clipRight == "60px")
    { alert("60 Pixel"); }
}
</SCRIPT>
<IMG ID="ID_Image" SRC="test.jpg">
<SELECT ID="ID_Select" onchange="AusschnittSetzen()">
    <OPTION VALUE=100>100 Pixel </OPTION>
    <OPTION VALUE=40>40 Pixel </OPTION>
    <OPTION VALUE=50>50 Pixel </OPTION>
    <OPTION VALUE=60>60 Pixel </OPTION>
</SELECT>
```

**.clipTop** obere Koordinate des Ausschnittes (Clipping Region) per currentStyle Objekt prüfen ob ein Ausschnitt vorliegt

Beispiel:

```
<SCRIPT>
function AusschnittSetzen()
{
    ID_Image.style.clip= "rect("
                        + ID_select.options(ID_select.selectedIndex).value
                        + ",100,100,0)";

    if (ID_Image.currentStyle.clipTop == "60px")
    { alert("60 Pixel"); }
}
</SCRIPT>
<IMG ID="ID_Image" SRC="test.jpg">
<SELECT ID="ID_Select" onchange="AusschnittSetzen()">
    <OPTION VALUE=100>100 Pixel </OPTION>
    <OPTION VALUE=40>40 Pixel </OPTION>
    <OPTION VALUE=50>50 Pixel </OPTION>
    <OPTION VALUE=60>60 Pixel </OPTION>
</SELECT>
```

**.closed** Zustand auf Geschlossenheit eines Fensters  
siehe Objekt window

**.code** Url der \*.class-Datei (kompilierter Javacode)

**.codeBase** Url der Komponente für Download der Komponente vom Server auf den Client  
optionale Versionsprüfung möglich (nicht bei Applet-Komponente) um unnötigen  
Download zu ersparen: vorheriger Abgleich der Version der Komponente auf Server und Client  
auch wenn der Versionsabgleich keinen Download ergab, wird immer HTTP Header-Transaktion  
ausgelöst

Beispiel:

```
<OBJECT ID="CommonDialog1"
    WIDTH=32
    HEIGHT=32
    CLASSID="....."
    CODEBASE="http://www.test.de/test.cab#Version=1,0,0,0"
>
</OBJECT>
```

**.codeType** Internet Media Type (Mimety) des Codes zum Objekt, das per OBJECT-Tag eingebunden wurde

**.color** Farbe des Objektes  
#rrggbb  
vordefinierter Farbname (browserspezifisch)  
rgb xyz                      x für rot, y für grün, z für blau, x bis z reelle Zahl

**.colorDepth** Anzahl der Bits pro Pixel für eine Farbe UND Verwendung destination device or buffer  
wird durch den Wert der Eigenschaft .bufferDepth überschrieben,  
wenn .bufferDepth mit Wert > 0  
per screen Objekt



<code>.colorDepth</code>	<p>Anzahl der Bits pro Pixel für eine Farbe UND Verwendung destination device or buffer wird durch den Wert der Eigenschaft <code>bufferDepth</code> überschrieben, wenn <code>.bufferDepth</code> mit Wert &gt; 0</p> <p>Behavior <code>.style.clientCaps</code></p>				
<code>.cols</code>	<p>Breite aller Frames eines Frameset (Breite des Frameset als Container)</p> <p>Beispiele:</p> <pre>&lt;FRAMESET COLS="40%, 60%"&gt;           // Summe muss immer 100% sein &lt;FRAMESET COLS="50, *, 80"&gt;           // 50 Pixel, 80 Pixel und restliche Fensterbreite</pre>				
<code>.cols</code>	<p>Anzahl der sichtbaren Zeichen in einer Zeile der TEXTAREA</p> <p>Hinweis: Es können mehr Zeichen in der TEXTAREA enthalten sein als sichtbar</p> <p>siehe <code>.rows</code> für Mehrzeiligkeit</p> <p><code>.wrap</code> für Wortumbruch</p> <p>Scrollleisten sind erzeugbar</p> <p>siehe <code>textarea</code> Objekt</p>				
<code>.cols</code>	<p>Anzahl der Spalten in der Tabelle</p> <p>wenn belegt so wird Tabelle schneller gerendert</p> <p>siehe Objekt <code>table</code></p> <p>Beispiel:</p> <pre>&lt;TABLE ID="ID_Tabelle" BORDER CELLSPACING=10&gt;   &lt;TR&gt;     &lt;TD&gt;Zelle 1&lt;/TD&gt;     &lt;TD&gt;Zelle 2&lt;/TD&gt;   &lt;/TR&gt; &lt;/TABLE&gt; &lt;BUTTON onclick="alert(ID_Tabelle.cols);"&gt;Anzahl&lt;/BUTTON&gt;</pre>				
<code>.colSpan</code>	<p>Anzahl der Spalten einer Zelle in einer Tabelle</p> <p>siehe Objekt <code>table.tr.td</code></p> <p>siehe Objekt <code>table.tr.th</code></p>				
<code>.Column</code>	<p>aktuelle Spalte des Zeichens im Textstream, also Nummer des Zeichens im Stream</p> <p>nur bei <b>zeichenweiser</b> Dateiverarbeitung verwendbar, das jedes gelesene newline-Zeichen den Wert von <code>.Column</code> auf 1 setzt</p> <p>Beispiel:</p> <pre>var DateiSystem = new ActiveXObject("Scripting.FileSystemObject"); var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0); DateiOffen.WriteLine("Test"); DateiOffen.Close(); DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0); DateiOffen.ReadLine(); alert(DateiOffen.Column); DateiOffen.Close();</pre>				
<code>.compatMode</code>	<p>Kompatibilität des IE 6.x zu CSS1 bzw. seinen Browservorgängern</p> <p>siehe <code>style</code> Objekt und Kodierung von <code>!DOCTYPE</code></p> <p>ab IE 6.x</p> <p><code>document.compatMode</code></p> <p>nur lesen</p> <table> <tr> <td>"BackCompat"</td><td>Kompilationsmodus ist aus IE 6.x Browser unterstützt nur CSS und ist damit kompatibel zu den Browservorgängern</td></tr> <tr> <td>"CSS1Compat"</td><td>Kompilationsmodus ist ein IE 6.x Browser unterstützt nur CSS1 und ist damit nicht kompatibel zu den Browservorgängern</td></tr> </table>	"BackCompat"	Kompilationsmodus ist aus IE 6.x Browser unterstützt nur CSS und ist damit kompatibel zu den Browservorgängern	"CSS1Compat"	Kompilationsmodus ist ein IE 6.x Browser unterstützt nur CSS1 und ist damit nicht kompatibel zu den Browservorgängern
"BackCompat"	Kompilationsmodus ist aus IE 6.x Browser unterstützt nur CSS und ist damit kompatibel zu den Browservorgängern				
"CSS1Compat"	Kompilationsmodus ist ein IE 6.x Browser unterstützt nur CSS1 und ist damit nicht kompatibel zu den Browservorgängern				
<code>.complete</code>	Zustand des Ladens des Objektes				
<code>.connectionType</code>	<p>Typ der genutzten Verbindung bzw. Offline-Status der Verbindung</p> <p>Behavior <code>.style.clientCaps</code></p>				
<code>.constructor</code>	<p>Bezeichner einer JScript-Klass (Objekttyp) oder eines privaten Konstruktors</p> <p>Anwendung: Ermittlung der Objektklasse/Konstruktors eines abgeleiteten Objektes</p> <p>Ableitung aus der Objektklasse</p> <p>per Anweisung <code>new</code> mit der Objektklasse als Konstruktor benötigt (siehe dort)</p> <p>nicht bei Script-Objekt <code>Math</code> möglich</p> <p>Ableitung aus privatem Konstruktor</p> <p>per Anweisung <code>new</code>, die den privaten Konstruktor verwendet</p> <p>JScript-Objektklassen sind z.B.</p>				





Array  
Boolean  
Date  
Function  
String

.constructor

Bezeichner einer JScript-Objektklasse (Objekttyp) oder eines privaten Konstruktors  
Anwendung: Ermittlung der Objektklasse/Konstruktors eines abgeleiteten Objektes  
Ableitung aus der Objektklasse  
per Anweisung new mit der Objektklasse als Konstruktor benötigt (siehe dort)  
nicht bei Script-Objekt Math möglich  
Ableitung aus privatem Konstruktor  
per Anweisung new, die den privaten Konstruktor verwendet  
JScript-Objektklassen sind z.B.  
Array  
Boolean  
Date  
Function

Beispiel 1 für Ableitung aus einem JScript-Objekt:

```
var Kette = new String("Hi");
alert( (Kette.constructor === String)); // liefert "true"
alert( (Kette.constructor === "String")); // liefert "false"
```

Beispiel 2 für Ableitung anhand privaten Konstruktors:

```
function TestFunktion()
{alert("Hallo");}

var ZeigerAufFunktion = new TestFunktion(); // bewirkt Ausführung von TestFunktion() also auch von alert()

// ZeigerAufFunktion(); // nicht möglich und bringt Fehlermeldung wegen fehlernder Instanz,
//                       // da keine Ableitung vom JScript-Objekt Function
// Kodierung ohne () bringt keinen Fehler, da als Variablendeklaration erkannt

alert(ZeigerAufFunktion.constructor === TestFunktion); // true
alert(TestFunktion.constructor === TestFunktion); // false
```

.content

Wert der Informationen laut Eigenschaften .httpEquiv und .name des meta Objektes

Beispiele:

Dokument alle 2 Sekunden neu laden  
<META HTTP-EQUIV="REFRESH" CONTENT=2>

Zeichensatz des Dokumentes  
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=Windows-1251">

Themenunterstützung abschalten  
<META HTTP-EQUIV="MSTHEMECOMPATIBLE" CONTENT="no">

um sicher zu gehen, daß immer die Seite mit garantiert aktuellem Stand geladen wird  
→ eventuell ist Proxyserver nicht aktuell  
<META HTTP-EQUIV="expires" CONTENT=sekunden\_wert>

sekunden\_wert:      Wartezeit in Sekunden bis zum nächsten Laden unter Umgehung  
des Proxyserver-Cache → 0, also immer umgehen und sofort laden

Suchmaschine beim Scannen des Dokumentes beeinflussen:

Suchmaschine darf alles tun  
<META HTTP-EQUIV="Robots" CONTENT="all">

Suchmaschine darf Dokument nichts scannen  
<META HTTP-EQUIV="Robots" CONTENT="noindex">

Suchmaschine darf Dokument scannen  
<META HTTP-EQUIV="Robots" CONTENT="index">

Suchmaschine darf Verweisen folgen  
<META HTTP-EQUIV="Robots" CONTENT="follow">

Suchmaschine darf Verweisen nicht folgen  
<META HTTP-EQUIV="Robots" CONTENT="nofollow">

.contentEditable

Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat)



Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.  
 "inherit" Default. Editierbarkeit von Parent geerbt  
 "false" Content nicht veränderbar  
 "true" Content ist veränderbar

Beispiel:

```
<HEAD>
<SCRIPT>
function EditierbarkeitWechseln()
{
    var Editierbarkeit = ID_Span1.isContentEditable;
    var Editierbarkeit_Negation = ! Editierbarkeit;
    ID_Span1.contentEditable = Editierbarkeit_Negation;
    ID_Span2.innerText = Editierbarkeit_Negation;

    if (Editierbarkeit_Negation==false)
    { ID_Button.innerText="editierbar machen"}
    else
    { ID_Button.innerText="nicht editierbar machen"}
}
</SCRIPT>
</HEAD>
<BODY onload="ID_Span2.innerText = ID_Span1.isContentEditable;">
    <BUTTON ID="ID_Button" onclick=" EditierbarkeitWechseln()">
        Klick mich
    </BUTTON>
    <SPAN ID="ID_Span1">Diesen Text editieren</SPAN>
    <SPAN ID="ID_Span2">Editierbarkeit</SPAN>
</BODY>
```

.contentWindow Referenz auf das zugehörige Fenster-Objekt laut Collection document.all  
 document.all.object.contentWindow

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
function LocationAendern()
{
    for(i=0;i<document.all.length;i++)
    {
        if (document.all[i].tagName=="IFRAME")
        {document.all[i].contentWindow.location = "http://www.test.com";}
    }
}
</SCRIPT>
</HEAD>
<BODY>
    <BUTTON onclick="LocationAendern();">Location aendern</BUTTON>
    <IFRAME SRC="http://www.test.de"></IFRAME>
</BODY>
</HTML>
```

.cookie Cookie des Dokumentes als Stringwert

Beispiel:

```
<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

    function LeseCookieWert(Name)
    // Name des zu lesenden Cookie ist String
    // liefert Cookiewert aus unescape() als String
    {
```



```

var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

// Feld der Cookies referenzieren
// erzeugt durch Separation anhand Semikolon
var CookieFeld = document.cookie.split("; ");
var AnzahlCookies = CookieFeld.length;

// Feldelement umfasst Name und Wert des Cookie
// Aufbau Cookie_Name = Cookie_Wert
// Separator ist Gleichheitszeichen
// Index 0 für Cookie_Name
// Index 1 für Cookie_Wert
var CookieNameUndWertAlsFeld;

// CookieFeld auslesen und auf Cookie laut Name prüfen
var Gefunden=false;
var Index = 0;
do
{
    // aktuelles Cookie lesen
    CookieNameUndWertAlsFeld = CookieFeld[Index].split("=");

    // und auf Name prüfen
    if (Name == CookieNameUndWertAlsFeld [0])
    {
        CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
        Gefunden=true;
    }
    else
    {Index++;}
}
while ( ( !Gefunden )
        && ( Index < AnzahlCookies)
        );

return CookieWert;
}
</SCRIPT>

```

**.cookieEnabled** Cookiesstatus lesen aber nicht verändern per navigator Objekt  
navigator.cookieEnabled  
false Cookies sind nicht erlaubt  
true Cookies sind erlaubt

**.cookieEnabled** Cookienutzbarkeit im Browser  
Behavior .style.clientCaps  
false Browser unterstützt keine Cookies  
true Browser unterstützt Cookies

**.cookieEnabled** Cookienutzbarkeit im Browser  
siehe Objekt window.clientInformation

**.coords** Koordinaten eines Objektes in Abhängigkeit zur Eigenschaft .shape

Beispiel 1:

```

<IMG SRC="test.gif" USEMAP="#MapName">
<MAP NAME="#MapName">
    <AREA SHAPE="rect" COORDS="0,0,33,33" HREF="test.gif">
    <AREA SHAPE="circle" COORDS="90,33,3" HREF="test1.gif">
</MAP>

```

Beispiel 2:

```

<MAP NAME="logischer_map_name"
<AREA
    NAME="name_des_verweissensitiven_bereiches_der_grafik"
    COORDS="koordinaten_liste_des_bereiches"
    HREF="url"
    SHAPE= "rect"
           oder "poly"
           oder "circle"
           oder "default"
    TARGET="logischer_window_name"
    onClick="eventhandler_1"
    onMouseOut="eventhandler_2"
    onMouseOver="eventhandler_3"

```



```

> .....
</MAP>

SHAPE:      rect      Rechteck
           poly      Vieleck
           circle     Kreis
COORDS: bei rect und poly "x1,y1,.....,xn,yn"
           circle      "x,y,r"
           x          Spalte in Pixel
           y          Zeile in Pixel
           r          Radius in Pixel
Hinweis: mehrere AREA sind möglich

```

Beispiel 3:

```

<IMG SRC="test.gif" WIDTH=504 HEIGHT=126 BORDER=0 ALT="Gesamtes Bild"
USEMAP="# freier_mappen_name"
>
<MAP NAME="freier_mappen_name">
<AREA SHAPE="rect"
COORDS="0,0,82,126"
ALT="Teilbereich 1"
HREF="/graphics/test.gif"
STYLE="....."
onxxx="....."
>
.....
<AREA ....>
</MAP>

```

.copyright

Copyright der Media-Datei auf der Timeline  
bei Advanced Stream Redirector (ASX)-Datei: Es wird der Text von COPYRIGHT des aktiven  
Eintrages geliefert und nicht den der Datei.  
Track entspricht playItem Objekt laut object.playlist.item() bzw. object.playlist.aktiveTrack  
Trackliste liegt in der Behavior-Collection .style.time2.playlist als Playliste:  
Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline  
aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:

```

var Kette = object.playlist.item(index).copyright;

index      Integer, ab 0

var CopyrightVariable = object.playlist.aktiveTrack.copyright;

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:VIDEO ID="ID_Video"
SRC="test.wmv"
STYLE="position:absolute;top:50px;height:100px"
>
</t:VIDEO>
<SPAN ID="ID_Span"
STYLE="position:absolute;top:165px;"
>
</SPAN>
<BUTTON ID="ID_Button"
onclick="ID_Span.innerText= ID_Video.copyright"
>
Klick
</BUTTON>
</BODY>
</HTML>

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">

```



```

<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()
    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"
        TIMEACTION="visibility"
        onend="ButtonUpdate();"
        ontrackchange="AnzeigeUpdate();"
        onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>
    <BR>

```



```

<SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

<BR>
<BUTTON ID="ID_Button1"
onclick="ID_Media.beginElement(); ButtonUpdate();"
>
    Start
</BUTTON>
<BUTTON ID="ID_Button2"
onclick="ID_Media.playList.nextTrack();"
>
    naechster Track
</BUTTON>
<BUTTON ID="ID_Button3"
onclick="ID_Media.playList.prevTrack();"
>
    vorhergehender Track
</BUTTON>
<BUTTON ID="ID_Button4"
onclick="ID_Media.endElement(); AnzeigeLoeschen();"
>
    Stop
</BUTTON>
</BODY>
</HTML>

```

.Count Anzahl der Elemente im Dictionary

.Count Anzahl der Elemente in der Collection FileSystemObject.Drives

.Count Anzahl der Elemente in der Collection FileSystemObject.Folder.Files

.Count Anzahl der Elemente in der Collection FileSystemObject.Folder.Folders

.cpuClass CPU-Klasse per navigator Objekt  
navigator.cpuClass  
"x86" Intel  
"68K" Motorola  
"Alpha" Digital  
"PPC" Motorola  
"Other" alles andere, auch Sun SPARC

.cpuClass CPU-Hersteller  
Behavior .style.clientCaps  
"x86" Intel  
"68K" Motorola  
"Alpha" Digital  
"PPC" Motorola  
"Other" alles andere, auch Sun SPARC

.cpuClass CPU-Hersteller  
siehe Objekt window.clientInformation  
"x86" Intel  
"68K" Motorola  
"Alpha" Digital  
"PPC" Motorola  
"Other" alles andere, auch Sun SPARC

.cssText Style-Attribut-Wert (Style-Regel, CSS-Attributwert)  
ab IE 5.x  
Hinweis: attributes Collection referenziert nicht Style-Attribute !

Beispiel:

```

<P ID="ID_P" STYLE="color:'green'; font-weight:bold">Test-Text</P>
<BUTTON onclick="alert(ID_P.style.cssText)"> CSS-Attributwerte anzeigen</BUTTON>

```

.ctrlKey CTRL-Tasten-Status  
Objekt event  
false CTRL-Taste ist nicht gedrückt  
true CTRL-Taste ist gedrückt

Beispiel:

```

<HEAD>
<SCRIPT>
    function InnerTextSetzen(Zeiger, Kette)
    {Zeiger.innerText=Kette;}

```



```

function init()
{
    InnerTextSetzen(ID_Span1,'false');
    InnerTextSetzen(ID_Span2,'false');
}
function CtrlDown()
{
    if (event.ctrlLeft)
    { InnerTextSetzen(ID_Span1,'true'); }
    else
    {
        if (event.ctrlKey)
        { InnerTextSetzen(ID_Span2,'true'); }
    }
}

function CtrlUp()
{
    if (!event.ctrlKey)
    {
        InnerTextSetzen(ID_Span1,'false');
        InnerTextSetzen(ID_Span2,'false');
    }
}
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="CtrlDown();" onkeyup="CtrlUp();">
    <TABLE>
        <TR>
            <TD><I>Linke CTRL-Taste gedrueckt</I></TD>
            <TD><I>Rechte CTRL-Taste gedrueckt</I></TD>
        </TR>
        <TR>
            <TD ALIGN="center">
                <SPAN ID="ID_Span1"></SPAN>
            </TD>
            <TD ALIGN="center">
                <SPAN ID="ID_Span2"></SPAN>
            </TD>
        </TR>
    </TABLE>
</BODY>

```

.ctrlLeft      linken CTRL-Taste Status  
Objekt event  
nicht für Win9x  
nur für Dokument, das den Fokus hat  
false linke CTRL-Taste ist nicht gedrückt  
true rechte CTRL-Taste ist gedrückt

Beispiel:

```

<HEAD>
<SCRIPT>
function InnerTextSetzen(Zeiger, Kette)
{ Zeiger.innerText=Kette; }

function init()
{
    InnerTextSetzen(ID_Span1,'false');
    InnerTextSetzen(ID_Span2,'false');
}
function CtrlDown()
{
    if (event.ctrlLeft)
    { InnerTextSetzen(ID_Span1,'true'); }
    else
    {
        if (event.ctrlKey)
        { InnerTextSetzen(ID_Span2,'true'); }
    }
}

function CtrlUp()
{

```



```

        if (!event.ctrlKey)
        {
            InnerTextSetzen(ID_Span1,'false');
            InnerTextSetzen(ID_Span2,'false');
        }
    }
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="CtrlDown();" onkeyup="CtrlUp();">
    <TABLE>
        <TR>
            <TD><I>Linke CTRL-Taste gedrueckt</I></TD>
            <TD><I>Recchte CTRL-Taste gedrueckt</I></TD>
        </TR>
        <TR>
            <TD ALIGN="center">
                <SPAN ID="ID_Span1"></SPAN>
            </TD>
            <TD ALIGN="center">
                <SPAN ID="ID_Span2"></SPAN>
            </TD>
        </TR>
    </TABLE>
</BODY>

```

**.currentFrame** Nummer des aktuellen Frame einer Media-Datei auf der Timeline  
nicht alle Media-Typen unterstützen Frames  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO ID="ID_Media"
            SRC="test.avi"
            STYLE="width:175px; height:150px;"
    >
    </t:VIDEO>
    <BR>
    <SPAN ID="ID_Span"
        CLASS="time_line_Klasse"
        DUR="0.01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Media.currentFrame);"
    >
    </SPAN>
    <BR>
    <BUTTON ID="ID_Button"
        onclick="ID_Media.beginElement();"
    >
        Restart
    </BUTTON>
</BODY>
</HTML>

```

**.currentItem** Zeiger auf aktuellen Media-Eintrag (Objekt MediaItem)  
Eintrag stammt aus der Playliste, wenn diese existent ist, und ist der aktuelle Eintrag der Playliste  
siehe Behavior .style.mediaBar

**.data** Url der Daten des Objektes

**.dataFld** Datenquelle-Name vergeben (ID)

Beispiel 1:

```

<TABLE DATASRC="#ice_cream">
    <TR>
        <TD>
            <INPUT TYPE=TEXT DATAFLD="flavor">
        </TD>
    </TR>

```





&lt;/TABLE&gt;

Beispiel 2:

```
<SELECT DATASRC="#Anker" DATAFLD="KartenTyp">
  <OPTION>Visa
  <OPTION>Mastercard
  <OPTION>American Express
  <OPTION>Diner's Club
  <OPTION>Sparkasse
</SELECT>
```

.dataFormatAs	Datenquelle-Anzeigeart
"text"	Default. Data als Text anzeigen
"html"	Data als HTML anzeigen
"localized-text"	ab IE 5.01 Data in den lokalen Einstellungen des Clients anzeigen

Beispiel:

```
<DIV DATAFLD="Column2" DATAFORMATAS="html"></DIV>
```

.dataPageSize	Anzahl der sichtbaren Datensätze auf einer Tabellenseite
	Anzahl der Sätze pro Dataset-Anzeige
	siehe Objekt table

Beispiel:

Datensätze liegen in der Datei adress.txt  
 Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument  
 hat folgenden Inhalt:

1. Satz	<b>vorname;name;telefon</b>
ab 2. Satz	Guericke;"Otto, von";0815 Willmann;Theo;1234 Xantippe;Isa;5678 Arktin;Richard;1055

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
```

```
var SortierRichtung = true;
```

```
function Sortieren(FeldBezeichner)
```

```
{
  // Sortierrichtung festlegen
  var Kette = "-"; // Annahme
  if (SortierRichtung) {Kette = "+";}

```

```
  // nächste Sortierung umgekehrt
  SortierRichtung = !SortierRichtung;

```

```
  // sortieren
  ID_Datenbank.Sort= Kette + FeldBezeichner;

```

```
  // und das Ergebnis anzeigen
  ID_Datenbank.Reset();
}
```

```
function vorwaerts(AnzahlSaetze)
```

```
{
  // nächste Seite anzeigen
  document.all.ID_Tabelle.nextPage();

```

```
  // und Satzzeiger korrigieren
  for (var i = 0; i < AnzahlSaetze; i++)
  {

```

```
    if (ID_Datenbank.recordset.AbsolutePosition !=
        ID_Datenbank.recordset.RecordCount)
    {ID_Datenbank.recordset.MoveNext();}
  }

```

```
  if (ID_Datenbank.recordset.AbsolutePosition == ID_Datenbank.recordset.RecordCount)
  {alert("Letzter Datensatz erreicht!");}
}
```

```
function rueckwaerts(AnzahlSaetze)
```

```
{
  // vorhergehende Seite anzeigen

```



```

        document.all.ID_Tabelle.previousPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSätze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition > 1)
            {ID_Datenbank.recordset.MovePrevious();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == 1)
        {alert("Erster Datensatz erreicht!");}
    }
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
        CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
    <PARAM NAME="DataURL" VALUE="adress.txt">
    <PARAM NAME="UseHeader" VALUE="True">
    <PARAM NAME="FieldDelim" VALUE=";">
</OBJECT>

<TABLE ID="ID_Tabelle"
        DATASRC="#ID_Datenbank"
        DATAPAGESIZE=1
>
    <THEAD>
        <TR>
            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
            <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
        </TR>
    </THEAD>
    <TBODY>
        <TR>
            <TD><DIV DATAFLD="vorname"></DIV></TD>
            <TD><DIV DATAFLD="name"></DIV></TD>
            <TD><DIV DATAFLD="telefon"></DIV></TD>
        </TR>
    </TBODY>
</TABLE>
<BR>
<INPUT
        TYPE="button"
        VALUE="Zurueck"
        onclick=rueckwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->

<INPUT
        TYPE="button"
        VALUE="Vorwaerts"
        onclick=vorwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->
</BODY>
</HTML>

```

.dataSrc                      Datenquelle als Anker festlegen

Beispiel:

```

<TABLE DATASRC="#anker">
    <TR>
        <TD>
            <INPUT TYPE=TEXT DATAFLD="customer_name">
        <TD>
    </TR>
</TABLE>

```

.DateCreated                      Datum und Zeit der Ordnererstellung liefern

Beispiel:

```

var OrdnerName                      = "c:\\windows\\desktop\\";
var DateiSystem                    = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                          = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateCreated);

```

.DateCreated                      Datum und Zeit der Dateierstellung liefern



Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateCreated);
```

.DateLastAccessed Datum und Zeit des letzten Ordnerzugriffes liefern

Beispiel:

```
var OrdnerName      = "c:\\windows\\desktop\\";
var DateiSystem     = new ActiveXObject("Scripting.FileSystemObject");
var Ordner          = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateLastAccessed);
```

.DateLastAccessed Datum und Zeit des letzten Dateizugriffes liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateLastAccessed);
```

.DateLastModified Datum und Zeit der letzten Ordneränderung liefern

Beispiel:

```
var OrdnerName      = "c:\\windows\\desktop\\";
var DateiSystem     = new ActiveXObject("Scripting.FileSystemObject");
var Ordner          = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateLastModified);
```

.DateLastModified Datum und Zeit der letzten Dateiänderung liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateLastModified);
```

.decelerate Verlangsamung des Elementes auf der Timeline  
hat keinen Einfluss auf die Dauer der Timeline  
auch wirksam bei Wiederholungen per Attribute .repeatCount oder .repeatDur  
Summe der Werte der Attribute .accelerate und .decelerate darf nicht 1 überschreiten  
wenn ja, so werden beide Attribute ignoriert, also nicht verwendet  
siehe Objekt currTimeState und Behavior .style.time2  
siehe .accelerate

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<DIV ID="ID_Div" CLASS="time_line_klasse" STYLE=".....">
</DIV>
<t:ANIMATE TARGETELEMENT="ID_Div"
ATTRIBUTENAME="left"
TO="400"
DUR="3"
decelerate="1"
REPEATCOUNT="3"
>
</t:ANIMATE>
</BODY>
</HTML>
```

.declare Zeichenkette für Declare-Funktionalität des Objektes, das per OBJECT-Tag eingebunden wurde

.defaultCharset regionaler Standardzeichensatz (Charakter-Set) zum Encoden eines Objektes im Dokument

.defaultChecked Selektionsstatus des Checkbox-Control bzw. Radio Button-Control bzw. Radio Button-Control  
bezüglich Standard-Selektion  
(Objekt input checkbox bzw. Objekt input radio)  
wird bei Checkbox-Control auch verändert durch Eigenschaft .indeterminate  
true Default



	Control-Element hat Standardselektion
	false Control-Element hat keine Standardselektion
.defaultSelected	Default-Selektionsstatus der Option, also Objektes document.select.option des Objektes document.select standardgemäß ist immer die oberste Option selektiert es kann aus einer Optionengruppe immer nur genau 1 Option selektiert sein
	true Default
	false Option ist selektiert
	false Option ist nicht selektiert
.defaultStatus	Standard-Text der Statuszeile (nicht der aktuelle Text) siehe .status Verwendung in Eventhandler-Funktion: Es muss <b>return true;</b> kodiert werden siehe Objekt window
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars z.B. Reset des Formular löst Initialisierung des Elementes aus

Beispiel für INPUT-Objekt und dessen Varianten:

Wert ist immer String

für Input-Elemente (input Objekt) gelten folgende Standardwerte:

INPUT type=checkbox	on
INPUT type=reset	Reset
INPUT type=submit	Submit Query

alle anderen Input-Elemente haben keinen Standardwert

Hinweis: sendbar sind folgende Werte:

INPUT type=checkbox	selektierter Wert
INPUT type=file	Dateiname laut Eingabe
INPUT type=hidden	selektierter Wert einer Box-Control (selektierte Option)
INPUT type=password	Eingabewert
INPUT type=radio	selektierter Wert
INPUT type=reset	das Label des Elementes (falls Label existent ist)
INPUT type=submit	das Label des Elementes (falls Label existent ist)
INPUT type=text	Eingabewerte

Werte können gelesen und geschrieben werden  
nur lesen bei INPUT type=file

.defaultValue	Vorbelegung der TEXTAREA sichtbare Auswirkungen nur bei Instanzierung des Objektes Formular-Reset siehe textarea Objekt
.defer	Pars-Status des Scriptes per script Objekt download eines Scriptes kann beschleunigt werden, wenn es vom Parsen zurückgestellt wird
.description	Beschreibung des Run-Time-Errors deprecated: dafür Eigenschaft .message verwenden, die indentische Funktion hat siehe error JScript-Objekt
.designMode	Editierbarkeit des Dokumentes (Manipulierbarkeit) z.B. per Ausführung von Javascript
.deviceXDPI	Aktuelle Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch ) auf Bildschirm per screen Objekt

Beispiel 1:

```
<SCRIPT>
function ScaleFactorX()
{
    var ScaleFactor = screen.deviceXDPI / screen.logicalXDPI;
    return ScaleFactor;
}
</SCRIPT>
```

Beispiel 2:

```
<SCRIPT>
function fnScaleManually()
{
    // normale DPI-Auflösung (Dot per Inch)
    var constNorm = 96;
```



```

        // Zoomen
        if ( (screen.deviceXDPI == screen.logicalXDPI)
            && (screen.deviceXDPI > constNorm)
            )
        {
            document.body.style.zoom = constNorm / screen.logicalXDPI;
        }
    }
</SCRIPT>

```

**.deviceYDPI** Aktuelle Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm per screen Objekt

Beispiel 1:

```

<SCRIPT>
function ScaleFactorY()
{
    var ScaleFactor = screen.deviceXDPI / screen.logicalYDPI;
    return ScaleFactor;
}
</SCRIPT>

```

Beispiel 2:

```

<SCRIPT>
function fnScaleManually()
{
    // normale DPI-Auflösung (Dot per Inch)
    var constNorm = 96;

    // Zoomen
    if ( (screen.deviceYDPI == screen.logicalYDPI)
        && (screen.deviceYDPI > constNorm)
        )
    {
        document.body.style.zoom = constNorm / screen.logicalYDPI;
    }
}
</SCRIPT>

```

**.dialogArguments** Zeiger auf Objekt window.dialogArguments als Argumente für modale oder nicht modale Dialoge ab IE 5.x wird per.showModalDialog() bzw .showModelessDialog() instanziiert siehe Objekt window

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
function Anzeige()
{
    // Zeiger auf die Formularelemente holen
    var FormularElemente = ID_Formular.elements;

    // Formulardaten in einem privaten Objekt kapseln als Argument für modalen Dialog
    var FormularDaten = new Object();
    FormularDaten.firstName = FormularElemente.ID_Input1.value;
    FormularDaten.lastName = FormularElemente.ID_Input2.value;

    // Fenster als modalen Dialog anzeigen und dabei Eigenschaft .dialogArguments füllen
    // Mit der Übergabe der Daten erfolgt das Öffnen des neuen Fenster, das
    // sich auf die namensgleichen Eigenschaften von FormularDaten
    // beruft, deren Bezeichner mit in den Argumenten übergeben werden
    window.showModalDialog( "test.htm",
        FormularDaten,
        "dialogHeight:300px; dialogLeft:200px;"
    );
}
</SCRIPT>
</HEAD>
<BODY>
<FORM ID= "ID_Formular">
    Vorname:
    <INPUT TYPE="text" NAME="ID_Input1" VALUE="Vorname">
    <BR>
    Nachname:
    <INPUT TYPE="text" NAME="ID_Input2" VALUE="Nachname">

```



```
</FORM>
<BR>
<BUTTON onclick="Anzeige();" >Formulardaten im modalen Dialog anzeigen</BUTTON>
</BODY>
</HTML>
```

test.htm enthält:

```
<HTML>
<HEAD>
<SCRIPT>
    function Anzeigen()
    {
        var DialogArgumente = window.dialogArguments;

        // Es müssen namensidentische Bezeichner der Eigenschaften von
        //     DialogArgumente verwendet werden:
        //         firstName und lastName werden in der
        //             aufrufenden Webseite definiert
        //             und müssen hier ebenfalls verwendet werden
        document.writeln("Vorname = " + DialogArgumente.firstName);
        document.write("Nachname = " + DialogArgumente.lastName);
    }
</SCRIPT>
</HEAD>
<BODY onload="Anzeigen();">
</BODY>
</HTML>
```

<code>.dialogHeight</code>	Höhe des Dialogfensters, das mit Methoden <code>.showModalDialog()</code> bzw. <code>.showModelessDialog()</code> erzeugt wurde
	ab IE 5.x
	siehe Objekt <code>window</code>

Beispiel:

```
<HEAD>
<SCRIPT>
function ZeigeModalenDialog()
{
    event.srcElement.blur();                // Focus dem QuellElement wegnehmen
    window.showModalDialog("test.htm",
        "",
        "dialogWidth:5cm; dialogHeight:10cm; dialogTop:0cm; dialogLeft:0cm"
    );
}
</SCRIPT>
</HEAD>
<BODY>
    <SELECT onchange="ZeigeModalenDialog()">
        <OPTION>Eintrag 1</OPTION>
        <OPTION>Eintrag 2</OPTION>
        <OPTION>Eintrag 3</OPTION>
    </SELECT>
</BODY>
```

<code>.dialogLeft</code>	X-Koordinate der linken oberen Ecke des Dialogfensters, das mit Methoden <code>.showModalDialog()</code> bzw. <code>.showModelessDialog()</code> erzeugt wurde
	ab IE 5.x
	siehe Objekt <code>window</code>

Beispiel:

```
<HEAD>
<SCRIPT>
function ZeigeModalenDialog()
{
    event.srcElement.blur();                // Focus dem QuellElement wegnehmen
    window.showModalDialog("test.htm",
        "",
        "dialogWidth:5cm; dialogHeight:10cm; dialogTop:0cm; dialogLeft:0cm"
    );
}
</SCRIPT>
</HEAD>
<BODY>
```



```

<SELECT onchange="ZeigeModalenDialog()">
  <OPTION>Eintrag 1</OPTION>
  <OPTION>Eintrag 2</OPTION>
  <OPTION>Eintrag 3</OPTION>
</SELECT>
</BODY>

```

**.dialogTop** Y-Koordinate der linken oberen Ecke des Dialogfensters, das mit Methoden  
 .showModalDialog() bzw. .showModelessDialog()  
 erzeugt wurde  
 ab IE 5.x  
 siehe Objekt window

Beispiel:

```

<HEAD>
<SCRIPT>
function ZeigeModalenDialog()
{
  event.srcElement.blur(); // Focus dem QuellElement wegnehmen
  window.showModalDialog("test.htm",
    "",
    "dialogWidth:5cm; dialogHeight:10cm; dialogTop:0cm; dialogLeft:0cm"
  );
}
</SCRIPT>
</HEAD>
<BODY>
  <SELECT onchange="ZeigeModalenDialog()">
    <OPTION>Eintrag 1</OPTION>
    <OPTION>Eintrag 2</OPTION>
    <OPTION>Eintrag 3</OPTION>
  </SELECT>
</BODY>

```

**.dialogWidth** Breite des Dialogfensters, das mit Methoden  
 .showModalDialog() bzw. .showModelessDialog()  
 erzeugt wurde  
 ab IE 5.x  
 siehe Objekt window

Beispiel:

```

<HEAD>
<SCRIPT>
function ZeigeModalenDialog()
{
  event.srcElement.blur(); // Focus dem QuellElement wegnehmen
  window.showModalDialog("test.htm",
    "",
    "dialogWidth:5cm; dialogHeight:10cm; dialogTop:0cm; dialogLeft:0cm"
  );
}
</SCRIPT>
</HEAD>
<BODY>
  <SELECT onchange="ZeigeModalenDialog()">
    <OPTION>Eintrag 1</OPTION>
    <OPTION>Eintrag 2</OPTION>
    <OPTION>Eintrag 3</OPTION>
  </SELECT>
</BODY>

```

**.dir** Umflussrichtung  
 "ltr" Default.  
 Umfluss von links nach rechts  
 "rtl" Umfluss von rechts nach links

**.direction** Start-Scrollrichtung des marquee Objektes  
 "left" Default  
 Scrollrichtung links  
 "right" Scrollrichtung rechts  
 "down" Scrollrichtung runter  
 "up" Scrollrichtung rauf

**.disabled** Interaktionsfähigkeit  
 nur wenn sichtbar so User-Interaktion möglich  
 true Element nicht interaktionsfähig



false      Default, Element ist interaktionfähig

Beispiel 1:

```
<STYLE ID="ID_Style">
.styletest{background-color: black; color: white;}
</STYLE>
<SCRIPT>
function Wechsel()
{
    if(ID_P.enablement == "enabled")
    {
        ID_P.enablement = "disabled";
        ID_Button2.value = "unsichtbar";
        ID_Style.disabled = true;
        ID_Button1.disabled = true;
    }
    else
    {
        ID_P.enablement = "enabled";
        ID_Button2.value = "sichtbar";
        ID_Style.disabled = false;
        ID_Button1.disabled = false;
    }
}
</SCRIPT>
<P ENABLEMENT="enabled" ID="ID_P" CLASS="styletest">
    Test Text
    <INPUT TYPE="button" ID="ID_Button1" CLASS="styletest"
        VALUE="Testen" onclick="alert('Test')"
    >
</P>
<INPUT TYPE="button" ID="ID_Button2" VALUE="Wechseln"
    onclick="Wechsel()"
>
```

Beispiel 2 für Sekundenbalken:

```
<HTML>
<HEAD>
<SCRIPT>
// ++++++ globale Variablen, die verändert werden können
var SoundUrl = "56sec.mid";
var SoundDauerInSekunden = 56; // Dauer muss exakt stimmen !

var PixelBreiteProBalkenErweiterung = 10;

// ++++++ Browser-Typ ermitteln
// Dieser Quellcode muss VOR allen anderen Routinen codiert sein, damit zuerst abgearbeitet
var ns = document.layers ? true : false;
var ie = document.all ? true : false;

// ++++++ Routinen der Sekundenzählung
var SekundenZahler = 0;
var SekundenZahlerTimeoutID = null;

function SekundenZaehlen() // wird durch Rekursion alle Sekunde neu gestartet
{
    // Zähler erhöhen
    SekundenZahler++;

    // visuelle Anzeige im Dokument auffrischen, also alle Audrucke der Style-Werte
    // neu berechnen und damit alle DIV's neu visualisieren
    document.recalc();
}

function SekundenZaehlen_Start()
{
    // prüfen ob Sekundenzählen nicht bereits läuft
    if (SekundenZahlerTimeoutID == null)
    {
        // nicht aktiv

        // Rekursion starten: SekundenZaehlen() wird permanent alle Sekunde aktiviert
        SekundenZahlerTimeoutID = setInterval("SekundenZaehlen()", 1000);
    }
}
```





```

    }

function SekundenZaehlen_Stop()
{
    // prüfen ob Sekunden zählen aktiv ist
    if (SekundenZahlerTimeoutID != null)
    {
        // aktiv, also stoppen
        clearInterval(SekundenZahlerTimeoutID);
        SekundenZahlerTimeoutID = null;
    }
}

// ++++++++ Routine zur Erzeugung Sound-Objekt
function SoundObjektErzeugen(SoundFileUrl, SoundFileDauerInSekunden)
{
    this.SoundFileUrl = SoundFileUrl;
    this.SoundFileDauerInMillisekundeSekunden = SoundFileDauerInSekunden * 1000;
    // Timerzeit für Rekursion
    this.SoundFileBeendet = true; // kein Sound aktiv
}

// ++++++++ Routinen zur Wiedergabe Sound-Objekt
var SoundTimeoutID=0;

function SoundAbspielen()
{
    // prüfen ob Sound nicht bereits aktiv ist
    if (SoundObjekt.SoundFileBeendet)
    {
        // Anzeige initialisieren
        SekundenAnzeigeInit();

        // Sekundenzähler starten, wobei das Zählen eigenständig und parallel erfolgt
        SekundenZaehlen_Start();

        // Sound erzeugen und sofort starten durch Url-Zuweisung
        ID_BGSound.src=SoundObjekt.SoundFileUrl;
        SoundObjekt.SoundFileBeendet=false;

        // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen
        SoundTimeoutID = setTimeout(
            "SoundAbspielen()",
            SoundObjekt.SoundFileDauerInMillisekundeSekunden
        );
    }
    else
    {
        // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

        // und Meldung
        var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;
        var TimerUngenauigkeit2 = TimerUngenauigkeit1 / SoundDauerInSekunden;
        alert(
            "Wiedergabe beendet\nUngenauigkeit des Timers = "
            + TimerUngenauigkeit1.toString() + " Sekunden\n"
            + "also " + TimerUngenauigkeit2.toString() + " Ticks pro Sekunde"
        );
    }
}

// ++++++++ Sekunden-Anzeige initialisieren
function SekundenAnzeigeInit()
{
    // ----- Variablen init
    SekundenZahler = 0;
    SekundenZahlerTimeoutID = null;

    // ----- visuelle Anzeige erzeugen

```



```

// - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
//      Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
//      Der Ausdruck liefert den Wert , welcher sofort das Layout der
//      DIV's beeinflusst.
//      Jeder Ausdruck besitzt den SekundenZähler als Komponente.
//      Damit ändert sich der Wert des Ausdrucks.
//      Für die Neuberechnung des Ausdrucks ist der Aufruf von
//      document.recal()
//      nötig.
//      Dieser Aufruf erfolgt in SekundenZahlen(), also permanent pro Sekunde.
//      Damit wird der Style-Wert permanent neu berechnet.
//      Damit visualisieren sich die DIV's permanent neu.

// Sekundenbalken in der Style-Eigenschaft width (Breite) mit Ausdruck belegen,
//      also dynamisch anzeigen
ID_DIV_Balken.style.setExpression( "width",
                                   "SekundenZähler * PixelBreiteProBalkenErweiterung"
                                   );

// Sekundenzähler in der Eigenschaft .innerText mit Ausdruck belegen,
//      also dynamisch anzeigen
ID_DIV_SekundenZähler.setExpression("innerText","SekundenZähler.toString()");

// - - - Messlatte statisch anzeigen
ID_DIV_MessLatte.style.width = SoundDauerInSekunden * PixelBreiteProBalkenErweiterung;
ID_DIV_MessLatte.innerText  = "Der Sound dauert "
                               + SoundDauerInSekunden.toString()
                               + " Sekunden";
}

// ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
if (ie)
{
    document.write('<BODY></BODY>');

    document.write( ' <BGSOUND ID= "ID_BGSound" LOOP="0">'

    document.write( ' <DIV ID="ID_DIV_Balken"'
                    + ' STYLE="background-color:lightblue"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_SekundenZähler"'
                    + ' STYLE="color:hotpink;font-weight:bold"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_MessLatte"'
                    + ' STYLE="color:white;background-color:gray"'
                    + '>'
                    + '</DIV>'

                    );

    // ++++++ Sound initialisieren und starten mit Laden des Dokumentes

    // ---- Sound-Objekt erzeugen anhand globaler Variablen
    SoundObjekt = new SoundObjektErzeugen(SoundUrl, SoundDauerInSekunden);

    // ---- Sound-Objekt wiedergeben
    SoundAbspielen(); // meldet wenn Wiedergabe beendet ist
}
</SCRIPT>
</HEAD>
<BODY>
<!-- BODY-Teil muss leer bleiben!-->
</BODY>
</HTML>

```

.disabledUI

Sichtbarkeit des User-Interfaces der Media Bar  
siehe Behavior .style.mediaBar



Beispiel:

```
<DIV ID="ID_Div"
  STYLE="behavior:url(#default#mediaBar)"
>
</DIV>
<INPUT TYPE=button
  VALUE='abspielen von test.asx'
  onclick=" ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
           ID_Div.disabledUI = true;
           "
>
```

.doctype	Referenz auf Dokumenttyp des Dokumentes document.doctype
.document	Zeiger auf das Objekt document im Fenster siehe Objekt window
.document	Zeiger auf das HTML-Dokument im Popup-Fenster, das per .show() instanziiert wird Es können damit alle Eigenschaften und Methoden des Objektes document referenziert werden siehe Objekt window.popup

Beispiel für diverse Meldungsfenster per **jeweiligem** PopUp-Fenster ( es kann immer nur genau 1 PopUp-Fenster angezeigt werden):

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">

    var PopUpFensterFeld = new Array();

    // nachfolgende Felder beschreiben die PopUp-Fenster und müssen
    // identische Anzahl der Feldelemente haben
    // Feld-Index ist die Nummer des PopUp-Fensters

    var PopUpFenster_RahmenStyle_Feld = new Array
    (
        "solid black 4px",
        "solid blue 3px",
        "solid green 2px"
    );

    var PopUpFenster_HintergrundFarbe_Feld = new Array
    (
        "yellow",
        "gray",
        "orange"
    );

    var PopUpFenster_Inhalt_Feld = new Array
    (
        "<B>Inhalt PopUpFenster 0<B>",
        "Inhalt PopUpFenster 1",
        "<TT>Inhalt PopUpFenster 2<TT>"
    );

    var PopUpFenster_X_Koordinate_Feld = new Array
    (
        100,
        150,
        200
    );

    var PopUpFenster_Y_Koordinate_Feld = new Array
    (
        150,
        200,
        250
    );

    var PopUpFenster_Breite_Koordinate_Feld = new Array
    (
        80,
        140,
```



```

        200
    );

    var PopUpFenster_Hoehe_Koordinate_Feld = new Array
    (
        100,
        140,
        180
    );

    var AnzahlPopUpFenster = PopUpFenster_Hoehe_Koordinate_Feld.length;

    function PopupFensterInstanzieren(NummerDesPopUpFensters, ZeigerAufElternFenster)
        // NummerDesPopUpFensters ab 0
    {
        PopUpFensterFeld[NummerDesPopUpFensters] =
            ZeigerAufElternFenster.createPopup();
    }

    function PopupFensterFuellen(NummerDesPopUpFensters)
    {
        // Body des Popup-Fensters gestalten
        var PopUpFenster_Body =
            PopUpFensterFeld[NummerDesPopUpFensters].document.body;

        PopUpFenster_Body.style.backgroundColor =
            PopUpFenster_HintergrundFarbe_Feld[NummerDesPopUpFensters];

        PopUpFenster_Body.style.border =
            PopUpFenster_RahmenStyle_Feld[NummerDesPopUpFensters];

        PopUpFenster_Body.innerHTML =
            PopUpFenster_Inhalt_Feld[NummerDesPopUpFensters];
    }

    function PopupFensterAnzeigen(NummerDesPopUpFensters,
        ObjektZuDemPopUpFensterRelativPositioniertIst
    )
    {
        // Popup-Fenster anzeigen und damit öffnen
        PopUpFensterFeld[NummerDesPopUpFensters].show(
            PopUpFenster_X_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Y_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Breite_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Hoehe_Koordinate_Feld[NummerDesPopUpFensters],
            ObjektZuDemPopUpFensterRelativPositioniertIst
        );
    }

    function PopupFensterSchliessen(NummerDesPopUpFensters)
    {
        var Zeiger = PopUpFensterFeld[NummerDesPopUpFensters];

        if (Zeiger.isOpen)
        { Zeiger.hide(); }
    }

    function Init()
    {
        // PopUpFenster instanzieren im aktuellen Fenster (window)
        for (var i = 0 ; i < AnzahlPopUpFenster; i++)
        {
            PopupFensterInstanzieren(i, window);
            PopupFensterFuellen(i);
        }
    }
</SCRIPT>
</HEAD>
<BODY onload="Init();">
    Durch Klick ausserhalb des Popup-Fensters wird dieses auch automatisch geschlossen.
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 anzeigen"

```



```

        onclick=" PopupFensterAnzeigen(0, document.body);"
    >
    <INPUT  TYPE=button
        VALUE="PopUpFenster 1 anzeigen"
        onclick=" PopupFensterAnzeigen(1, document.body);"
    >
    <INPUT  TYPE=button
        VALUE="PopUpFenster 2 anzeigen"
        onclick=" PopupFensterAnzeigen(2, document.body);"
    >
    <BR>
    <INPUT  TYPE=button
        VALUE="PopUpFenster 0 schliessen"
        onclick=" PopUpFensterSchliessen(0);"
    >
    <INPUT  TYPE=button
        VALUE="PopUpFenster 1 schliessen"
        onclick=" PopUpFensterSchliessen(1);"
    >
    <INPUT  TYPE=button
        VALUE="PopUpFenster 2 schliessen"
        onclick=" PopUpFensterSchliessen(2);"
    >
</BODY>
</HTML>

```

**.documentElement**                      Referenz auf Wurzelknoten (root node) des Dokumentes liefern  
 Beispiel:

```

<SCRIPT>
    function HoleHTML()
    {ID_Textarea.value= document.documentElement.innerHTML;}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS = 50 ROWS = 10>
</TEXTAREA>

```

**.domain**                                  Domain-Suffix des Dokumentes  
 Kommunikation von Dokumenten verschiedener Urls mit gemeinsamen Domainsuffix  
 document.domain

Beispiel:

**home.test.com** und **www.test.com** können kommunizieren,  
 wenn Kette auf "**test.com**" gesetzt wurde in den Dokumenten beider Urls

**.downloadCurrent**                      Anzahl der bisher downgeloadeten Bytes beim Laden einer Media-Datei auf der Timeline  
 (Laden des Daten-Stream in Form der Media-Datei)  
 nur für Media-Datei mit Datenfluss  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:MEDIA          ID="ID_Media"
                     SRC="test.asx"
                     SYNVBHAVIOR="locked"
                     TIMEACTION="display"
    >
    </t:MEDIA>
    <SPAN  ID="ID_Span"
        CLASS="time_line_klasse"
        DUR="0.1"
        REPEATCOUNT="indefinite"
        onrepeat="ID_Span.innerText= ID_Media.downloadCurrent;"
    >
        0
    </SPAN>
</BODY>
</HTML>

```

**.downloadTotal**                          Anzahl der insgesamt downgeloadeten Bytes nach dem kompletten Laden einer Media-Datei  
 auf der Timeline



(Laden des Daten-Stream in Form der Media-Datei)  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
SRC="test.asx"
SYNVBEHAVIOR="locked"
TIMEACTION="display"
>
</t:MEDIA>
<SPAN ID="ID_Span"
CLASS="time_line_klasse"
DUR="0.1"
REPEATCOUNT="indefinite"
onrepeat="ID_Span.innerText= ID_Media.downloadTotal;"
>
0
</SPAN>
</BODY>
</HTML>
```

.Drive Buchstaben des Laufwerkes liefern, auf dem der Ordner liegt

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Drive);
```

.Drive Buchstaben des Laufwerkes liefern, auf dem die Datei liegt

Beispiel:

```
var DateinameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.Drive);
```

.DriveLetter Laufwerksbuchstaben liefern (ohne : und \\ etc.)

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Buchstabe = Laufwerk.DriveLetter;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Buchstabe);
```

.DriveType Laufwerkstyp liefern

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Typ = Laufwerk.DriveType;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Typ);
```

.dropEffect Cursor-Layout bei Drop  
 wird von Eigenschaft .effectAllowed beeinflusst  
 nur Objekt event.dataTransfer  
 "copy" Copy-Cursor wird angezeigt  
 "link" Link-Cursor wird angezeigt  
 "move" Move-Cursor wird angezeigt  
 "none" Default, Standard-Cursor wird angezeigt

Beispiel für Clipboard-Nutzung mit Drag&Drop:

```
<HTML>
<HEAD>
<SCRIPT>
// ##### Quellobjekt #####
```



```

function EventHandlerFuerOnDragStart()
    // Quellobjekt löst Event aus:
    //      in Quelle wird markiert und selektiert
{
    // selektierte Quell-Daten in die Zwischenablage puffern
    //      (Puffer wird per window.event-Objekt verwaltet)
    //      Datentyp ist hier uninteressant, da für die Zwischenablage
    //      der Typ automatisch erkannt wird, also
    //      Speicherung der Daten in der Zwischenablage
    //      immer typgerecht ist
    var ZwischenAblage = window.event.dataTransfer;

    // und diese Daten als verschiebbar erklären:
    //      aus der Quelle in die Zwischenablage
    ZwischenAblage.effectAllowed = "move";
}

##### Zielobjekt #####
function EventHandlerFuerOnDragEnter
    // Zielobjekt löst Event aus
    // Maus über Ziel nach dem Draggen der Daten,
    //      UND Maustaste ist noch nicht losgelassen
    // also Zielobjekt wird mit den Daten betreten
{
    // Daten adressieren
    var ZwischenAblage = window.event.dataTransfer;

    // und diese Daten als verschiebbar erklären:
    //      aus der Zwischenablage in das Zielobjekt
    ZwischenAblage.dropEffect = "move";

    // Event-Handler-Rückkehrcode
    var EventObjekt = window.event;
    EventObjekt.returnValue = false;
}

function EventHandlerFuerOnDrop
    // Zielobjekt löst Event aus
    // Maus über Ziel nach dem Droppen der Daten,
    //      UND Maustaste wird losgelassen
{
    // Ziel adressieren, das mit ondrop-Ereignis behandelt werden soll
    var Ziel = window.event.srcElement;

    // Daten in der Zwischenablage adressieren und holen
    //      (Puffer wird per window.event-Objekt verwaltet)
    var ZwischenAblage = window.event.dataTransfer;

    // und Daten vom Texttyp aus der Zwischenablage im Ziel ablegen,
    //      da Ziel nur Textdaten empfangen kann
    Ziel.innerText += Daten.getData("text");

    // Event-Handler-Rückkehrcode
    var EventObjekt = window.event;
    EventObjekt.returnValue = false;
}

function EventHandlerFuerOnDragOver
    // Zielobjekt löst Event aus
{
    // nichts tun ausser Rückkehrcode des Handlers liefern
    var EventObjekt = window.event;
    EventObjekt.returnValue = false;
}
</SCRIPT>
</HEAD>
<BODY>
    <DIV ondragstart=" EventHandlerFuerOnDragStart ()"
    >
        Diesen Text markieren und draggen
    </DIV>
    <BR>
    <DIV
        ondragover="EventHandlerFuerOnDragOver()"
        ondragenter="EventHandlerFuerOnDragEnter()"

```



```

ondrop="EventHandlerFuerOnDrop()"
>
An diese Stelle den Text dropfen
</DIV>
</BODY>
</HTML>

```

.dur

Dauer Objektaktivitäten laut Eigenschaft .timeAction  
alternativ: Eigenschaft .end  
siehe Objekt currTimeState und Behavior .style.time2  
Wert im Time-Format des Behavior

z.B.	"h:min:s.f"
<b>nicht</b>	id.event
<b>nicht</b>	id.event+zeit_wert_als_string
Beispiele	
"25:45:10"	25 Stunden, 45 Minuten, 10 Sekunden
"45:35"	45 Minuten, 35 Sekunden
"45:00.275"	45 Minuten, 0,275 Sekunde
"10.5"	10,5 Sekunden

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter"
TYPE="fade"
DUR="8"
END="4"
TARGETELEMENT="ID_Div"
>
</t:TRANSITIONFILTER>
<DIV ID="ID_Div"
CLASS="time_line_klasse"
DUR="indefinite"
STYLE="background-image:url(test.gif); background-repeat: no-repeat;"
>
</DIV>
</BODY>

```

.dur

zeitliche Gesamtdauer des Übergangsfilters per .style.time2.transitionFilter Behavior-Objekt  
siehe Objekt currTimeState und Behavior .style.time2  
Timeformat des time2-Behavior

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter"
FROM="0.3"
BY="0.4"
TYPE="barWipe"
DUR="3"
TARGETELEMENT="ID_Div"
>
</t:TRANSITIONFILTER>
<DIV ID="ID_Div"
CLASS="time_line_Klasse"
DUR="indefinite"
STYLE="position:relative; left:20px; width:420px; height:100px;
background-image:url(test.gif); background-repeat: no-repeat;"
>
</DIV>
</BODY>
</HTML>

```





.duration	Dauer des MediaItem Objektes in Sekunden siehe Behavior .style.mediaBar																						
.dynsrc	Adresse von Videoclip oder VRML Hinweis: für statisches Bild bitte Eigenschaft .src verwenden !!																						
.E	eulersche Zahl siehe Script-Objekt Math																						
.effectAllowed	Art von Drag und Drop beeinflusst Eigenschaft .dropEffect nur Objekt event.dataTransfer <table> <tr><td>"copy"</td><td>kopieren</td></tr> <tr><td>"link"</td><td>verlinken</td></tr> <tr><td>"move"</td><td>verschieben</td></tr> <tr><td>"copyLink"</td><td>kopieren bzw. verlinken, je nach dem was Ziel zulässt</td></tr> <tr><td>"copyMove"</td><td>kopieren bzw. verschieben, je nachdem was Ziel zulässt</td></tr> <tr><td>"linkMove"</td><td>verschieben bzw. verlinken, je nachdem was Ziel zulässt</td></tr> <tr><td>"all"</td><td>alle Arten</td></tr> <tr><td>"none"</td><td>Ziel kann kein Dropping</td></tr> <tr><td></td><td>.dropEffect wird automatisch auf "none" gesetzt</td></tr> <tr><td>"uninitialized"</td><td>Default</td></tr> <tr><td></td><td>Standard-Dragging-Eigenschaft des Zieles verwenden</td></tr> </table>	"copy"	kopieren	"link"	verlinken	"move"	verschieben	"copyLink"	kopieren bzw. verlinken, je nach dem was Ziel zulässt	"copyMove"	kopieren bzw. verschieben, je nachdem was Ziel zulässt	"linkMove"	verschieben bzw. verlinken, je nachdem was Ziel zulässt	"all"	alle Arten	"none"	Ziel kann kein Dropping		.dropEffect wird automatisch auf "none" gesetzt	"uninitialized"	Default		Standard-Dragging-Eigenschaft des Zieles verwenden
"copy"	kopieren																						
"link"	verlinken																						
"move"	verschieben																						
"copyLink"	kopieren bzw. verlinken, je nach dem was Ziel zulässt																						
"copyMove"	kopieren bzw. verschieben, je nachdem was Ziel zulässt																						
"linkMove"	verschieben bzw. verlinken, je nachdem was Ziel zulässt																						
"all"	alle Arten																						
"none"	Ziel kann kein Dropping																						
	.dropEffect wird automatisch auf "none" gesetzt																						
"uninitialized"	Default																						
	Standard-Dragging-Eigenschaft des Zieles verwenden																						

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
// ##### Quellobjekt #####
function EventHandlerFuerOnDragStart()
// Quellobjekt löst Event aus:
//      in Quelle wird markiert und selektiert
{
// selektierte Quell-Daten in die Zwischenablage puffern
//      (Puffer wird per window.event-Objekt verwaltet)
//      Datentyp ist hier uninteressant, da für die Zwischenablage
//      der Typ automatisch erkannt wird, also
//      Speicherung der Daten in der Zwischenablage
//      immer typgerecht ist
var ZwischenAblage = window.event.dataTransfer;

// und diese Daten als verschiebbar erklären:
//      aus der Quelle in die Zwischenablage
ZwischenAblage.effectAllowed = "move";
}

// ##### Zielobjekt #####
function EventHandlerFuerOnDragEnter
// Zielobjekt löst Event aus
// Maus über Ziel nach dem Draggen der Daten,
//      UND Maustaste ist noch nicht losgelassen
//      also Zielobjekt wird mit den Daten betreten
{
// Daten adressieren
var ZwischenAblage = window.event.dataTransfer;

// und diese Daten als verschiebbar erklären:
//      aus der Zwischenablage in das Zielobjekt
ZwischenAblage.dropEffect = "move";

// Event-Handler-Rückkehrcode
var EventObjekt = window.event;
EventObjekt.returnValue = false;
}

function EventHandlerFuerOnDrop
// Zielobjekt löst Event aus
// Maus über Ziel nach dem Droppen der Daten,
//      UND Maustaste wird losgelassen
{
// Ziel adressieren, das mit ondrop-Ereignis behandelt werden soll
var Ziel = window.event.srcElement;
```



```

// Daten in der Zwischenablage adressieren und holen
//           (Puffer wird per window.event-Objekt verwaltet)
var ZwischenAblage = window.event.dataTransfer;

// und Daten vom Texttyp aus der Zwischenablage im Ziel ablegen,
//           da Ziel nur Textdaten empfangen kann
Ziel.innerText += Daten.getData("text");

// Event-Handler-Rückkehrcode
var EventObjekt = window.event;
EventObjekt.returnValue = false;
}

function EventHandlerFuerOnDragOver
// Zielobjekt löst Event aus
{
    // nichts tun ausser Rückkehrcode des Handlers liefern
    var EventObjekt = window.event;
    EventObjekt.returnValue = false;
}
</SCRIPT>
</HEAD>
<BODY>
    <DIV ondragstart=" EventHandlerFuerOnDragStart ()"
    >
        Diesen Text markieren und draggen
    </DIV>
    <BR>
    <DIV
        ondragover="EventHandlerFuerOnDragOver()"
        ondragenter="EventHandlerFuerOnDragEnter()"
        ondrop="EventHandlerFuerOnDrop()"
    >
        An diese Stelle den Text dropfen
    </DIV>
</BODY>
</HTML>

```

.enabled Sichtbarkeit des Media Bar Player  
Media Bar Player ist der Windows Media Player  
siehe Behavior .style.mediaBar

Beispiel:

```

<DIV ID="ID_Div"
STYLE="behavior:url(#default#mediaBar)"
>
    Div ist der Time-Container
</DIV>
<INPUT TYPE=button
VALUE='abspielen von test.asx'
onclick= "ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
ID_Div.disabledUI = true;
ID_Div.enabled = false;
"
>

```

.enctype Multipurpose Internet Mail Extensions (MIME) des Formulars für Encoding nach dem Senden  
der Daten  
ab IE 6.x

.end Objektaktivitäten laut Eigenschaft .timeAction beenden  
ab IE 6.x  
alternativ: Eigenschaft .dur  
siehe Objekt currTimeState und Behavior .style.time2  
Wert im Time-Format  
z.B. "h:min:s.f"  
id.event  
id.event+zeit\_wert\_als\_string

Beispiele für Kodierung von Time in der Eigenschaft .end:

```

object.end="objekt_zeiger.begin+10s" // warten bis Event "onbegin" zum Objekt laut
//           objekt_zeiger (z.B. laut ID-Attribut)
//           eintritt,
//           dann 10 Sekunden warten
//           dann Objekt laut object beenden

```



```

object.end="objekt_zeiger.focus+10s" // warten bis Event "onfocus" zum Objekt laut
//                                objekt_zeiger (z.B. laut ID-Attribut)
//                                eintritt,
//                                dann 10 Sekunden warten
//                                dann Objekt laut object beenden

object.end="2; objekt_zeiger.click+1" // 2 Sekunden nach dem Laden des Elternobjektes von
//                                object warten
//                                dann auf das Ereignis click zum Objekt laut
//                                objekt_zeiger warten
//                                dann 1 Sekunde warten
//                                dann Objekt laut object beenden

```

Hinweis: object laut ID-Attribut des Behavior-Objektes von .style.time2.

```

"25:45:10" 25 Stunden, 45 Minuten, 10 Sekunden
"45:35"    45 Minuten, 35 Sekunden
"45:00.275" 45 Minuten, 0,275 Sekunde
"10.5"     10,5 Sekunden

```

Beispiele für Kodierung von .end:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
CLASS=time_line_klasse
STYLE="COLOR:Red;"
BEGIN="0"
END="10"
TIMEACTION="visibility"
>
<H3>Test 1</H3>
</SPAN>
<SPAN ID="ID_Span2"
CLASS=time_line_klasse
STYLE="COLOR:Blue;"
BEGIN="3"
END="10"
TIMEACTION="visibility"
>
<H3>Test 2</H3>
</SPAN>
<SPAN ID="ID_Span3"
CLASS=time_line_klasse
STYLE="COLOR:Green;"
BEGIN="6"
END="10"
TIMEACTION="visibility"
>
<H3>Test 3</H3>
</SPAN>
</BODY>
</HTML>

```

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter"
TYPE="fade"
DUR="8"
END="4"
TARGETELEMENT="ID_Div"

```



```

>
</t:TRANSITIONFILTER>
<DIV ID="ID_Div"
      CLASS="time_line_klasse"
      DUR="indefinite"
      STYLE="background-image:url(test.gif); background-repeat: no-repeat;"
>
</DIV>
</BODY>
</HTML>

```

**.endSync**                      Ende der Animation von Elementen in einem gemeinsamen Time-Container  
                                  bei Ende der Timeline des Eltern-Time-Containers  
                                  Verhalten des Eltern-Time-Containers bezüglich der Animation seiner Kind-Elemente  
                                  siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:EXCL ID="parent"
        ENDSYNC="ID_Div2"
>
    <DIV ID="ID_Div1"
          CLASS="time_line_klasse"
          BEGIN="0"
          DUR="2"
    >
        Zeile 1
    </DIV>
    <DIV ID="ID_Div2"
          CLASS="time_line_klasse"
          BEGIN="2"
          DUR="2"
    >
        Zeile 2
    </DIV>
    <DIV ID="ID_Div3"
          CLASS="time_line_klasse"
          BEGIN="3"
          DUR="2"
    >
        Zeile 3
    </DIV>
</t:EXCL>
</BODY>
</HTML>

```

**.event**                      On-Eventbezeichner mit angefügten Klammernpaar  
                                  Script soll das Event verwalten  
                                  per script Objekt  
                                  immer mit Eigenschaft `.htmlFor` kodieren

Beispiel:

```

<SCRIPT ID="ID_Script" FOR="ID_Botton" EVENT="onclick()">
var Text1 = "Pferdchen hue ! "
var Text2 = "Pferdchen brrr !"

if (ID_Botton.innerText == Text1)
{ID_Botton.innerText = Text2; }
else
{
    if (ID_Botton.innerText == Text2)
    { ID_Botton.innerText = Text1; }
}
</SCRIPT>
<BUTTON ID="ID_Botton" onmouseout="alert(ID_Script.event)">Hue oder Brrr ? </BUTTON>

```

**.event**                      Zeiger auf das Objekt event im Fenster  
                                  siehe Objekt `window`



`.expando`                      Wirksamkeit von Attributen ein/aus  
                                  true        ein, Default  
                                  false      aus

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    document.expando = false; // für gesamtes Dokument abschalten
</SCRIPT>
</HEAD>
<BODY>
<DIV>
    <SPAN ID="ID_Span" UNSELECTABLE="on">
        Dieser Text ist <B>selektierbar !!<B>
    </SPAN>
</DIV>
</BODY>
</HTML>
```

`.expires`                      Zeitstempel der Daten im UserDataStore (User-Cache) per `.style.userData` Behavior  
                                  Zeitstempel als Verfallszeitpunkt für automatische Löschung gecachter Daten:  
                                  Daten werden **automatisch** durch den Browser gelöscht, wenn das Datum der Daten anhand des  
                                  Zeitstempels als verfallen erkannt wurde, also der aktuelle Zeitpunkt laut PC-Uhr jünger ist, als  
                                  der Zeitstempel.  
                                  String im UTC (Universal Time Coordinate)-Format

Beispiel:

```
<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //        es können diverse Cachennamen definiert und somit Versionen von Cache
    //        verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //        es können diverse Attribute definiert und somit Versionen von Input-Daten
    //        verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache save
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
```



```

        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

.external                      Zeiger auf das Objekt window.external im Fenster

.face                          Familie des Font-Typs (Font-Face) z.B. Courier

Beispiel:

```

<FONT ID="ID_Font" FACE="Arial">
<SCRIPT>
    alert(ID_Font.face);
    ID_Font.face = 'Courier';
    alert(ID_Font.face);
</SCRIPT>

```

.fgColor                      Vordergrundfarbe (Textfarbe) im Dokument  
#rrggbg Standard ist #000000  
vordefinierter Farbname (browserspezifisch)

.fileCreatedDate              Datum der Dokumenterstellung

Beispiel 1:

```
"Monday, December 08, 2000"
```

Beispiel 2:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

.fileModifiedDate            Datum der letzten Dokumentveränderung

Beispiel:

```
"Monday, December 08, 2000"
```

.Files                          Zeiger auf die interne Collection FileSystemObject.Folder.Files  
Collection kann nur z.T. über das JScript-Objekt Enumerator angesprochen werden

.fileSize                      Größe des Dokumentes

.FileSystem                   Typ des Dateisystems auf dem Laufwerk liefern

Beispiel:



```

var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk         = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Dateisystem = Laufwerk.FileSystem;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Dateisystem);

```

.fileUpdatedDate Datum des letzten Datei-Updates

Beispiel:

"Monday, December 08, 2000"

.fill Aktion eines Elementes zum Zeitpunkt des Ende der Timeline des Elementes aber  
bevor die Timeline des Elternelementes endet  
ist Ersatz für die Eigenschaft endHold, die deprecated ist und nicht mehr verwendet  
werden darf !!  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel 1:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:EXCL ID="ID_Excl"
BEGIN="0"
DUR="15"
>
<DIV ID="ID_Div"
CLASS="time_line_klasse"
BEGIN="0"
DUR="10"
FILL="freeze"
>
</DIV>
</t:EXCL>
</BODY>
</HTML>

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<DIV STYLE="height:100px">
<t:SEQ ID="ID_Seq"
REPEATCOUNT="indefinite" >
<t:MEDIA ID="ID_Media1"
SRC = "test1.jpg"
STYLE="position:absolute;"
DUR="3"
TIMECONTAINER="par"
FILL="transition"
>
<t:TRANSITIONFILTER TYPE="fade"
DUR="2"
>
</t:TRANSITIONFILTER>
</t:MEDIA>
<t:MEDIA ID="ID_Media2"
SRC = "test2.jpg"
STYLE="position:absolute;"
DUR="3"
TIMECONTAINER="par"
FILL="transition"
>
<t:TRANSITIONFILTER ID="ID_Transfilter"

```



TYPE="ClockWipe"  
DUR="2"

```
>
</t:TRANSITIONFILTER>
</t:MEDIA>
</t:SEQ>
</DIV>
</BODY>
</HTML>
```

**.firstChild** Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

Beispiel:

```
<SCRIPT>
var ZeigerAufErstesKind = Liste.firstChild; // liefert Referenz auf Listenelement 1
</SCRIPT>
<BODY>
<UL ID = "Liste">
<LI> Listenelement 1
<LI> Listenelement 2
<LI> Listenelement 3
</UL>
</BODY>
```

**.fontSmoothingEnabled** Fontglättung auf Bildschirm  
per screen Objekt  
false Default  
keine Glättung  
true Glättung ist aktiv

**.form** Zeiger auf das Formular (Formular als Container)  
ab IE 6.x für Elemente fieldSet, label, legend

**.frame** Art des Rahmens um eine Tabelle  
siehe Objekt table  
"void" Standard, kein Rahmen  
"above" Rahmen oberhalb  
"below" Rahmen unterhalb  
"border" Rahmen auf allen Seiten  
"box" Rahmen auf allen Seiten  
"hsides" Rahmen oben und unten  
"lhs" Rahmen links  
"rhs" Rahmen rechts  
"vsides" Rahmen links und rechts

**.frameBorder** Borderanzeige ein/aus beim Frame  
"0" oder "no" Anzeige aus  
"1" oder "yes" Default  
Anzeige ein  
Hinweis: wenn Anzeige aus, so wird Eigenschaft .borderColor ignoriert

**.frameElement** Zeiger auf Frame bzw. IFrame, die im Fenster liegen  
mit diesem Zeiger können im Fenster alle Eigenschaften und Methoden des Frame bzw. IFrame referenziert werden  
siehe Objekt window

Beispiel:

```
<SCRIPT LANGUAGE="JScript">
var FrameZeiger = window.frameElement;
FrameZeiger.src = "http://www.test.de ";
</SCRIPT>
```

**.frames** Zeiger auf die Collection document.frames im HTML-Dokument, das im Fenster angezeigt wird  
siehe Objekt window

**.frameSpacing** Zwischenraum in Pixel zwischen 2 benachbarten Frames

**.FreeSpace** Freien Speicher in Bytes auf Laufwerk ermitteln

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_FreierPlatz = Laufwerk.FreeSpace;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_FreierPlatz);
```





`.from` Startwert zur Werterhöhung bei Elemente-Animation(en) auf der Timeline  
per `.additive` oder `.accumulate`  
für die Objekte `animate`, `animateMotion` und `animateColor` gilt:  
`.from` wird von `.path` und `.values` überschrieben  
Achtung: Im Objekt `animatecolor` (t:ANIMATECOLOR) ist `.values` auch als Farbliste definiert !  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft `.width`) nach rechts:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function SpanInhaltInit(Kette)
    {
        // Zustand
        ID_Span1.innerText = "";

        // Kumulationsart
        ID_Span2.innerText =Kette;

        // Zaehler auf 1
        ID_Span3.innerText = "1";

        // DIV-Text festlegen
        ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
        if (Kette == "sum")
        {
            // Wertkumulation von .width
            ID_Div.innerText += " genau 1x kumulativ um ";
            ID_Div.innerText += ID_Animate.by
            ID_Div.innerText += " mal "
            ID_Div.innerText += ID_Animate.repeatCount;
        }
        else
        {
            // keine Wertkumulation von .width
            ID_Div.innerText += ID_Animate.repeatCount;
            ID_Div.innerText += " mal um ";
            ID_Div.innerText += ID_Animate.by
        }

        ID_Div.innerText += " aus";
    }

    function KumulationAus()
    {
        // Element darf nicht aktiv sein !!

        // ERST .accumulate definieren
        SpanInhaltInit("none");
        ID_Animate.accumulate="none";

        // DANACH Animation starten
        ID_Animate.beginElement();
    }

    function KumulationEin()
    {
        // Element darf nicht aktiv sein !!

        // ERST .accumulate definieren
        SpanInhaltInit("sum");
        ID_Animate.accumulate="sum";

        // DANACH Animation starten
        ID_Animate.beginElement();
    }

    function Anzeige()
```



```

    {
        ID_Span1.innerText = "Animation ist beendet ";
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE
        ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        FROM="150px"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    </t:ANIMATE>

    animierter DIV
    <DIV ID="ID_Div"
        CLASS="time_line_klasse"
        STYLE="position:absolute;
            top:125px;
            left:25px;
            height:100px;
            width:125px;
            border:solid black 2px;
            "
    >
    </DIV>
    <BR>

    Zustand der Animation:
    <SPAN ID="ID_Span1"></SPAN>
    <BR>

    Kumulationsart:
    <SPAN ID="ID_Span2"></SPAN>
    <BR>

    Durchlaufzaehler:
    <SPAN ID="ID_Span3" ></SPAN>
    <BR>

    <BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
    <BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
</BODY>
</HTML>

```

.from Start-Prozentsatz des kompletten Überganges bei Start der Animation des Übergangsfilters  
per .style.time2.transitionFilter Behavior-Objekt  
Start des Übergangsfilters mit bereits teilweise vollendetem Übergang  
nicht zusammen mit Eigenschaft .value kodieren, sonst wird .from ignoriert  
siehe Objekt currTimeState und Behavior .style.time2  
Ziffernfolge Floating point  
Wert numerisch von 0 bis 1  
Standard ist "0.0" oder "0"  
0 entspricht 0 % des kompletten Überganges  
1 entspricht 100% des kompletten Überganges  
Bsp.: 0.3 entspricht: Mit **bereits** 30% vollendeten kompletten Übergang die Animation starten

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:TRANSITIONFILTER
        ID="ID_Transfilter"
        FROM="0.3"
        BY="0.4"

```



```

        TYPE="barWipe"
        DUR="3"
        TARGETELEMENT="ID_Div"
    >
</t:TRANSITIONFILTER>

<DIV ID="ID_Div"
      CLASS="time_line_Klasse"
      DUR="indefinite"
      STYLE="position:relative; left:20px; width:420px; height:100px;
            background-image:url(test.gif); background-repeat: no-repeat;
            "
    >
</DIV>
</BODY>
</HTML>

```

**.fromElement** Referenz auf Maus-Eventauslösendes Quell-Element bei Mausbewegung  
nicht für Event ondragleave  
Objekt event

Beispiel:

```

<SCRIPT>
function TesteMaus(Zeiger)
{
    if( ! Zeiger.contains(event.fromElement) )
    {alert("Maus über Button erkannt");}
}
</SCRIPT>
<BUTTON onmouseover=" TesteMaus(this)">Ueberfahre mich mit der Maus</BUTTON>

```

**.galleryImg** Toobar "My Pictures Photo Support image toolbar" ein/ausschalten für aktuelles Image  
true oder "yes" Default  
Toolbar ein  
false oder "no" Toolbar aus

Hinweis: bei Image-Mappe (USEMAP, ISMAP) wird Toolbar immer eingeschaltet  
permanentes Einstellen der Toolbar für gesamte Webseite auch per META-Tag

HTTP-EQUIV="imagetoolbar" CONTENT="no" oder false bzw. "yes" oder true

Beispiel:

```

<IMG SRC="mypicture.jpg" HEIGHT="100px" WIDTH="100px" GALLERYIMG="yes">

```

**.hasAudio** Media-Datei mit Audioinhalt auf der Timeline  
nicht alle Media-Typen können Audio beinhalten  
Stummschaltung oder Lautstärkeinstellungen sind dabei egal  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Anzeige()
{
    alert('hasAudio: ' + ID_Media.hasAudio);
}
</SCRIPT>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
        SRC="test.wmv"
        BEGIN="0"
        FILL="remove"
        onmediacomplete="Anzeige ();"
    >
</t:MEDIA>
</BODY>
</HTML>

```

**.hasDownloadProgres** Start des Donloades einer Media-Datei auf der Timeline  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:



```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:VIDEO ID="ID_Video"
SRC="test.avi"
SYNCHBEHAVIOR="locked"
>
</t:VIDEO>
<SPAN ID="ID_Span"
CLASS="time_line_klasse"
DUR="0.1"
REPEATCOUNT="indefinite"
onrepeat="ID_Span.innerText= ID_Video.hasDownloadProgress;"
>
</SPAN>
</BODY>
</HTML>

```

**.hash** Teil des Wertes der Eigenschaft **.href** also Teil der Url als Anker  
Teil folgt direkt hinter dem Nummernkreuz # und ohne Nummernkreuz  
lesen: ist Leerkette wenn kein # gefunden

Beispiel 1:  
if (document.location.hash == "") {...}

**.hasLayout** Objekt mit Style-Layout  
ab IE 5.5  
Style-Layout wird erzeugt durch Kodierung einer Style-Eigenschaft (siehe style Objekt)  
oder durch Setzen der Eigenschaft **.contentEditable** auf true  
Folgende Objekte haben immer ein Style-Layout:  
BODY, IMG, INPUT, TABLE, TD  
Objekte mit

<b>.style.display</b>	auf inline-block
<b>.style.height</b>	
<b>float</b>	auf left oder right
<b>.style.position</b>	auf absolute
<b>.style.width</b>	
<b>.style.writingMode</b>	auf tb-rl
<b>.style.zoom</b>	

false Default.  
Objekt hat kein Layout  
true Objekt hat Layout

**.hasMedia** Objekt ist HTML-Media-Objekt  
siehe Objekt **currentTimeState** und Behavior **.style.time2**

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Anzeige()
{alert('hasMedia = ' + ID_Media.hasMedia);}
</SCRIPT>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
BEGIN="0"
SRC="/test /media/test.wmv"
FILL="remove"
onmediacomplete="Anzeige();"
>
</ t:MEDIA>
</BODY>
</HTML>

```

**.hasNextItem** aktueller Eintrag in der Playliste (Objekt **PlaylistInfo**) hat einen Nachfolger-Eintrag  
siehe Behavior **.style.mediaBar**  
false Default



kein Nachfolger  
true mit Nachfolger

.hasPlayList Verfügbarkeit der Behavior-Collection .style.time2.playList für Element auf der Timeline, also ob Element eine Playliste hat  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Anzeige()
{alert('hasPlayList: ' + ID_Media.hasPlayList); }
</SCRIPT>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
SRC="test.wmv"
BEGIN="0"
FILL="remove"
onmediacomplete="Anzeige();"
>
</t:MEDIA>
</BODY>
</HTML>
```

.hasVisual Media-Datei mit visuellem Inhalt auf der Timeline  
visuelle Daten werden auf dem Bildschirm sichtbar  
nicht alle Media-Typen können sichtbare Daten beinhalten  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Anzeige()
{
alert('hasAudio: ' + ID_Media.hasVisual);
}
</SCRIPT>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
SRC="test.wmv"
BEGIN="0"
FILL="remove"
onmediacomplete="Anzeige ();"
>
</t:MEDIA>
</BODY>
</HTML>
```

.height Höhe des Objektes in Pixel

.height Auflösung des Bildschirms in Höhe, also Anzahl der vertikalen Pixel  
per screen Objekt

.height Auflösung des Bildschirms in Höhe, also Anzahl der vertikalen Pixel  
Behavior .style.clientCaps

.hideFocus Focussierbarkeit  
true Focus ist nicht möglich  
false Default  
Focus ist möglich

Beispiel:

```
<BUTTON>fokussierbar</BUTTON>
<BUTTON HIDEFOCUS="true">nicht fokussierbar</BUTTON>
```



.higher Abbruch bzw. Pausierung der aktiven Animation durch Start der Animation eines höher priorisierten Kindes  
 Time-Container ist Behavior-Objektes .style.time2.excl  
 Folge der in HTML kodierten Objekte t:PRIORITYCLASS entspricht der absteigenden Folge der Prioritäten der Animation.  
 Jedes Kind ist ein t:PRIORITYCLASS  
 darf **kein** weiteres Behavior-Objekt .style.time2.priorityClass besitzen  
 wenn mehrere pausierende Kinder existieren:  
 Das zuletzt pausierte Kind startet zuerst.  
 Das zuerst pausierte Kind startet zuletzt.  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:EXCL ID="ID_Excl"
BEGIN="0;indefinite;"
>
<t:PRIORITYCLASS>
<SPAN ID="ID_Span1"
CLASS="time_line_Klasse"
BEGIN="5"
DUR="5"
>
Test1
</SPAN>
</t:PRIORITYCLASS>
<t:PRIORITYCLASS HIGHER="stop">
<SPAN ID="ID_Span2"
CLASS="time_line_Klasse"
BEGIN="0"
DUR="10"
>
Test2
</SPAN>
</t:PRIORITYCLASS>
</t:EXCL>
<BR>
<BUTTON ID="ID_Button" onclick="ID_Excl.beginElement();">
</BUTTON>
</BODY>
</HTML>
```

.history Zeiger auf das Objekt history (Verlauf)  
 siehe Objekt window

.host **Hostname** und **Port** einer Location oder Url in der Form "**hostname:port**"

Beispiel:

```
document.location.host;
```

.hostname **Hostname** einer Location oder Url in der Form "**hostname:port**"  
 kann Domain oder IP sein

.href Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag)  
 siehe auch Eigenschaften .rel und .rev

Beispiel 1:

```
document.location.href;
```

Beispiel 2 für globalen Style per HEAD:

```
<HEAD>
<STYLE>
.an {text-decoration: underline overline; color:blue;}
.aus {text-decoration: none; color:black;}
</STYLE>
</HEAD>
<BODY>
<A HREF="test.htm"
```



```

        CLASS="aus"
        onmouseover="this.className='an';"
        onmouseout="this.className='aus';"
    >
    </A>
</BODY>

```

Beispiel 3 für Abschaltung Rahmen um Link:

Immer, wenn Sie auf einen Link oder einen Button klicken, zieht der Internet Explorer einen kleinen, gestrichelten Rahmen darum, der erst beim Klick auf ein anderes Objekt wieder verschwindet. Dieser Rahmen ist vor allem bei Imagemaps störend. Der HTML-Code für das Objekt (z.B. einen Link) ist onclick="this.blur()" zu erweitern.

```
<a href="seite.htm" onclick="this.blur()">Link</a>
```

Beispiel 4 für Webseite mit eigenem Icon ab IE 5.x:

```
<LINK REL="SHORTCUT ICON" HREF="name.ico">
```

name.ico: name ist beliebig, auch mit Pfad

ICO-Datei: 32x32 Pixel mit 16 Farben

oder 16x16 Pixel mit 256 Farben

ist BMP-Datei, die zu ICO-Datei konvertiert werden muss

(kann nicht jedes Grafik-Programm, aber Microsoft bietet dafür IconPro an)

.href                      Url einer externen Style-Sheet-Ressource-Datei (externe Style-Datei \*.css)  
siehe Objekt styleSheet

.hreflang                Sprachcode des Objektes laut RFC1766

Beispiel:

```
<A HREF="www.test.com" HREFLANG="en-US">Link</A>
```

.hspace                   horizontaler Abstand in Pixel zum Elternobjekt

.htmlFor                   ID des Label **nur für die Realisierung der Beschriftung**  
ist nicht das ID laut ID-Attribut  
per label Objekt

.htmlFor                   Referenz auf Objekt, das das Event laut Eigenschaft .event verwalten soll  
und dafür einen Scriptaufruf im Ereignishandler onxxx besitzt  
per script Objekt  
immer mit Eigenschaft .event kodieren

Beispiel:

```

<SCRIPT ID="ID_Script" FOR="ID_Botton" EVENT="onclick()">
    var Text1 = "Pferdchen hue ! "
    var Text2 = "Pferdchen brrr !"

    if (ID_Botton.innerText == Text1)
    {ID_Botton.innerText = Text2; }
    else
    {
        if (ID_Botton.innerText == Text2)
        { ID_Botton.innerText = Text1; }
    }
</SCRIPT>
<BUTTON ID="ID_Botton" onmouseout="alert(ID_Script.event)">Hue oder Brrr ? </BUTTON>

```

.htmlText                   HTML-Text im Textbereich  
nicht den Plaintext-Anteil liefern  
per textrange Objekt  
nur unter Windows 32-Bit

.httpEquiv                Informationen des HTTP Response Header per meta Objekt  
Beschaffung der Informationen per Serverfunktionen  
HttpQueryInfo und QueryInfo  
Dieser Kopf wird von Suchmaschine, Server und Browser verwendet  
der Wert der Information wird in der Eigenschaft .content abgelegt

Beispiele:

```

Dokument alle 2 Sekunden neu laden
<META HTTP-EQUIV="REFRESH" CONTENT=2>

```

```

Zeichensatz des Dokumentes
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=Windows-1251">

```



Themenunterstützung abschalten

```
<META HTTP-EQUIV="MSTHEMECOMPATIBLE" CONTENT="no">
```

HTML-Seiten-Refresh für WebCam

```
<META HTTP-EQUIV="refresh" CONTENT="sekundenzahl">
```

```
<META HTTP-EQUIV="Expires" CONTENT="Tue, 04 Dec 1997 21:29:026 GMT">
```

Datum egal, aber auf jedenfall älter als aktuelles, damit die HTML-Seite als verfallen gilt

```
<META HTTP-EQUIV="Pragma" CONTENT="nocache">
```

Seite nicht in Browsercache speichern

```
<META HTTP-EQUIV="Cache-Control" CONTENT="nocache">
```

Seite nicht aus Browsercache laden

um sicher zu gehen, daß immer die Seite mit garantiert aktuellem Stand geladen wird

→ eventuell ist Proxyserver nicht aktuell

```
<META HTTP-EQUIV="expires" CONTENT=sekunden_wert>
```

sekunden\_wert: Wartezeit in Sekunden bis zum nächsten Laden unter Umgehung des Proxyserver-Cache → 0, also immer umgehen und sofort laden

**.id** Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)  
ID-Attribut in HTML: Wert ist String alphanumerisch  
muss mit Buchstaben beginnen  
Unterstrich \_ verwendbar  
kann in " " bzw. ' ' kodiert werden, muss aber nicht  
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).  
Zeiger aus ID bilden var Zeiger = eval(object.id);  
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

Beispiel:

```
<SCRIPT>
function Anzeige(ObjektZeiger)
{alert('ID = ' + ObjektZeiger.id);}
</SCRIPT>
<TABLE BORDER COLS=3>
<TR>
<TD ID="ID_Zelle1" onclick=" Anzeige(this);">Zelle 1</TD>
<TD ID="ID_Zelle2" onclick=" Anzeige(this);">Zelle 2</TD>
<TD ID="ID_Zelle3" onclick=" Anzeige(this);">Zelle 3</TD>
</TR>
</TABLE>
```

**IMMEDIATEEND** wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software

**.implementation** Referenz auf Objekt implementation zum Dokument

**.indeterminate** Grauzustand (Dimmed) **und** Selektiertheit des Checkbox-Control

Achtung: verändert Eigenschaften .checked .status und .defaultChecked

verändert **nicht** Anwählbarkeit durch User (Attribut DISABLED, Eigenschaft .disabled)  
also ist der Focus weiterhin veränderbar !

wenn User auf per Eigenschaft .indeterminate angegrautes

Checkbox-Control klickt, so wird

Angegrautheit aufgehoben (.indeterminate auf false)

und der Status geändert (.status etc. neu gesetzt)

false	Default
	nicht selektiert <b>und</b> nicht grau
true	selektiert <b>und</b> grau

**.index** laufende Nummer der Option, also Objektes document.select.option, in einer List-Box als document.select Objekt

**.index** Index des Objektes playItem in der Behavior-Collection .style.time2.playList  
Track entspricht playItem Objekt  
Trackliste liegt in der Behavior-Collection .style.time2.playList als Playliste:  
Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:

```
var Wert = object.playList.index; // Wert Integer, ab 0
```

```
var Wert = object.playList.aktiveTrack.index; // Wert Integer, ab 0
```





Beispiel 2:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Anzeige()
    {
        alert('Index: ' + ID_Media.playlist.activeTrack.index);
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:MEDIA          ID="ID_Media"
                      SCR="test.asx"
                      BEGIN="0;"
    >
    </t:MEDIA>
    <BUTTON          ID="ID_Button"
                    onclick="Anzeige();"
    >
    </BUTTON>
</BODY>
</HTML>
```

.index  
Index des Playlisten-Eintrages in der Behavior-Collection .style.time2.playlist  
Diese Eigenschaft ist eigentlich nur sinnig für die Kodierung zum aktiven Track..  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
var PlaylistElement = zeiger_auf_timer_objekt.playlist.item(Wert);
// Wert      Index in der Behavior-Collection .style.time2.playlist
//              ab 0
```

.....

```
var VergessenerIndex = PlaylistElement.index;

var IndexAktiverTrack = zeiger_auf_timer_objekt.playlist.activeTrack.index;

// zeiger_auf_timer_objekt      laut ID-Attribut
```

.innerHTML  
Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag  
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>  
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,  
also nach dem Laden des Objektes  
aber wirksam erst mit parsen des HTML-Endetags  
Plain-Text  
HTML-Elemente je nach Objekt möglich  
Script: muss mit DEFER-Attribut kodiert sein  
schreiben: wenn Bereich nicht leer, so komplett überschreiben  
wenn Bereich leer so einfügen  
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD,  
TITLE, TR.

Beispiel 1:

```
<P onmouseover="this.innerHTML+=<B>on Mouse Over erkannt</B>"
  onmouseout="this.innerHTML+=<I>on Mouse Out erkannt</I>">
  <I>Dieser Text wird veraendert</I>
</P>
```

Beispiel 2:

```
<HTML>
<SCRIPT>
    function Veraendern()
    {
        // Scriptcode erzeugen zur Laufzeit
        var ScriptText = "<SCRIPT DEFER>"; // muss DEFER sein !!!
        ScriptText = ScriptText
            + "function Test(){ alert('Hallo von Script.')}";
        ScriptText = ScriptText
            + "</SCRIPT" + ">";

        // neuer Text erzeugen zur Laufzeit
```



```

var TextMitHTML = "<input type=button onclick="
                  + "Test()" // Aufruf der Funktion im Script
                  + " value=Click Me'><BR>";

// Text und Script als neuer Inhalt des Div
ID_Div.innerHTML = TextMitHTML + ScriptText;
    }
</SCRIPT>
<BODY onload="Veraendern();">
    <DIV ID="ID_Div">Alter Inhalt</DIV>
</BODY>
</HTML>

```

Beispiel 3: Anstelle der Alert-Box wird oft ein per STYLE-Attribut wohl positionierter DIV benutzt

```

<DIV ID="ID_Div" STYLE="...">Hier erscheint der Kommentar</DIV>
in dem ein Text mit oder ohne HTML-Tags angezeigt wird. Dieser DIV wird in seiner Eigenschaft
ID_Div.innerHTML durch ein anderes HTML-Element gefüllt, das einen Kommentar loswerden will.
ID_Div.innerHTML="<B>Das ist ein fetter Kommentar</B>";
Hinweis: Die Status-Zeile des Browserfenster sollte nur den Status-Meldungen des Browsers vorbehalten sein.

```

Beispiel 4 für Ersatz von document.write() durch die Eigenschaft .innerHTML (nur IE):

```

<SCRIPT>
var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild ">';

function Laden()
{
    ID_Div.innerHTML=Kette; // wird sofort geparkt, also IMG-Tag ausgeführt
                           // ( anstelle von eval()
                           // bzw. document.write() )

    ID_IMG.SRC="";
    ID_IMG.style.display="block";

    ID_IMG.onerror= IMGAltTextAufFehlerMeldung; // ohne ()
}

function IMGAltTextAufFehlerMeldung ()
{
    ID_IMG.alt="Das Bild konnte nicht geladen werden.";
    return true;
}
</SCRIPT>
<INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
<DIV ID="ID_Div"></DIV>

```

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag  
 ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>  
 dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,  
 also nach dem Laden des Objektes  
 aber wirksam erst mit parsen des HTML-Endetags  
 nur Plain-Text also keine HTML-Elemente und kein Script  
 schreiben: wenn Bereich nicht leer, so komplett überschreiben  
 wenn Bereich leer so einfügen  
 nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

Beispiel 1:

```

<P ID="ID_P">Dieser Text wird verändert</P>
<BUTTON onclick=" ID_P.innerText='Neuer Text'">Neuer Text</BUTTON>
<BUTTON onclick=" ID_P.innerText= Dieser Text wird verändert">Reset</BUTTON>

```

Beispiel 2: Anstelle der Alert-Box wird oft ein per STYLE-Attribut wohl positionierter DIV benutzt

```

<DIV ID="ID_Div" STYLE="...">Hier erscheint der Kommentar</DIV>
in dem ein Text ohne HTML-Tags angezeigt wird. Dieser DIV wird in seiner Eigenschaft
ID_Div.innerText durch ein anderes HTML-Element gefüllt, das einen Kommentar loswerden will.
ID_Div.innerText="Das ist ein Kommentar";
Hinweis: Die Status-Zeile des Browserfenster sollte nur den Status-Meldungen des Browsers vorbehalten sein.

```

Beispiel 3 für Sound mit Sekundenanzeige:

```

<HTML>
<HEAD>
<SCRIPT>
// ++++++ globale Variablen, die verändert werden können
var SoundUrl = "56sec.mid";

```



```

var SoundDauerInSekunden    = 56; // Dauer muss exakt stimmen !

var PixelBreiteProBalkenErweiterung = 10;

// ++++++++ Browser-Typ ermitteln
// Dieser Quellcode muss VOR allen anderen Routinen codiert sein, damit zuerst abgearbeitet
var ns = document.layers ? true : false;
var ie = document.all ? true : false;

// ++++++++ Routinen der Sekundenzählung
var SekundenZahler          = 0;
var SekundenZahlerTimeoutID = null;

function SekundenZaehlen()    // wird durch Rekursion alle Sekunde neu gestartet
{
    // Zähler erhöhen
    SekundenZahler++;

    // visuelle Anzeige im Dokument auffrischen, also alle Audrucke der Style-Werte
    // neu berechnen und damit alle DIV's neu visualisieren
    document.recalc();
}

function SekundenZaehlen_Start()
{
    // prüfen ob Sekundenzählen nicht bereits läuft
    if (SekundenZahlerTimeoutID == null)
    {
        // nicht aktiv

        // Rekursion starten: SekundenZaehlen() wird permanent alle Sekunde aktiviert
        SekundenZahlerTimeoutID = setInterval("SekundenZaehlen()", 1000);
    }
}

function SekundenZaehlen_Stop()
{
    // prüfen ob Sekundenzählen aktiv ist
    if (SekundenZahlerTimeoutID != null)
    {
        // aktiv, also stoppen
        clearInterval(SekundenZahlerTimeoutID);
        SekundenZahlerTimeoutID = null;
    }
}

// ++++++++ Routine zur Erzeugung Sound-Objekt
function SoundObjektErzeugen(SoundFileUrl, SoundFileDauerInSekunden)
{
    this.SoundFileUrl          = SoundFileUrl;
    this.SoundFileDauerInMillisekundeSekunden = SoundFileDauerInSekunden * 1000;
                                                    // Timerzeit für Rekursion
    this.SoundFileBeendet      = true; // kein Sound aktiv
}

// ++++++++ Routinen zur Wiedergabe Sound-Objekt
var SoundTimeoutID=0;

function SoundAbspielen()
{
    // prüfen ob Sound nicht bereits aktiv ist
    if (SoundObjekt.SoundFileBeendet)
    {
        // Anzeige initialisieren
        SekundenAnzeigeInit();

        // Sekundenzähler starten, wobei das Zählen eigenständig und parallel erfolgt
        SekundenZaehlen_Start();

        // Sound erzeugen und sofort starten durch Url-Zuweisung
        ID_BGSound.src=SoundObjekt.SoundFileUrl ;
        SoundObjekt.SoundFileBeendet=false;

        // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen

```



```

        SoundTimeoutID = setTimeout(
            "SoundAbspielen()",
            SoundObjekt.SoundFileDauerInMillisekundeSekunden
        );
    }
    else
    {
        // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

        // und Meldung
        var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;
        var TimerUngenauigkeit2 = TimerUngenauigkeit1 / SoundDauerInSekunden;
        alert(
            "Wiedergabe beendet\nUngenauigkeit des Timers = "
            + TimerUngenauigkeit1.toString()+ " Sekunden\n"
            + "also " + TimerUngenauigkeit2.toString()+ " Ticks pro Sekunde"
        );
    }
}

// ++++++ Sekunden-Anzeige initialisieren
function SekundenAnzeigeInit()
{
    // ---- Variablen init
    SekundenZahler      = 0;
    SekundenZahlerTimeoutID = null;

    // ---- visuelle Anzeige erzeugen
    // - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
    //      Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
    //      Der Ausdruck liefert den Wert , welcher sofort das Layout der
    //      DIV's beeinflusst.
    //      Jeder Ausdruck besitzt den SekundenZahler als Komponente.
    //      Damit ändert sich der Wert des Ausdrucks.
    //      Für die Neuberechnung des Ausdrucks ist der Aufruf von
    //      document.recal()
    //      nötig.
    //      Dieser Aufruf erfolgt in SekundenZaehlen(), also permanent pro Sekunde.
    //      Damit wird der Style-Wert permanent neu berechnet.
    //      Damit visualisieren sich die DIV's permanent neu.

    // Sekundenbalken in der Style-Eigenschaft width (Breite) mit Ausdruck belegen,
    //      also dynamisch anzeigen
    ID_DIV_Balken.style.setExpression( "width",
        "SekundenZahler * PixelBreiteProBalkenErweiterung"
    );

    // Sekundenzähler in der Eigenschaft .innerText mit Ausdruck belegen,
    //      also dynamisch anzeigen
    ID_DIV_SekundenZahler.setExpression("innerText","SekundenZahler.toString()");

    // - - - Messlatte statisch anzeigen
    ID_DIV_MessLatte.style.width = SoundDauerInSekunden * PixelBreiteProBalkenErweiterung;
    ID_DIV_MessLatte.innerText  = "Der Sound dauert "
        + SoundDauerInSekunden.toString()
        + " Sekunden";
}

// ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
if (ie)
{
    document.write('<BODY></BODY>');

    document.write(
        '<BGSOUND ID= "ID_BGSound" LOOP="0">'
    );

    document.write(
        '<DIV ID="ID_DIV_Balken"'
        + 'STYLE="background-color:lightblue"'
        + '>'
        + '</DIV>'
    );
}

```



```

        + '<BR>'
    );

    document.write(    '<DIV    ID="ID_DIV_SekundenZahler"'
        +              'STYLE="color:hotpink;font-weight:bold"'
        + '>'
        + '</DIV>'
        + '<BR>'
    );

    document.write(    '<DIV    ID="ID_DIV_MessLatte"'
        +              'STYLE="color:white;background-color:gray"'
        + '>'
        + '</DIV>'
    );

    // ++++++ Sound initialisieren und starten mit Laden des Dokumentes

    // ----- Sound-Objekt erzeugen anhand globaler Variablen
    SoundObjekt = new SoundObjektErzeugen(SoundUrl, SoundDauerInSekunden);

    // ----- Sound-Objekt wiedergeben
    SoundAbspielen(); // meldet wenn Wiedergabe beendet ist
}
</SCRIPT>
</HEAD>
<BODY>
    <!-- BODY-Teil muss leer bleiben -->
</BODY>
</HTML>

```

**.isContentEditable**      Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)  
 Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.  
 false   Content nicht editierbar  
 true    Content editierbar

Beispiel:

```

<HEAD>
<SCRIPT>
    function EditierbarkeitWechseln()
    {
        var Editierbarkeit = ID_Span1.isContentEditable;
        var Editierbarkeit_Negation = ! Editierbarkeit;
        ID_Span1.contentEditable = Editierbarkeit_Negation;
        ID_Span2.innerText = Editierbarkeit_Negation;

        if (Editierbarkeit_Negation == false)
        { ID_Button.innerText="editierbar machen"}
        else
        { ID_Button.innerText="nicht editierbar machen"}
    }
</SCRIPT>
</HEAD>
<BODY onload=" ID_Span2.innerText = ID_Span1.isContentEditable;">
    <BUTTON ID="ID_Button" onclick=" EditierbarkeitWechseln()">
        Klick mich
    </BUTTON>
    <SPAN ID="ID_Span1">Diesen Text editieren</SPAN>
    <SPAN ID="ID_Span2">Editierbarkeit</SPAN>
</BODY>

```

**.isDisabled**      Interaktionsfähigkeit  
 nur wenn sichtbar, so User-Interaktion möglich  
 false    User kann mit Objekt interagieren  
 true     User kann mit Objekt nicht interagieren

Beispiel:

```

<HEAD>
<SCRIPT>
    function SichtbarkeitWechseln()
    {
        var Sichtbarkeit = ID_Span1.isDisabled;
        var Sichtbarkeit_Negation = ! Sichtbarkeit;
        ID_Span1.isDisabled = Sichtbarkeit_Negation;
        ID_Span2.innerText = Sichtbarkeit_Negation;
    }

```



```

        if (Sichtbarkeit_Negation == false)
        { ID_Button.innerText="interaktiv machen"}
        else
        { ID_Button.innerText="nicht interaktiv machen"}
    }
</SCRIPT>
</HEAD>
<BODY onload=" ID_Span2.innerText = ID_Span1.isDisabled;">
    <BUTTON ID="ID_Button" onclick="SichtbarkeitWechseln()">
        Klick mich
    </BUTTON>
    <SPAN ID="ID_Span1">Diesen Text editieren</SPAN>
    <SPAN ID="ID_Span2">Sichtbarkeit</SPAN>
</BODY>

```

.isMap server-seitige Image-Map

.isMultiLine Mehrzeiligkeit des Objekthinhaltes  
 false Objekt hat genau 1 Zeile  
 true Objekt hat mindestens 1 Zeile

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Antworten(Wert)
    {Wert ? alert("Ja") : alert("Nein");}
</SCRIPT>
</HEAD>
<BODY>
    <P>
        <INPUT type="text" ID="ID_Input" VALUE="Test">
        <BUTTON onclick="Antworten(ID_Input.isMultiLine);">
            Mehrzeiligkeit eines INPUT TYPE=text
        </BUTTON>
    </P>
    <P>
        TEXTAREA:
        <TEXTAREA ID="ID_Textarea">
            Test
        </TEXTAREA>
        <BUTTON onclick="Antworten(ID_Textarea.isMultiLine);">
            Mehrzeiligkeit eines Textbereiches
        </BUTTON>
    </P>
</BODY>
</HTML>

```

.isMuted Elemente-Status der Audio-Stummschaltung in der Timeline  
 siehe Objekt currTimeState und Behavior .style.time2  
 true Audio ist stummgeschaltet  
 false Audio ist hörbar

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Stummschaltung()
    {
        if (ID_Video.currTimeState.isActive)
        { ID_Span.innerText=' stumm = ' + ID_Video.currTimeState.isMuted;}
        else
        { ID_Span.innerText=' stumm =';}
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:media ID="ID_Video"
        SRC="/test /media/test.wmv"
        BEGIN="indefinite;"
        FILL="remove"
    >

```



```

</ t:media>
<BR><BR>
<SPAN ID="ID_Span">stumm =</SPAN>
<BR><BR>
<BUTTON onclick="ID_Video.beginElement();Stummschaltung();">
    Start
</BUTTON>
<BUTTON onclick="ID_Video.endElement();Stummschaltung();">
    Stop
</BUTTON>
<BR><BR>
<B>Volume control:</B>&nbsp;
<INPUT TYPE="radio"
    onpropertychange=
        "ID_Video.mute='true'; Stummschaltung();"
>
    Mute
<INPUT TYPE="radio" CHECKED
    onpropertychange=
        "ID_Video.mute='false'; ID_Video.volume=100; Stummschaltung();"
>
    100% Volume
</BODY>
</HTML>

```

.isOn

```

Elemente-Status aktiv oder halten in der Timeline
Stauts aktiv: Objekt kann auf Events reagieren
Status halten: Attribut Fill muss auf "freeze" oder "hold" gesetzt sein
ermittelbar per eigenschaft .state
Objekt kann nicht auf Events reagieren
zu Attribut FILL:
    wenn "hold" oder "freeze" dann wartet Element auf
                                die Synchronisation mit der
                                Timeline des Elternobjektes
    wenn "hold"   Element kann keine Events verarbeiten
                                und ist inaktiv
    wenn "freeze" Element ist aktiv
siehe Objekt currTimeState und Behavior .style.time2

```

Beispiel:

```
<HTML xmlns:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function WindowOnLoad()
    {window.setInterval(UpdateZeit, 10);}

    function UpdateZeit()
    {
        if (ID_Animation.currTimeState.isOn == true )
        { ID_Span.innerHTML="aktiv";}
        else
        { ID_Span.innerHTML="gehalten";}
    }

    window.onload = WindowOnLoad;
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onrepeat="innerText=parseInt(ID_Animation.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <t:EXCL ID="ID_excl"
        BEGIN="indefinite"
        DUR="5"
        REPEATCOUNT="5"
    >
```



```

        <t:ANIMATEMOTION
            ID="ID_Animation"
            TARGETELEMENT="ID_Div"
            TO="200,0"
            BEGIN="0"
            DUR="2"
            AUTOREVERSE="true"
        >
    </t:ANIMATEMOTION>
</t:EXCL>
<DIV ID="ID_Div"
    CLASS="time"
    STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<SPAN ID="ID_Span">
    not
</SPAN>
<BUTTON onclick="ID_excl.beginElement();">Start</BUTTON>
<BUTTON onclick="ID_excl.endElement();">Stop</BUTTON>
</BODY>
</HTML>

```

.isOpen

prüfen ob ein per .createPopup() instanziiertes Popup-Fenster angezeigt wird  
 siehe Objekt window.popup  
 true, so angezeigt  
 false, so nicht angezeigt

Beispiel für diverse Meldungsfenster per **jeweiligem** PopUp-Fenster ( es kann immer nur genau 1 Popup-Fenster angezeigt werden):

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">

    var PopUpFensterFeld = new Array();

    // nachfolgende Felder beschreiben die PopUp-Fenster und müssen
    // identische Anzahl der Feldelemente haben
    // Feld-Index ist die Nummer des PopUp-Fensters

    var PopUpFenster_RahmenStyle_Feld = new Array
    (
        "solid black 4px",
        "solid blue 3px",
        "solid green 2px"
    );

    var PopUpFenster_HintergrundFarbe_Feld = new Array
    (
        "yellow",
        "gray",
        "orange"
    );

    var PopUpFenster_Inhalt_Feld = new Array
    (
        "<B>Inhalt PopUpFenster 0<B>",
        "Inhalt PopUpFenster 1",
        "<TT>Inhalt PopUpFenster 2<TT>"
    );

    var PopUpFenster_X_Koordinate_Feld = new Array
    (
        100,
        150,
        200
    );

    var PopUpFenster_Y_Koordinate_Feld = new Array
    (
        150,
        200,
        250
    );

```





```

var PopUpFenster_Breite_Koordinate_Feld = new Array
(
    80,
    140,
    200
);

var PopUpFenster_Hoehe_Koordinate_Feld = new Array
(
    100,
    140,
    180
);

var AnzahlPopUpFenster = PopUpFenster_Hoehe_Koordinate_Feld.length;

function PopupFensterInstanzieren(NummerDesPopUpFensters, ZeigerAufElternFenster)
// NummerDesPopUpFensters ab 0
{
    PopUpFensterFeld[NummerDesPopUpFensters] =
        ZeigerAufElternFenster.createPopup();
}

function PopupFensterFuellen(NummerDesPopUpFensters)
{
    // Body des Popup-Fensters gestalten
    var PopupFenster_Body =
        PopUpFensterFeld[NummerDesPopUpFensters].document.body;

    PopupFenster_Body.style.backgroundColor =
        PopUpFenster_HintergrundFarbe_Feld[NummerDesPopUpFensters];

    PopupFenster_Body.style.border =
        PopUpFenster_RahmenStyle_Feld[NummerDesPopUpFensters];

    PopupFenster_Body.innerHTML =
        PopUpFenster_Inhalt_Feld[NummerDesPopUpFensters];
}

function PopupFensterAnzeigen(NummerDesPopUpFensters,
    ObjektZuDemPopUpFensterRelativPositioniertIst
)
{
    // Popup-Fenster anzeigen und damit öffnen
    PopUpFensterFeld[NummerDesPopUpFensters].show(
        PopUpFenster_X_Koordinate_Feld[NummerDesPopUpFensters],
        PopUpFenster_Y_Koordinate_Feld[NummerDesPopUpFensters],
        PopUpFenster_Breite_Koordinate_Feld[NummerDesPopUpFensters],
        PopUpFenster_Hoehe_Koordinate_Feld[NummerDesPopUpFensters],
        ObjektZuDemPopUpFensterRelativPositioniertIst
    );
}

function PopupFensterSchliessen(NummerDesPopUpFensters)
{
    var Zeiger = PopUpFensterFeld[NummerDesPopUpFensters];

    if (Zeiger.isOpen)
    { Zeiger.hide(); }
}

function Init()
{
    // PopUpFenster instanzieren im aktuellen Fenster (window)
    for (var i = 0 ; i < AnzahlPopUpFenster; i++)
    {
        PopupFensterInstanzieren(i, window);
        PopupFensterFuellen(i);
    }
}
</SCRIPT>
</HEAD>

```



```

<BODY onload="Init();">
    Durch Klick ausserhalb des Popup-Fensters wird dieses auch automatisch geschlossen.
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 anzeigen"
        onclick=" PopupFensterAnzeigen(0, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 anzeigen"
        onclick=" PopupFensterAnzeigen(1, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 anzeigen"
        onclick=" PopupFensterAnzeigen(2, document.body);"
    >
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 schliessen"
        onclick=" PopupFensterSchliessen(0);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 schliessen"
        onclick=" PopupFensterSchliessen(1);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 schliessen"
        onclick=" PopupFensterSchliessen(2);"
    >
</BODY>
</HTML>

```

.isPaused

Elemente-Status pausieren in der Timeline  
 siehe Objekt currTimeState und Behavior .style.time2

true	Element pausiert wartet auf Aktivierung kann keine anderen Events verarbeiten
false	Default Element pausiert nicht und ist aktiv

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Pausieren()
    {
        if (ID_Animation.currTimeState.isPaused == true)
        {ID_Span.innerHTML="pausiert";}
        else
        { ID_Span.innerHTML="aktiv";}
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID=ID_Span"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onrepeat="innerText=parseInt(ID_Animation.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <t:EXCL ID="ID_excl"
        BEGIN="indefinite"
        DUR="5"
        REPEATCOUNT="5"
        onbegin="Pausieren();"
        onend=" ID_Span.innerHTML='pausiert';
    >
        <t:ANIMATEMOTION ID="ID_Animation"
            TARGETELEMENT="ID_Div"
            TO="200,0"

```



```

        BEGIN="0"
        DUR="2"
        AUTOREVERSE="true"
    >
</t:ANIMATEMOTION>
</t:EXCL>
<DIV ID="ID_Div"
    CLASS="time"
    STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<SPAN ID="ID_Span">not</SPAN>
<BUTTON onclick="ID_excl.beginElement();">
    Start
</BUTTON>
<BUTTON onclick="ID_excl.pauseElement();Pausieren();">
    Pause
</BUTTON>
<BUTTON onclick="ID_excl.resumeElement();Pausieren();">
    Resume
</BUTTON>
<BUTTON onclick="ID_excl.endElement();ID_Span.innerHTML='pausiert!'">
    Stop
</BUTTON>
</BODY>
</HTML>

```

.IsReady                      Bereitschaft des Laufwerkes für Zugriffe z.B. ob Medium im Laufwerk liegt

Beispiel:

```

var DateiSystem                = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung     = DateiSystem.GetDriveName("C:");
var Laufwerk                  = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName       = Laufwerk.VolumeName;
var Laufwerk_Bereitschaft     = Laufwerk.IsReady;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Bereitschaft);

```

.IsRootFolder                prüfen auf Root

Beispiel:

```

var OrdnerName                = "c:\\windows\\desktop\\";
var DateiSystem               = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                    = DateiSystem.GetFolder(OrdnerName);
var Zahler                    = 0;
if (!Ordner.IsRootFolder)
{
    do
    {
        Ordner = Ordner.ParentFolder;
        Zahler++;
    }
    while (!Ordner.IsRootFolder);
}

alert("Ordner liegt " + Zahler + " Ebenen unter der Root");

```

.isStreamed                  Media-Datei mit Datenfluss (Data Stream) auf der Timeline  
nicht alle Media-Typen unterstützen Datenstrom  
siehe Objekt currTimeState und Behavior .style.time2  
false      hat keinen Datenfluss  
true       hat Datenfluss

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO ID="ID_Video"
        SRC="test.avi"
        SYNCBEHAVIOR="locked"
    >
    </t:VIDEO>
    <SPAN ID="ID_Span"

```



```

        CLASS="time_line_klasse"
        DUR="0.1"
        REPEATCOUNT="indefinite"
        onrepeat="ID_Span.innerText= ID_Video.isStreamed;"
    >
</SPAN>
</BODY>
</HTML>

```

**.isTextEdit** Erzeugbarkeit eines Textbereiches  
z.B. bei

- BUTTON
- TEXTAREA
- INPUT BUTTON
- INPUT HIDDEN
- INPUT PASSWORD
- INPUT RESET
- INPUT SUBMIT
- INPUT TEXT

false Textbereich nicht erzeugbar  
true Textbereich erzeugbar

**.javaEnabled** Verfügbarkeit der Microsoft Virtual Machine (Microsoft VM) für Java  
Achtung: Aufgrund eines Rechtsstreites mit dem Unternehmen Sun entwickelt Microsoft ab IE 6.x die MS VM nicht mehr weiter, hält die VM aber noch per Download verfügbar. Für eine aktuellere Version muss der User selbständig beim Unternehmen Sun eine Lizenz für Java erwerben und die zugehörige Software installieren (inklusive Updates). Es reicht dabei die Run-Time-Version der Virtuellen Java Maschine von Sun. Ob allerdings Sun-Java-Standards in Microsoft-Produkten implementiert werden, ist nicht gesichert.

Behavior **.style.clientCaps**  
false Microsoft VM ist nicht verfügbar  
true Microsoft VM ist verfügbar

**.keyCode** Unicode der Taste, die das Event auslöst per onkeydown, onkeyup und onkeypress  
Objekt event

**.keySplines** Wert eines Pixel-Intervalles zur Werterhöhung bei 2D-Elemente-Animation(en) auf der Timeline  
per **.additive** oder **.accumulate**  
benötigt **.calcMode** auf "spline"  
**.keyTimes**  
**.values** oder **.path**  
Achtung: Im Objekt **animatecolor** (t:ANIMATECOLOR) ist **.values** auch als Farbliste definiert !  
siehe Objekt **currTimeState** und Behavior **.style.time2**

Beispiel 1:

KEYSPLINES="0 1 .5 1;5 1 0 1"

Quadrupel	Koordinaten			
	x1	y1	x2	y2
1	0	1	.5	1
2	5	1	0	1

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
.div_Klasse{position:absolute; top:195px; height:100px; width:150px; border:solid black 2px;}
</STYLE>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
CLASS="time_line_klasse"
DUR=".1"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=( ID_Animatemotion.currTimeState.simpleTime);"
>
0
</SPAN>
<BR>
<SPAN ID=" ID_Span2"
CLASS="time_line_klasse"
DUR=".1"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=( ID_Animatemotion.currTimeState.activeTime);"

```



```

>
0
</SPAN>
<DIV ID="ID_Div"
CLASSE="div_klasse"
>
animierter Div
</DIV>
<t:ANIMATEMOTION ID="ID_Animatemotion"
BEGIN="1; ID_Button.click+1"
DUR="6s"
AUTOREVERSE="true"
CALCMODE="spline"
KEYSPLINES="0 1 .5 1;.5 1 0 1"
KEYTIMES="0;.5;1"
VALUES="25,0;250,50;500,0"
TARGETELEMENT="ID_Div"
FILL="hold"
>
</t:ANIMATEMOTION>
<BUTTON ID="ID_Button">Restart</BUTTON>
</BODY>
</HTML>

```

.keyTimes                      Zeitwert eines Pixel-Intervalles zur Werterhöhung bei 2D-Elemente-Animation(en) auf der Timeline  
                                  per .additive oder .accumulate  
 benötigt .calcMode auf "spline"  
                  .keySplines  
                  .values oder .path  
 Achtung: Im Objekt animatecolor (t:ANIMATECOLOR) ist .values auch als Farbliste definiert !  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
.div_Klasse{position:absolute; top:195px; height:100px; width:150px; border:solid black 2px;}
</STYLE>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
CLASS="time_line_klasse"
DUR="1"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=( ID_Animatemotion.currTimeState.simpleTime);"
>
0
</SPAN>
<BR>
<SPAN ID="ID_Span2"
CLASS="time_line_klasse"
DUR="1"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=( ID_Animatemotion.currTimeState.activeTime);"
>
0
</SPAN>
<DIV ID="ID_Div"
CLASSE="div_klasse"
>
animierter Div
</DIV>
<t:ANIMATEMOTION ID="ID_Animatemotion"
BEGIN="1; ID_Button.click+1"
DUR="6s"
AUTOREVERSE="true"
CALCMODE="spline"
KEYSPLINES="0 1 .5 1;.5 1 0 1"
KEYTIMES="0;.5;1"
VALUES="25,0;250,50;500,0"
TARGETELEMENT="ID_Div"

```



```

        >
        </t:ANIMATEMOTION>
        <BUTTON ID="ID_Button">Restart</BUTTON>
    </BODY>
</HTML>

```

**.label** Label einer Option, also Objektes document.select.option des Objektes document.select

**.lang** Sprache für Anzeige von Sonderzeichen etc.

**.language** Sprache für Script festlegen  
 gross klein egal  
 "JScript" (Default)  
 "javascript"  
 "vbs" oder "vbscript"  
 "XML" ab IE 5.x

**.lastChild** Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

Beispiel:

```

<SCRIPT>
    var ZeigerAufLetztesKind = Liste.lastChild; // liefert Referenz auf Listenelement 3
</SCRIPT>
<BODY>
    <UL ID = "Liste">
        <LI> Listenelement 1
        <LI> Listenelement 2
        <LI> Listenelement 3
    </UL>
</BODY>

```

**.lastModified** Datum der letzten Dokumentveränderung

**.latestMediaTime** Wartezeit für Start der Wiedergabe eines Media-Elementes auf der Timeline  
 nicht für Media-Datei mit Datenfluss  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO ID="ID_Video"
            SRC="test.avi"
            SYNCBEHAVIOR="locked"
    >
    </t:VIDEO>
    <SPAN ID="ID_Span"
        CLASS="time_line_klasse"
        DUR="0.1"
        REPEATCOUNT="indefinite"
        onrepeat="ID_Span.innerText= ID_Video.latestMediaTime;"
    >
    </SPAN>
</BODY>
</HTML>

```

**.left** linke Pixelposition des Rechteckes um ein Objekt  
 auch textrectangle Objekt

Beispiel für Nicht-TextRectangle-Objekt:

```

<SCRIPT>
    function Anzeigen(ObjektZeiger)
    {
        var Rechteck = ObjektZeiger.getBoudingClientRect();

        alert(
            "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
            + "\n"
            + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
        );
    }

```



```
</SCRIPT>
<P onclick="Anzeigen(this)">
```

Beispiel für TextRectangleObjekt:

```
<SCRIPT>
function Anzeigen(ObjektZeiger)
{
    var TextRectangleCollection = document.body.ObjektZeiger.TextRange.getClientRects();

    // .getClientRects() liefert immer Collection der textrectangle Objekte !!!

    var Rechteck = TextRectangleCollection[0];

    alert(    "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
            + "\n"
            + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
            );
}
</SCRIPT>
<BODY>
<DIV onclick="Anzeigen(this)">
    Test
</DIV>
</BODY>
```

.leftMargin	Left Margin in Pixel des Dokumentes Margin: Abstand des Aussenrandes eines Objektes zur Umgebung Padding: Abstand des Objekthinhaltes zum Aussenrand
.length	Anzahl der Feldelemente also Feldlänge z.B. bei Collection
.length	Anzahl der Zeichen im Kommentar (comment-Objekt), ab IE 6.x
.length	Anzahl der Argumente im Script-Objekt arguments als Eigenschaft eines Funktionsobjektes (siehe Script-Objekt Function und Anweisung function) Wert ist identisch mit dem Wert der Eigenschaft .length des Script-Objektes Function

Beispiel für Einlesen der Werte der Argumente aus dem Funktionsaufruf:

```
var GlobalesFeld = new Array();

function FunktionsBezeichner()
{

    // Argumenteliste der Funktion lokal zur Funktion referenzieren
    var ArgumentenListeAlsFeld = FunktionsBezeichner.arguments;

    // Anzahl der Argumente
    var ArgumenteAnzahl = ArgumentenListeAlsFeld.length;

    if ( ArgumenteAnzahl > 0 )
    {
        // Arrgumentenliste auslesen
        for (var i = 0; i < ArgumenteAnzahl; i++)
        {GlobalesFeld[i] = ArgumentenListeAlsFeld[i];}
    }

    // hier die weiteren Anweisungen der Funktion
}
```

.length	Anzahl der Elemente in einem Feld der Objektklasse Script-Objekt Array
.length	Wert laut zeiger_auf_funktion.arguments.length siehe Script-Objekt Function und Script-Objekt arguments und Anweisung function
.length	Anzahl der Zeichen im String bzw. String-Literal Integer, ab 0 wenn 0 so Leerkette siehe Script-Objekt String

Beispiel:

```
var StringLiteral ="StringLiteral";
StringLiteral.length      liefert 13
"StringLiteral".length    liefert 13
```



**.length** Anzahl aller Frames bzw. IFrames im Fenster laut Collection document.frames  
siehe Objekt window

**.Line** Anzahl der Zeilen im Textstream  
sinnvoll nur verwendbar, wenn mindestens 1 newline-Zeichen in der Datei auftaucht

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.ReadAll();
alert(DateiOffen.Line);
DateiOffen.Close();
```

**.link** Farbe eines Links des Dokumentes  
document.body.link  
"#rrggbb"  
vordefinierte Farbname (browserspezifisch)

**.linkColor** Farbe von noch nicht benutzten Links im Dokument  
document.body.linkColor  
"#rrggbb" Standard ist "#0000FF"  
vordefinierter Farbname (browserspezifisch)

**.LN2** Logarithmus zur Basis 2  
siehe Script-Objekt Math

**.LN10** Logarithmus zur Basis 10  
siehe Script-Objekt Math

**.location** Zeiger auf das gleichnamige Objekt

Beispiel 1:

```
<HTML>
<HEAD>
<SCRIPT>
function LocationAendern()
{
    for(i=0;i<document.all.length;i++)
    {
        if (document.all[i].tagName=="IFRAME")
        {document.all[i].contentWindow.location = "http://www.test.com";}
    }
}
</SCRIPT>
</HEAD>
<BODY>
<BUTTON onclick="LocationAendern();">Location aendern</BUTTON>
<IFRAME SRC="http://www.test.de"></IFRAME>
</BODY>
</HTML>
```

Beispiel 2:

```
<HEAD>
<SCRIPT>
function Entfernen()
{
    // versuche den Text zu entfernen
    try
    {
        var KindZeigerAufTextImDiv = ID_Div.children(0);
        ID_Div.removeChild(KindZeigerAufTextImDiv);
        // Achtung: Der Text ist noch sichtbar !!!!
    }
    // oder fange das Ereignis des bereits entfernten Textes ein
    // und behandle das Ereignis
    catch(x)
    {
        alert(
            "Text wurde entfernt !\n"
            + "Das Dokument muss neu geladen werden !
        );
        document.location.reload();
    }
}
```





```

</SCRIPT>
</HEAD>
<BODY>
    <DIV ID="ID_Div" onclick=" Entfernen()">
        Klick, um diesen Text zu entfernen !
    </DIV>
</BODY>

```

Beispiel 3: Quelltext anzeigen: Öffnet lokalen Standard-Text-Editor z.B. Notepad

```
<INPUT TYPE="button" VALUE="Quelltext ansehen" onclick="window.location = 'view-source:' + window.location.href">
```

.LOG2E                      Logarithmus von E (eulersche Zahl) zur Basis 2  
siehe Script-Objekt Math

.LOG10E                     Logarithmus von E (eulersche Zahl) zur Basis 10  
siehe Script-Objekt Math

.logicalXDPI                Standard-Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch ) auf Bildschirm  
per screen Objekt

Beispiel 1:

```

<SCRIPT>
function ScaleFactorX()
{
    var ScaleFactor = screen.deviceXDPI / screen.logicalXDPI;
    return ScaleFactor;
}
</SCRIPT>

```

Beispiel 2:

```

<SCRIPT>
function fnScaleManually()
{
    // normale DPI-Auflösung (Dot per Inch)
    var constNorm = 96;

    // Zoomen
    if ( (screen.deviceXDPI == screen.logicalXDPI)
        && (screen.deviceXDPI > constNorm)
        )
    {
        document.body.style.zoom = constNorm / screen.logicalXDPI;
    }
}
</SCRIPT>

```

.logicalYDPI                Standard-Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch ) auf Bildschirm  
per screen Objekt

Beispiel 1:

```

<SCRIPT>
function ScaleFactorY()
{
    var ScaleFactor = screen.deviceXDPI / screen.logicalYDPI;
    return ScaleFactor;
}
</SCRIPT>

```

Beispiel 2:

```

<SCRIPT>
function fnScaleManually()
{
    // normale DPI-Auflösung (Dot per Inch)
    var constNorm = 96;

    // Zoomen
    if ( (screen.deviceYDPI == screen.logicalYDPI)
        && (screen.deviceYDPI > constNorm)
        )
    {
        document.body.style.zoom = constNorm / screen.logicalYDPI;
    }
}
</SCRIPT>

```

.longDesc                   Uniform Resource Identifier (URI) zur einer "long description" umwandeln

LONGTRANSITION           wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software



.loop	Anzahl der Wiederholungen
	nur Objekt BGSOUND bzw. IMG mit Sound bzw. INPUT-Element mit Sound
	-1           endlos
	0           genau 1 mal
	> 0        Anzahl der Wiederholungen
	Standard ist 1
	1 entspricht t 0

Beispiel 1:

<BGSOUND SRC="file:///c:/test/wav/test.wav">	genau 1 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP>	genau 1 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP=>	genau 1 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP=0>	genau 1 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP=1>	genau 1 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP=10>	genau 10 mal
<BGSOUND SRC="file:///c:/test/wav/test.wav" LOOP=-1>	endlos

Beispiel 2:

```
<HEAD>
<SCRIPT>
    function KanalVerteilung(Wert)
    { ID_bgsound.balance = Wert; }

    function GenauEinmal()
    {
        ID_bgsound.loop = 1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }

    function Endlos()
    {
        ID_bgsound.loop = -1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>
    <BUTTON onclick="Endlos()"></BUTTON>
    <BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung(10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung (0)"></BUTTON>
    <B>Volume control:</B>&nbsp;
    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-10;"
    >Mute

    < INPUT TYPE="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-7;"
    >25% Volume

    < INPUT TYPE="radio" NAME ="ID_RadioButtonGruppe" CHECKED
        onpropertychange="ID_bgsound.volume=-5;"
    >50% Volume

    < INPUT TYPE="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-2;"
    >75% Volume

    < INPUT TYPE="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=0;"
    >100% Volume
</BODY>
```

.loop	Anzahl der Scrollaktionen des marquee Objektes
0	endlos scrollen
	in Script nicht kodierbar sonst Fehlermeldung
-1	Default
	endlos scrollen
> 0	Anzahl der Scrollaktionen
	null nicht kodierbar sonst Fehlermeldung

Beispiel:

<BODY>



```

<MARQUEE BEHAVIOR="alternate" LOOP=2 onstart="alert('onstart-Ereignis erkannt')">
  Marquee Text
</MARQUEE>
</BODY>

```

**.lower** Abbruch bzw. Pausierung der aktiven Animation durch Start der Animation eines niedriger priorisierten Kindes  
 Time-Container ist Behavior-Objektes `.style.time2.excl`  
 Folge der in HTML kodierten Objekte `t:PRIORITYCLASS` entspricht der absteigenden Folge der Prioritäten der Animation.  
 Jedes Kind ist ein `t:PRIORITYCLASS`  
 darf **kein** weiteres Behavior-Objekt `.style.time2.priorityClass` besitzen  
 wenn mehrere pausierende Kinder existieren:  
 Das zuletzt pausierte Kind startet zuerst.  
 Das zuerst pausierte Kind startet zuletzt.  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
  <t:EXCL ID="ID_Excl"
    BEGIN="0;indefinite;"
  >
    <t:PRIORITYCLASS>
      <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        BEGIN="5"
        DUR="5"
      >
        Test1
      </SPAN>
    </t:PRIORITYCLASS>
    <t:PRIORITYCLASS LOWER="never">
      <SPAN ID="ID_Span2"
        CLASS="time_line_klasse"
        BEGIN="0"
        DUR="10"
      >
        Test2
      </SPAN>
    </t:PRIORITYCLASS>
  </t:EXCL>
  <BR>
  <BUTTON ID="ID_Button" onclick="ID_Excl.beginElement();">
  </BUTTON>
</BODY>
</HTML>

```

**.lowsrc** Url des Images in geringerer Auflösung

**.marginHeight** Abstand in Pixel vom unteren zum oberen Rand des Objektes  
 Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
 Padding: Abstand des Objekthinhaltes zum Aussenrand

**.marginWidth** Abstand in Pixel vom linken zum rechten Rand des Objektes  
 Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
 Padding: Abstand des Objekthinhaltes zum Aussenrand

**.maxLength** Maximale Anzahl der durch User eingebaren Zeichen in einem Text-Control

**MAX\_VALUE** maximaler numerischer endlicher Wert  $> 0$  oder  $< 0$ , den Javascript zulässt:  $1.79E+308$   
 Werte über **MAX\_VALUE** sind unendlich (`Number.POSITIVE_INFINITY` bzw. `Number.NEGATIVE_INFINITY`)

kann als Wert zugewiesen werden  
 siehe Script-Objekt `Number`  
 nur lesen



**.media** Media-Typ für Ausgabe eines Objektes  
 "screen" Ausgabe auf Bildschirm  
 "print" Ausgabe auf Printmedium oder Druckvorschau auf Bildschirm  
 "all" Default  
 Ausgabe auf alle Geräte

**.mediaDur** Länge der Media-Datei in Sekunden (Länge bei der Wiedergabe) auf der Timeline  
 Ereignis onmediacomplete muss ausgelöst also die Media-Datei komplett geladen worden sein  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:VIDEO ID="ID_Video"
SRC="test.avi"
SYNCBEHAVIOR="locked"
>
</t:VIDEO>
<SPAN ID="ID_Span"
CLASS="time_line_klasse"
DUR="0.1"
REPEATCOUNT="indefinite"
onrepeat="ID_Span.innerText= ID_Video.mediaDur;"
>
</SPAN>
</BODY>
</HTML>
```

**.mediaHeight** aktuelle Höhe des Media-Elementes in Pixels auf der Timeline  
 Standardgemäß wird die Elementhöhe aus der Höhenangabe in der Media-Datei verwendet  
 (ansonsten STYLE-Angabe zu .style.height im Element kodieren)  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:VIDEO ID="ID_Video"
SRC="test.avi"
SYNCBEHAVIOR="locked"
>
</t:VIDEO>
<SPAN ID="ID_Span"
CLASS="time_line_klasse"
DUR="0.1"
REPEATCOUNT="indefinite"
onrepeat="ID_Span.innerText= ID_Video.mediaHeight;"
>
</SPAN>
</BODY>
</HTML>
```

**.mediaWidth** aktuelle Breite des Media-Elementes in Pixels auf der Timeline  
 Standardgemäß wird die Elementhöhe aus der Breitenangabe in der Media-Datei verwendet  
 (ansonsten STYLE-Angabe zu .style.width im Element kodieren)  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
```



```

<t:VIDEO      ID="ID_Video"
              SRC="test.avi"
              SYNCBEHAVIOR="locked"
>
</t:VIDEO>
<SPAN ID="ID_Span"
      CLASS="time_line_klasse"
      DUR="0.1"
      REPEATCOUNT="indefinite"
      onrepeat="ID_Span.innerText= ID_Video.mediaWidth;"
>
</SPAN>
</BODY>
</HTML>

```

**.menuArguments** Zeiger auf dasjenige offene Fenster, indem ein Eintrag aus dem Kontextmenü ausgeführt wurde  
siehe Objekt `window.external`

Beispiel:

```

<SCRIPT LANGUAGE = "JavaScript">
    var ZeigeraufWindow                                = window.external.menuArguments;

    // beispielsweise einen selektierten Text im Fenster verarbeiten
    var ZeigerAufDokuemntImWindow                      = ZeigeraufWindow.document;
    var SelektionImDokument                           = ZeigerAufDokuemntImWindow.selection;
    var TextBereichImDokument                         = SelektionImDokument.createRange();
    var SelektierterTextImTextBereich                 = TextBereichImDokument.text;

    if (SelektierterTextImTextBereich.length == 0)
    { TextBereichImDokument.text = "Einfuege Text"; }
    else
    { TextBereichImDokument.text = SelektierterTextImTextBereich.toUpperCase(); }
</SCRIPT>

```

**.message** Beschreibung des Run-Time-Errors  
identisch mit Eigenschaft `.description`, die aber deprecated ist  
siehe error JScript-Objekt

Beispiel:

```

function AlterErmitteln(Wert)
{
    if(Wert < 0)
    { throw new Error("Fehler: Altersangabe muss >= 0 sein"); }
}

try
{ AlterErmitteln (-5); } // aktiviert throw
catch(e)
{ alert (e.message); }

```

**.method** Methode des Sendens/Empfangens der Daten an/von den Server bei Formular

"get"	Daten vom Server holen und an die Url laut Eigenschaft <code>.action</code> anhängen und Url öffnen, wenn Url ein Anker ist
	Url und angehängte Daten max. 2048 Bytes
"post"	Daten an Server senden per HTTP-Post-Transaktion physischer Umfang der Daten ist unbegrenzt

Beispiel für Festplattenverzeichnis unter Windows auslesen:

```

<BODY>
<FORM ACTION="file:///C:/"
      METHOD="GET"
>
    <INPUT TYPE="submit" VALUE="Root von LW C anzeigen!">
</FORM>
</BODY>

```

**.Methods** Liste der vom Objekt unterstützten HTTP-Methoden

**.mimeType** MIME-Typ (Multipurpose Internet Mail Extension-Typ) des Media-Elementes auf der Timeline  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>

```



```
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
  <t:VIDEO          ID="ID_Video"
                   SRC="test.avi"
                   SYNCBEHAVIOR="locked"
  >
  </t:VIDEO>
  <SPAN   ID="ID_Span"
         CLASS="time_line_klasse"
         DUR="0.1"
         REPEATCOUNT="indefinite"
         onrepeat="ID_Span.innerText= ID_Video.mimeType;"
  >
  </SPAN>
</BODY>
</HTML>
```

<code>.mimeTypes</code>	Zeiger auf Collection <code>mimeTypes</code> siehe Objekt <code>navigator</code>
-------------------------	---

MIN_VALUE	minimaler numerischer endlicher Wert > 0, den Javascript zulässt: 5E-324 Werte unter MIN_VALUE sind immer 0 kann als Wert zugewiesen werden siehe Script-Objekt Number nur lesen
-----------	--

.mode	Ergebnis der Animation des Übergangsfilters per .style.time2.transitionFilter Behavior-Objekt, also Sichtbarkeit bzw. Unsichtbarkeit <b>nach Vollendung</b> des Filters siehe Objekt currTimeState und Behavior .style.time2
"in"	Default Übergang hinein mit fortschreitender Animation wird Element sichtbar, das nach der Animation <b>voll sichtbar</b> sein soll
"out"	Übergang hinaus mit fortschreitender Animation wird Element unsichtbarer, das nach der Animation <b>nicht mehr</b> sichtbar sein soll

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

    <t:TRANSITIONFILTER          ID="ID_Transfilter1"
                                TYPE="barWipe"
                                DUR="3"
                                TARGETELEMENT="ID_Div1"
                                MODE="in"

    >
    </t:TRANSITIONFILTER>

    <t:TRANSITIONFILTER          ID="ID_Transfilter2"
                                TYPE="barWipe"
                                DUR="3"
                                TARGETELEMENT="ID_Div2"
                                MODE="out"

    >
    </t:TRANSITIONFILTER>

    <DIV          ID="ID_Div1"
                CLASS="time_line_klasse"
                BEGIN="0"
                DUR="indefinite"
                STYLE= "position:relative; left:20px; width:420px; height:100px;
                        background-image:url(test.gif); background-repeat: no-repeat;
                        "

    >
    </DIV>

    <DIV          ID="ID_Div2"
```



```

        CLASS="time_line_klasse"
        BEGIN="0"
        DUR="indefinite"
        STYLE= "position:relative; left:20px; width:420px; height:100px;
                background-image:url(test.gif); background-repeat: no-repeat;
                "
    >
</DIV>
</BODY>
</HTML>

```

MODULATE wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software

MOTIFNAME wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software

.multiple	Multiple Auswahl bei document.select
false	Default keine multiple Auswahl
true	multiple Auswahl möglich

Beispiel:

```
<SELECT ID="ID_select" MULTIPLE>
    <OPTION>Auswahl 1</OPTION>
    <OPTION> Auswahl 2</OPTION>
    <OPTION> Auswahl 3</OPTION>
</SELECT>
<BUTTON onclick="ID_select.multiple=false">keine multiple Auswahl</BUTTON>
<BUTTON onclick="ID_select.multiple=true">multiple Auswahl</BUTTON>
```

<code>.mute</code>	Audio aktiv oder aus (stumm) auf der Timeline wenn Eigenschaft für body Objekt kodiert, so alle Audio-Elemente im Body davon betroffen beeinflusst nicht die Systemlautstärke-Regelung von Windows oder anderen Soundmixern siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
<code>false</code>	Default Sound ist an
<code>true</code>	Sound ist aus

Beispiel für Abspielen einer Sounddatei per Timeline als Alternative zum Objekt bgsound:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

    <t:MEDIA                ID="ID_Media"
                           BEGIN="indefinite;"
                           SRC="test.wmv"
                           FILL="remove"
                           onmediacomplete="Timer2.beginElement();"
    >
    </t:MEDIA>

    <BR>

    <BUTTON onclick="ID_Media.beginElement();">Start</BUTTON>
    <BUTTON onclick="ID_Media.endElement();">Stop</BUTTON>

    <BR>

    <B>Volume control:</B>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~

    <INPUT  TYPE ="radio" NAME="ID_RadioButtonGruppe"
           onpropertychange="ID_Media.mute='true';"
    >Mute

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
           onpropertychange="ID_Media.mute='false'; ID_Media.volume=25;"
    >25% Volume

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe" CHECKED
           onpropertychange="ID Media.mute='false'; ID Media.volume=50;"
    >50% Volume

    </BODY>
</HTML>
```



```

>50% Volume

< INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_Media.mute='false'; ID_Media.volume=75;"
>75% Volume

< INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_Media.mute='false'; ID_Media.volume=100;"
>100% Volume
</BODY>
</HTML>

```

.name                    Name des Objektes (nicht ID !!!)  
                          muss beim Formular für alle zu sendenden Felder kodiert sein !!  
                          Element darf nicht per Methode .createElement() erzeugt worden sein

.name                    Informationen diverser Arten per meta Objekt  
                          der Wert der Information wird in der Eigenschaft .content abgelegt

Beispiele:

```

Quelltextdokumentation
<META NAME="ProgID" CONTENT="word.document">
<META NAME="Template" CONTENT="C:\test\html.dot">

```

```

Anzahl der Tage nach deren Ablauf Suchmaschine erneut scannen darf
<META NAME="Revisit-After " CONTENT="20 days">           days ist festkodiert !!!

```

```

Autor
<META NAME="Author " CONTENT="Ich">

```

```

Datum des Dokumentes
<META NAME="Date" CONTENT="Mittwoch, 26. April 2000">

```

```

Dokumentbeschreibung
<META NAME="Description" CONTENT="freier text der von suchmaschinen indiziert wird">

```

```

Schlagwortliste
<META NAME="Keywords" LANG="xx" CONTENT="ich, du, er, sie, es, wir, ihr sie">

```

Hinweis: für xx bitte einsetzen	en	für Englisch
	de	für Deutsch
	fr	für Französisch

.name                    Name des MediaItem Objektes  
                          siehe Behavior .style.mediaBar

.name                    Ausnahmetyp des Run-Time-Error liefern  
                          siehe error JScript-Objekt  
                          privater Run-Time-Error  
                          "Error"  
                          Standard-Run-Time-Error

"ConversionError"	Konvertierungsfehler
"RangeError"	Bereichfehler z.B. Feldlänge negativ
"ReferenceError"	Referenz-Fehler z.B. Feld mit negativer Anzahl von Feldelementen soll erzeugt werden
"RegExpError"	Fehler bei regular Expression (regExp-Objekt)
"SyntaxError"	
"TypeError"	Typfehler z.B. bei Wertzuweisung
"URIError"	falscher Uniform Resource Indicator (URI) z.B. bei encode()

Beispiel:

```

function AlterErmitteln(Wert)
{
    if(Wert < 0)
    {throw new Error("Fehler: Altersangabe muss >= 0 sein");}
}

try
{AlterErmitteln (-5);} // aktiviert throw
catch(e)
{alert (e.message + " " + e.name);}

```

.name                    physischer Fenstertitel laut open()  
                          entspricht Wert des Attributes TARGET eines Links  
                          Text des Fenstertitels (Text in Titelzeile des Fensters) laut document.title  
                          siehe Objekt window





Beispiel:

```
window.name="MyWindow";
window.open("file.htm","Frame1");
```

.Name Ordnerbezeichner in voller Länge (nicht 8.3 Bezeichner)

Beispiel:

```
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner               = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Name);
```

.Name Dateibezeichner in voller Länge (nicht 8.3 Bezeichner)

Beispiel:

```
var DateinameMitPfad     = "c:\\test.txt";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Datei               = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.Name);
```

.nameProp Dateiname-Teil einer Url ohne Path und ohne Protokoll liefern

Beispiel:

```
<SCRIPT>
function Init()
{ ID_A.innerText= ID_A.nameProp; } // test.img

window.onload=Init; // ohne () da sonst sofort ausgeführt
</SCRIPT>
<A ID="ID_A" HREF="http://www.test.de/test.img "></A>
```

NaN nicht numerischer Wert (Not a Number)

kann als Wert zugewiesen werden

Bsp: var Wert = Number.NaN;

Hinweis: Methoden parseFloat() und parseInt() liefern NaN, wenn Parameter nicht numerisch ist.

Vergleich mit NaN ist nicht zulässig: Dafür ist die Methode .isNaN() zu verwenden.

siehe Script-Objekt Number

nur lesen

.navigator Zeiger auf das Objekt navigator

siehe Objekt window

NEGATIVE\_INFINITY entspricht Negativ-Unendlich

kann als Wert zugewiesen werden

Hinweis für numerische Operationen

Multiplikation	Wert mal unendlich ergibt unendlich unendlich mal unendlich ergibt unendlich 0 * unendlich ergibt NaN NaN * unendlich ergibt NaN
Division	Wert durch unendlich ergibt 0 unendlich durch Wert ergibt unendlich unendlich durch unendlich ergibt NaN Division durch 0 nicht erlaubt
Vorzeichen	wird wie üblich ermittelt

siehe Script-Objekt Number

nur lesen

.nextItem Zeiger auf nächsten Eintrag in der Playliste (Objekt PlaylistInfo)

siehe Behavior .style.mediaBar

Beispiel:

```
<DIV ID="ID_Div"
STYLE="behavior:url(#default#mediaBar)"
>
</DIV>
<INPUT TYPE=button
VALUE='abspielen von test.asx'
onclick= "ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
ID_Div.disabledUI = true;
ID_Div.enabled = false;
"
>
```

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

Beispiel:

```
<SCRIPT>
```



```

var ErstesKind_Index = 0;      // Index ab 0
var ZeigerAufNachfolgerKind = Liste.childNodes(ErstesKind_Index).nextSibling;
                                // liefert Referenz auf Listenelement 2

</SCRIPT>
<BODY>
    <UL ID = "Liste">
        <LI> Listenelement 1
        <LI> Listenelement 2
        <LI> Listenelement 3
    </UL>
</BODY>

```

**.nodeName** String als Name des Kindes (Knoten, Node, Element)  
also TAG-Bezeichner, Attribut-Name; #text für Anker

Beispiel:

```

<SCRIPT>
var ErstesKind_Index = 0;      // Index ab 0
var ErstesKind_Name = Liste.childNodes(ErstesKind_Index).nodeName;
alert(ErstesKind_Name);       // liefert den Tagnamen 'LI' von Listenelement 1
                                // Hinweis: Listenelement 1 ist der Wert des Kindes
</SCRIPT>
<BODY>
    <UL ID = "Liste">
        <LI> Listenelement 1
        <LI> Listenelement 2
        <LI> Listenelement 3
    </UL>
</BODY>

```

**.nodeType** Knotentyp laut attributes Collection

1	für Element-Knoten
2	für Attribut-Knoten
3	für Textknoten
4	für CDATA-Abschnitt
5	für Entity-Referenz
6	für Entity-Knoten
7	für Processing Instruction
8	für Kommentar-Knoten
9	für das Dokument
10	für den Dokumenttyp
11	für ein Dokument-Fragment
12	für Notation

Beispiel 1:

```
var KnotenTypVonBODY = document.body.nodeType;
```

Beispiel 2:

```

var ZeigerAuf_B_Tag = document.createElement("B");      // B-Tag erzeugen
document.body.insertBefore(ZeigerAuf_B_Tag);
                                // B-Tag in den BODY-Teil des Dokument einfügen,
                                // also in die Baumstruktur
var KnotenTypDes_B_Tag = ZeigerAuf_B_Tag.nodeType;

```

**.nodeValue** Knotenwert (Wert des Kindes, Node, Elementes)  
nur für Text- und Attribut-Elemente  
nicht für Element-Knoten (Knotentyp 1)

Beispiel:

```

<SCRIPT>
function KnotenWertAendern( ZeigerAufListe,
                             IndexVonListenElement,      // immer ab 0
                             Zeichenkette                  // Listenlement muss Text sein
                           )
{
    var ReturnWert=false;      // Annahme: Änderung schlägt fehl

    // prüfen auf UL-Tag
    if (ZeigerAufListe.nodeName == "UL")
    {
        // Anzahl der Listenelemente holen
        var AnzahlListenelemente= ZeigerAufListe.childNodes.length;
                                                // immer ab 1

        // und Anzahl und Index prüfen
        if ( (AnzahlListenelemente > 0)      // immer ab 1
            && (IndexVonListenElement >= 0)   // immer ab 0
            && (IndexVonListenElement < AnzahlListenelemente)
        )
        {
            // ...
        }
    }
}

```



```

    )
    {
        // Zeiger auf das Listenelement laut Index holen
        var ZeigerAufListenElement =
            ZeigerAufListe.childNodes(IndexVonListenElement);

        // existiert das Listenelement ?
        if (ZeigerAufListenElement)
        {
            // ZeigerAufListenElement ist nicht null

            // Listenelement ist Textelement ?
            if (ZeigerAufListenElement.nodeType == 3)
            {
                ZeigerAufListenElement.nodeValue =
                    Zeichenkette;

                ReturnWert = true;
            }
        }
    }

    return ReturnWert;
}
</SCRIPT>
<UL ID="Liste" onclick=" KnotenWertAendern(this, 0, 'Listenelement Neu')">
    <LI>Listenelement alt
</UL>

```

**.noHref**                      HREF-Wirkung (Eigenschaft .href) ein/ aus bei Click auf den Link  
                               false        Default  
     Link ist wirksam bei click  
                               true        Link ist unwirksam bei click

**.noResize**                      Frame-Größenänderung ein/aus  
                                      Größenänderung z.B. durch Maus etc.  
                                      per frame Objekt  
                               false Default                      Größe änderbar  
                               true                      Größe nicht änderbar

**.noWrap**                      Wortumbruch einstellen  
                               false        Default.  
     Browser bricht den Text automatisch um  
                               true        Browser bricht den Text nicht um

**.number**                      Nummer des Run-Time-Error  
                                      siehe error JScript-Objekt

Beispiel:

```

function AlterErmitteln(Wert)
{
    if(Wert < 0)
    {throw new Error(8888,"Fehler: Altersangabe muss >= 0 sein");}
}

try
{AlterErmitteln (-5);} // aktiviert throw
catch(e)
{alert (e.message + " " + e.number);}

```

**.object**                      Zeiger auf das Objekt laut Applet bzw. laut Objekt object

**.offscreenBuffering**              alle Objekt in aktueller Seite im Offscreen zeichnen bevor sie sichtbar werden  
                                      siehe Objekt window

**.offsetHeight**                      Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem  
    des Elternobjektes (.offsetParent)

Beispiel 1:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
    var Faktor = 4;

    function startClock()

```



```

    {
        window.setInterval("Clock_Tick()", 1000);
        runClock();
    }

function runClock()
{
    var DatumHeute = Date();
    var ZeitHeute = DatumHeute.substring(11,19);
    var ElternHoehe = document.body.offsetHeight;
    var ElternBreite = document.body.offsetWidth;

    if ( (ElternHoehe * Faktor) > ElternBreite )
    { ElternHoehe = ElternBreite / Faktor;}

    document.all. ID_P.innerText = ZeitHeute;
    document.all. ID_P.style.fontSize = ElternHoehe;
}
</SCRIPT>
</HEAD>
<BODY onload="startClock()">
    <P ID="ID_P">&nbsp;</P>
</BODY>
</HTML>

```

Beispiel 2:

```

<DIV ID="ID_Div" STYLE="overflow:scroll; width:200; height:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientHeight)">client height</BUTTON>
<BUTTON onclick="alert(ID_Div.offsetHeight)">offset height</BUTTON>

```

**.offsetLeft** X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

Beispiel:

```

<SCRIPT>
function Anzeige(ObjektZeiger oObject)
{
    var ElternZeiger = ObjektZeiger.offsetParent;
    var ElternBreite = ElternZeiger.clientWidth;

    var KindOffsetLeft = ID_Div.offsetLeft;

    if ( KindOffsetLeft > ElternBreite )
    {alert("nach rechts scrollen");}
}
</SCRIPT>
<BUTTON onclick="Anzeige(this)">Klick mich</BUTTON>
<DIV ID="ID_Div" STYLE="position:absolute; top:200; left:1200;"> </DIV>

```

**.offsetParent** Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
    function Anzeige()
    {
        var ZeigerAufKind = document.all.ID_TD;

        alert( "TD liegt an Position (Left, Top) ("
            + ZeigerAufKind.offsetLeft
            + ", "
            + ZeigerAufKind.offsetTop
            + ")\n"
            + "Eltern als Container von TD ist "
            + ZeigerAufKind.offsetParent.tagName
            );
    }
</SCRIPT>
</HEAD>
<BODY onload="Anzeige()">
    <TABLE BORDER=1 ALIGN=right>
        <TR>
            <TD ID="ID_TD" >TD als Kind</TD>

```



```

        </TR>
    </TABLE>
</BODY>
</HTML>

```

**.offsetTop** Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Anzeige(ObjektZeiger)
    {
        var ElternZeiger = ObjektZeiger.offsetParent;
        var ElternHoehe = ElternZeiger.clientHeight;

        var KindOffsetTop = ObjektZeiger.offsetTop;

        if (KindOffsetTop > ElternHoehe)
        {alert("Testtext nicht sichtbar. Fenster erweitern für Sichtbarkeit");}
        else
        {alert("Testtext ist sichtbar");}
    }
</SCRIPT>
</HEAD>
<BODY onload="window.resizeTo(430,250)" onclick="Anzeige(ID_Div)" SCROLL=NO>
<DIV STYLE="position:absolute;left:20px">
    Irgendwo in das Fenster klicken
</DIV>
<DIV ID="ID_Div" STYLE=
    "position:absolute;"
    + "left:50px;top:300px;width:280px;color:lightGreen;"
    + "font-size:large;font-weight:bold;"
    + "background-color:hotPink;font-family:Arial"
>
    Testtext
</DIV>
</BODY>
</HTML>

```

**.offsetWidth** X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

Beispiel 1:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
    var Faktor = 4;

    function startClock()
    {
        window.setInterval("Clock_Tick()", 1000);
        runClock();
    }

    function runClock()
    {
        var DatumHeute = Date();
        var ZeitHeute = DatumHeute.substring(11,19);
        var ElternHoehe = document.body.offsetHeight;
        var ElternBreite = document.body.offsetWidth;

        if ( (ElternHoehe * Faktor) > ElternBreite )
        { ElternHoehe = ElternBreite / Faktor;}

        document.all. ID_P.innerText = ZeitHeute;
        document.all. ID_P.style.fontSize = ElternHoehe;
    }
</SCRIPT>
</HEAD>
<BODY onload="startClock()">
    <P ID="ID_P">&nbsp;&nbsp;&nbsp;</P>
</BODY>
</HTML>

```



Beispiel 2:

```
<DIV ID="ID_Div" STYLE="overflow:scroll; width:200; height:100">Test</DIV>
<BUTTON onclick="alert(ID_Div.clientWidth)">client width</BUTTON>
<BUTTON onclick="alert(ID_Div.offsetWidth)">offset width</BUTTON>
```

.offsetX

X-Koordinate Maus relativ zum Objekt, das das Event auslöst  
Objekt event

Beispiel:

```
<HEAD>
<SCRIPT>
    function AlertAnzeige()
    {
        var Kette = "X= " + window.event.offsetX + "\r";
        Kette += "Y= " + window.event.offsetY + "\r";
        alert(Kette);
    }

    function StatusZeileAnzeigen()
    {
        var Kette = "X= " + window.event.offsetX + " ";
        Kette += "Y= " + window.event.offsetY;
        window.status = Kette;
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV STYLE="position:absolute;top:200px;left:300px;"
        onmousemove="StatusZeileAnzeigen()" ondblclick="AlertAnzeige()">
    </DIV>
</BODY>
```

.offsetY

Y-Koordinate Maus relativ zum Objekt, das das Event auslöst  
Objekt event

Beispiel:

```
<HEAD>
<SCRIPT>
    function AlertAnzeige()
    {
        var Kette = "X= " + window.event.offsetX + "\r";
        Kette += "Y= " + window.event.offsetY + "\r";
        alert(Kette);
    }

    function StatusZeileAnzeigen()
    {
        var Kette = "X= " + window.event.offsetX + " ";
        Kette += "Y= " + window.event.offsetY;
        window.status = Kette;
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV STYLE="position:absolute;top:200px;left:300px;"
        onmousemove="StatusZeileAnzeigen()" ondblclick="AlertAnzeige()">
    </DIV>
</BODY>
```

.onLine

Onlinestatus des Browsers ermitteln per navigator Objekt  
keine Überprüfung auf Netzwerkverbindung  
navigator.onLine  
true     ist offline  
false    ist online

.onLine

System-Online-Status (nicht Netzwerk-Online-Status !!!)  
siehe Objekt window.clientInformation  
true     ist offline  
false    ist online  
Hinweis: Eigenschaft .connectionType des Behavior clientCaps zeigt den Status der  
genutzten Verbindung an

.onOffBehavior

deprecated ab IE 5.x  
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound



**.opener** Zeiger auf Elternfenster, das **open()** enthält und das Kind-Fenster öffnet, welches in dieser **.opener**-Eigenschaft den Zeiger auf das Elternfenster enthält  
 Bezug im geöffneten Fenster auf Instanzen des Aufrufers ist möglich  
 Bezug des Aufrufers auf geöffnetes Fenster ist möglich  
 Hinweis: Bei FRAME und IFRAME anstelle opener den Zeiger parent verwenden  
 Öffnet ein Frame / IFrame ein Fenster, das damit nicht im Frameset läuft, so ist der Bezug vom Fenster aus auf das Frameset oder auf einen Frame im Frameset per **window.opener.parent** möglich, solange opener existiert.  
 siehe Objekt window

Beispiel:

```
function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
    FensterZeiger.close();
}

....

// Fenster öffnen
var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");
var FensterDokumentZeiger = FensterZeiger.document;

var Kette= '<HTML>'
+ '<HEAD></HEAD>'
+ '<BODY>'
+ '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
+ '<BR>'
+ '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
+ ' onclick="opener.OnClickHandler();"'
+ '>'
+ '</BODY>'
+ '</HTML>';

FensterDokumentZeiger.open("text/html");
FensterDokumentZeiger.write(Kette);
FensterDokumentZeiger.close();
```

**.openState** Status des Media Bar Player bezüglich Playlist, Codec, Lizenz und Individualisierung  
 Media-Datei kann haben  
 Codec (Ländernummer)  
 Lizenz bei käuflichem Erwerb durch den User  
 Individualisierung auf den User  
 Media Bar Player ist der Windows Media Playersiehe Behavior .style.mediaBar

Beispiel:

```
<DIV ID="ID_Div"
STYLE="behavior:url(#default#mediaBar)"
onopenstatechange="alert(ID_Div.openState);"
>
</DIV>
<INPUT TYPE=button
VALUE='abspielen von test.asx'
onclick=" ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
ID_Div.disabledUI = true;
"
>
```

**.options** Zeiger auf die Collection document.select.options des document.select Objektes  
 Feld der Zeiger aller document.select.option Objekte

**.origin** Bezugspunkt der Animation eines Elementes auf der Timeline  
 nur für Objekt animatemotion (t:ANIMATEMOTION)  
 und .additive muss "replace" sein  
 siehe Objekt currTimeState und Behavior .style.time2

**.outerHTML** Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag  
 wirksam mit parsen des Ende-Tag  
 nur nach kompletten Einlesen des Dokumentes nutzbar  
 schreiben: wenn Bereich gefüllt so komplett überschreiben  
 Plain-Text  
 HTML-Elemente möglich  
 nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.

Beispiel:

```
<SCRIPT>
```



```

function ListeAendern()
{
    var ListenElement = event.srcElement;

    if ( (ListenElement.tagName != "UL")
        && (ListenElement.tagName != "LI")
        )
    {
        alert("ListenElement.outerHTML + " zur Liste hinzufügen");
        ID_Div.innerHTML += ListenElement.outerHTML + "<BR>";
    }
}
</SCRIPT>
<UL onclick = " ListeAendern()">
    <LI><B>Listenelement 1</B>
    <LI><I> Listenelement 2</I>
    <LI><U> Listenelement 3</U>
    <LI><STRIKE> Listenelement 4</STRIKE>
</UL>
<DIV ID="ID_Div"></DIV>

```

**.outerText** Referenz auf den gesamten Plain-Text im Objekt  
 nur nach kompletten einlesen des Dokumentes nutzbar  
 nur Plain-Text  
 schreiben: immer komplett überschreiben  
 nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR

Beispiel:

```

<DIV ID="ID_Div">
    <P ID="ID_P">Dieser Text wird veraendert</P>
</DIV>
<BUTTON onclick="ID_P.outerText='ändern'">
    Aendern
</BUTTON>
<BUTTON onclick="ID_Div.innerHTML='<P ID=ID_P>Dieser Text wird veraendert</P>'">
    Reset
</BUTTON>

```

**.ownerDocument** Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde

Beispiel:

```

<SCRIPT>
    var ElternDokument = SpanTag.ownerDocument;
</SCRIPT>
<BODY>
    <SPAN ID="SpanTag">Hallo !</SPAN>
</BODY>

```

**.owningElement** Referenz auf den im StyleSheet implementierten Style als Kindobjekt  
 Hinweis: Es sind die Eigenschaften und Methoden des style Objektes referenzierbar  
 siehe Objekt styleSheet

Beispiel:

```

for ( var i = 0; i < document.styleSheets.length; i++ )
{
    if ( document.styleSheets(i).owningElement.tagName == "STYLE" )
    {
        for ( var j = 0; j < document.styleSheets(i).imports.length; j++ )
        {
            alert( "Importierter StyleSheet (Style-Regel) Nr " + j
                + " hat HREF = " + document.styleSheets(i).imports(j).href
            );
        }
    }
}

```

**.parent** Zeiger auf das **in der Fensterhierarchie übergeordnete** Fenster, das aber nicht das .open() zum  
 Kindfenster enthalten muss  
 z.B. Zeiger auf das Fenster, das die FRAMESET-Deklaration enthält,  
 oder das diesem Fenster übergeordnet ist  
 Test auf Existenz von parent per if (parent != self) ....  
 siehe Objekt window

**.parentElement** Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM  
 var Zeiger = document.body.parentElement;





**.ParentFolder** Bezeichner des Eltern-Ordners liefern, in dem der Ordner liegt

Beispiel 1:

```
var OrdnerName          = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner               = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ParentFolder);
```

Beispiel 2:

```
var OrdnerName          = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner               = DateiSystem.GetFolder(OrdnerName);
var Zahler               = 0;
if (!Ordner.IsRootFolder)
{
    do
    {
        Ordner = Ordner.ParentFolder;
        Zahler++;
    }
    while (!Ordner.IsRootFolder);

    alert("Ordner liegt " + Zahler + " Ebenen unter der Root");
}
```

**.ParentFolder** Bezeichner des Ordners liefern, in dem die Datei liegt

Beispiel:

```
var DateinameMitPfad     = "c:\\test.txt";
var DateiSystem           = new ActiveXObject("Scripting.FileSystemObject");
var Datei                = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.ParentFolder);
```

**.parentNode** Referenz auf Elternknoten innerhalb der DOM-Hierarchie

Beispiel 1:

```
<SCRIPT>
    var ElternZeiger = SpanTag.parentNode;
</SCRIPT>
<BODY>
    <SPAN ID="SpanTag">Hallo !</SPAN>
</BODY>
```

Beispiel 2:

document.body.parentNode wird null liefern, da BODY das oberste Objekt

Beispiel 3:

```
var ZeigerAuf_B_Tag = document.createElement("B");    // erzeugen
document.body.insertBefore(ZeigerAuf_B_Tag);          // einfügen
var ElternZeiger = ZeigerAuf_B_Tag.parentNode;        // Zeiger auf BODY
```

**.parentStyleSheet** Referenz auf das den StyleSheet implementierende Objekt (Elternobjekt)  
siehe Objekt styleSheet

**.parentTextEdit** Textbereich des Elternobjektes referenzieren

Beispiel:

```
<SCRIPT>
    function Selektion()
    {
        var ZeigerAufSelektionsQuelle = window.event.srcElement ;

        if (!ZeigerAufSelektionsQuelle.isTextEdit)
        { ZeigerAufSelektionsQuelle = ZeigerAufSelektionsQuelle.parentTextEdit; }

        if (ZeigerAufSelektionsQuelle != null)
        {
            var ZeigerAufTextBereich =
                ZeigerAufSelektionsQuelle.createTextRange();

            ZeigerAufTextBereich.moveToElementText(window.event.srcElement);
            ZeigerAufTextBereich.collapse();
            ZeigerAufTextBereich.expand("SelektionsText");
            ZeigerAufTextBereich.select();
        }
    }
</SCRIPT>
```



.parentTimeBegin      Startzeit als Offset von der Startzeit der Eltern-Timeline  
                          siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function ZeitHolen()
{
    ID_Span.innerHTML +=
        ID_Div.currTimeState.parentTimeBegin + ' Sekunden';
}
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
CLASS="time_line_klasse"
DUR=".01"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
>0
</SPAN>
<BR>
<SPAN ID="ID_Span2"
CLASS="time_line_klasse"
DUR=".01"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=parseInt(ID_excl.currTimeState.activeTime);"
>0
</SPAN>
<BR>
<t:EXCL ID="ID_excl "
REPEATDUR="indefinite"
onbegin=" ZeitHolen();"
>
    <DIV ID="ID_Div"
CLASS="time_line_klasse"
BEGIN="1"
DUR="5"
>
    </DIV>
</t:EXCL>
<BR>
<SPAN ID="ID_Span">
0 Sekunden
</SPAN>
</BODY>
</HTML>
```

.parentTimeEnd      Endezeit als Offset von der Startzeit der Eltern-Timeline  
                          ist zugleich Summe der Werte laut Eigenschaften  
                          parentTimeBegin und activeDur  
                          Hinweis zu activeDur: inklusive aller Wiederholungen und Autoreverse  
                          siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function ZeitHolen()
{
    ID_Span.innerHTML +=
        ID_Div.currTimeState.parentTimeEnd + ' Sekunden';
}
</SCRIPT>
```



```

</HEAD>
<BODY>
  <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
  >
    0
  </SPAN>
  <BR>
  <SPAN ID="ID_Span2"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_excl.currTimeState.activeTime);"
  >
    0
  </SPAN>
  <BR>
  <t:EXCL ID="ID_excl"
        REPEATDUR="indefinite"
        onbegin="ZeitHolen();"
  >
    <DIV ID="ID_Div"
          CLASS="time_line_klasse"
          BEGIN="4"
          DUR="10"
          AUTOREVERSE="true"
    >
    </DIV>
  </t:EXCL>
  <BR>
  <SPAN ID="ID_Span">
    0 Sekunden
  </SPAN>
</BODY>
</HTML>

```

**.parentWindow**      Referenz auf Elternfenster des Dokumentes  
 document.parentWindow

**.path**      Liste von Werten für 2D-Vektorgraphik-Animation eines Elementes auf der Timeline  
 nur für Objekt `animatemotion` (t:ANIMATEMOTION)  
 siehe `.values` für 2D-Animation anhand absoluter Koordinaten im Grafiksystem  
 wenn `.calcMode` auf "paced" so werden Move To-Kommandos ignoriert  
 für die Objekte `animate`, `animateMotion` und `animateColor` gilt:  
     `.path` wird von keiner Eigenschaft überschrieben  
     `.path` überschreibt `.by` `.from` `.to` `.values`  
 Achtung: Im Objekt `animatecolor` (t:ANIMATECOLOR) ist `.values` auch als Farbliste definiert !  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel 1:

`path="M 0 0 L 100 0 v 100 h -100 V 0"`

erreichte Position absolut zum Ursprung 0,0			
Operation	X-Ordinate	Y-Ordinate	
Pfadanfang	0	0	
Line ziehen	100	0	
vertikale Linie	100	100	um 100 Pixel nach oben +100
horizontale Linie	0	100	um 100 Pixel nach links - 100
vertikale Linie	0	0	

Hinweis: Anstelle von "V 0" hätte auch Z oder z kodiert werden können.

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

```



```

<DIV ID="ID_Div1"
      CLASS="time_line_klasse"
      STYLE="position:absolute;top:50px;left:50px;height:100px;width:150px;
            border:solid black 1px;
            "
>
    animierter Div mit Vektorgraphik
</DIV>
<t:ANIMATEMOTION TARGETELEMENT="ID_Div1"
                  BEGIN="ID_Button.click"
                  PATH="M 0 0 L 100 0 v 100 h -100 V 0"
                  DUR="3"
>
</t:ANIMATEMOTION>
<DIV ID="ID_Div2"
      CLASS="time_line_klasse"
      STYLE="position:absolute;top:50px;left:250px;height:100px;width:150px;
            border:solid black 1px;
            "
>
    animierter Div mit Vektorgraphik
</DIV>
<t:ANIMATEMOTION TARGETELEMENT="ID_Div2"
                  BEGIN="ID_Button.click"
                  PATH="m 0 0 L 100 0 100 100 0 100 z"
                  DUR="3"
>
</t:ANIMATEMOTION>
<BUTTON ID="ID_Button">Starten/Restarten</BUTTON>
</BODY>
</HTML>

```

.Path Pfad des Ordners liefern in voller Länge (nicht 8.3 Pfad)

Beispiel:

```

var OrdnerName      = "c:\\windows\\desktop\\";
var DateiSystem     = new ActiveXObject("Scripting.FileSystemObject");
var Ordner          = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Path);

```

.Path Pfad der Datei liefern in voller Länge (nicht 8.3 Pfad)

Beispiel:

```

var DateinameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.Path);

```

.Path Pfad eines Laufwerkes ermitteln in voller Länge (nicht 8.3 Pfad)

Beispiel:

```

var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk         = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Pfad     = Laufwerk.Path;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Pfad);

```

.pathname Datei und Pfad eines Objektes

.peers

Abbruch bzw. Pausierung der aktiven Animation durch Start der Animation eines Kindes  
Time-Container ist Behavior-Objektes .style.time2.excl

Es gibt **genau** ein t:PRIORITYCLASS, in dem **kein** weiteres Behavior-Objekt  
.style.time2.priorityClass kodiert sein darf.

wenn mehrere pausierende Kinder existieren:

Das zuletzt pausierte Kind startet zuerst.

Das zuerst pausierte Kind startet zuletzt.

siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

```



```

<t:EXCL ID="ID_Excl"
  BEGIN="0;indefinite;"
>
  <t:PRIORITYCLASS PEERS="pause">
    <SPAN ID="ID_Span1"
      CLASS="time_line_Klasse"
      BEGIN="0"
      DUR="10"
    >
      Test1
    </SPAN>
    <SPAN ID="ID_Span2"
      CLASS="time_line_Klasse"
      BEGIN="5"
      DUR="3"
    >
      Test2
    </SPAN>
  </t:PRIORITYCLASS>
</t:EXCL>
<BR>
<BUTTON ID="ID_Button" onclick="ID_Excl.beginElement();">
</BUTTON>
</BODY>
</HTML>

```

.PI                      Zahl Pi  
                          siehe Script-Objekt Math

.platform                Name des Betriebssystems per navigator Objekt  
                          navigator.platform  
                          "HP-UX"                HP Unix  
                          "MacPPC"                Macintosh PowerPC  
                          "Mac68K"                Macintosh 68K  
                          "SunOS"                 Solaris  
                          "Win32"                Microsoft Windows 32-Bit  
                          "Win16"                Microsoft Windows 16-Bit  
                          "WinCE"                Microsoft Windows CE

Beispiel für prüfen auf ActiveX beim IE:

```

var ActiveXAktiv = false;

var NavigatorObjekt = window.navigator;                      // Zeiger

var BrowserArt        = NavigatorObjekt.userAgent;            // String
var IEerkannt        = (BrowserArt.indexOf('IE') > -1);       // true, so IE erkannt

var Plattform         = NavigatorObjekt.platform;            // String
var Win32Erkannt     = (Plattform == "Win32");               // true, so Win32-Bit erkannt

ActiveXAktiv         = (IEerkannt && Win32Erkannt);         // true, so IE und Win32-Bit erkannt,
                                                                 // also ActiveX möglich

```

.platform                Windows-Betriebssystem  
                          Behavior .style.clientCaps  
                          "Win32"                Windows 32-Bit  
                          "Win16"                Windows 16-Bit  
                          "WinCE"                Windows CE

.platform                Betriebssystem  
                          siehe Objekt window.clientInformation  
                          "HP-UX"                HP Unix  
                          "MacPPC"                Macintosh PowerPC  
                          "Mac68K"                Macintosh 68K  
                          "SunOS"                Solaris  
                          "Win32"                Microsoft Windows 32-Bit  
                          "Win16"                Microsoft Windows 16-Bit  
                          "WinCE"                Microsoft Windows CE

.player                    Wiedergabeplayer einem Media-Element zuordnen für Wiedergabe auf der Timeline  
                          Player ist eine Softwarekomponente, die im System und im Browser installiert sein muss  
                          ab IE 6.x werden keine Plugins mehr unterstützt sondern nur noch ActiveX-Controls  
                          z.B. ist {6BF52A52-394A-11d3-B153-00C04F79FAA6} das ActiveX-Control für den  
                          Windows Media-Player 7.1



siehe Objekt currTimeState und Behavior .style.time2

Beispiel für Wiedergabe per Media Bar-Player:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY ID="ID_Body">
    <t:PAR SYNCBEHAVIOR="locked">
        <t:MEDIA
            ID="ID_Media"
            SRC="test.mid"
            PLAYER="{52ca3bcf-3b9b-419e-a3d6-5d28c0b0b50c}"
            SYNCMASTER="true"
            SYNCBEHAVIOR="locked"
            BEGIN="1"
        >
        </t:MEDIA>
        <t:ANIMATEMOTION
            ID="ID_Animatemotion"
            BEGIN="ID_Media.begin"
            DUR="9"
            TARGETELEMENT="ID_Div1"
            PATH="M 0 0 L 100 300"
            FILL="freeze"
        >
        </t:ANIMATEMOTION>
        <t:SEQ ID="ID_Seq"
            STYLE="font-size:18pt;color:#ff0000"
            SYNCBEHAVIOR="locked"
            BEGIN="ID_Media.begin"
            FILL="freeze"
        >
            <DIV ID="ID_Div1"
                CLASS="time_line_klasse"
                DUR="1.5"
            >
                Diese MIDI-File
            </DIV>
            <DIV ID="ID_Div2"
                CLASS="time_line_klasse"
                DUR="1.5"
            >
                wird mit dem
            </DIV>
            <DIV ID="ID_Div3"
                CLASS="time_line_klasse"
                DUR="2"
            >
                Media Bar-Player
            </DIV>
            <DIV ID="ID_Div4"
                CLASS="time_line_klasse"
                DUR="2"
            >
                abgespielt.
            </DIV>
        </t:SEQ>
    </t:PAR>
    <DIV ID="ID_Div5"
        CLASS="time_line_klasse"
        STYLE="background-color:#FFCC00;position:absolute;height:40;width:120;
            top:70;left:10;font-size:18;border-style:solid"
    >
        animierter DIV waehrend der Wiedergabe der MIDI-File
    </DIV>
</BODY>
</HTML>
```

Beispiel für Wiedergabe per Windows Media Player:



```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY ID="ID_Body">
    <t:PAR SYNCBEHAVIOR="locked">
        <t:MEDIA ID="ID_Media"
            SRC="test.mid"
            PLAYER="{6BF52A52-394A-11d3-B153-00C04F79FAA6}"
            SYNCMASTER="true"
            SYNCBEHAVIOR="locked"
            BEGIN="1"
        >
    </t:MEDIA>
    <t:ANIMATEMOTION ID="ID_Animatemotion"
        BEGIN="ID_Media.begin"
        DUR="9"
        TARGETELEMENT="ID_Div1"
        PATH="M 0 0 L 100 300"
        FILL="freeze"
    >
</t:ANIMATEMOTION>
    <t:SEQ ID="ID_Seq"
        STYLE="font-size:18pt;color:#ff0000"
        SYNCBEHAVIOR="locked"
        BEGIN="ID_Media.begin"
        FILL="freeze"
    >
        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            DUR="1.5"
        >
            Diese MIDI-File
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            DUR="1.5"
        >
            wird mit dem
        </DIV>
        <DIV ID="ID_Div3"
            CLASS="time_line_klasse"
            DUR="2"
        >
            Windows Media-Player
        </DIV>
        <DIV ID="ID_Div4"
            CLASS="time_line_klasse"
            DUR="2"
        >
            abgespielt.
        </DIV>
    </t:SEQ>
    </t:PAR>
    <DIV ID="ID_Div5"
        CLASS="time_line_klasse"
        STYLE="background-color:#FFCC00;position:absolute;height:40;width:120;
            top:70;left:10;font-size:18;border-style:solid
        "
    >
        animierter DIV wachrend der Wiedergabe der MIDI-File
    </DIV>
</BODY>
</HTML>

```

.playerObject

Zeiger auf den Wiedergabeplayer, der das Media-Element auf der Timeline wiedergibt  
 besonders zu verwenden, wenn Player über ActiveX-Control im Browser implementiert ist  
 Zeiger ermöglicht, die Methoden und Eigenschaften des Players anzusprechen  
 siehe Playerbeschreibung des Herstellers vom Player  
 (im Falle von Microsoft sind die Methoden und Eigenschaften  
 im jeweiligen SDK der Player-Version zu finden)



	Playerart laut <code>.player</code> siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>						
<code>.playlistInfo</code>	Zeiger auf die Playliste (Objekt <code>PlaylistInfo</code> ) siehe Behavior <code>.style.mediaBar</code>						
<code>.playState</code>	Wiedergabe-Status des Media Bar Player (Wiedergabe der Media-Datei) Status wird verändert durch Aktionen des Users mit dem Player Media Bar Player ist der Windows Media Player Es kann zu jedem Zeitpunkt nur genau 1 Media-Datei wiedergegeben werden, es sei denn, man öffnet mehrere Browser-Instanzen. Die Wiedergabe ist nicht möglich, wenn der User in der Media Bar per Links navigiert, da damit die						
aktuelle	Url in der Media Bar geändert wird gegenüber der Url der gerade wiedergegebenen Media-Datei. siehe Methoden <code>.playNext()</code> <code>.playURL()</code> <code>.stop()</code> siehe Behavior <code>.style.mediaBar</code> 0 undefiniert 1 Wiedergabe ist gestoppt 2 Wiedergabe pausiert 3 Player gibt gerade einen Daten-Stream wider 4 Player springt geraden den Nachfolger-Daten-Stream an 5 Player springt geraden den Vorgänger-Daten-Stream an 6 Player puffert gerade Media-Daten 7 Player wartet gerade auf Daten-Stream 8 Player hat das Ende der Media-Datei erkannt, also das Ende des letzten Daten-Stream in der Media-Datei 9 Player bereitet gerade eine neue Media-Wiedergabe vor 10 Player ist bereit für Wiedergabe						
<code>.plugins</code>	Zeiger auf Collection plugin siehe Objekt navigator						
<code>.port</code>	<b>Port</b> einer Location oder Url in der Form "hostname: <b>port</b> " basierend auf dem aktuellen <b>Protokoll</b> laut Eigenschaft <code>.protocol</code> z.B. <table> <tr> <th>Standard-Ports</th><th>Protokoll</th></tr> <tr> <td>21</td><td>FTP</td></tr> <tr> <td>80</td><td>HTTP</td></tr> </table>	Standard-Ports	Protokoll	21	FTP	80	HTTP
Standard-Ports	Protokoll						
21	FTP						
80	HTTP						

Beispiel:

```

<SCRIPT>
function getPort()
{
    alert ("FTP: " + ID_A1.port + "\n" + "HTTP: " + ID_A2.port);
}
</SCRIPT>
<A ID="ID_A1" HREF="ftp://www.test.de" onclick="getPort();">Link1</A>
<A ID="ID_A2" HREF="http://www.test.de" onclick="getPort();">Link2</A>

```

POSITIVE_INFINITY	entspricht Positiv-Unendlich kann als Wert zugewiesen werden Hinweis für numerische Operationen						
	<table> <tr> <td>Multiplikation</td><td>           Wert mal unendlich ergibt unendlich            unendlich mal unendlich ergibt unendlich            0 * unendlich ergibt NaN            NaN * unendlich ergibt NaN         </td></tr> <tr> <td>Division</td><td>           Wert durch unendlich ergibt 0            unendlich durch Wert ergibt unendlich            unendlich durch unendlich ergibt NaN            Division durch 0 nicht erlaubt         </td></tr> <tr> <td>Vorzeichen</td><td>wird wie üblich ermittelt</td></tr> </table>	Multiplikation	Wert mal unendlich ergibt unendlich unendlich mal unendlich ergibt unendlich 0 * unendlich ergibt NaN NaN * unendlich ergibt NaN	Division	Wert durch unendlich ergibt 0 unendlich durch Wert ergibt unendlich unendlich durch unendlich ergibt NaN Division durch 0 nicht erlaubt	Vorzeichen	wird wie üblich ermittelt
Multiplikation	Wert mal unendlich ergibt unendlich unendlich mal unendlich ergibt unendlich 0 * unendlich ergibt NaN NaN * unendlich ergibt NaN						
Division	Wert durch unendlich ergibt 0 unendlich durch Wert ergibt unendlich unendlich durch unendlich ergibt NaN Division durch 0 nicht erlaubt						
Vorzeichen	wird wie üblich ermittelt						
	siehe Script-Objekt Number nur lesen						

<code>.previousSibling</code>	Referenz auf das Vorgängerkind
-------------------------------	--------------------------------

Beispiel:

```

<SCRIPT>
var AktuellesKind_Index = 1; // Index ab 0
var VorgaengerKind_Zeiger = Liste.childNodes(AktuellesKind_Index).previousSibling;
// Listenelement 1 wird referenziert
</SCRIPT>
<BODY>
<UL ID = "Liste">
  <LI> Listenelement 1
  <LI> Listenelement 2
  <LI> Listenelement 3
</UL>

```





&lt;BODY&gt;

.progress                   Fortschrittangabe entlang der Timeline  
                               inklusive Repeats etc.  
                               siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<SCRIPT LANGUAGE="JScript">
function update()
{
    var Fortschritt = movie.currTimeState.progress;
    var Kette = Fortschritt.toString();

    if (movie.currTimeState.stateString != "holding")
    {
        // Element wird nicht gehalten
        if (Fortschritt != 0)
        {
            if (Kette.substr(2,1) == "0")
            {
                Kette = Kette.substr(3,1); // 1 bis 9
            }
            else
            {
                Kette = Kette.substr(2,2); // 10 bis 99
            }
        }
        else
        { Kette = "0"; }
    }
    else
    {
        // Element wird gehalten
        Kette = "100";
    }

    alert(Kette + "%");
}
</SCRIPT>
```

.propertyIsEnumerable   prüfen ob Stringwert in einem Objekt als Menge von String-Elementen enthalten ist  
                               **und** ob das Objekt mit der Anweisung for in verarbeitet werden kann

Beispiel:

```
var Feld = new Array("Apfel", "Banane", "Zitrone");
alert( (Feld.propertyIsEnumerable("Apfel"));       // true
```

.propertyName           Name der Eigenschaft, die wertmässig verändert wurde durch onpropertychange Event  
                               auch Style-Eigenschaft etc.  
                               Objekt event

Beispiel:

```
<HEAD>
<SCRIPT>
function ValueAendern()
{ ID_Input1.value = "Neuer Wert von VALUE";}

function FarbeAendern()
{ ID_Input2.style.backgroundColor = "aqua";}

function Anzeige()
{alert(event.propertyName + " wurde verändert");}

</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE=button ID="ID_Input1"
VALUE="Click um VALUE zu ändern"
onclick="ValueAendern()"
onpropertychange="Anzeige()"
>
<INPUT TYPE=button ID="ID_Input2"
VALUE="Click um Farbe zu ändern"
onclick="FarbeAendern()"
onpropertychange="Anzeige()"
>
</BODY>
```



`.protocol`                      Protokoll-Teil einer Url inklusive http und ftp liefern  
                                  nur lesen bei Objekten document  
                                  img  
                                  location

Beispiel:

location.protocol liefert z.B. http: und immer inklusive dem Doppelpunkt

`.prototype`                      Zeiger auf den Prototyp-Bereich im Objekt, der per Prototyping erweitert wird  
                                  Objekt ist entweder Script-Objekt oder bereits instanziiertes Objekt:  
                                  innerhalb eines Konstruktors ist `.prototype` nicht zu kodieren  
                                  Prototyping eines Objektes verändert den Umfang der Standard-Methoden und -Eigenschaften zum Objekt  
                                  zur Laufzeit der Scriptmaschine.  
                                  Script-Objekte können nur erweitert werden.  
                                  Für private Objekte, die per `new`-Anweisung mit privatem Konstruktor erzeugt wurden, wird leider **nicht**  
                                  die Eigenschaft `.prototype` erzeugt ! Prototyping kann also nur innerhalb des Konstruktors  
                                  erfolgen und nicht nachträglich per Eigenschaft `.prototype` .  
                                  Nur für Objekte, die aus einem vordefiniertem Objekt abgeleitet sind, existiert die Eigenschaft `.prototype` .  
                                  siehe auch `.isPrototypeOf()` und `.hasOwnProperty()`

Beispiel für Erweiterung des Script-Objektes Array, das die Eigenschaft `.prototype` besitzt:

```
function MaximumErmitteln( )
{
    var max = this[0];    // this referenziert die Objekt-Instanz, in dem die Methode vorhanden ist,
                        // also Variable Feld

    for (var i = 1; i < this.length; i++)
    {
        if (max < this[i])
        {max = this[i];}
    }

    return max;
}

// neue Methode per Prototyping hinzufügen zum JScript-Objekt Array
Array.prototype.NeueArrayMethode = MaximumErmitteln;    // ohne () kodieren !

// Feld vom Array-Typ erzeugen mit der Methode .NeueArrayMethode()
var Feld = new Array(1, 2, 3, 4, 5, 6);    // 6 numerische Elemente

// neue Methode des JScript-Objektes Array aufrufen
alert(Feld.NeueArrayMethode());
```

Beispiel für private Datenstruktur-Objekt mit Methode anhand einer privaten Konstruktor-Methode:

Es wird die Eigenschaft `.prototype` **nicht** erzeugt !

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
<!--
function datenstruktur_ausgeben()
{
    with (document)
    {
        write(person.vorname + " " + person.nachname + "<BR>");
        write(person.strasse + " " + person.nummer + "<BR>");
        write(person.plz + " " + person.ort + "<BR>");
        //      Erika Mustermann
        //      Musterstrasse 1
        //      .....
    }

    for (i in person)
    {
        document.write(i + ": " + person[i] + "<BR>");
        //      vorname: Erika
        //      nachname: Mustermann
        //      .....
    }
}

function datenstruktur_erzeugen(vorname, nachname, strasse, nummer, plz, ort, methode)
{
    this.vorname = vorname;
    this.nachname = nachname;
    this.strasse = strasse;
```



```

        this.nummer = nummer;
        this.plz = plz;
        this.ort = ort;
        this.methode = methode;
    }

    person = new datenstruktur_erzeugen
    (
        "Erika",                // Vorname
        "Mustermann",          // Nachname
        "Musterstrasse",        // Strasse
        "1",                    // Nummer
        "10000",                // PLZ
        "Musterstadt",          // Ort
        datenstruktur_ausgeben  // ohne () kodieren
    );

    // alternativ auch kodierbar:
    // var person = {
    //     vorname:"Erika",        // Vorname
    //     nachname:"Mustermann",  // Nachname
    //     strasse:"Musterstrasse", // Strasse
    //     nummer:"1",            // Nummer
    //     plz:"10000",            // PLZ
    //     ort:"Musterstadt"       // Ort
    //     methode:datenstruktur_ausgeben // Ausgabemethode ohne () kodieren
    // };

    // Achtung: Eigenschaft .prototype wird leider nicht (automatisch) erzeugt und ist somit nicht anwendbar !

    alert(person.vorname + "\n" + person.methode);    // person.methode ohne () kodieren !
    // -->
    </SCRIPT>
    </HTML>

```

**.pseudoClass**      Pseudoklasse einer Seite oder @page Regel per Objekt page

":first"	Regel gehört der 1. Seite
":footer"	Regel gehört der Fußnote
":header"	Regel gehört der Kopfnote
":left"	Regel gehört der linken Seite
":left : header"	Regel gehört der Kopfnote Seite links
":left : footer"	Regel gehört der Fußnote Seite links
":right"	Regel gehört der rechten Seite
":right:header"	Regel gehört der Kopfnote Seite rechts
":right:footer"	Regel gehört der Fußnote Seite rechts

Beispiel: @page : left {font-weight:bold;font\_style:italic;}

**.rating**      Rating der Media-Datei auf der Timeline

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO ID="ID_Video"
            SRC="test.wmv"
            STYLE="position:absolute;top:50px;height:100px"
    >
    </t:VIDEO>
    <SPAN ID="ID_Span"
        STYLE="position:absolute;top:165px;"
    >
    </SPAN>
    <BUTTON ID="ID_Button"
        onclick="ID_Span.innerText= ID_Video.rating"
    >
        Klick
    </BUTTON>
</BODY>
</HTML>

```

**.readOnly**      Editierbarkeit eines Text-Controls



Editierung bedeutet Focuserhalt !  
 false      Default  
             Objekt ist **nicht** read-only  
                     also ist es editierbar  
                     kann also Focus erhalten  
 true        Objekt ist read-only  
                     also ist es nicht editierbar  
                     kann kein also Focus erhalten

.readOnly                      Art der Implementation des StyleSheets im Dokument ermitteln  
                                   siehe Objekt styleSheet

.readyState                    aktueller Status des Objektes beim Füllen des Objektes mit Daten  
                                   "uninitialized"      Objekt ist nicht initialisiert  
                                   "loading"            Objekt ist nicht initialisiert aber lädt gerade Daten zur Initialisierung  
                                   "loaded"            Objekt hat alle Daten komplett geladen und ist komplett initialisiert  
                                   "interactive"        Objekt kann vom User interaktiv verwendet werden zum Füllen mit Daten  
                                   "complete"            Object hat alle Daten geladen und ist mit diesen komplett initialisiert

Beispiel:

alert(document.body.readyState);

.recordNumber                Datenquelle-Satznummer eines Datenfeldes

.recordset                    Zeiger des Standard-Record-Set in einem Datasource-Objekt (DSO)  
                                   Methode . namedRecordset() liefert den Zeiger für Datenquellen-Objekt mit Namen  
                                   also eines beliebige Mitglieders im Datasource-Objekt (DSO)

.referrer                      Url der direkt vorhergehenden Seite der aktuellen Seite  
                                   aktuelle Seite muss durch Link angesprungen sein  
                                   Eingabe in Adresszeile oder per Menü Datei-Öffnen etc. wird nicht verwendet  
                                   ist Leerkette, wenn aktuelle Seite nicht durch Link von der vorherigen aktiviert wurde

.rel                            Beziehung zwischen Objekt in Quellseite und Nachfolger-Zielseite laut Objekt  
                                   Objekt ist ein Link z.B. A-Tag oder Link-Tag  
                                   nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen  
                                   beim durchforsten der Seiten und Nachfolgerseiten  
                                   vorrangig kodiert in <A> oder <LINK>  
                                   es **muss** zugleich die Eigenschaft .href kodiert und mit **gültigen** Wert belegt sein  
                                   nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)

Beispiel für Webseite mit eigenem Icon ab IE 5.x:

<LINK    REL="SHORTCUT ICON" HREF="name.ico">

name.ico: name ist beliebig, auch mit Pfad

ICO-Datei:                      32x32 Pixel mit 16 Farben  
                                   oder            16x16 Pixel mit 256 Farben  
                                   ist BMP-Datei, die zu ICO-Datei konvertiert werden muss  
                                   (kann nicht jedes Grafik-Programm, aber Microsoft bietet dafür IconPro an)

Beispiel für Seitenelemente vom Druck ausschließen (ab IE 4.x und NS 6.x):

Seitenteile, die nicht gedruckt werden sollen, in <DIV CLASS="keindruck"> und </DIV>  
 oder in <SPAN CLASS="keindruck"> und </SPAN> einschliessen.

zwischen <HEAD> und </HEAD> einfügen:

<LINK REL="stylesheet" MEDIA="print" HREF="print.css">

print.css enthält nur

.keindruck {display:none;}

Beispiel für Seitenelemente nur beim Druck anzeigen (ab IE 4.x):

Seitenteile, die nur auf Ausdrucken sichtbar sein sollen, in <DIV CLASS="nurdruck"> und </DIV>  
 oder in <SPAN CLASS="nurdruck"> und </SPAN> einschliessen

zwischen <HEAD> und </HEAD> einfügen:

<LINK REL="stylesheet" MEDIA="screen" HREF="screen.css">

screen.css enthält nur

.nurdruck {display:none;}

.repeatCount                aktuelle Nummer der Wiederholung bei Loop



siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
function Anzeige()
{
    ID_Div1.innerText = " Aktuelle Wiederholung: "
                        + (ID_Animation.currTimeState.repeatCount + 1);
                        // repeatCount ab 0, Anzeige ab 1
}
</SCRIPT>
</HEAD>
<BODY>
<DIV ID="ID_Div1">Aktuelle Wiederholung: 1</DIV>
<DIV ID="ID_Div2"
CLASS="time"
STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<t:ANIMATEMOTION ID="ID_Animation"
TARGETELEMENT="ID_Div2"
TO="150,0"
BEGIN="0;indefinite"
DUR="1"
AUTOREVERSE="true"
REPEATCOUNT="5"
onrepeat="Anzeige()"
>
</t:ANIMATEMOTION>
<BR>
<BUTTON ID="ID_Button"
onclick= "ID_Div1.innerText='Aktuelle Wiederholung: 1';"
+ "ID_Animation.beginElement();"
>
Click to restart
</BUTTON>
</BODY>
</HTML>
```

.repeatDur

Anzahl der Dauer in Sekunden für Wiederholung auf der Timeline  
verlangt kodierte Eigenschaft .dur oder .repeat  
darf nicht kodiert werden mit der Eigenschaft .repeatCount  
siehe Objekt currTimeState und Behavior .style.time2  
"indefinite" für endlos (Default)

Wert im Time-Format des Behavior

z.B. "h:min:s.f"  
"0" entspricht "indefinite"

**nicht** id.event  
**nicht** id.event+zeit\_wert\_als\_string

Beispiele:

"25:45:10"	25 Stunden, 45 Minuten, 10 Sekunden
"45:35"	45 Minuten, 35 Sekunden
"45:00.275"	45 Minuten, 0,275 Sekunde
"10.5"	10,5 Sekunden

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:SEQ ID="ID_Seq"
REPEATDUR="27"
BEGIN="0"
>
<DIV ID="ID_Div1"
CLASS="time_line_klasse"
```



```

        DUR="3"
    >
        3 Sekunden lang anzeigen
    </DIV>
    <DIV ID="ID_Div2"
        CLASS="time_line_klasse"
        DUR="4"
    >
        4 Sekunden lang anzeigen
    </DIV>
</t:SEQ>
</BODY>
</HTML>

```

**.restart** generelle Restartmöglichkeit eines Elementes auf der Timeline  
ersetzt die Eigenschaft `eventRestart`, da diese deprecated ist und nicht mehr verwendet werden darf !!  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:PAR ID="ID_Par"
        BGIN="indefinite"
        DUR="20"
    >
        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            STYLE="...."
            END="6"
            RESTART="never"
        >
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            STYLE="...."
            BEGIN="2"
            END="8"
            RESTART="always"
        >
        </DIV>
    </t:PAR>
    <BUTTON onclick="ID_Par.beginElement();"
    >
        Start
    </BUTTON>
    <BUTTON onclick="ID_Par.endElement();"
    >
        Stop
    </BUTTON>
    <BUTTON onclick="ID_Div1.beginElement();"
    >
        Restart DIV1
    </BUTTON>
    <BUTTON onclick="ID_Div2.beginElement();"
    >
        Restart DIV2
    </BUTTON>
</BODY>
</HTML>

```

**.returnValue** Returnwert für den Eventhandler festlegen anstelle von `return`-Anweisung  
Achtung: Wenn kodiert, so wird eine ebenfalls im Eventhandler kodierte die Anweisung  
`return ....;`  
ignoriert  
Objekt `event`  
`true` Default. Action des Events wurde ausgeführt  
`false` Action des Events wurde nicht ausgeführt

**.returnValue** Returnwert des Dialog-Fensters, das mit Methoden  
`.showModalDialog()` bzw. `.showModelessDialog()`  
erzeugt wurde



wird gefüllt durch .showModalDialog() bzw. .showModelessDialog()  
siehe Objekt window

**.rev** Beziehung zwischen Objekt in Quellseite und Vorgänger-Zielseite laut Objekt  
Objekt ist ein Link z.B. A-Tag oder Link-Tag  
nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen  
beim durchforsten der Seiten und Vorgängerseiten  
vorrangig kodiert in <A> oder <LINK>  
es **muss** zugleich die Eigenschaft .href kodiert und mit **gültigen** Wert belegt sein  
nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)

**.right** rechte Pixelposition des Rechteckes um ein Objekt  
auch textrectangle Objekt

Beispiel für Nicht-TextRectangle-Objekt:

```
<SCRIPT>
function Anzeigen(ObjektZeiger)
{
    var Rechteck = ObjektZeiger.getBoundingClientRect();

    alert(      "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
              + "\n"
              + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
            );
}
</SCRIPT>
<P onclick="Anzeigen(this)">
```

Beispiel für TextRectangleObjekt:

```
<SCRIPT>
function Anzeigen(ObjektZeiger)
{
    var TextRectangleCollection = document.body.ObjektZeiger.TextRange.getClientRects();

    // .getClientRects() liefert immer Collection der textrectangle Objekte !!!

    var Rechteck = TextRectangleCollection[0];

    alert(      "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
              + "\n"
              + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
            );
}
</SCRIPT>
<BODY>
<DIV onclick="Anzeigen(this)">
    Test
</DIV>
</BODY>
```

**.rightMargin** Right Margin in Pixel des Dokumentes  
Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
Padding: Abstand des Objekthinhaltes zum Aussenrand  
document.body.rightMargin

**.RootFolder** Root des Laufwerkes liefern, also mit ":\"  
Beispiel:

```
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk         = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Root     = Laufwerk.RootFolder;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Root);
```

**.rowIndex** Index der Tabellenzeile in der Collection table.rows  
siehe Objekt table.tr

**.rows** Höhe aller Frames eines Frameset (Höhe des Frameset als Container)

Beispiel:

```
<FRAMESET ROWS="40%, 60%">      // Summe muss immer 100% sein
<FRAMESET ROWS="50, *, 80">      // 50 Pixel, 80 Pixel und restliche Fensterhöhe
```



.rows	Anzahl der sichtbaren Zeilen der TEXTAREA siehe textarea Objekt
.rowSpan	Anzahl der Zeilen einer Zelle in einer Tabelle siehe Objekt table.tr.td siehe Objekt table.tr.th
.rules	Art der sichtbaren inneren Rahmen einer Tabelle Art der sichtbaren Rahmen zwischen den Tabellenelementen Art des sichtbaren Rahmens um Tabelle siehe Objekt table "all" Rahmen um alle Zeilen und Spalten "cols" Trennlinie zwischen allen Spalten "groups" horizontale Trennlinie zwischen allen THEAD, TBODY's und TFOOT vertikale Trennlinie zwischen allen COLGROUP "none" keine Rahmen und Trennlinien zwischen Tabellenelementen "rows" Trennlinie zwischen allen Zeilen "" keine Rahmen generell (auch nicht um Tabelle)

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
<THEAD>
  <TR>
    <TD>Kopf Zelle 1</TD>
    <TD>Kopf Zelle 2</TD>
  </TR>
</THEAD>
<TBODY>
  <TR>
    <TD>Body Zeile 1 Zelle 1</TD>
    <TD>Body Zeile 1 Zelle 2</TD>
  </TR>
  <TR>
    <TD>Body Zeile 2 Zelle 1</TD>
    <TD>Body Zeile 2 Zelle 2</TD>
  </TR>
  <TR>
    <TD>Body Zeile 3 Zelle 1</TD>
    <TD>Body Zeile 3 Zelle 2</TD>
  </TR>
</TBODY>
<TFOOT>
  <TR>
    <TD>Fuss Zelle 1</TD>
    <TD>Fuss Zelle 2</TD>
  </TR>
</TFOOT>
</TABLE>
<BUTTON onclick="ID_Tabelle.rules="";">keine Rahmen</BUTTON>
<BUTTON onclick="ID_Tabelle.rules="none";">none</BUTTON>
<BUTTON onclick="ID_Tabelle.rules="all";">all</BUTTON>
<BUTTON onclick="ID_Tabelle.rules="cols";">cols</BUTTON>
<BUTTON onclick="ID_Tabelle.rules="groups";">groups</BUTTON>
<BUTTON onclick="ID_Tabelle.rules="rows";">rows</BUTTON>
```

.scheme	Schema der Interpretation des Wertes laut Eigenschaft .content per meta Objekt z.B. von Datum und Zeit oder Tag-Content-Beschreibung
---------	--

Beispiele:

```
<META scheme="USA" name="date" content="04-05-1962">
<META scheme="Europe" name="date" content="05-04-1962">
```

Tag ID beschreiben

```
<META scheme="customer" name="id" content="test">
```

.scope	Gruppierung, in der eine Zelle liegt siehe Objekt table.tr.td siehe Objekt table.tr.th "row" Zelle enthält Header-Informationen zur Zeile "col" Zelle enthält Header-Informationen zur Spalte "rowgroup" wie "row" aber zur Gruppe aus Zeilen "colgroup" wie "col" aber zur Gruppe aus Spalten
--------	--

.scopeName	Namensraum laut XMLNS-Attribut
------------	--------------------------------





Beispiel:

```
<HTML XMLNS:InetSDK='http://www.test.de'>
<STYLE>
    @media all {InetSDK:HalloDu { behavior:url (simple.htc) }}
</STYLE>
<SCRIPT>
    function window.onload()      // überschreibt Standard window.onload-Routien !!!!
    {
        // Statuszeile als Ausgabebereich nutzen
        window.status = 'scopeName = ' + Hallo.scopeName + ' ';
        + ' tagUrn = ' + Hallo.tagUrn;
    }
</SCRIPT>
<BODY>
    <InetSDK:HelloWorld ID=Hallo></InetSDK:HalloDu>
</BODY>
</HTML>
```

.screen	Zeiger auf das Objekt screen siehe Objekt window
.screenLeft	X-Koordinate der linken oberen Fensterecke (ohne Bar, Menü, Scrollbalken, Statuszeile etc) bezüglich linke obere Ecke (0,0) vom Bildschirm siehe Objekt window
.screenTop	Y-Koordinate der linken oberen Fensterecke (ohne Bar, Menü, Scrollbalken, Statuszeile etc) bezüglich linke obere Ecke (0,0) vom Bildschirm siehe Objekt window
.screenX	X-Koordinate Maus relativ zum Bildschirm Objekt event
.screenY	Y-Koordinate Maus relativ zum Bildschirm Objekt event
.scroll	Scrollbar-Anzeige ein/aus vor IE 6.x nur BODY-Objekt ab IE 6.x alle HTML-Elemente wenn DOCTYPE im Kopf des Dokumentes kodiert "yes" Default sichtbar "no" nicht sichtbar
.scrollAmount	Scrollschrittweite in Pixel des marquee Objektes Standard ist 6
.scrollDelay	Wartezeit in Millisekunden <b>und</b> laut Eigenschaft .trueSpeed zwischen zwei Scrollschritten eines marquee Objektes Scrollgeschwindigkeit, Fließgeschwindigkeit in Millisekunden wenn .trueSpeed auf false (Standard) , so wenn Wert < 60 so 60 verwendet = 1 Sekunde wenn Wert > 59 so Wert verwendet in Millisekunden wenn .trueSpeed auf true, so Wartezeit = Wert in Millisekunden Standard ist 85 Millisekunden wenn .trueSpeed auf false (Standard) , so Wartezeit = 85 Millisekunden da > 60 wenn .trueSpeed auf true, so Wartezeit = 85 Millisekunden Belegung von .trueSpeed ist egal
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes

Beispiel:

```
<DIV ID="ID_Div" STYLE="overflow:scroll; height=100; width=250; text-align:left">
</DIV>
<INPUT TYPE=button VALUE="anzeigen"
onclick="javascript:alert('Scrollhöhe = ' + ID_Div.scrollHeight);"
>
```

.scrolling	Scrollenbar erzeugen "auto" Default automatisch Scrollbar erzeugt, wenn nötig "no" nie Scrollbar erzeugen, also kein Scrollen "yes" immer Scrollbar erzeugen, also Scrollen
------------	---



**.scrollLeft** Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes  
 nur nutzbar nach dem kompletten Laden des Dokumentes  
 immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

Beispiel:

```
<DIV STYLE="position:absolute; width:200px; height:100px; overflow:scroll"
  onclick="alert(this.scrollLeft);"
>
  <SPAN STYLE="width:250px"> . . . </SPAN>
</DIV>
```

**.scrollTop** Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes  
 nur nutzbar nach dem kompletten Laden des Dokumentes  
 immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

Beispiel:

```
<DIV STYLE="position:absolute; width:200px; height:100px; overflow:scroll"
  onclick="alert(this.scrollTop)"
>
  <SPAN STYLE="width:250px"> . . . </SPAN>
</DIV>
```

**.scrollWidth** Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes

Beispiel:

```
<SCRIPT>
function Anzeige()
{
    var ScrollBreite = ID_Div.scrollWidth;
    var OffsetBreite = ID_Div.offsetWidth;
    var Differenz = iScrollWidth - iOffsetWidth;

    alert( "OffsetWidth = " + OffsetBreite
        + "\nScrollWidth = " + ScrollBreite
        + "\nDifferenz = " + Differenz
    );
}
</SCRIPT>
<DIV ID="ID_Div" STYLE="overflow:scroll;height=200;width=250;text-align:left">
</DIV>
<INPUT TYPE=button VALUE="anzeigen" onclick="Anzeige()">
```

**.search** Teil des Wertes des Eigenschaft .href  
 Teil folgt direkt dem Fragezeichen  
 wird als Querystring oder Formdata bezeichnet  
 hat nichts mit der Suche auf Webseiten zu tun  
 Hinweis: Fragezeichen-Anhang an der HREF dient zur Übergabe von String-Werten einer Webseite an eine andere:  
 Quellseite besitzt HREF mit " ...?....."  
 ruft Zielseite mit diesem HREF auf  
 Zielseite wurde von Quellseite aufgerufen  
 liest den Teil von HREF hinter dem ? als Zeichenkettendaten

Beispiel:

```
document.location.search;
```

**.sectionRowIndex** Index der Tabellenzeile in der Collection  
 table.tBody.rows  
 bzw. table.tFoot.rows  
 bzw. table.tHead.rows  
 Zeile muss im **existierenden** Tabellenteil TOBODY bzw. TFOOT bzw. THEAD liegen  
 siehe Objekt table.tr

**SECURITY** temporäre Sicherheit des IFRAME  
 wird an Kinder weitervererbt  
 ab IE 6.0  
 "restricted"  
 entspricht als hätte der User im Browser-Menü "Extra-Internet Optionen-Sicherheit" den Eintrag "Programme und Dateien in einem IFRAME starten" auf "Deaktivieren" gesetzt (falls nicht durch User bereits getan)  
 bewirkt somit die Sperrung der Ausführung von Scripten wie javascript, vbscript **und** Protokollen in URLs (z.B. http), die im Source-File kodiert sind und im IFRAME aktiviert werden sollen  
 Hinweis: wenn restricted, so alle untergeordneten IFRAME ebenfalls restricted

Beispiel 1:

wenn restricted, so nachfolgender Script-Code im FRAME und IFRAME nicht ausgeführt:



```

<IFRAME SECURITY="restricted">
  <A HREF="javascript:alert('funktioniert nicht bei FRAME oder IFRAME!');">
    JavaScript Link
  </A>
</IFRAME>

```

sondern für den Link ein neues Fenster geöffnet

Beispiel 2:

wenn restricted, so nachfolgender Script-Code im IFRAME nicht ausgeführt:

```

<IFRAME SECURITY="restricted" SRC="http://www.microsoft.com"></IFRAME>

```

sondern für den Link ein neues Fenster geöffnet

.segmentDur

Dauer der Timeline der Wiederholungen laut autoReverse  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function ZeitHolen()
{
  ID_Span3.innerText += ID_Span1.currTimeState.segmentDur;
  ID_Span4.innerText += ID_Span2.currTimeState.segmentDur;
}
</SCRIPT>
</HEAD>
<BODY>
<t:EXCL onbegin="ZeitHolen();">
  <t:PRIORITYCLASS PEERS="pause">
    <SPAN ID="ID_Span1"
      CLASS="time_line_klasse"
      BEGIN="0"
      DUR="10"
    >
      <H3>Animation 1</H3>
      <P>im Wechsel mit Animation 2</P>
    </SPAN>
    <SPAN ID="ID_Span2"
      CLASS="time_line_klasse"
      BEGIN="5"
      DUR="5"
    >
      <H3>Animation 2</H3>
      <P>im Wechsel mit Animation 1</P>
    </SPAN>
  </t:PRIORITYCLASS>
</t:EXCL>
<BR>
<SPAN ID="ID_Span3"></SPAN>&nbsp;&nbsp;&nbsp;Sekunden
<SPAN ID="ID_Span4"></SPAN>&nbsp;&nbsp;&nbsp;Sekunden
</BODY>
</HTML>

```

.segmentTime

aktueller Zeitpunkt in der Timeline der Wiederholungen laut autoReverse  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
function WindowOnLoad()
{
  // Timer mit Intervall von 100 Millisekunden
  window.setInterval(Anzeigen, 100);
}

function Anzeigen()
{

```



```

        ID_Div2.innerHTML = "simpleTime:&nbsp;"
                          + (ID_Animation.currTimeState.simpleTime);

        ID_Div3.innerHTML = "segmentTime:&nbsp;"
                          + (ID_Animation.currTimeState.segmentTime);

        ID_Div4.innerHTML = "activeTime:&nbsp;"
                          + (ID_Animation.currTimeState.activeTime);
    }

    function Wiederholen()
    {
        ID_Div1.innerHTML = "repeatCount:&nbsp;"
                          + (ID_Animation.currTimeState.repeatCount + 1);
                          // repeatCount ab 0, aber Anzeige ab 1
    }

    window.onload = WindowOnLoad;    // mit () so sofort ausgeführt
                                      // ohne () so reiner Zeiger

</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <DIV ID="ID_Div1">repeatCount:&nbsp;1</DIV>
        <DIV ID="ID_Div2">simpleTime:&nbsp;0</DIV>
        <DIV ID="ID_Div3">segmentTime:&nbsp;0</DIV>
        <DIV ID="ID_Div4">activeTime:&nbsp;0</DIV>
    </DIV>

    <DIV ID="ID_DivTimeLine"
        CLASS="time_line_klasse"
        STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;" >
    </DIV>
    <t:ANIMATEMOTION          ID="ID_Animation"
                              TARGETELEMENT="ID_DivTimeLine"
                              TO="340,40"
                              DUR="2"
                              AUTOREVERSE="true"
                              REPEATCOUNT="5"
                              FILL="freeze"
                              onrepeat="Wiederholen()"
    >
    </t:ANIMATEMOTION>
</BODY>
</HTML>

```

SEGMENTTYPE	wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software
.selected	<p>Selektionsstatus der Option, also Objektes document.select.option des Objektes document.select standardgemäß ist immer die oberste Option selektiert es kann aus einer Optionengruppe immer nur genau 1 Option selektiert sein</p> <p>true      Option ist selektiert false     Default           Option ist nicht selektiert</p>
.selectedIndex	<p>Index der Option in der Collection document.select.options des Objektes select jedes option Objekt eines select Objektes wird im Feld, das Teil des select Objektes ist, referenziert es kann immer nur eine Option innerhalb einer Optionsgruppe selektiert sein</p>
.selector	Seiten-Selektor per Objekt page
.self	Referenz auf das aktuelle Fenster oder den aktuellen Frame
.self	<p>Zeiger auf das <b>aktuelle</b> Fenster                                   innerhalb der Fensterhierarchie                                   im FRAME</p> <p>ist synonym zu window siehe Objekt window</p>
.SerialNumber	<p>Seriennummer des Laufwerkes liefern</p> <p><b>Achtung: Die Seriennummer lässt eine eindeutige Identifizierung der Festplattenhardware zu vorallem dann, wenn zusätzlich Userdaten zum PC bekannt sind.</b></p>

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
```



```

var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk              = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName   = Laufwerk.VolumeName;
var Laufwerk_SerienNummer = Laufwerk.SerialNumber;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " "
      + Laufwerk_SerienNummer);

```

**.shape**                      Form/Gestalt eines Objektes  
                               siehe auch Eigenschaft .coords  
                               "circ" oder "circle"              Kreisform  
                               "poly" oder "polygon"            Vieleck (Polygon)  
                               "rect" oder "rectangle"          Viereck

**.ShareName**                      öffentlicher Netzwerkname des Laufwerkes liefern

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk      = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_ShareName = Laufwerk.ShareName;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_ShareName);

```

**.shiftKey**                      SHIFT-Tasten-Status  
                               Objekt event  
                               false      SHIFT-Taste ist nicht gedrückt  
                               true        SHIFT-Taste ist gedrückt

Beispiel:

```

<HEAD>
<SCRIPT>
function InnerTextSetzen(Zeiger, Kette)
{Zeiger.innerText=Kette;}

function init()
{
    InnerTextSetzen(ID_Span1,'false');
    InnerTextSetzen(ID_Span2,'false');
}

function ShiftDown()
{
    if (event.shiftLeft)
    { InnerTextSetzen(ID_Span1,'true'); }
    else
    {
        if (event.shiftKey)
        { InnerTextSetzen(ID_Span2,'true'); }
    }
}

function ShiftUp()
{
    if (!event.shiftKey)
    {
        InnerTextSetzen(ID_Span1,'false');
        InnerTextSetzen(ID_Span2,'false');
    }
}
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="ShiftDown();" onkeyup="ShiftUp();">
    <TABLE>
        <TR>
            <TD><I>Linke Shift-Taste gedrueckt</I></TD>
            <TD><I>Recchte Shift-Taste gedrueckt</I></TD>
        </TR>
        <TR>
            <TD ALIGN="center">
                <SPAN ID="ID_Span1"></SPAN>
            </TD>
            <TD ALIGN="center">
                <SPAN ID="ID_Span2"></SPAN>
            </TD>
        </TR>
    </TABLE>
</BODY>

```



.shiftLeft                    SHIFT-Taste links Status  
                              nur für Dokument mit Focus  
                              Objekt event  
                              false      linke SHIFT-Taste ist nicht gedrückt  
                              true       linke SHIFT-Taste ist gedrückt

Beispiel:

```
<HEAD>
<SCRIPT>
function InnerTextSetzen(Zeiger, Kette)
{Zeiger.innerText=Kette;}

function init()
{
    InnerTextSetzen(ID_Span1,'false');
    InnerTextSetzen(ID_Span2,'false');
}
function ShiftDown()
{
    if (event.shiftLeft)
    { InnerTextSetzen(ID_Span1,'true'); }
    else
    {
        if (event.shiftKey)
        { InnerTextSetzen(ID_Span2,'true'); }
    }
}

function ShiftUp()
{
    if (!event.shiftKey)
    {
        InnerTextSetzen(ID_Span1,'false');
        InnerTextSetzen(ID_Span2,'false');
    }
}
}
</SCRIPT>
</HEAD>
<BODY onload="document.body.focus(); init()" onkeydown="ShiftDown();" onkeyup="ShiftUp();">
    <TABLE>
        <TR>
            <TD><I>Linke Shift-Taste gedrueckt</I></TD>
            <TD><I>Recchte Shift-Taste gedrueckt</I></TD>
        </TR>
        <TR>
            <TD ALIGN="center">
                <SPAN ID="ID_Span1"></SPAN>
            </TD>
            <TD ALIGN="center">
                <SPAN ID="ID_Span2"></SPAN>
            </TD>
        </TR>
    </TABLE>
</BODY>
```

.ShortName                    Ordnerbezeichner als 8.3 Bezeichner liefern

Beispiel:

```
var OrdnerName                = "c:\\windows\\desktop\\";
var DateiSystem              = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                    = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ShortName);
```

.ShortName                    Dateibezeichner als 8.3 Bezeichner liefern

Beispiel:

```
var DateinameMitPfad         = "c:\\test.txt";
var DateiSystem              = new ActiveXObject("Scripting.FileSystemObject");
var Datei                    = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.ShortName);
```

.ShortPath                    Pfad des Ordners als 8.3 Pfad liefern

Beispiel:

```
var OrdnerName                = "c:\\windows\\desktop\\";
var DateiSystem              = new ActiveXObject("Scripting.FileSystemObject");
```



```
var Ordner = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ShortPath);
```

.ShortPath                      Pfad der Datei als 8.3 Pfad liefern

Beispiel:

```
var DateinameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.ShortPath);
```

.simpleDur                      Dauer einer Wiederholung  
Wiederholung erzeugt per Eigenschaft .autoReverse auf true setzen  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
function Anzeige1()
{
    ID_Div1.innerText = " Anzahl Wiederholungen: "
                      + (ID_Animation.currTimeState.repeatCount + 1);
                      // repeatCount ab 0, Anzeige ab 1
}

function Anzeige2()
{
    ID_Span.innerHTML = " Dauer jeder Wiederholung: "
                      + (ID_Animation.currTimeState.simpleDur)
                      + ' Sekunden';
}
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="ID_Div0"
CLASS="time_line_klasse"
DUR=".01"
REPEATCOUNT="indefinite"
FILL="hold"
onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
>
0
</SPAN>
<BR>
<DIV ID="ID_Div1"
STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
Anzahl Wiederholungen: 1
</DIV>
<SPAN ID="ID_Span">Dauer jeder Wiederholung: </SPAN>
<DIV ID="ID_Div2"
CLASS="time_line_klasse" ></DIV>
<t:ANIMATEMOTION ID="ID_Animation"
TARGETELEMENT=" ID_Div1"
TO="325,0"
BEGIN="0; indefinite;"
DUR="2"
AUTOREVERSE="true"
REPEATCOUNT="5"
onrepeat="Anzeige1()"
>
</t:ANIMATEMOTION>
<BR>
<BUTTON ID="ID_Button1" onclick="Anzeige2()">
Anzeige Dauer
</BUTTON>
<BUTTON ID="ID_Button2" onclick=" ID_Animation.beginElement();">
Click to restart
</BUTTON>
</BODY>
</HTML>
```



.simpleTime                    aktueller Punkt auf Timeline  
                                  siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT LANGUAGE="JScript">
    function WindowOnLoad()
    {
        // Timer mit Intervall von 100 Millisekunden
        window.setInterval(Anzeigen, 100);
    }

    function Anzeigen()
    {
        ID_Div2.innerHTML = "simpleTime:&nbsp;"
                        + (ID_Animation.currTimeState.simpleTime);

        ID_Div3.innerHTML = "segmentTime:&nbsp;"
                        + (ID_Animation.currTimeState.segmentTime);

        ID_Div4.innerHTML = "activeTime:&nbsp;"
                        + (ID_Animation.currTimeState.activeTime);
    }

    function Wiederholen()
    {
        ID_Div1.innerHTML = "repeatCount:&nbsp;"
                        + (ID_Animation.currTimeState.repeatCount + 1);
        // repeatCount ab 0, aber Anzeige ab 1
    }
    window.onload = WindowOnLoad; // mit () so sofort ausgeführt
                                // ohne () so reiner Zeiger

</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <DIV ID="ID_Div1">repeatCount:&nbsp;1</DIV>
        <DIV ID="ID_Div2">simpleTime:&nbsp;0</DIV>
        <DIV ID="ID_Div3">segmentTime:&nbsp;0</DIV>
        <DIV ID="ID_Div4">activeTime:&nbsp;0</DIV>
    </DIV>

    <DIV ID="ID_DivTimeLine"
        CLASS="time_line_klasse"
        STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;" >
    </DIV>
    <t:ANIMATEMOTION
        ID="ID_Animation"
        TARGETELEMENT="ID_DivTimeLine"
        TO="340,40"
        DUR="2"
        AUTOREVERSE="true"
        REPEATCOUNT="5"
        FILL="freeze"
        onrepeat="Wiederholen()"
    >
    </t:ANIMATEMOTION>
</BODY>
</HTML>
```

.size                    Fontgröße  
                          1 bis 7  
                          1        kleinste Größe  
                          7        größte Größe

.size                    Dimension eines Input-Objektes in Zeichenanzahl

Beispiel:

```
<INPUT TYPE=text SIZE=33>
```





**.size** Anzahl der Zeilen in einer List-Box als select Objekt

**.Size** Ordnergrösse in Bytes liefern

Beispiel:

```
var OrdnerName      = "c:\\windows\\desktop\\";
var DateiSystem     = new ActiveXObject("Scripting.FileSystemObject");
var Ordner           = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Size);
```

**.Size** Dateigrösse in Bytes liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.Size);
```

**.sourceIndex** Index des Objektes in der Collection document.all

Beispiel:

```
<SCRIPT>
function Handler()
{
    // Element referenzieren das Event verarbeitet
    var ElementZeiger =event.srcElement;

    var Index= ElementZeiger.sourceIndex;
    var TagName= ElementZeiger.tagName;

    if(TagName=="!"){TagName="COMMENT";}

    ID_TD1.innerHTML=Index;
    ID_TD2.innerHTML=TagName;

    if (Index-1>0)
    {
        TagName=document.all[Index-1].tagName;

        if(TagName=="!"){TagName="COMMENT";}

        ID_TD3.innerHTML=sTagName;
    }
    else
    { ID_TD3.innerHTML="nicht vorhanden";}

    if (Index+1<document.all.length)
    {
        TagName=document.all[Index+1].tagName;

        if(TagName=="!"){TagName="COMMENT";}

        ID_TD4.innerHTML=TagName;
    }
    else
    { ID_TD4.innerHTML="nicht vorhanden"; }
}
</SCRIPT>
<BODY onmousemove="Handler()">
<TABLE>
<TR><TD>Source Index:</TD><TD ID="ID_TD1"></TD></TR>
<TR><TD>Objekt-Name:</TD><TD ID="ID_TD2"></TD></TR>
<TR><TD>Vorhergehendes Objekt:</TD><TD ID="ID_TD3"></TD></TR>
<TR><TD>Naechstes Objekt:</TD><TD ID="ID_TD4"></TD></TR>
</TABLE>
</BODY>
```

**.sourceURL** Url der Media-Datei des MediaItem Objektes  
siehe Behavior .style.mediaBar

**.span** Anzahl der Spalten innerhalb einer Spaltengruppe einer Tabelle  
siehe Objekt table.col und Objekt table.colGroup

Beispiel:

```
<TABLE BORDER="2">
<COLGROUP SPAN="3" STYLE="color:green;background:black">
<COL SPAN="2" STYLE="color:red">
```



&lt;/COLGROUP&gt;

....

&lt;/TABLE&gt;

.specified

prüfen ob Objekt Attribute hat  
 true, so Attribute ist vorhanden  
 false, so keine Attribute vorhanden

Beispiel:

```
<SCRIPT>
function Anzeigen()
{
    var FeldAllerAttribute = ID_List.attributes;
    alert(FeldAllerAttribute(0).nodeName);    // Knotenname

    for( var i=0; i< FeldAllerAttribute.length; i++)
    {
        // neues LI-Element als Knoten der Liste anhängen
        var NeuesListenElement = document.createElement("LI");
        ID_List.appendChild(NeuesListenElement);

        // Wert des neuen LI-Elementes ist Text, also Textknoten
        var WertNeuesListenElement = document.createTextNode(
            i
            + " "
            + FeldAllerAttribute(i).nodeName
            + " = "
            + FeldAllerAttribute(i).nodeValue
        );
        NeuesListenElement.appendChild(WertNeuesListenElement);

        // auf specified prüfen
        if(FeldAllerAttribute(i).nodeValue != null )
        {
            alert(    FeldAllerAttribute(i).nodeName
                    + " specified: "
                    + FeldAllerAttribute(i).specified    // boolean
            );
        }
    }
}
</SCRIPT>
<UL ID="ID_List" onclick = " Anzeigen()">
    <LI>Klick mich
</UL>
```

.speed

aktuelle Wiedergabegeschwindigkeit des Kindes (Run time speed)  
 Beispiel: Eltern mit 50 % Geschwindigkeit  
 Kind mit 50 % Geschwindigkeit der Eltern  
 also aktuelle Wiedergabegeschwindigkeit = 25 %  
 siehe Objekt currTimeState und Behavior .style.time2  
 Floating-point  
 in Prozent

Beispiel 1:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:PAR SPEED="2" >
    <t:KIND ID="ID_Kind"
        BEGIN="1"
        SRC="http://www.test.de/media/test.wmv"
        SPEED="1.5"
    >
    </t:KIND
</t:PAR>
<BUTTON ID="ID_Button1"
    CLASS="time_line_klasse"
    REPEATCOUNT="indefinite"
    onclick="alert( 'Run time speed des Kindes: ' + ID_Kind.currTimeState.speed );">
```



```

        Run time speed des Kindes
    </BUTTON>
    <BUTTON
        ID="ID_Button2"
        CLASS="time_line_klasse"
        REPEATCOUNT="indefinite"
        onclick="alert( 'SPEED-Attribut des Kindes: ' + ID_Kind.speed );"
    >
        SPEED-Attribut des Kindes
    </BUTTON>
</BODY>
</HTML>

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<SCRIPT>
    function Aendern()
    {
        ID_Video.updateMode = ID_Select1.options.value;
        ID_Video.speed      = ID_Select1.options.value;
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:VIDEO
        ID="ID_Video"
        SRC="test.avi"
        UPDATEMODE="reset"
        STYLE="width:175px; height:150px;"
    >
    </t:VIDEO>
    <BR>
    updateMode waehlen:
    <SELECT NAME="ID_Select1">
        <OPTION VALUE="auto">Auto</OPTION>
        <OPTION VALUE="reset" SELECTED>Reset</OPTION>
    </SELECT>
    <BR>
    Geschwindigkeit waehlen:
    <SELECT NAME="ID_Select2">
        <OPTION VALUE="0.25">25%</OPTION>
        <OPTION VALUE="0.50">50%</OPTION>
        <OPTION VALUE="0.75">75%</OPTION>
        <OPTION VALUE="1" SELECTED>100% </OPTION>
        <OPTION VALUE="2" SELECTED>200% </OPTION>
    </SELECT>
    <BR>
    <BUTTON ID="ID_Button1" onClick="Aendern();">
        Geschwindigkeit aendern
    </BUTTON>
    <BUTTON ID="ID_Button1" onClick="document.body.beginElement();">
        Restart
    </BUTTON>
</CENTER>
</BODY>
</HTML>

```

.SQRT1\_2      Quadratwurzel von 0,5 also von 1 dividiert durch Quadratwurzel von 2  
siehe Script-Objekt Math

.SQRT2      Quadratwurzel von 2  
siehe Script-Objekt Math

.src      Url der Daten z.B. vom Image in normaler Auflösung

Hinweis: Eine Kodierung der Url ohne Einschluss in " " bzw. ' ' kann in der Regel nur erfolgen, wenn die Url bereits ein Wert vom Format der Eigenschaft .src ist. Empfehlung: Per new ein Objekt erzeugen und dann .src belegen.  
.src bzw. das SRC-Attribut kann mit einer Url belegt werden, die innerhalb " " bzw. ' ' kodiert wurde. Alternativ ist der Wert der Eigenschaft .src eines anderen Objektes zuweisbar, aber ohne Kodierung von " " bzw. ' '.



Beispiel: 

```
var Pfad='./test/';
var BildName = 'testbild.jpg';
var BildObjekt1 = new Image(10,20);
BildObjekt1.src= "" + Pfad + BildName + "";
BildObjekt1.src= Pfad + BildName; // könnte Fehler erzeugen bzw. nicht ausgeführt werden
var BildObjekt2 = new Image(100,200);
BildObjekt2.src = BildObjekt1.src;
BildObjekt2.src = "" + BildObjekt1.src + ""; // könnte Fehler erzeugen bzw. nicht ausgeführt werden
```

Beispiel: 

```
<BGSOUND SRC="file:///c:/test/wav/test.wav">
```

**.src**                      Url einer Media-Datei auf der Timeline  
nur lesen beim Objekt laut `object.playList.item()` bzw. `object.playList.aktiveTrack`

**.srcElement**              Referenz auf Objekt das das Event auslöst  
Objekt event

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
    function selectWord()
    {
        var ZeigerAufObjekt = window.event.srcElement ;

        if (!ZeigerAufObjekt.isTextEdit)
        { ZeigerAufObjekt = window.event.srcElement.parentTextEdit;}

        if (ZeigerAufObjekt != null)
        {
            var TextBereich = ZeigerAufObjekt.createTextRange();
            TextBereich.moveToElementText(window.event.srcElement);
            TextBereich.collapse();
            TextBereich.expand("word");
            TextBereich.select();
        }
    }
</SCRIPT>
```

Beispiel 2:

```
<STYLE>
    .normal {background-color:"#FFFFFF"; color:"#000000"; font-weight:normal; font-size:8pt; font-family:Arial;}
    .accessible { background-color:"beige"; font-weight:bold; font-size:10pt;}
</STYLE>
<SCRIPT>
    function StyleWechsel()
    {
        // Input-Objekt
        event.srcElement.className="accessible"; // Input-Objekt

        // Label-Objekt
        var ZeigerAuf Label =eval(event.srcElement.id + "_Label ");
        // ID ist "ID_Input"
        //      also "ID_Input" + "_Label" = "ID_Input_Label"
        //      Zeichenkettenoperation leider nötig, wenn mehrere
        //      Objekte mit Eventerzeugung vorhanden wären
        //      und ebenfalls nötig wegen Bezug im Label auf das
        //      ID des Objektes, das Label haben soll
        ZeigerAuf Label.className="accessible";
    }
</SCRIPT>
<LABEL FOR="ID_Input" oInput" CLASS="normal" ID="ID_Input_Label">Text eingeben</LABEL>
<INPUT TYPE="text" CLASS="normal" onfocus="StyleWechsel()" ID="ID_Input">
```

Beispiel 3:

```
<HTML>
<HEAD>
<SCRIPT>
    function HandlerFuerOnMoveStart()
    {
        // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
        var ZeigerAufObjektMitEvent = event.srcElement;

        ZeigerAufObjektMitEvent.style.backgroundColor = "green";
        ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
    }
</SCRIPT>
```



```

function HandlerFuerOnMove ()
{
    ID_Span1.innerHTML = event.srcElement.offsetLeft;
    ID_Span2.innerHTML = event.srcElement.offsetTop;
}

function HandlerFuerOnMoveEnd()
{
    // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
    var ZeigerAufObjektMitEvent = event.srcElement;

    ZeigerAufObjektMitEvent.style.backgroundColor = "red";
    ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
}

// 2-D Positionierung einschalten
document.execCommand("2D-position",false,true);
</SCRIPT>
</HEAD>
<BODY onmovestart="HandlerFuerOnMoveStart();"
onmove="HandlerFuerOnMove();"
onmoveend="HandlerFuerOnMoveEnd();"
>
    offsetLeft = <SPAN ID="ID_Span1"></SPAN>
    <BR>
    offsetTop = <SPAN ID="ID_Span2"></SPAN>
    <BR>
    <DIV CONTENTEDITABLE="true">
        <DIV STYLE=
            "position:absolute;width:300px;height:100px; background-color:red;"
        >
            bewegbarer DIV
        </DIV>
    </DIV>
</BODY>
</HTML>

```

.standby                    Zeichenkette für Standby-Funktionalität des per OBJECT-Tag eingebundenen Objektes

.start                    Start eines Videoclips oder VRML-Datei

.state                    Status der Timeline – Variante 1  
 siehe Objekt currTimeState und Behavior .style.time2  
 0 Timeline inaktiv  
 1 Timeline aktiv  
 2 Timeline verarbeitet media file  
     Element ist aktiv  
     nur für Element das media file verarbeitet  
 3 Timeline sucht einen Punkt im media file  
     Element ist aktiv  
     nur für Element das media file verarbeitet  
 4 Timeline ist holding für das aktuelle Element und Element wartet auf Ende der Timeline  
     der Eltern  
     Element ist nicht aktiv  
 Hinweis zu media file: Audio, Video  
     zu Attribut FILL: wenn "hold" oder freeze" dann wartet Element auf  
     die Synchronisation mit der  
     Timeline des Elternobjektes  
     wenn "hold"    Element    kann keine Events verarbeiten  
     ist inaktiv  
     wenn "freeze"   Element ist aktiv

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Anzeige()
    {
        switch (ID_Video.currTimeState.state)
        {
            case 1:

```



```

        if (ID_Video.currTimeState.isPaused == true)
        {
            ID_Button2.disabled = true;
            ID_Button3.disabled = true;
            ID_Button4.disabled = false;
            ID_Button5.disabled = false;
        }
        else
        {
            ID_Button2.disabled = true;
            ID_Button3.disabled = false;
            ID_Button4.disabled = true;
            ID_Button5.disabled = false;
        }
        break;
    case 0:
        ID_Button2.disabled = false;
        ID_Button3.disabled = true;
        ID_Button4.disabled = true;
        ID_Button5.disabled = true;
        break;
    case 4:
        ID_Button2.disabled = false;
        ID_Button3.disabled = true;
        ID_Button4.disabled = true;
        ID_Button5.disabled = true;
        break;
    }
}
</SCRIPT>
<SCRIPT FOR="document" EVENT="onclick" LANGUAGE="JScript">
    Anzeige();
</SCRIPT>
</HEAD>
<BODY onload=" Anzeige()">
    <t:VIDEO ID="ID_Video"
        CLASS="time_line_klasse"
        BEGIN="indefinite"
        STYLE="position:absolute;top:90px;height:150px;"
        FILL="remove"
        SRC="/test /media/movie.avi"
    >
</ t:VIDEO>
<SPAN ID="ID_Span"
    STYLE="position:absolute;top:255px;"
>
    Status: 0
</SPAN>
<P STYLE="position:absolute;top:280px;">
    <BUTTON ID="ID_Button1"
        onclick=
            "ID_Span.innerText='Status: ' + ID_Video.currTimeState.state"
    >
        Aktueller Status
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Video.beginElement();"
    >
        Beginn
    </BUTTON>
    <BUTTON ID="ID_Button3"
        onclick="ID_Video.pauseElement();"
    >
        Pause
    </BUTTON>
    <BUTTON ID="ID_Button4"
        onclick="ID_Video.resumeElement();"
    >
        Resume
    </BUTTON>
    <BUTTON ID="ID_Button5"
        onclick=
            "ID_Video.endElement();ID_Span.innerText='Status: 0'"
    >

```



```

                Stop
            </BUTTON>
        </P>
    </BODY>
</HTML>

```

.stateString                      Status der Timeline – Variante 2  
                                   siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Anzeige()
    {
        switch (ID_Video.currTimeState.stateString)
        {
            case "active":
                if (ID_Video.currTimeState.isPaused == true)
                {
                    ID_Button2.disabled = true;
                    ID_Button3.disabled = true;
                    ID_Button4.disabled = false;
                    ID_Button5.disabled = false;
                }
                else
                {
                    ID_Button2.disabled = true;
                    ID_Button3.disabled = false;
                    ID_Button4.disabled = true;
                    ID_Button5.disabled = false;
                }
                break;

            case "inactive":
                ID_Button2.disabled = false;
                ID_Button3.disabled = true;
                ID_Button4.disabled = true;
                ID_Button5.disabled = true;
                break;

            case "holding":
                ID_Button2.disabled = false;
                ID_Button3.disabled = true;
                ID_Button4.disabled = true;
                ID_Button5.disabled = true;
                break;
        }
    }
</SCRIPT>
<SCRIPT FOR="document" EVENT="onclick" LANGUAGE="JScript">
    Anzeige();
</SCRIPT>
</HEAD>
<BODY onload=" Anzeige()">
    <t:VIDEO ID="ID_Video"
        CLASS="time_line_klasse"
        SRC="test.avi"
        BEGIN="indefinite"
        FILL="remove"
        STYLE="position:absolute;top:90px;height:150px;"
    >
    </ t:VIDEO>
    <SPAN ID="ID_Span"
        STYLE="position:absolute;top:255px;"
    >
        Status: 0
    </SPAN>
    <P STYLE="position:absolute;top:280px;">
        <BUTTON ID="ID_Button1"
            onclick=

```



```

        "ID_Span.innerText='Status: ' + ID_Video.currTimeState.state"
    >
        Aktueller Status
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Video.beginElement();"
    >
        Beginn
    </BUTTON>
    <BUTTON ID="ID_Button3"
        onclick="ID_Video.pauseElement();"
    >
        Pause
    </BUTTON>
    <BUTTON ID="ID_Button4"
        onclick="ID_Video.resumeElement();"
    >
        Resume
    </BUTTON>
    <BUTTON ID="ID_Button5"
        onclick=
            "ID_Video.endElement();ID_Span.innerText='Status: 0'"
    >
        Stop
    </BUTTON>
</P>
</BODY>
</HTML>

```

**.status** Selektionsstatus eines Control-Elementes  
 Control-Element: kann durch User interaktiv verändert werden  
 wird bei Ceckbox-Control auch verändert durch Eigenschaft **.indeterminate**  
 bei Statusveränderung wird Event **onpropertychange** erzeugt  
**false** Default außer bei textArea Objekt  
           Control ist nicht selektiert  
**true** Control ist selektiert  
**null** Control ist nicht initialisiert  
           Default bei textArea Objekt

Beispiel:

```

<INPUT TYPE=checkbox
        ID="ID_InputCheckbox"
        CHECKED
        DISABLED
    >
    <SPAN STYLE="font-weight:bold"
        onclick="ID_InputCheckbox.status=false"
    >
        ablehnen
    </SPAN>
    <SPAN STYLE="font-weight:bold"
        onclick="ID_InputCheckbox.status=true"
    >
        annehmen
    </SPAN>

```

**.status** enthält aktuellen Text der Statuszeile des Fensters (nicht den Standardtext)  
 siehe **.defaultStatus**  
 Verwendung in Eventhandler-Funktion: Es muss **return true**; kodiert werden.  
 siehe Objekt **window**

Beispiel für Text buchstabenweise in Statuszeile anzeigen:

Der aktuelle Statuszeilentext wird als Teilkette von Position 0 bis **zeichen\_nr** angezeigt, wobei **zeichen\_nr** pro Anzeige um 1 erhöht wird.

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var statuszeile    = new Array();
    statuszeile[0]     = "Text0";
    statuszeile[1]     = "Text1";
    statuszeile[2]     = "Text2";

```





```

var statuszeile_nr = 0;
var zeichen_nr = 0;

function textanzeigen ()
{
    if (position < statuszeile[statuszeile_nr].length) // Länge ab 1
    {
        // nächste Teilkette des aktuellen Statuszeilentextes anzeigen
        window.status = statuszeile[statuszeile_nr].substring(0, zeichen_nr);

        // nächste Position
        position++;
        setTimeout("textanzeigen()", 100);
    }
    else
    {
        // aktuelle Statuszeilentext komplett anzeigen
        window.status = statuszeile[statuszeile_nr];

        // nächste Statuszeile adressieren
        statuszeile_nr++;
        if (statuszeile_nr >= statuszeile.length)
        { statuszeile_nr = 0; }

        // Position 0 einstellen
        zeichen_nr = 0;

        // Start der buchstabenweise Anzeige
        setTimeout("textanzeigen()", 1000);
    }
}

// -->
</SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>

```

Beispiel für automatisch wechselnder Text in der Statuszeile:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var statuszeile = new Array();
    statuszeile[0] = "Text0";
    statuszeile[1] = "Text1";
    statuszeile[2] = "Text2";

    var statuszeile_nr = 0;

    function textanzeigen()
    {
        window.status = statuszeile[statuszeile_nr];
        statuszeile_nr++;
        if (statuszeile_nr >= statuszeile.length) {statuszeile_nr = 0;}
        setTimeout("textanzeigen()", 1000);
    }

// -->
</SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>

```

Beispiel für dauerhaftes Scrollen eines Textes in der Statuszeile:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var scrolltext_gesamt = "Dieser Text scrollt!";
    var scrolltext = "";
    var zahler = scrolltext_gesamt.length;

```



```

function scrollen()
{
    if (zahler == scrolltext_gesamt.length)
    {
        zahler = 0;
        scrolltext = scrolltext_gesamt;
    }
    else
    {
        zahler++; // 1 bis scrolltext_gesamt.length
        // Blank vorsetzen, also Kette um 1 Zeichen verlängern
        scrolltext = " "+scrolltext;

        // rausgerutschtes Zeichen abschneiden
        scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
        // Angaben ab 0
    }

    window.status = scrolltext;

    setTimeout("scrollen()", 100);
}
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="scrollen()">
</BODY>
</HTML>

```

Beispiel für einen Scrolltext in der Statusleiste, der angehalten wird, wenn Mauszeiger sich über einen Linkt bewegt:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var scrolltext_gesamt = "Dieser Text scrollt!";
    var scrolltext = "";
    var zahler = scrolltext_gesamt.length;
    var id;

    function scrollen()
    {
        if (zahler >= scrolltext_gesamt.length)
        {
            zahler = 0;
            scrolltext = scrolltext_gesamt;
        }
        else
        {
            zahler++; // 1 bis scrolltext_gesamt.length
            // Blank vorsetzen, also Kette um 1 Zeichen verlängern
            scrolltext = " "+scrolltext;

            // rausgerutschtes Zeichen abschneiden
            scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
            // Angaben ab 0
        }

        window.status = scrolltext;
        id=setTimeout("scrollen()", 100);
    }

    function scrollen_stoppen()
    {clearTimeout(id);}

// -->
</SCRIPT>
</HEAD>

<BODY onLoad="scrollen()">
    Bewegen Sie den Mauspfel &uuml;ber diesen
    <A      HREF="http://www.test.de"
        onMouseOver="scrollen_stoppen()"

```



```

        onMouseOut="scrollen()"
    >
    Link
</A>
</BODY>
</HTML>

```

Beispiel für Anzeige eines Textes zu einem Hyperlink (HREF) für eine feste Zeitspanne in der Statuszeile:

```

<HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var id;
    var anzeige_aktiv = false;
    var anzeige_zeit = 3000;      // 3000 Millisekunden = 3 Sekunden

    function anzeige_starten(href_text)
    {
        if(anzeige_aktiv)
        {clearTimeout(id);}

        window.status = href_text;

        id = setTimeout("anzeige_stoppen()", anzeige_zeit);

        anzeige_aktiv = true;

        return true;           // Wichtig !!!
    }

    function anzeige_stoppen ()
    {
        anzeige_aktiv = false;

        window.status = "";
    }
-->
</SCRIPT>
</HEAD>
<BODY>
    <A HREF= ... onMouseOver="javascript:return anzeige_starten('Hinweistext...')">
    ...
    </A>
</BODY>
</HTML>

```

Beispiel zur Anzeige eines Formularfeld-Hinweises in der Statuszeile:

```

<HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
    function hinweis_anzeigen(hinweis_text)
    {window.status = hinweis_text;}

    function hinweis_loeschen()
    {window.status = "";}
-->
</SCRIPT>
</HEAD>
<BODY>
    <FORM>
        <INPUT TYPE=TEXT
            onFocus="hinweis_anzeigen('Hinweis');"
            onBlur="hinweis_loeschen();"
        >
    </FORM>
</BODY>
</HTML>

```

Beispiel für blinkende Anzeige mit Zeitspanne von Text in der Statuszeile:

```

<HTML>

```



```

</HEAD>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
    var zeitspanne = 6000;          // 6 Sekunden blinken lassen, danach
                                   //      Statuszeile löschen
    var anzeigezeit = 500;         // 0,5 Sekunden warten nach Setzen
                                   //      bzw. Löschen der Statuszeile
    var id_blinken = null;         // muss auf null initialisiert sein !!

    function blinken_timer_loeschen
    {
        if (id_blinken != null)
        {
            clearTimeout(id_blinken); // blinken stoppen falls aktiv
            id_blinken = null;
        }
    }

    function blinken_stop()
    {
        blinken_timer_loeschen;
        window.status="";
    }

    function blinken_start(status_text)
    {
        blinken_timer_loeschen;
        blinken(true, status_text);          // Blinken anstossen für
                                              // paralleles Arbeiten per Timer
        setTimeout("blinken_stop()",zeitspanne); // warten und danach blinken stoppen
    }

    function blinken(ein_aus, text)
    {
        if (ein_aus)
        {
            window.status = text;
            ein_aus=false;    // im nächsten Aufruf die Statuszeile löschen
        }
        else
        {
            window.status = "";
            ein_aus=true;     // im nächsten Aufruf die Statuszeile setzen
        }

        id_blinken = setTimeout("blinken(" + ein_aus + ")", anzeigezeit)
                          // nächsten Aufruf nach Ablauf der anzeigezeit starten
    }
}
-->
</SCRIPT>
</HEAD>
<BODY ... onLoad="blinken_start('Ich blinke !')">
</BODY>
</HTML>

```

STYLE                      direkt im HTML-Element kodierter Style (Inline-Style)  
Hinweis: für Scripting ist das Style-Objekt zu nutzen

.SubFolders                Zeiger auf die interne Collection FileSystemObject.Folder.Folders  
Collection kann nur z.T. über das JScript-Objekt Enumerator angesprochen werden

.subtype                   visuelle Erscheinungsform des Überganges per .style.time2.transitionFilter Behavior-Objekt,  
also visuelle Art und Weise des Überganges  
optional kodierbar zur Eigenschaft .type des Behavior (bis auf 1 Ausnahme --> siehe unten)  
siehe Objekt currTimeState und Behavior .style.time2  
Übersicht zu Wert von .subtype bei gegebenen Wert von .type

<u>Wert</u>	<u>Eigenschaft .type</u>	<u>Wert</u>	<u>Eigenschaft .subtype</u>
"starWipe"		"fivePoint"	
			<b>muss immer kodiert sein</b>
			<b>wenn .type kodiert wurde</b>
"barWipe"		"leftToRight"	
		oder	"topToBottom"
"barnDoorWipe"		"vertical"	
		oder	"horizontal"



"irisWipe"	oder	"rectangle" "diamond"
"ellipseWipe"		"circle"
"clockWipe"		"clockwiseTwelve"
"fanWipe"		"centerTop"
"snakeWipe"		"topLeftHorizontal"
"spiralWipe"		"topLeftClockwise"
"pushWipe"		"fromLeft"
"slideWipe"		"fromLeft"
"fade"		"crossfade"

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter1"
TYPE="barWipe"
SUBTYPE="leftToRight"
DUR="3"
TARGETELEMENT="ID_Div1"
>
</t:TRANSITIONFILTER>

<t:TRANSITIONFILTER ID="ID_Transfilter2"
TYPE="starWipe"
SUBTYPE="fivePoint"
DUR="3"
TARGETELEMENT="ID_Div1"
>
</t:TRANSITIONFILTER>

<t:TRANSITIONFILTER ID="ID_Transfilter3"
TYPE="barnDoorWipe"
DUR="3"
TARGETELEMENT="ID_Div2"
FROM="0"
TO="1"
CALCMODE="linear"
MODE="in"
>
</t:TRANSITIONFILTER>

<DIV STYLE="height:170px;">

<DIV ID="ID_Div1"
CLASS="time_line_klasse"
STYLE="position:absolute; top:150px; left:20px; background-color:#3366CC;
padding:10px; height:80; color:white;
"
>
Test1
</DIV>

<DIV ID="ID_Div2"
CLASS="time_line_klasse"
STYLE="position:absolute; top:185px; left:60px; background-color:#FFCC00;
padding:10px; height:80;
"
>
Test2
</DIV>
</DIV>
</BODY>
</HTML>
```

.summary

Kommentar in einer Tabelle, der nicht gerendert wird  
siehe Objekt table

Beispiel:



```

<TABLE ID="ID_Tabelle" BORDER CELSPACING=10 SUMMARY="Kommentar">
  <TR>
    <TD>Zelle 1</TD>
    <TD>Zelle 2</TD>
  </TR>
</TABLE>

```

**.syncBehavior** Synchronisation der Timeline eines Elementes mit der Timeline des Elternelementes, sinnvoll vor allem bei Time-Container (Gruppen von Objekten)

Es muss die Eigenschaft **.syncmaster** kodiert sein. Ob das Element diese Eigenschaft unterstützt, ist der Beschreibung zum jeweiligen Objekt zu entnehmen.

siehe Objekt **currTimeState** und Behavior **.style.time2**

siehe **.syncTolerance** und **.syncmaster**

**"canSlip"** keine Synchronisation

sinnvoll bei Problemen der Geschwindigkeit des Internets

**"locked"** Element muss sich synchronisieren

sinnvoll wenn Kindelement erst animiert werden soll, wenn Elternelement komplett dazu in der Lage ist: Es wird automatisch pausiert, bis Synchronisation erreicht ist.

Beispiel für Wiedergabe per Windows Media Player:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY ID="ID_Body">
  <t:PAR ID="ID_Par"
    SYNCBEHAVIOR="locked"
  >
    <t:MEDIA ID="ID_Media"
      SRC="test.mid"
      PLAYER="{6BF52A52-394A-11d3-B153-00C04F79FAA6}"
      SYNCMASTER="true"
      SYNCBEHAVIOR="locked"
      BEGIN="1"
    >
    </t:MEDIA>
    <t:ANIMATEMOTION ID="ID_Animatemotion"
      BEGIN="ID_Media.begin"
      DUR="9"
      TARGETELEMENT="ID_Div1"
      PATH="M 0 0 L 100 300"
      FILL="freeze"
    >
    </t:ANIMATEMOTION>
    <t:SEQ ID="ID_Seq"
      STYLE="font-size:18pt;color:#ff0000"
      SYNCBEHAVIOR="locked"
      BEGIN="ID_Media.begin"
      FILL="freeze"
    >
      <DIV ID="ID_Div1"
        CLASS="time_line_klasse"
        DUR="1.5"
      >
        Diese MIDI-File
      </DIV>
      <DIV ID="ID_Div2"
        CLASS="time_line_klasse"
        DUR="1.5"
      >
        wird mit dem
      </DIV>
      <DIV ID="ID_Div3"
        CLASS="time_line_klasse"
        DUR="2"
      >
        Windows Media-Player
      </DIV>
      <DIV ID="ID_Div4"

```



```

        CLASS="time_line_klasse"
        DUR="2"
    >
        abgespielt.
    </DIV>
</t:SEQ>
</t:PAR>
<DIV ID="ID_Div5"
    CLASS="time_line_klasse"
    STYLE="background-color:#FFCC00;position:absolute;height:40;width:120;
        top:70;left:10;font-size:18;border-style:solid
    ">
    >
        animierter DIV waehrend der Wiedergabe der MIDI-File
    </DIV>
</BODY>
</HTML>

```

.syncMaster      Synchronisierung der Animation des im Container liegenden Elementes auf Timeline  
 (Container ist Master in der Synchronisierung), wobei ein Master **nur** genau  
 ein zu synchronisierendes Element haben darf (Eineindeutigkeit).  
 nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior)  
 ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf!  
 siehe Objekt currTimeState und Behavior .style.time2  
 siehe .syncTolerance und .syncBehavior  
 false      Default  
             Container synchronisiert nicht  
 true      Container muss synchronisieren

Beispiel für Wiedergabe per Windows Media Player:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY ID="ID_Body">
    <t:PAR ID="ID_Par"
        SYNCBEHAVIOR="locked"
    >
        <t:MEDIA ID="ID_Media"
            SRC="test.mid"
            PLAYER="{6BF52A52-394A-11d3-B153-00C04F79FAA6}"
            SYNCMASTER="true"
            SYNCBEHAVIOR="locked"
            BEGIN="1"
        >
        </t:MEDIA>
        <t:ANIMATEMOTION ID="ID_Animatemotion"
            BEGIN="ID_Media.begin"
            DUR="9"
            TARGETELEMENT="ID_Div1"
            PATH="M 0 0 L 100 300"
            FILL="freeze"
        >
        </t:ANIMATEMOTION>
        <t:SEQ ID="ID_Seq"
            STYLE="font-size:18pt;color:#ff0000"
            SYNCBEHAVIOR="locked"
            BEGIN="ID_Media.begin"
            FILL="freeze"
        >
            <DIV ID="ID_Div1"
                CLASS="time_line_klasse"
                DUR="1.5"
            >
                Diese MIDI-File
            </DIV>
            <DIV ID="ID_Div2"
                CLASS="time_line_klasse"
                DUR="1.5"
            >

```



```

        wird mit dem
    </DIV>
    <DIV ID="ID_Div3"
        CLASS="time_line_klasse"
        DUR="2"
    >
        Windows Media-Player
    </DIV>
    <DIV ID="ID_Div4"
        CLASS="time_line_klasse"
        DUR="2"
    >
        abgespielt.
    </DIV>
</t:SEQ>
</t:PAR>
<DIV ID="ID_Div5"
    CLASS="time_line_klasse"
    STYLE="background-color:#FFCC00;position:absolute;height:40;width:120;
        top:70;left:10;font-size:18;border-style:solid
    "
>
    animierter DIV waehrend der Wiedergabe der MIDI-File
</DIV>
</BODY>
</HTML>

```

**.syncTolerance** zeitliche Toleranz in Sekunden für Zwangs-Synchronisation von Elementen auf der Timeline  
 also wenn Eigenschaft **.syncBehavior** auf "locked" gesetzt ist  
 sinnvoll vor allem bei Time-Container (Gruppen von Objekten)  
 siehe Objekt **currTimeState** und Behavior **.style.time2**  
 siehe **.syncBehavior** und **.syncMaster**  
 Wert im Time-Format des Behavior  
 z.B. "h:min:s.f"  
**nicht** id.event  
**nicht** id.event+zeit\_wert\_als\_string

Beispiele:

"25:45:10"	25 Stunden, 45 Minuten, 10 Sekunden
"45:35"	45 Minuten, 35 Sekunden
"45:00.275"	45 Minuten, 0,275 Sekunde
"10.5"	10,5 Sekunden

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Anzeige()
    { alert(ID_Media.syncTolerance; }
</SCRIPT>
</HEAD>
<BODY>
    <t:PAR ID="ID_Par"
        TIMEACTION="display"
    >
        <t:MEDIA ID="ID_Media"
            SCR="test.wmv"
            BEGIN="0"
            DUR="20s"
            TIMEACTION="display"
            SYNCMASTER="true"
            SYNCBEHAVIOR="locked"
            onmediacomplete="Anzeige()"
        >
        </t:MEDIA>
    </t:PAR>
</BODY>

```

**.systemBitrate** wird hier nicht erklärt

**.systemCaptions** wird hier nicht erklärt





`.systemLanguage` Sprache festlegen für das Objekt

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:SWITCH ID="ID_Switch">
<SPAN systemLanguage="es">Spanisch</SPAN>
<SPAN systemLanguage="pt">Portugiesisch</SPAN>
<SPAN systemLanguage="en">Englisch</SPAN>
</t:SWITCH>
<t:AUDIO ID="ID_Audio"
SRC="fun.wav"
SYSTEMLANGUAGE="mi, en"
>
</ t:AUDIO>
</BODY>
</HTML>
```

`.systemLanguage` Standard-Sprache des Betriebssystems (Sprache der Installation) per navigator Objekt  
navigator.systemLanguage  
z.B. "en" oder "de"

`.systemLanguage` Standardsprache des Betriebssystems  
Behavior .style.clientCaps  
z.B. "en" oder "de"

`.systemLanguage` Standardsprache des Betriebssystems  
siehe Objekt window.clientInformation  
z.B. "en" oder "de"

`.systemOverdubOrSubtitle` wird hier nicht erklärt

`.tabIndex` Index des Elementes in der Tab-Tasten-Folge  
für Anspringen des Dokumentes  
Anspringen verbunden mit Focus erhalten  
--> Ereignisse werden ausgelöst !!  
unter IE 5.x  
onblur, onfocus  
ab IE 5.x  
onblur, onfocus, onkeydown, onkeypress, onkeyup

Anspringen default per TAB-Taste  
für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT,  
SELECT, TEXTAREA

Anspringen default nicht per TAB-Taste  
für APPLET, DIV, FRAMESET, SPAN, TABLE, TD

Beispiele:

```
<INPUT TYPE="text" TABINDEX="1">
<INPUT TYPE="text" > Deafult ist 0
<INPUT TYPE="text" TABINDEX="2">
<INPUT TYPE="submit" TABINDEX="-1">
<UL>
<LI TABINDEX="1">Tab Item 1</LI>
<LI TABINDEX="2">Tab Item 2</LI>
<LI TABINDEX="3">Tab Item 3</LI>
<LI TABINDEX="4">Tab Item 4</LI>
<LI TABINDEX="5">Tab Item 5</LI>
</UL>
```

`.tagName` Tag-Bezeichner des Objektes

Beispiel:

```
<SCRIPT>
var ID_Wert = window.prompt("Bitte ID eines Tags eingeben:");
if (ID_Wert != null)
{alert(document.all[ID_Wert].tagName) }
</SCRIPT>
```



.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut

Beispiel:

```
<HTML XMLNS:InetSDK='http://www.test.de'>
<STYLE>
    @media all {InetSDK:HalloDu { behavior:url (simple.htc) }}
</STYLE>
<SCRIPT>
    function window.onload()      // überschreibt Standard window.onload-Routien !!!!
    {
        // Statuszeile als Ausgabebereich nutzen
        window.status = 'scopeName = ' + Hallo.scopeName + ' ';
        + ' tagUrn = ' + Hallo.tagUrn;
    }
</SCRIPT>
<BODY>
    <InetSDK:HelloWorld ID='Hallo'></InetSDK:HalloDu>
</BODY>
</HTML>
```

.target

Name des Ziel-Fenster bzw. Ziel-Frame

Name des Window oder Frame

bei Window laut Methode .open()

vordefinierte Namen sind:

"_blank"	neues leeres Fenster
	Fenster hat keinen Namen
"_media"	Media Bar
	ab IE 6.x
"_parent"	Elternfenster bzw. Elternframe
"_search"	Suchfenster
	ab IE 5.x
"_self"	Default
	aktuelles Fenster bzw. Frame
"_top"	oberste Fenster bzw. Frame

Wenn Zielfenster bzw. Zielframe nicht existent, soneues Fenster eröffnet mit dem Namen laut Eigenschaft .target

Beispiel:

```
<A HREF="test.htm" TARGET="_top">im obersten Fenster anzeigen</A>
```

.targetElement

ID des zu animierenden Elementes auf der Timeline

muss für Kindelement **immer** kodiert werden, wenn nicht das Elternelement animiert werden soll  
muss nicht kodiert werden, wenn kein Elternelement vorliegt

Achtung: Objekt body ist das oberste Elternelement im Dokument  
und somit haben Elemente innerhalb BODY immer Eltern

Empfehlung: immer kodieren

ist der Bezug des Time-Container auf das zu animierende Element

siehe Objekt currTimeState und Behavior .style.time2

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft .width) nach rechts:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function SpanInhaltInit(Kette)
    {
        // Zustand
        ID_Span1.innerText = "";

        // Kumulationsart
        ID_Span2.innerText =Kette;

        // Zaehler auf 1
        ID_Span3.innerText ="1";

        // DIV-Text festlegen
        ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
        if (Kette == "sum")
        {
            // Wertkumulation von .width
            ID_Div.innerText += " genau 1x kumulativ um ";
            ID_Div.innerText += ID_Animate.by
```



```

        ID_Div.innerText += " mal "
        ID_Div.innerText += ID_Animate.repeatCount;
    }
    else
    {
        // keine Wertkumulation von .width
        ID_Div.innerText += ID_Animate.repeatCount;
        ID_Div.innerText += " mal um ";
        ID_Div.innerText += ID_Animate.by
    }

    ID_Div.innerText += " aus";
}

function KumulationAus()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("none");
    ID_Animate.accumulate="none";

    // DANACH Animation starten
    ID_Animate.beginElement();

}

function KumulationEin()
{
    // Element darf nicht aktiv sein !!

    // ERST .accumulate definieren
    SpanInhaltInit("sum");
    ID_Animate.accumulate="sum";

    // DANACH Animation starten
    ID_Animate.beginElement();

}

function Anzeige()
{
    ID_Span1.innerText = "Animation ist beendet ";
}
</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE
        ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    <t:ANIMATE>

animierter DIV
<DIV
    ID="ID_Div"
    CLASS="time_line_klasse"
    STYLE= "position:absolute;
            top:125px;
            left:25px;
            height:100px;
            width:125px;
            border:solid black 2px;
            "
    >
</DIV>
<BR>

```

Zustand der Animation:



```
<SPAN ID="ID_Span1"></SPAN>
<BR>
```

Kumulationsart:

```
<SPAN ID="ID_Span2"></SPAN>
<BR>
```

Durchlaufzaehler:

```
<SPAN ID="ID_Span3"></SPAN>
<BR>
```

```
<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
```

```
</BODY>
</HTML>
```

.text                      Textfarbe (Vordergrundfarbe) des Dokumentes  
                               "#rrggbb"  
                               vordefinierte Farbname (browserspezifisch)

Beispiel:

```
<BODY ID="ID_Body" >
<BUTTON onmouseover=" ID_Body.text='green'">GREEN</BUTTON>
```

.text                      Text eines Objektes  
                               nur Plaintext

.text                      Plain-Text im Textbereich  
                               per textrange Objekt  
                               Dokument muss komplett geladen sein  
                               nur unter Windows 32-Bit

.text                      interner Kettenwert einer Option, also Objektes document.select.option des Objektes document.select  
                               ändert nicht den angezeigten Text hinter <OPTION ...>  
                               ist nur ein interner Wert und beim Formular der gesendete Wert der Option  
                               Plain-Text

Beispiel:

```
<SCRIPT>
function Erweitern()
{
    for ( var i=0; i < ID_Select.options.length; i++)
    { ID_Select.options[i].text+=" ist eine Stadt"; }
}

function Anzeigen()
{
    var Kette="";

    for (var i=0; i < ID_Select.options.length;i++)
    {
        Kette += i
                + "angezeigter Wert ="      + ID_Select.options[i].value
                + " zu sendender Wert ="    + ID_Select.options[i].text
                + "\n";
    }

    alert(Kette);
}
</SCRIPT>
<SELECT ID="ID_Select" onchange="Anzeigen()">
  <OPTION VALUE="DE">Berlin
  <OPTION VALUE="F">Paris
  <OPTION VALUE="GB">London
</SELECT>
<INPUT TYPE=button VALUE="Alles anzeigen" onclick="Erweitern()">
```

.tfoot                      Zeiger auf das Objekt table.tFoot

.thead                      Zeiger auf das Objekt table.tHead

.timeAction                  Aktion des Objektes in der Timeline  
                               Aktion nur ausführbar wenn Objekt aktiv ist UND Timeline aktiv ist  
                               siehe Objekt currTimeState und Behavior .style.time2  
                               ab IE 5.5                      "class:classname1 [classname2 ...]"  
    Liste mit Blanktrennung



	nur wenn Objekt aktiv, so Klassen verwendet
"display"	nur wenn Objekt aktiv, so sichtbar inaktives Objekt wird aus dem Layout entfernt Umfluss verändert sich somit Default nur für t:SEQ
"style"	nur wenn Objekt aktiv, so in Inline-Style verwendet (falls kodiert) Inline-Style per STYLE-Attribut
"visibility"	nur wenn Objekt aktiv, so sichtbar inaktives Objekt bleibt im Layout Umfluss wird nicht verändert Default außer für t:SEQ
"none"	entspricht nicht kodierte Eigenschaft

Beispiel 1:

```
<SPAN CLASS="redText boldFont"
STYLE="behavior:url(#default#time2)"
BEGIN="5"
TIMEACTION="class:redText"
>
  Dieser Text ist rot, solange Element aktiv ist
</SPAN>
```

Beispiel 2:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<P ID="ID_P1"
CLASS="time-line_klasse"
BEGIN="2"
DUR="5"
>
  Test
</P>
<P ID="ID_P2"
CLASS="time_line_klasse"
BEGIN="3"
DUR="6"
TIMEACTION="display"
>
  Test
</P>
<P ID="ID_P3"
CLASS="time_line_klasse"
BEGIN="4" DUR="7"
TIMEACTION="visibility"
>
  Test
</P>
<P ID="ID_P4"
CLASS="time_line_klasse"
BEGIN="1"
DUR="3"
TIMEACTION="hidden"
>
  Test
</P>
<P ID="ID_P5"
CLASS="time_line_klasse"
BEGIN="1"
DUR="3"
TIMEACTION="none"
>
  Test
</P>
<H1 ID="ID_H1"
CLASS="time_line_klasse"
BEGIN="0"
DUR="11"
TIMEACTION="style"
STYLE="Color:Red;"
```



```

>
    Test
</H1>
</BODY>
</HTML>

```

**.timeAll** Zeiger auf die Collection document.body.timeAll  
Feld aller Zeiger per Behavior .style.time2 verwalteter Elemente der Webseite (getimte Elemente)  
siehe Objekt currTimeState und Behavior .style.time2

**.timeContainer** Typ der Timeline des Objektes  
siehe Objekt currTimeState und Behavior .style.time2  
"excl" genau ein Objekt kann agieren  
"none" Default  
entspricht nicht kodierter Eigenschaft  
"par" alle Objekte agieren parallel  
"seq" alle Objekte agieren in 1 sequentieller Folge

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <MARQUEE          ID="ID_Marquee"
                      CLASS="time_line_klasse"
                      TIMECONTAINER="seq"
                      REPEATCOUNT="indefinite"
    >
        <IMG ID="ID_Img1" CLASS="time_line_klasse" DUR="4" SRC="test1.gif" ALT="Test1">
        <IMG ID="ID_Img2" CLASS="time_line_klasse" DUR="4" SRC="test2.gif" ALT="Test2">
        <IMG ID="ID_Img3" CLASS="time_line_klasse" DUR="4" SRC="test3.gif" ALT="Test3">
    </MARQUEE>
</BODY>
</HTML>

```

**.timeParent** Zeiger auf das Eltern-Timeline  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    window.onload = Rekursion;    // ohne () kodieren !!

    var ZeigerAufElternTimeline = ID_Animate.timeParent;

    function Rekursion()
    {window.setInterval(Aktualisieren,100); }

    function Aktualisieren()
    {
        ID_Div.innerText = ZeigerAufElternTimeline.dur;
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV          ID="ID_Div"
                  CLASS="time_line_klasse"
                  STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
    >
    </DIV>
    <t:EXCL ID="ID_Excl"
           CLASS="time_line_klasse"
           BEGIN="0"
           DUR="indefinite"
    >
        <t:ANIMATEMOTION          ID="ID_Animate"
                                TARGETELEMENT="ID_Div"

```



```

TO="375,0"
BEGIN="0"
DUR="3"
AUTOREVERSE="true"
REPEATCOUNT="indefinite"
>
</t:ANIMATEMOTION>
</t:EXCL>
<BUTTON ID="ID_Button" onclick="ZeigerAufElternTimeline.DUR='30s';">
    Dauer der Eltern-Timeline
</BUTTON>
<BODY>
</HTML>

```

.timeStartRule deprecated  
 Startpunkt der Timeline  
 "onDocLoad" Default  
 Timeline beginnt nach dem kompletten Laden des Dokumentes

.title Fenstertitel des Dokumentes  
 siehe Objekt document

Beispiel für permanenten Fenstertitel-Wechsel:

```

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    var titel_texte_feld = [
        "Titel 1",
        "Titel 2",
        "Titel 3",
        "Titel 4"
    ];

    var wechsel_tempo = 2500;    // Zeit in ms zwischen zwei Schritten
    var zahler = 0;
    var id;

    function start()
    {
        titel_texte_feld[titel_texte_feld.length] = document.title;
        // beim ersten Aufruf wird der Original-Titel mit gespeichert
        // Länge ab 1, aber Index ab 0
        // Länge == Index +1, an desssen Position der Original-Titel abgelegt wird

        window.setTimeout("wechseln()", wechsel_tempo);
    }

    function wechseln()
    {
        document.title = titel_texte_feld [zahler];
        zahler ++;

        if(zahler >= texte.length)
        {zahler = 0}

        id = window.setTimeout("wechseln()", wechsel_tempo);
    }
}
-->
</SCRIPT>

```

Beispiel für Fenstertitelzeile mit scrollendem Text:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
//
//      In der aktuellen Fenster-Titelzeile einen freien Text durch endloses Scrollen von links nach rechts vorsetzen
//
//*****
//
//      nachfolgende Variablen sind vom Programmierer zu setzen
//
//*****
var VorsetzText_Wert      = "Freien Text vorsetzen durch scrollen und warten ";

```



```

var ScrollGeschwindigkeit = "150"; // in Millisekunden, immer > 0, wird nicht geprüft
var DummyLeerzeichenAnzahl = 10; // immer > 0, wird nicht geprüft
// Die Leerzeichen werden nicht sichtbar angezeigt, aber gescrollt.
// So entsteht eine Wartezeit (pro Leerzeichen laut
// ScrollGeschwindigkeit)
// für die vollständige Anzeige von VorsetzText NACH dem
// vollständigem Vorsetzen per Scrollen von links nach
// rechts, ehe der Scroller wieder von vorn beginnt

//*****
//
// nachfolgender Code darf nicht verändert werden
// wird anstelle von onLoad innerhalb BODY verwendet
//
//*****
// rechtsbündige Auffüllung mit DummyLeerzeichen
for (i=0; i <=DummyLeerzeichenAnzahl; i++)
{VorsetzText_Wert+=" ";}

// Länge NACH Auffüllung ermitteln
var VorsetzText_Laenge=VorsetzText_Wert.length;

// globale Variablen für EndlosScrollen(), die dort laufend manipuliert werden,
// also hier initialisiert werden müssen
var Zahler="0";
var VorsetzText="";
//-->
</SCRIPT>
</HEAD>

<BODY>
<SCRIPT LANGUAGE="JavaScript1.2">

//*****
//
// nachfolgender Code darf nicht verändert werden
//
//*****

function EndlosScrollen()
{
    // endloses Scrollen

    if (
        (document.all)
        || (document.getElementById)
    )
    {
        // VorsetzText um nächsten Buchstaben aus dem VorsetzText_Wert erweitern
        // (einschliesslich DummyLeerzeichen)
        VorsetzText+=VorsetzText_Wert.charAt(Zahler); // charAt ab Null

        // Fenster-Titelzeile neu belegen
        document.title=VorsetzText;

        Zahler++;
        if (Zahler==VorsetzText_Laenge)
        {
            Zahler="0";
            VorsetzText="";
        }

        setTimeout("EndlosScrollen()",ScrollGeschwindigkeit);
    }
}

window.onload=EndlosScrollen; // ohne () kodieren, damit nicht sofort sondern beim Laden ausgeführt wird
//-->
</SCRIPT>
</BODY>
</HTML>

```

.title                      Tooltip-Text bei Mouse over über Objekt  
                              Plain-Text, max. 1024 Zeichen

Beispiel:





```

<SCRIPT>
    function TestSetzen(ObjektZeiger)
    {
        ObjektZeiger.title="Tooltip";
        return;
    }
</SCRIPT>
<SPAN onmouseover="TextSetzen(this)">Testtext</SPAN>

```

.title  
 Titel der Media-Datei auf der Timeline  
 bei Advanced Stream Redirector (ASX)-Datei: Es wird der Text von TITLE des aktiven  
 Eintrages geliefert und nicht den der Datei.  
 Track entspricht playItem Objekt laut object.playList.item() bzw. object.playList.aktiveTrack  
 Trackliste liegt in der Behavior-Collection .style.time2.playList als Playliste:  
 Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline  
 aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
 Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei

Beispiel 1:

```

var Kette = object.playList.item(index).title;

        index      Integer, ab 0

var Kette = object.playList.aktiveTrack.title;

```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:VIDEO          ID="ID_Video"
                     SRC="test.wmv"
                     STYLE="position:absolute;top:50px;height:100px"
    >
    </t:VIDEO>
    <SPAN ID="ID_Span"
        STYLE="position:absolute;top:165px;"
    >
    </SPAN>
    <BUTTON ID="ID_Button"
        onclick="ID_Span.innerText= ID_Video.title "
    >
        Klick
    </BUTTON>
</BODY>
</HTML>

```

Beispiel 3:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

```



```

    }
}

function AnzeigeUpdate()
{
    ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
    ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
    ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
    ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
    ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
    ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
    ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
    ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
}

function AnzeigeLoeschen()
{
    ID_Span1.innerText = "Titel: ";
    ID_Span2.innerText = "Autor: ";
    ID_Span3.innerText = "Abstract: ";
    ID_Span4.innerText = "Copyright: ";
    ID_Span5.innerText = "Filename: ";
    ID_Span6.innerText = "Banner: ";
    ID_Span7.innerText = "Banner Abstract: ";
    ID_Span8.innerText = "Banner MoreInfo: ";
}
}
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"
        TIMEACTION="visibility"
        onend="ButtonUpdate();"
        ontrackchange="AnzeigeUpdate();"
        onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON ID="ID_Button1"
        onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Media.playlist.nextTrack();"
    >
        naechster Track
    </BUTTON>
    <BUTTON ID="ID_Button3"
        onclick="ID_Media.playlist.prevTrack();"
    >
        vorhergehender Track
    </BUTTON>
    <BUTTON ID="ID_Button4"
        onclick="ID_Media.endElement(); AnzeigeLoeschen();"

```



```

>
    Stop
</BUTTON>
</BODY>
</HTML>

```

**.title** Titel des StyleSheets  
siehe Objekt styleSheet

**.to** Endwert zur Werterhöhung bei Elemente-Animation(en) auf der Timeline  
per **.additive** oder **.accumulate**  
für die Objekte **animate**, **animateMotion** und **animateColor** gilt:  
**.to** wird von **.path** und **.values** überschrieben  
**.to** überschreibt **.by**  
Achtung: Im Objekt **animatecolor** (t:ANIMATECOLOR) ist **.values** auch als Farbliste definiert !  
siehe Objekt **currTimeState** und Behavior **.style.time2**

Beispiel: pixelweise Ausdehnung eines DIV in seiner Breite (Style-Eigenschaft **.width**) nach rechts:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function SpanInhaltInit(Kette)
    {
        // Zustand
        ID_Span1.innerText = "";

        // Kumulationsart
        ID_Span2.innerText =Kette;

        // Zaehler auf 1
        ID_Span3.innerText ="1";

        // DIV-Text festlegen
        ID_Div.innerText="Dieser DIV dehnt sich in der Breite ";
        if (Kette == "sum")
        {
            // Wertkumulation von .width
            ID_Div.innerText += " genau 1x kumulativ um ";
            ID_Div.innerText += ID_Animate.by
            ID_Div.innerText += " mal "
            ID_Div.innerText += ID_Animate.repeatCount;
        }
        else
        {
            // keine Wertkumulation von .width
            ID_Div.innerText += ID_Animate.repeatCount;
            ID_Div.innerText += " mal um ";
            ID_Div.innerText += ID_Animate.by
        }

        ID_Div.innerText += " aus";
    }

    function KumulationAus()
    {
        // Element darf nicht aktiv sein !!

        // ERST .accumulate definieren
        SpanInhaltInit("none");
        ID_Animate.accumulate="none";

        // DANACH Animation starten
        ID_Animate.beginElement();
    }

    function KumulationEin()
    {
        // Element darf nicht aktiv sein !!

```



```

// ERST .accumulate definieren
SpanInhaltInit("sum");
ID_Animate.accumulate="sum";

// DANACH Animation starten
ID_Animate.beginElement();
}

function Anzeige()
{
    ID_Span1.innerText = "Animation ist beendet ";
}
</SCRIPT>
</HEAD>
<BODY>
    <t:ANIMATE
        ID="ID_Animate"
        TARGETELEMENT="ID_Div"
        ATTRIBUTENAME="width"
        TO="425px"
        BY="150px"
        DUR="3"
        REPEATCOUNT="3"
        BEGIN="indefinite"
        FILL="freeze"
        onend="Anzeige();"
        onrepeat="ID_Span3.innerText= ID_Animate.currTimeState.repeatCount + 1;"
    >
    <t:ANIMATE>

animierter DIV
<DIV ID="ID_Div"
    CLASS="time_line_klasse"
    STYLE= "position:absolute;
        top:125px;
        left:25px;
        height:100px;
        width:125px;
        border:solid black 2px;
        "
    >
</DIV>
<BR>

Zustand der Animation:
<SPAN ID="ID_Span1"></SPAN>
<BR>

Kumulationsart:
<SPAN ID="ID_Span2"></SPAN>
<BR>

Durchlaufzaehler:
<SPAN ID="ID_Span3"></SPAN>
<BR>

<BUTTON ID="ID_Button1" onclick="KumulationAus()">Kumulation aus </BUTTON>
<BUTTON ID="ID_Button2" onclick="KumulationEin()">Kumulation ein</BUTTON>
</BODY>
</HTML>

```

.to

Ende-Prozentsatz des kompletten Überganges bei Ende der Animation des Übergangsfilters  
 per .style.time2.transitionFilter Behavior-Objekt  
 Ende des Übergangsfilters mit nur teilweise vollendetem Übergang  
 nicht zusammen mit Eigenschaften .by und .value kodieren, sonst wird .to ignoriert  
 siehe Objekt currTimeState und Behavior .style.time2  
 Ziffernfolge Floating point  
 Wert numerisch von 0 bis 1  
 Standard ist "1.0" oder "1"  
 0 entspricht 0 % des kompletten Überganges  
 1 entspricht 100% des kompletten Überganges  
 Bsp.: 0.3 entspricht: Mit **nur** 30 % des kompletten Überganges die Animation beenden

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
```



```

<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:TRANSITIONFILTER ID="ID_Transfilter1"
FROM="0.3"
TO="0.7"
TYPE="barWipe"
DUR="3"
TARGETELEMENT="ID_Div"
>
</t:TRANSITIONFILTER>

<DIV ID="ID_Div"
CLASS="time_line_klasse"
DUR="9"
STYLE=" width:400px; height:100px;background:#CC3333;color:#FFFFFF;"
>
<SPAN ID="ID_Span"
STYLE="position:relative; top:20px;margin:160;"
>
Test
</SPAN>
</DIV>
</BODY>
</HTML>

```

**.toElement** Referenz auf Maus-Eventauslösendes Ziel-Element bei Mausbewegung  
nicht für Event ondragleave  
Objekt event

Beispiel:

```

<SCRIPT>
function Aendern()
{ ID_Span.innerHTML=window.event.toElement.tagName; }
</SCRIPT>
<SPAN onmouseout="Aendern()">
<P>Ueberfahre mich mit Maus</P>
<P>Zielelement =
<SPAN ID="ID_Span"></SPAN>
</P>
</SPAN>

```

**.top** obere Pixelposition des Rechteckes um ein Objekt  
auch textrectangle Objekt

Beispiel für Nicht-TextRectangle-Objekt:

```

<SCRIPT>
function Anzeigen(ObjektZeiger)
{
var Rechteck = ObjektZeiger.getBoudingClientRect();

alert(      "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
+ "\n"
+ "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
);
}
</SCRIPT>
<P onclick="Anzeigen(this)">

```

Beispiel für TextRectangleObjekt:

```

<SCRIPT>
function Anzeigen(ObjektZeiger)
{
var TextRectangleCollection = document.body.ObjektZeiger.TextRange.getClientRects();

// .getClientRects() liefert immer Colletion der textrectangle Objekte !!!

var Rechteck = TextRectangleCollection[0];

```



```

        alert(      "oben links (x,y) = " + Rechteck.left + " " + Rechteck.top
                    + "n"
                    + "unten rechts (x,y) = " + Rechteck.right + " " + Rechteck.bottom
                    );
    }
</SCRIPT>
<BODY>
<DIV onclick="Anzeigen(this)">
    Test
</DIV>
</BODY>

```

.top	Zeiger auf das oberste Fenster in der Fenster-Hierarchie siehe Objekt window
.topMargin	Top Margin in Pixel des Dokumentes Margin: Abstand des Aussenrandes eines Objektes zur Umgebung Padding: Abstand des Objekthinhaltes zum Aussenrand document.body.topMargin
.TotalSize	Gesamten Speicher in Bytes auf Laufwerk ermitteln
Beispiel:	<pre> var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject"); var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:"); var Laufwerk         = DateiSystem.GetDrive(Laufwerk_Bezeichnung); var Laufwerk_VolumeName = Laufwerk.VolumeName; var Laufwerk_TotalerPlatz = Laufwerk.TotalSize; alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalerPlatz); </pre>
TRANSITIONTYPE	wird hier nicht erklärt, da Komponenten von MS DirectMusic-Software
.trueSpeed	zeitliche Takteinheit zur Erzeugung der Pause laut Eigenschaft .scrollDelay eines marquee Objektes false Default Takteinheit = 60 Millisekunden = 1 Sekunde wenn .scrollDelay < 60 so Sekundentakt wenn .scrollDelay > 59 so Millisekundentakt true Takteinheit = immer 1 Millisekunde
.type	Bezeichner des Events, also Eventart Objekt event ist immer der Eventname OHNE den Präfix "on" Bsp. Kette ist "click" für das Ereignis onclick
.type	MIME-Typ des Objektes (Multipurpose Internet Mail Extension) wenn die Eigenschaft .classid nicht kodiert wird, dann immer .type kodieren bzw. die im Browserstandard enthaltenen MIME verwenden Hinweis: MIME wird von Simple Mail Transfer Protocol (SMTP) benutzt z.B. bei Audio, Images, Video, Texten
.type	Art des Buttons (Standard-Behavior (Verhalten) des Buttons) "button" Default. Kommando-Button "reset" Reset-Button wenn im Formular so Funktionalität für Formular-Reset analog zum input-reset-Objekt Achtung: zugleich NAME-Attribut kodieren ! "submit" Submit-Button wenn im Formular so Funktionalität für Formular-Sendeanalog zum input-submit-Objekt (senden der Werte laut .name und .value) Achtung: zugleich NAME-Attribut kodieren !
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT) z.B. im Formular Achtung: Soll der Wert eines Control-Elementes im Formular gesendet werden, so muss für das Element das NAME-Attribut kodiert sein (Wert des Name-Attributs muss elementspezifisch sein: siehe z.B. Checkbox-Control-Element). Gesendet werden in der Regel Wert und Name. Das Name-Attribut ist wichtig für die Auswertung des Wertes des Control-Elementes: Name-Attribut hat dazu die Funktionalität wie das ID-Attribut. Hinweis: Wenn Control-Element ein ID-Attribut erlaubt, dann ist ID auch zur Referenzierung verwendbar z.B. wert_laut_id_attribut.value



Hinweis: Wert des Control-Elementes

nur Stringwerte möglich

Standardwerte laut Eigenschaft .defaultValue

(Eigenschaft nur per Script nutzbar, da keine HTML-Attribut verfügbar)

INPUT type=checkbox

on

INPUT type=reset

Reset

INPUT type=submit

Submit Query

alle anderen Input-Elemente haben keinen Standardwert

Wert laut .value (auch HTML-Attribut vorhanden)

folgenden Werte sind sendbar:

INPUT type=checkbox

selektierter Wert

INPUT type=file

Dateiname laut Eingabe

INPUT type=hidden

selektierter Wert einer

Box-Control (selektierte

Option)

INPUT type=password

Eingabewert

INPUT type=radio

selektierter Wert

INPUT type=reset

das Label des Elementes

(falls Label existent ist)

INPUT type=submit

das Label des Elementes

(falls Label existent ist)

INPUT type=text

Eingabewerte

Werte sind les- und schreibbar

nur lesen bei INPUT type=file

kann in " " kodiert werden, muss aber nicht

"button"

Button-Control

"checkbox"

Checkbox-Control

jedes Element mit **seinem** spezifischen Namen

erhält intern eine fortlaufende Nummer

Achtung: Wenn Elemente des Checkbox-Control

mit gemeinsamen Namen, so

haben diese eine gemeinsame Nummer.

Mehrfachauswahl (Mehrfachselektion) möglich

"file"

Objekt zum Upload einer Datei

(Upload aus Sicht des Client = Browser

Download aus Sicht des PC-User und seiner Festplatte)

Die Datei kann auf dem PC gespeichert werden

(Browser öffnet automatisch eine

Dialogbox zur Angabe der

Speicherortes).

"hidden"

Hidden-Control

"image"

Image-Control das angeklickt werden kann

(Alternative zu einem Link mit Bild)

Achtung: Es löst **sofort** Senden des Formulars aus,

falls Ereignis onclick nicht abgefangen wird !

gesendet wird name.x

name.y

mit name als Wert des Attributes NAME

x und y als Pixelkoordinaten der

linken oberen Ecke des Bildes

gesendet wird **nicht** der Wert laut VALUE-Attribut !

"password"

Text-Control mit veränderter Anzeige eingegebener

Zeichen: Anzeige durch Dummyzeichen

"radio"

Radio-Button-Control

Gruppierung möglich: Gruppe = Radio-Button-

Controls mit gemeinsamen Namen

jede Gruppe hat einen spezifischen Namen

gesendet wird nur der Wert des selektierten

Radio-Button-Control laut

VALUE-Attribut

"reset"

Button-Control mit Funktionalität der Initialisierung

des Formulars auf Standardwerte

Label möglich oder per VALUE-Attribut kodierbar

Eventbehandlung: siehe Formular

"submit"

Button-Control mit Funktionalität des Sendens des Formulars

Label möglich oder per VALUE-Attribut kodierbar

Eventbehandlung: siehe Formular

"text"

Default

einzeiliges Texteingabe-Control

.type

Type des TEXTAREA-Control

siehe textarea Objekt



.type	Mimetype des Scriptes per script Objekt Mimetype wird von der Scriptmaschine zum Parsen verwendet Achtung: Wert muss passend zum Wert der Eigenschaft .language sein ! "text/ecmascript" ECMA-Script "text/JScript" Microsoft JScript "text/javascript" JavaScript "text/vbs" VB-Script "text/vbscript" VB-Script (wie text/vbs) "text/xml" XML.
.type	prüfen auf multiples select Objekt "select-multiple" multiple Auswahl möglich Attribut .multiple ist true "select-one" Default keine multiple Auswahl Attribut .multiple ist false (Standard für .multiple)
.type	Type der Selektion per document.selection Objekt des Dokumentes siehe auch Methoden .createRange() .createControlRange() und .createTextRange() Objekt textrange "none" keine Selektion vom Typ text oder control wird auch geliefert, wenn Cursor auf einen leeren Bereich der Webseite gesetzt wird "text" Text wurde selektiert (siehe document.selection.textrange Collection) typisch für Markierung von Text auf Webseite, um diesen dann in den notepad zu laden "control" Control-Element(e) wurde(n) selektiert z.B. Elemente zum Resize des Dokumentes ( siehe document.selection.controlRange Collection)

Beispiel:

```
<BODY onclick="alert(document.selection.type)">
TestText
</BODY>
```

.type
Sprache des Cascading Style Sheets (CSS) des style Objektes

.type
MIME-Typ eines Media-Elementes auf der Timeline  
Media-Datei zum Element laut Eigenschaft .src

.type
visuelle Erscheinungsform des Überganges per .style.time2.transitionFilter Behavior-Objekt,  
also visuelle Art und Weise des Überganges  
optional kodierbar ist zusätzlich die Eigenschaft .subtype des Behavior  
(bis auf 1 Ausnahme --> siehe unten)  
siehe Objekt currTimeState und Behavior .style.time2  
Übersicht zu Wert von .subtype bei gegebenen Wert von .type

Wert	Eigenschaft	.type	Wert	Eigenschaft	.subtype
"starWipe"			"fivePoint"		
				<b>muss immer kodiert sein</b>	
				<b>wenn .type kodiert wurde</b>	
"barWipe"			"leftToRight"		
		oder	"topToBottom"		
"barnDoorWipe"			"vertical"		
		oder	"horizontal"		
"irisWipe"			"rectangle"		
		oder	"diamond"		
"ellipseWipe"			"circle"		
"clockWipe"			"clockwiseTwelve"		
"fanWipe"			"centerTop"		
"snakeWipe"			"topLeftHorizontal"		
"spiralWipe"			"topLeftClockwise"		
"pushWipe"			"fromLeft"		
"slideWipe"			"fromLeft"		
"fade"			"crossfade"		

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
```





```

<BODY>
  <t:TRANSITIONFILTER ID="ID_Transfilter1"
    TYPE="barWipe"
    SUBTYPE="leftToRight"
    DUR="3"
    TARGETELEMENT="ID_Div1"
  >
</t:TRANSITIONFILTER>

  <t:TRANSITIONFILTER ID="ID_Transfilter2"
    TYPE="starWipe"
    SUBTYPE="fivePoint"
    DUR="3"
    TARGETELEMENT="ID_Div1"
  >
</t:TRANSITIONFILTER>

  <t:TRANSITIONFILTER ID="ID_Transfilter3"
    TYPE="barnDoorWipe"
    DUR="3"
    TARGETELEMENT="ID_Div2"
    FROM="0"
    TO="1"
    CALCMODE="linear"
    MODE="in"
  >
</t:TRANSITIONFILTER>

  <DIV STYLE="height:170px;">

    <DIV ID="ID_Div1"
      CLASS="time_line_klasse"
      STYLE="position:absolute; top:150px; left:20px; background-color:#3366CC; padding:10px;
        height:80; color:white;
      "
    >
      Test1
    </DIV>

    <DIV ID="ID_Div2"
      CLASS="time_line_klasse"
      STYLE="position:absolute; top:185px; left:60px; background-color:#FFCC00;
        padding:10px; height:80;
      "
    >
      Test2
    </DIV>
  </DIV>
</BODY>
</HTML>

```

.type                      Mimetyp des StyleSheets  
                              siehe Objekt styleSheet

.Type                      Beschreibung zum Suffix des Ordnerbezeichners liefern laut Registry

Beispiel:

```

var OrdnerName                = "c:\\windows\\desktop\\";
var DateiSystem              = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                    = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Type);

```

.Type                      Beschreibung zum Suffix des Dateibezeichners liefern laut Registry  
                              z.B. Suffix ist .TXT    Beschreibung ist "Text Document"

Beispiel:

```

var DateinameMitPfad         = "c:\\test.txt";
var DateiSystem              = new ActiveXObject("Scripting.FileSystemObject");
var Datei                    = DateiSystem.GetFile(DateinameMitPfad);
alert(Datei.Type);

```

.typeDetail                Type der Selektion per document.selection Objekt des Dokumentes, wobei die Hostanwendung  
                              diese Eigenschaft unterstützen und mit Wert belegen muss

.uniqueID                   durch den Browser automatisch-generiertes ID des Objektes  
                              Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde



kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

Beispiel:

```
<PUBLIC:ATTACH EVENT="onload" FOR="window" ONEVENT="init()"/>
<SCRIPT LANGUAGE="JScript">
    function init()
    {
        var newTextAreaID = element.document.uniqueID;
        element.document.body.insertAdjacentHTML ( "beforeEnd",
            "<P><TEXTAREA STYLE='height: 200 ;'"
            + "width: 350' ID=" + newTextAreaID
            + "></TEXTAREA></P>"
        );
    }
</SCRIPT>
```

UNSELECTABLE      Selektionsfähigkeit eines Objektes  
                       "off" Default. selektierbar  
                       "on" nicht selektierbar

Beispiel:

```
<SPAN UNSELECTABLE="on" >
    Dieser Text kann nicht selektiert werden
    <TEXTAREA WRAP="PHYSICAL" ROWS="5"
        STYLE="font-weight: bold;"
    >
        Dieser Text kann selektiert werden
    </TEXTAREA>
    Dieser Text kann nicht selektiert werden
</SPAN>
```

.updateInterval      Refresh-Intervall des Bildschirmes: aus dem Puffer neu schreiben  
                       per screen Objekt

.updateMode          Art des Updates der Eigenschaften eines Elementes nach dem Start des Elementes auf der Timeline  
                       folgende Eigenschaften können geupdatet werden:

.autoReverse  
 .begin  
 .dur  
 .end  
 .fill  
 .repeatCount  
 .repeatDur  
 .speed

siehe Objekt currTimeState und Behavior .style.time2  
 "Auto"      automatisches Update  
               immer ausgeführt per Methode .beginElement()  
 "Reset"      Default  
               Reset auf Initialwert nach einer Veränderung

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<SCRIPT>
    function Aendern()
    {
        ID_Video.updateMode = ID_Select1.options.value;
        ID_Video.speed        = ID_Select1.options.value;
    }
</SCRIPT>
</HEAD>
<BODY>
    <t:VIDEO                    ID="ID_Video"
                             SRC="test.avi"
                             UPDATEMODE="reset"
                             STYLE="width:175px; height:150px;"
    >
    </t:VIDEO>
    <BR>
    updateMode waehlen:
    <SELECT NAME="ID_Select1">
```



```

        <OPTION VALUE="auto">Auto</OPTION>
        <OPTION VALUE="reset" SELECTED>Reset</OPTION>
    </SELECT>
    <BR>
    Geschwindigkeit waehlen:
    <SELECT NAME=" ID_Select2">
        <OPTION VALUE="0.25">25%</OPTION>
        <OPTION VALUE="0.50">50%</OPTION>
        <OPTION VALUE="0.75">75%</OPTION>
        <OPTION VALUE="1" SELECTED>100% </OPTION>
        <OPTION VALUE="2" SELECTED>200% </OPTION>
    </SELECT>
    <BR>
    <BUTTON ID="ID_Button1" onClick="Aendern();">
        Geschwindigkeit aendern
    </BUTTON>
    <BUTTON ID="ID_Button2" onClick="document.body.beginElement();">
        Restart
    </BUTTON>
</BODY>
</HTML>

```

.URL	Url des Dokumentes Achtung: Gross-und kleinschreibung wird unterschieden !!! ist identisch mit location.href
.URL	Url laut einem Kommando aus einer Advanced Streaming Format (ASF)-Datei von Element auf der Timeline es muss Ereignis onURLFlip aufgetreten sein siehe Objekt currTimeState und Behavior .style.time2 window.event.URL
.URLUnencoded	Url des Dokumentes ohne Zeichenencoding
.urn	Uniform Resource Name (URN) des Dokumentes
.useMap	Url oder Anker für client-seitige Image-Map
.userAgent	Browser-Codename <b>und</b> –Version per navigator Objekt navigator.userAgent z.B. IE 6.0 liefert unter <b>Windows XP</b> "Mozilla/4.0 (compatible; MSIE 6.0; <b>Windows NT 5.1</b> )"

Beispiel 1:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        if(navigator.userAgent.indexOf("Windows NT 5.1")>-1)
        {..... }
    }
</SCRIPT>

```

Beispiel 2 für prüfen auf ActiveX beim IE:

```

var ActiveXAktiv = false;

var NavigatorObjekt = window.navigator;           // Zeiger

var BrowserArt     = NavigatorObjekt.userAgent;    // String
var IEerkannt      = (BrowserArt.indexOf('IE') > -1); // true, so IE erkannt

var Plattform      = NavigatorObjekt.platform;     // String
var Win32Erkannt   = (Plattform == "Win32");       // true, so Win32-Bit erkannt

ActiveXAktiv       = (IEerkannt && Win32Erkannt);  // true, so IE und Win32-Bit erkannt,
                                                    // also ActiveX möglich

```

.userAgent	HTTP User-Agent siehe Objekt window.clientInformation z.B. IE 6.0 liefert unter <b>Windows XP</b> "Mozilla/4.0 (compatible; MSIE 6.0; <b>Windows NT 5.1</b> )"
------------	---

.userLanguage	vom User eingestellte Sprache des Betriebssystems (nicht die Sprache der Installation,
---------------	--



	sondern die regionale Sprache des Betriebssystems)
	per navigator Objekt navigator.userAgent z.B. "en" oder "de"
.userLanguage	Sprache des Betriebssystems laut UserEinstellung Behavior .style.clientCaps z.B. "en" oder "de"
.userLanguage	Sprache des Betriebssystems laut UserEinstellung siehe Objekt window.clientInformation z.B. "en" oder "de"
.userProfile	Zeiger auf Collection userProfile siehe Objekt navigator
.vAlign	Lage der CAPTION einer Tabelle es darf nur 1 CAPTION zur Tabelle existieren nach Änderung ist ein Tabellen-Refresh per Methode .refresh() notwendig siehe Objekt table.caption "top" Überschrift am Kopf der Tabelle Standard "bottom" Überschrift am Fuss der Tabelle
.vAlign	vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes "middle" zentriert, Standard "baseline" Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des <b>benachbarten</b> Objektes "bottom" am Fuss "top" am Kopf
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar

Beispiel für Input-Objekt und seine Varianten:

für Input-Elemente (input Objekt) gelten folgende Standardwerte:

INPUT type=checkbox	on
INPUT type=reset	Reset
INPUT type=submit	Submit Query

alle anderen Input-Elemente haben keinen Standardwert

für Input-Elemente (input Objekt) sind folgenden Werte sendbar:

INPUT type=checkbox	selektierter Wert
INPUT type=file	Dateiname laut Eingabe
INPUT type=hidden	selektierter Wert einer Box-Control (selektierte Option)
INPUT type=password	Eingabewert
INPUT type=radio	selektierter Wert
INPUT type=reset	das Label des Elementes (falls Label existent ist)
INPUT type=submit	das Label des Elementes (falls Label existent ist)
INPUT type=text	Eingabewerte

Werte sind les- und schreibbar  
nur lesen bei INPUT type=file

Beispiel für option objekt eines select objektes:

```
<SCRIPT>
function Anzeigen()
{
    var Kette = "Auswahl = " + ID_select.options(ID_select.selectedIndex).value;
    alert(Kette);
}
</SCRIPT>
<SELECT ID="ID_select" onchange = "Anzeigen()">
    <OPTION VALUE="1">Auswahl 1 </OPTION>
    <OPTION VALUE="2">Auswahl 2 </OPTION>
    <OPTION VALUE="3">Auswahl 3 </OPTION>
</SELECT>
```

Beispiel für Zeichenkette auf Wertebereich der Zeichen prüfen:



```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function pruefe_zeichenkette(zeichenkette, wertebereich)
{
    // wertebereich ist Zeichenkette z.B. "0123456789 -+/,()"

    var rueckgabewert = true;          // Annahme: Zeichenkette hat NUR Zeichen
                                      //                               aus dem Wertebereich

    var zeichen_aus_zeichenkette;

    //      Zeichenkette zeichenweise analysieren: Jedes Zeichen mit Wertebereich vergleichen
    for (var i = 0; i < zeichenkette.length; i++)
    {
        zeichen_aus_zeichenkette = zeichenkette.charAt(i);

        if (wertebereich.indexOf(zeichen_aus_zeichenkette) == -1)
        {
            rueckgabewert = false;
            break;    // ungültiges Zeichen gefunden, also abbrechen
        }
    }

    return rueckgabewert;
}

function pruefe_eingabe(zu_pruefende_zeichenkette)
{
    if (pruefe_zeichenkette(zu_pruefender_zeichenkette, "0123456789 -+/,()") )
    {alert("Eingabe ist korrekt !");}
    else
    {alert("Eingabe ist nicht korrekt !"); }
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>

    Telefon:
    <INPUT    TYPE="text"
              NAME="Telefon"
              VALUE=""

    >
    <INPUT    TYPE="button"
              VALUE="Ueberpruefen"
              onclick="pruefe_eingabe(this.form.Telefon.value)">

</FORM>
</BODY>
</HTML>
```

`.values` 2D-Animation bei der Elemente-Animation(en) auf der Timeline  
per `.additive` oder `.accumulate`  
anhand absoluter Koordinaten im Grafiksystem  
siehe `.path` für Vektorgraphik-Animation anhand realtiver und absoluter Koordinaten  
Werteliste korrespondiert zu der Werteliste der Eigenschaft `.keyTimes`  
benötigt `.calcMode` auf "spline" `.keySplines` und `.keyTimes`  
für die Objekte `animate`, `animateMotion` und `animateColor` gilt:  
`.values` wird von `.path` überschrieben  
`.values` überschreibt `.by` `.from` `.to`  
Achtung: Im Objekt `animatecolor` (`t:ANIMATECOLOR`) ist `.values` auch als Farbliste definiert !  
siehe Objekt `currentTimeState` und Behavior `.style.time2`

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
.div_Klasse{position:absolute; top:195px; height:100px; width:150px; border:solid black 2px;}
</STYLE>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
CLASS="time_line_klasse"
DUR="1"
```



```

        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=( ID_Animatemotion.currTimeState.simpleTime);"
    >
        0
    </SPAN>
    <BR>
    <SPAN ID=" ID_Span2"
        CLASS="time_line_klasse"
        DUR=".1"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=( ID_Animatemotion.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <DIV ID="ID_Div"
        CLASSE="div_klasse"
    >
        animierter Div
    </DIV>
    <t:ANIMATEMOTION ID="ID_Animatemotion"
        BEGIN="1; ID_Button.click+1"
        DUR="6s"
        AUTOREVERSE="true"
        CALCMODE="spline"
        KEYSPLINES="0 1 .5 1;.5 1 0 1"
        KEYTIMES="0;.5;1"
        VALUES="25,0;250,50;500,0"
        TARGETELEMENT="ID_Div"
        FILL="hold"
    >
    </t:ANIMATEMOTION>
    <BUTTON ID="ID_Button">Restart</BUTTON>
</BODY>
</HTML>

```

.values

Farbwerte für die Elemente-Animation(en) auf der Timeline  
 per .additive oder .accumulate  
 nur für Objekt animatecolor (t:ANIMATECOLOR)  
 Achtung: Nur im Objekt animatecolor (t:ANIMATECOLOR) ist .values auch als Farbliste definiert !  
 Es existieren als 2 kodierbare Varianten von .values.  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
    <t:PAR ID="ID_Par"
        BEGIN="0"
        DUR="10"
        FILL="hold"
    >
        <t:ANIMATECOLOR ID="ID_Animatecolor1"
            TARGETELEMENT="ID_Div1"
            ATTRIBUTENAME="background-color"
            VALUES="#0000FF;cyan"
            BEGIN="0"
            DUR="5"
            FILL="hold"
        >
        </t:ANIMATECOLOR>
        <t:ANIMATECOLOR ID="ID_Animatecolor2"
            TARGETELEMENT="ID_Div2"
            ATTRIBUTENAME="background-color"
            VALUES="cyan ;#0000FF"
            BEGIN="0"
            DUR="5"
            FILL="hold"
        >
        </t:ANIMATECOLOR>
    >

```



```

</t:PAR>
<DIV ID="ID_Div1"
      CLASS="time_line_Klasse"
      STYLE="position: absolute; left: 68px; width: 279px; top: 260px; height: 217px;
            border: 1px solid black; background-color: green;
            "
      >
        animierter Div
</DIV>
<DIV ID="ID_Div2"
      CLASS="time_line_klasse"
      STYLE="position: absolute; left: 112px; width: 188px; top: 318px; height: 98px;
            padding-left: 3; background-color: gray;
            "
      >
        animierter Div
</DIV>
</BODY>
</HTML>

```

.values

Animationsschritte als Prozentsatz des kompletten Überganges  
 per .style.time2.transitionFilter Behavior-Objekt  
 nicht zusammen mit Eigenschaften .by und .to und .from kodieren, da diese sonst ignoriert werden  
 siehe Objekt currTimeState und Behavior .style.time2  
 Liste aus per Semikolon getrennten Elementen  
 Element:  
     Ziffernfolge Floating point  
     Wert numerisch von 0 bis 1  
     0 entspricht 0 % des kompletten Überganges  
     1 entspricht 100% des kompletten Überganges  
 Elementefolge:      muss numerisch aufsteigend sein  
                       erstes Element muss nicht 0.0 sein  
                       letztes Element muss nicht 1.0 sein

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>

  <t:TRANSITIONFILTER ID="ID_Transfilter1"
    BEGIN="ID_Div1.begin"
    TYPE="barWipe"
    DUR="5"
    TARGETELEMENT="ID_Div1"
    VALUES=".1;.17;.27;.37;.47;.56;.65;.71;.82;.92;1.0"
    CALCMODE="discrete"
  >
</t:TRANSITIONFILTER>

  <t:TRANSITIONFILTER ID="ID_Transfilter2"
    BEGIN="ID_Div1.begin"
    TYPE="barWipe"
    DUR="5"
    TARGETELEMENT="ID_Div2"
    VALUES=".1;.17;.27;.37;.47;.56;.65;.71;.82;.92;1.0"
    CALCMODE="linear"
  >
</t:TRANSITIONFILTER>
<BR>
<INPUT TYPE="button" ID="ID_Button" VALUE="Start Transition">

  <DIV ID="ID_Div1"
    CLASS="time_line_Klasse"
    BEGIN="ID_Button.click"
    DUR="indefinite"
    STYLE="position: relative; left: 20px; width: 420px; height: 100px;
          background-image: url(test.gif); background-repeat: no-repeat;
          "
  >
</DIV>

```



```
<DIV      ID="ID_Div2"
          CLASS="time_line_Klasse"
          BEGIN="ID_Button.click"
          DUR="indefinite"
          STYLE="position:relative; left:20px; width:420px; height:100px;
                background-image:url(test.gif); background-repeat: no-repeat;
                "
>
</DIV>
</BODY>
</HTML>
```

.vcard_name	<p>Werte für vCard für Autocomplete (Autovervollständigung) bei Formular per Text-Controls</p> <p>z.B. input text Objekt</p> <p>anhand vCard-Schemas analog zur Visitenkarte</p> <p>Userdaten werden als Profil gespeichert</p> <p>bei aktivem Autocomplete wird der Wert vom NAME-Attribut</p> <p>nicht verwendet wenn VCARD_NAME ist kodiert</p> <p>verwendet wenn VCARD_NAME nicht ist kodiert</p> <p>bei inaktivem Autocomplete werden</p> <p>weder der Wert des NAME-Attribut</p> <p>noch der Wert von VCARD_NAME verwendet</p> <p>siehe auch Eigenschaft .autocomplete</p>
-------------	--

Beispiel: Der in die Textbox eingegebene Wert wird als Email-Adresse aufgefasst  
(und gespeichert falls Autocomplete aktiv ist)  
es wird nicht "CustomerEmail" gesendet sondern der Textboxinhalt als vCard.Email-Schema

```
<INPUT TYPE = text
NAME= "CustomerEmail"
VCARD_NAME = "vCard.Email"
>
```

.version	Document Type Definition-Version (DTD-Version) ab IE 5.x
----------	---

.vLink	Farbe eine VLINK des Dokumentes document.body.vLink "#rrggbb" vordefiniertes Farbname (browserspezifisch)
--------	--

<code>.vlinkColor</code>	Farbe bereits geklickter Links des Dokumentes
<code>document.body.vlinkColor</code>	
<code>"#rrggbb"</code>	Standard "#800080"
	vordefinierter Farbname (browserspezifisch)

.volume      aktuelle Wiedergabe-Lautstärke (Run time volume) per Objekt bgsound, wenn die Media-Datei nicht selbst Lautstärke-Einstellungen während der Wiedergabe vornimmt:  
                  Bsp.:      MIDI      Volume ohne Wirkung  
                               WAVE      Volume soll Wirkung haben  
                  Hinweis: eventuelle Veränderung der Regler in der Windows-Lautstärke-Regelung beachten.  
                  -10 bis 0  
                  0    maximal laut

```
<HEAD>
<SCRIPT>

function KanalVerteilung(Wert)
{ ID_bgsound.balance = Wert; }

function GenauEinmal()
{
    ID_bgsound.loop = 1;
    ID_bgsound.src = ID_bgsound.src; // restart
}

function Endlos()
{
    ID_bgsound.loop = -1;
    ID_bgsound.src = ID_bgsound.src; // restart
}

</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
```





```

<BUTTON onclick="GenauEinmal()"></BUTTON>
<BUTTON onclick="Endlos()"></BUTTON>
<BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
<BUTTON onclick="KanalVerteilung(10)"></BUTTON>
<BUTTON onclick="KanalVerteilung (0)"></BUTTON>
<B>Volume control:</B>&nbsp;
<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-10;"
>Mute

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-7;"
>25% Volume

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED
onpropertychange="ID_bgsound.volume=-5;"
>50% Volume

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-2;"
>75% Volume

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=0;"
>100% Volume
</BODY>

```

.volume

Lautstärke eines Audio-Elementes auf der Timeline relativ zur Lautstärke des Elternelementes für alle Audio-Elemente des body Objektes auch möglich  
Achtung: Bei Verwendung von DirectMusic wirkt sich das auf den DirectMusic-Player aus !  
siehe Objekt currTimeState und Behavior .style.time2  
siehe Objekt bgsound  
Prozent der Lautstärke der Eltern  
0 bis 100  
0 stumm  
100 Lautstärke wie die der Eltern

Beispiel für Verwendung von DirectMusic:

```
<t:AUDIO VOLUME="100" PLAYER=dmusic SRC="sample.sgt">
```

Beispiel für Abspielen einer Sounddatei per Timeline als Alternative zum Objekt bgsound:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:MEDIA ID="ID_Media"
BEGIN="indefinite;"
SRC="test.wmv"
FILL="remove"
onmediacomplete="Timer2.beginElement();"
>
</t:MEDIA>

<BR>

<BUTTON onclick="ID_Media.beginElement();">Start</BUTTON>
<BUTTON onclick="ID_Media.endElement();">Stop</BUTTON>

<BR>

<B>Volume control:</B>&nbsp;

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_Media.mute='true';"
>Mute

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_Media.mute='false'; ID_Media.volume=25;"
>25% Volume

```



```

< INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED
        onpropertychange="ID_Media.mute='false'; ID_Media.volume=50;"
>50% Volume

< INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_Media.mute='false'; ID_Media.volume=75;"
>75% Volume

< INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_Media.mute='false'; ID_Media.volume=100;"
>100% Volume
</BODY>
</HTML>

```

**.VolumeName**                      Volumenbezeichner des Laufwerkes

Beispiel:

```

var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk             = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName  = Laufwerk.VolumeName;
var Laufwerk_TotalerPlatz = Laufwerk.TotalSize;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalerPlatz);

```

**.vspace**                              vertikaler Abstand in Pixel zum Elternobjekt

**.wheelDelta**                      liefert Umdrehung und Richtung in der das Mausexplorer gedreht wurde  
Verwendung bei Event onmousewheel  
ab IE 6.x  
Objekt event  
ist Faktor von 120 Grad-Umdrehung  
    > 0 so Mausexplorer vom User weggedreht  
    < 0 so Mausexplorer zum User gedreht

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Drehen()
    { ID_Span.innerText=event.wheelDelta; }
</SCRIPT>
</HEAD>
<BODY>
    <DIV onmousewheel="Drehen()">Maus hierauf setzen und dann Mausexplorer drehen ...>
        <SPAN ID="ID_Span"></SPAN>
    </DIV>
</BODY>
</HTML>

```

**.width**                              Breite des Objektes in Pixel  
Integer in Pixels für absolute Breite, >=0  
    bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel  
String      Ziffernfolge eines Integer mit nachfolgendem % für Breite als  
    Anteil der Breite des Elternobjektes  
    z.B. 10%

**.width**                              Auflösung des Bildschirms in Breite, also Anzahl der horizontalen Pixel  
per screen Objekt

**.width**                              Auflösung des Bildschirms in Breite, also Anzahl der horizontalen Pixel  
Behavior .style.clientCaps

**.wrap**                              Wortumbruch der TEXTAREA  
siehe textarea Objekt

"soft"      Standard  
    Wortumbruch aktiv  
    im Formular wird nicht gesendet:  
        Zeilenumbruch  
        Zeilenvorschub

"hard"      Wortumbruch aktiv  
    im Formular wird gesendet:  
        Zeilenumbruch  
        Zeilenvorschub



"off" Wortumbruch nicht aktiv

.x X-Koordinate Maus relativ zum Eltern-Objekt ab IE 5.x  
 zum Fenster unter IE 5.x  
 X-Koordinate ist relativ zum BODY-Element wenn  
 Objekt, das das Event auslöst, absolut positioniert ist (z.B. per Style)  
 oder Eltern nicht absolut positioniert sind  
 Objekt event  
 -1 wenn Maus außerhalb des Fensters

.XMLDocument Referenz auf XML-Dokument (XML-DOM)  
 Beispiel:  

```
<HTML>
<HEAD>
<SCRIPT>
  var ZeigerAufXMLDocument = ID_Div.XMLDocument;
</SCRIPT>
</HEAD>
<BODY>
  <DIV ID="ID_Div">
  </DIV>
</BODY>
</HTML>
```

XMLNS Namensraum für Benutzer-Tags zu einem HTML-Tag  
 ab IE 5.x  
 nur für das Tag HTML möglich  
 Namensraum: Präfix des Benutzer-Tags  
 oder Uniform Resource Name (URN)

Beispiel 1:  

```
<HTML xmlns:Prefix1 xmlns:Prefix2="www.microsoft.com">
```

Beispiel 2:  

```
<HTML xmlns:MSIE>
<HEAD>
  <STYLE>
    @media all { MSIE\clientCaps {behavior:url(#default#clientcaps);} }
  </STYLE>
</HEAD>
<BODY>
  <MSIE:CLIENTCAPS ..... >
</BODY>
```

.XSLDocument Referenz auf den obersten Knoten des XSL-Dokumentes (Style-Sheet-Dokument)  
 var Zeiger = document.XSLDocument;

.y Y-Koordinate Maus relativ zum Eltern-Objekt ab IE 5.x  
 zum Fenster unter IE 5.x  
 Y-Koordinate ist relativ zum BODY-Element wenn  
 Objekt, das das Event auslöst, absolut positioniert ist (z.B. per Style)  
 oder Eltern nicht absolut positioniert sind  
 Objekt event  
 -1 wenn Maus außerhalb des Fensters

### Methoden

.abs() Absolutbetrag  
 siehe Script-Objekt Math

.acos() Arcus Cosinus im Bogenmass  
 liefert Wert von 0 bis PI  
 siehe Script-Objekt Math

.activeTimeToParentTime() Wert der aktiven Timeline des Elementes in den korrespondierenden Wert der Timeline  
 der Eltern konvertieren  
 per Eigenschaft .isActive das Element auf Aktivsein prüfen  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:  

```
<HTML xmlns:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
  function Anzeige()
```



```

    { alert(ID_Media.syncTolerance; }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onrepeat="innerText=parseInt(t1.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <t:EXCL ID="ID_EXCL"
        CLASS="time_line_klasse"
        DUR="10s"
        AUTOREVERSE="true"
    >
        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            BEGIN="1s"
            DUR="3s"
            TIMEACTION="visibility"
        >
            Erste Zeile
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            BEGIN="4s"
            DUR="3s"
            TIMEACTION="visibility"
        >
            Zweite Zeile
        </DIV>
    </t:EXCL>
    <BR>
    <BUTTON ID="ID_Button1"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onclick="alert(
            'Punkt in der Timeline der Eltern: '
            + ID_Div1.activeTimeToParentTime(
                ID_EXCL.currTimeState.activeTime
            )
        );
        onrepeat=
            'innerText=Erste Zeile activeTimeToParentTime bei '
            + parseInt(ID_EXCL.currTimeState.activeTime)
            + ' Sekunden';
    >
        Erste Zeile activeTimeToParentTime bei 0 Sekunden
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button2"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onclick="alert(
            'Punkt in der Timeline der Eltern: '
            + ID_Div1.activeTimeToParentTime(
                ID_EXCL.currTimeState.activeTime
            )
        );
        onrepeat=
            'innerText=Zweite Zeile activeTimeToParentTime bei '
            + parseInt(ID_EXCL.currTimeState.activeTime)
            + ' Sekunden';
    >
        Zweite Zeile activeTimeToParentTime bei 0 Sekunden
    </BUTTON>
</BODY>
</HTML>

```



`.activeTimeToSegmentTime()` Wert der aktiven Timeline des Elementes in den korrespondierenden Wert der Timeline des Segmentes konvertieren  
 per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
window.onload = Rekursion; // ohne () kodieren !!

function Rekursion()
{window.setInterval(Anzeige, 100); }

function Anzeige()
{
    ID_Span1.innerHTML = "&nbsp;activeTimeToSegmentTime:&nbsp;"
                        + (ID_Animate.activeTimeToSegmentTime(
                            ID_Div.currTimeState.activeTime
                        ));

    ID_Span2.innerHTML = "&nbsp;segmentTime:&nbsp;"
                        + (ID_Animate.currTimeState.segmentTime);
}
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="ID_Span0"
CLASS="time_line_klasse"
DUR="1"
REPEATCOUNT="indefinite"
onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
>
0
</SPAN>
<DIV ID="ID_Div"
CLASS="time_line_klasse"
STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<t:ANIMATEMOTION ID="ID_Animate"
TARGETELEMENT="ID_Div"
TO="250,0"
DUR="3"
AUTOREVERSE="true"
>
</t:ANIMATEMOTION>
<SPAN ID="ID_Span1">
activeTimeToSegmentTime:
</SPAN>
<BR>
<SPAN ID="ID_Span2">
segmentTime:
</SPAN>
</BODY>
</HTML>
```

`.add()` ein bereits erzeugtes Element einer Collection hinzufügen  
 hinzufügen erst nach dem kompletten Laden des Dokumentes  
 Element erzeugen per Methode `.createElement()`

Beispiel:

```
<SCRIPT>
var ZeigerAufNeueOption = document.createElement("OPTION");
ID_Select.options.add(ZeigerAufNeueOption);
ZeigerAufNeueOption.innerText = "Option 2";
ZeigerAufNeueOption.Value = "2";
</SCRIPT>
<SELECT ID="ID_Select" >
<OPTION VALUE="1">Option 1</OPTION>
</SELECT>
```



**.Add()** Date zum Dictionary Objekt hinzufügen

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}

// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // und Meldung ob Date vorhanden ist
    alert(SchlüsselVorhanden(DatenSchlüssel));
}
```

**.Add()** Ordner der Collection Collection FileSystemObject.Folder.Folders hinzufügen  
Ordner darf in der Collection nicht bereits existieren (sonst Fehler erzeugt)

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder(OrdnerName);
var DateiSystem_FoldersCollection = Ordner.SubFolders;
var NeuerOrdner = DateiSystem_FoldersCollection.Add("Neuer Ordner");
```

**.addBehavior()** DHTML-Verhaltenseigenschaft einem Element hinzufügen  
Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig).  
ab IE 5.x bis unter IE 5.5

Beispiel:

```
<SCRIPT>
var FeldDerEigenschaftenID = new Array(); // für removeBehavior
var FeldDerTagsLlimDokument = new Array ();
var FeldDerTagsLlimDokument_Laenge = 0;

function EigenschaftHinzufuegen()
{
    FeldDerTagsLlimDokument = document.all.tags ("LI");
    FeldDerTagsLlimDokument_Laenge = FeldDerTagsLlimDokument.length;
    for (var i=0; i < FeldDerTagsLlimDokument_Laenge; i++)
    {
        var EigenschaftenID // immer neu anlegen wegen Zeigerprüfung
        = FeldDerTagsLlimDokument [i].addBehavior ("hilite.htc");

        if (iEigenschaftenID)
        {FeldDerEigenschaftenID[i] = EigenschaftenID;}
    }
}

function EigenschaftEntfernen()
{
    for (var i=0; i < FeldDerTagsLlimDokument_Laenge; i++)
    {FeldDerEigenschaftenID[i].removeBehavior (FeldDerEigenschaftenID[i]); }
}
</SCRIPT>
<A HREF="javascript:EigenschaftHinzufuegen()">Eigenschaft hinzufuegen</A>
<A HREF="javascript:EigenschaftEntfernen()">Eigenschaft entfernen</A>
```

**.AddChannel()** Dialogbox zur Channel-Einstellung öffnen um einen Channel zu installieren (Microsoft Active Channel)  
erzeugt im Fehlerfall eine Dialogbox und das Event onerror  
siehe Objekt window.external

Beispiel:

```
window.external.AddChannel("http://test /file.cdf");
```

**.addComponentRequest()** Komponente zum Download anfordern und nach dem Download installieren  
Behavior .style.clientCaps

**.AddDesktop()** eine Webseite oder Bild zum installierten Microsoft Active Desktop hinzufügen



wenn Active Desktop nicht installiert, so passiert nichts  
siehe Objekt window.external

Beispiel:

```
window.external.AddDesktopComponent("http://www.test.de","website",100,100,200,200);
```

.AddFavorite()                      Dialogbox zum Hinzufügen einer Url in die Favoritenliste für den User öffnen  
siehe Objekt window.external

Beispiel:

```
window.external.AddFavorite(location.href, document.title);
```

.addImport()                      StyleSheet aus externer CSS-Datei importieren und als Element der Collection styleSheet.imports erzeugen (entspricht @import url() innerhalb STYLE im HEAD)  
siehe Objekt styleSheet und Collection styleSheet.imports

.addPageRule()                    StyleSheet importieren, als ob er wie ein in das Dokument direkt kodierter Style implementiert wurde, und als Element der Collection styleSheet.pages erzeugen (entspricht @page vom Objekt page)  
siehe Objekt styleSheet und Collection styleSheet.pages

Beispiel:

```
function TextFaerben ()
{document.styleSheets[0].addPageRule("DIV B", "color:blue", 0);}

<DIV onmouseover="TextFaerben ();">
  <B>dieser Text wird per CSS mit blauer Fabrbe angezeigt</B>
</DIV>
```

.addReadRequest()                Eintrag in Queue für Lesezugriff (Request Queue) erzeugen  
siehe Objekt navigator.userProfile und Autovervollständigung im Internet Explorer

Beispiel :

```
// Request Queue füllen

//      es soll der Wert zu "vcard.displayname" ermittelt werden
navigator.userProfile.addReadRequest("vcard.displayname");
//      es soll der Wert zu " vcard.gender " ermittelt werden
navigator.userProfile.addReadRequest("vcard.gender");

// Datenermitteln per Queue
navigator.userProfile.doReadRequest(12, "Daten von unbekannt verwendet");

// Queue leeren
navigator.userProfile.clearRequest();
```

.addRule()                        StyleSheet importieren, als ob er wie ein in das Dokument direkt kodierter Style implementiert wurde, und als Element der Collection styleSheet.rules erzeugen (entspricht xx { ...} innerhalb STYLE im HEAD)  
siehe Objekt styleSheet und Collection styleSheet.rules

Beispiel:

```
function TextFaerben ()
{document.styleSheets[0].addRule("DIV B", "color:blue", 0);}

<DIV onmouseover="TextFaerben ();">
  <B>dieser Text wird per CSS mit blauer Fabrbe angezeigt</B>
</DIV>
```

.alert()                          Dialogbox erzeugen  
1.                                Anzeige von:  
                                    Ausrufungszeichen-Symbol  
                                    variablen String  
                                    OK-Button und  
2.                                warten auf Drücken des OK-Button  
siehe Objekt window

.anchor()                        HTML-Anker <A> als HTML-Kette erzeugen in der Form "<A NAME='....'>text</A>" und ohne weitere Attribute ID, HREF etc.  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral        ="Sichtbarer Ankertext"
var HTML_Kette         = StringLiteral.anchor("WertDesNAMEAttributes");
HTML_Kette             ="Sichtbarer Ankertext".anchor("WertDesNAMEAttributes");

entspricht <A NAME="WertDesNAMEAttributes">Sichtbarer Ankertext</A>
```

```
eval(HTML_Kette);                    erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);        zeigt den Anker an und erzeugt Eintrag in der Collection document.anchors
```



**.appendChild()** Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern  
 DOM wird geändert  
 Zeiger wird zugleich immer am Ende der Collection `childNodes` angehängen  
 Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde

Beispiel 1:

```
var ZeigerAufDiv=document.createElement("DIV");
document.body.appendChild(ZeigerAufDiv);
```

Beispiel 2:

```
<SCRIPT>
function Anhaengen()
{
    var ZeigerAufNeuesLI oNewNode = document.createElement("LI");
    Liste.appendChild(ZeigerAufNeuesLI);           // dem BODY anhängen
                                                    // also sichtbar werdend
    ZeigerAufNeuesLI.innerText="Listenelement 5";
}
</SCRIPT>
<BODY>
    <UL ID = "Liste" >
        <LI>Listenelement 1
        <LI>Listenelement 2
        <LI>Listenelement 3
        <LI>Listenelement 4
    </UL>
    <INPUT TYPE = "button" VALUE = "Anhaengen "
           onclick = " Anhaengen()">
</BODY>
```

**.appendData()** String an das Ende des Objektes anhängen

**.apply()** ein anderes Objekt anstelle des arguments Script-Objekt verwenden  
 siehe Script-Objekt Funktion und Script-Objekt arguments und Anweisung function

**.applyElement()** Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern  
 DOM wird geändert  
 Element kann selbst Kinder haben  
 Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde  
 Achtung: Wenn Element per Methode `.createElement()` erzeugt wurde, aber nicht bereits im Dokumentenbaum eingebunden ist, so wird die Eigenschaft `.innerHTML` gelöscht !

Beispiel:

```
<SCRIPT>
function Hinzufuegen()
{
    var Zeiger = document.createElement("LI");
    Liste.applyElement(Zeiger);
}
</SCRIPT>
<UL ID = Liste>
    <LI>Listenelement 1
    <LI>Listenelement 2
    <LI>Listenelement 3
    <LI>Listenelement 4
</UL>
<INPUT TYPE="button" VALUE=" Hinzufuegen" onclick=" Hinzufuegen()">
```

**.asin()** Arcus Sinus im Bogenmass  
 liefert Wert von  $-\pi/2$  bis  $+\pi/2$   
 siehe Script-Objekt Math

**.assign()** neues Dokument zuweisen und laden  
 im Verlauf (History) wird ein neuer Eintrag hinzugefügt (history Objekt)  
 Altes Dokument ist per Vorwärts- und Zurück-Button einstellbar.

**.atan()** Arcus Tangens im Bogenmass  
 liefert Wert von  $-\pi/2$  bis  $+\pi/2$   
 siehe Script-Objekt Math

**.atan2()** Arcus Tangens im Bogenmass  
 siehe Script-Objekt Math

**.atEnd()** prüfen auf Erreichen des Ende der Collection





auf Zustand undefined eines Elementes der Collection  
 auf leere Collection  
 verwenden mit .moveNext() innerhalb einer Schleife  
 siehe Enumerator JScript-Objekt

Beispiel Laufwerke auf dem PC des Users ermitteln:

```
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

for ( ; ! DateiSystem_Laufwerke.atEnd(); DateiSystem_Laufwerke.moveNext() )
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        {
            LaufwerkName = "[Drive not ready]";
        }
    }

    Kette += LaufwerkName + "\n";
}
alert (Kette);
```

.attachEvent()      Einschalten des Registrieren eines Events durch Eventhandler  
 Hinweis: Abschalten mit Methode .detachEvent()  
 Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider  
           NICHT verkettet sondern in **Zufallsfolge**, es sei denn  
           die Handler prüfen ihre Aufruffolge (muss programmiert werden)  
 true            Einschalten erfolgreich  
 false          Einschalten nicht möglich gewesen

Beispiel:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor      = style.color;
            runtimeStyle.color = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {

```



```

        if (event.srcElement == element)
        {
            runtimeStyle.color = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu);           // nicht () kodieren !!!
        detachEvent ('onmouseout', CursorNormal);         // nicht () kodieren !!
    }

    attachEvent ('onmouseover', CursorNeu);
    attachEvent ('onmouseout', CursorNormal);
</SCRIPT>

```

`.attachEvent()` Einschalten des Registrieren eines Events durch Eventhandler  
 Hinweis: Abschalten mit Methode `.detachEvent()`  
 Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider  
 NICHT verkettet sondern in **Zufallsfolge**, es sei denn  
 die Handler prüfen ihre Aufruffolge (muss programmiert werden)  
 Vor dem Neubelegen immer den Standard-Eventhandler retten und diesen  
**nach** `.detachEvent()` wieder einbinden (siehe Beispiel).

siehe Objekt window

Beispiel 1:

```

<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor = style.color;
            runtimeStyle.color = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {
        if (event.srcElement == element)
        {
            runtimeStyle.color = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu); // nicht () kodieren !!!
        window.onmouseover = Rette_Standard_Eventhandler_OnMouseOver;

        detachEvent ('onmouseout', CursorNormal); // nicht () kodieren !!
        window.onmouseout = Rette_Standard_Eventhandler_OnMouseOut;
    }

    var Rette_Standard_Eventhandler_OnMouseOver = window.onmouseover;
    attachEvent ('onmouseover', CursorNeu);

    var Rette_Standard_Eventhandler_OnMouseOut = window.onmouseout;
    attachEvent ('onmouseout', CursorNormal);
</SCRIPT>

```

Beispiel 2:

```

function ResizeHandler()           // Homepage neu laden
{window.history.go(0);}

// Standardhandler retten
var RetteHandler_Resize = window.onresize;

// und neu belegen
window.onresize = ResizeHandler;   // Homepage neu laden
                                   // ohne () kodieren, damit nicht sofort ausgeführt wird

```



```
// und Start des Abfangens vom resize-Event
window.attachEvent('onresize', ResizeHandler);
```

.....

```
// irgendwann abschalten der Eventüberwachung
window.detachEvent('onresize', ResizeHandler);
```

```
// und Standardhandler wieder einbinden
window.onresize = RetteHandler_Resize;
```

`.AutoCompleteSaveForm()` permanentes Speichern von Daten eines Formulars in den AutoComplete-Data Store  
**Diese Methode ist mit Vorsicht zu genießen und richtet sich nach dem aktuellen Einstellungen von AUTOCOMPLETE, die der User in den Browser-Optionen treffen kann.**  
 Gespeichert werden die Werte von INPUT's im Formular, die das Attribut NAME besitzen,  
 wobei auch INPUT TYPE=password speicherbar ist  
 Hinweis: Im Formular-Tag ist ebenfalls NAME zu kodieren  
 NAME-Attribut ist auch Voraussetzung für das Senden von Formulardaten an den Server der Webseite  
 AutoComplete-Data Store kann vom User gelöscht werden (auch komponentenweise).

Beispiel:

```
<SCRIPT>
function Speichern()
{
    // erst speichern
    window.external.AutoCompleteSaveForm(ID_Formular);
    // dann löschen
    ID_Formular.ID_Input1.value="";
    ID_Formular.ID_Input2.value="";
}
</SCRIPT>
<FORM NAME="ID_Formular">
  Dieser Text wird gespeichert:
  <INPUT TYPE="text" NAME="ID_Input1">
  Dieser Text wird nicht gespeichert:
  <INPUT TYPE="text" NAME="ID_Input2" AUTOCOMPLETE="off">
</FORM>
<INPUT TYPE="button" VALUE="Speichern" onclick="Speichern()">
```

`.AutoScan()` Url einer beliebigen Webseite festlegen, die als Fehlermeldung vom Autoscan angezeigt wird  
 Webseite wird aufgerufen, wenn Autoscan nicht erfolgreich war  
 erklärt den Misserfolg  
**muss immer erreichbar, also anzeigbar sein**  
 Standardwebseite auf Festplatte des User-PC vorhanden (falls der User diese nicht manuell gelöscht hat)  
 Autoscan: Suchen nach einer anzuzeigenden Web-Seite im Internet durch den Browser nach Eingabe der Url (auch mit Autovervollständigung)  
 Voraussetzungen für Autoscan:  
 es muss aktiv sein  
 Autoscan leider nur möglich für Domains mit  
 "www" am Anfang der Url  
 Suffixe der Url laut Registry-Eintrag  
 HKEY\_LOCAL\_MACHINE\software\microsoft\internet explorer\main\urltemplate  
 (standardgemäß steht dort .com, .org, .net, und .edu )  
 Suche in der Reihenfolge der Einträge laut dem Registry-Eintrag  
 Hinweis: Der Registry-Eintrag ist manuell änderbar:  
 Bsp.: .de als Eintrag hinzufügen an gewünschte Position  
 siehe Objekt window.external

Beispiel:

```
window.external.AutoScan("test","error.htm","_main");
// für www.test.xxx mit xxx als Domain-Suffix laut Registry-Eintrag (siehe oben)
```

`.back()` Aktivierung einer zur aktuell besuchten Seite im Verlauf des Surfens davorliegenden Seite

`.beginElement()` Element auf der Timeline starten, also aktivieren  
 identisch mit Wirkung, wenn Startzeit des Elementes erreicht wird auf der Timeline  
 Alle Kinder des Elementes erhalten Information über Start des Elternelementes und sind somit korrekt auf der Timeline des Elternelementes.  
 per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
 siehe Objekt `currTimeState` und `Behavior.style.time2`  
 -1 entspricht wie wenn nicht kodiert



entspricht Druck des Back-Buttons  
 je kleiner umso weiter zuvorliegend  
 entspricht Selektion aus der Verlaufsliste

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:SEQ ID="ID_Seq"
CLASS="time_line_klasse"
>
<SPAN ID="ID_Span1"
CLASS="time_line_klasse"
TIMEACTION="display"
DUR="1"
>
5...
</SPAN>
<SPAN ID="ID_Span2"
CLASS="time_line_klasse"
TIMEACTION="display"
DUR="1"
>
4...
</SPAN>
<BR>
<IMG ID="ID_Img"
SRC="test.gif"
CLASS="time_line_klasse"
DUR="indefinite"
>
</t:SEQ>
<BR>
<BUTTON ID="ID_Button" onclick="ID_Seq.beginElement();">
Start der Timeline
</BUTTON>
</BODY>
</HTML>
```

.beginElementAt() Element auf der Timeline zu einem spezifischen Zeitpunkt ab Start der Timeline starten (aktivieren)  
 also mit Wartezeit ab Beginn der Timeline des Elementes  
 wenn das Elternelement diesen Zeitpunkt schon erreicht hat, so startet das Kindelement sofort,  
 also ohne Wartezeit  
 wenn zusätzlich ein weiterer Start (oder mehrere Starts)  
 mit identischem oder kleinerem Zeitpunkt kodiert, so wird dieser zusätzliche ignoriert  
 mit größerem Zeitpunkt kodiert, so wird dieser Start auch ausgeführt  
 also Neustart nach einem erfolgten Start  
 per Eigenschaft .isActive das Element auf Aktivsein prüfen  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Starten()
{
// prüfen ob Elternobjekt, also body, den Startzeitpunkt schon erreicht hat
// bzw. der Startwert falsch ist
if ( (ID_Input.value <= document.body.currTimeState.activeTime)
|| (ID_Input.value==null)
)
{
alert('Startwert falsch!');
ID_Input.value="";
ID_Input.focus();
return;
}
}
```



```

        else
        { ID_Excl.beginElementAt(ID_Input.value); }
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <SPAN ID="ID_Span2"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Excl.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <t:EXCL ID="ID_Excl"
        BEGIN="0"
        CLASS="time_line_klasse"
        DUR="27"
    >
        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            BEGIN="1"
            DUR="5"
        >
            Erste Zeile
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            BEGIN="6"
            DUR="5"
        >
            Zweite Zeile
        </DIV>
    </t:EXCL>
    <INPUT type="text" ID="ID_Input" SIZE="3" >
    <BUTTON ID="ID_Button" onclick="Starten()">
        Start
    </BUTTON>
</BODY>
</HTML>

```

.big()                      HTML-Tag <BIG> erzeugen in der Form <BIG>text</BIG>  
 siehe Script-Objekt String

Beispiel:

```

var StringLiteral    ="Text der BIG wird"
var HTML_Kette      = StringLiteral.big();
HTML_Kette           = "Text der BIG wird".big();

```

entspricht <BIG>Text der BIG wird</BIG>

eval(HTML\_Kette);                      erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
 document.write(HTML\_Kette);

.blink()                      HTML-Tag <BLINK> erzeugen in der Form <BLINK>text</BLINK>  
 siehe Script-Objekt String

Beispiel:

```

var StringLiteral    ="Text der BLINK wird"
var HTML_Kette      = StringLiteral.blink();
HTML_Kette           = "Text der BLINK wird".blink();

```

entspricht <BLINK>Text der BLINK wird</BLINK>



eval(HTML\_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
document.write(HTML\_Kette);

.blur() Element den Focus wegnehmen und Event onblur auslösen  
Der Focus wird nicht automatisch auf irgend ein anderes Element gesetzt !  
vor IE 5.0 TABINDEX-Attribut muss kodiert sein  
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein

.blur() Element den Focus wegnehmen und Event onblur auslösen  
Der Focus wird **nicht automatisch** auf irgend ein anderes Element gesetzt !  
vor IE 5.0 TABINDEX-Attribut muss kodiert sein  
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein  
siehe Objekt window

.bold() HTML-Tag <B> erzeugen in der Form <B>text</B>  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral = "Text der BOLD wird"
var HTML_Kette = StringLiteral.bold();
HTML_Kette = "Text der BOLD wird".bold();

entspricht <B>Text der BOLD wird</B>
```

eval(HTML\_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
document.write(HTML\_Kette);

Boolean() konvertiert einen Wert nach Boolean

.BuildPath() PATH-Variable im Dateisystem erweitern durch Anhängen  
keine Erweiterung des Dateisystems per Ordner (dafür gibt es andere Methoden)

.call() eine Methode eines anderen Objektes aufrufen: Argumentenliste beachten  
siehe Script-Objekt Funktion und Script-Objekt arguments und Anweisung function

.ceil() nächste ganze Zahl oberhalb zahl ermitteln  
Bsp: ceil(1.1) ergibt 2  
ceil(1) ergibt 1  
siehe Script-Objekt Math

.charAt() Zeichen liefern unter Nutzung des Indexes  
siehe Script-Objekt String

Beispiel 1:

```
var StringLiteral = "Text"
StringLiteral.charAt(0) liefert "T"
StringLiteral.charAt(5) liefert Leerkette
"Text".charAt(0) liefert "T"
```

Beispiel 2 Zeichenkette auf Wertebereich der Zeichen prüfen:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function pruefe_zeichenkette(zeichenkette, wertebereich)
{
    // wertebereich ist Zeichenkette z.B. "0123456789 -+/,()"

    var rueckgabewert = true; // Annahme: Zeichenkette hat NUR Zeichen
                             // aus dem Wertebereich

    var zeichen_aus_zeichenkette;

    // Zeichenkette zeichenweise analysieren: Jedes Zeichen mit Wertebereich vergleichen
    for (var i = 0; i < zeichenkette.length; i++)
    {
        zeichen_aus_zeichenkette = zeichenkette.charAt(i);

        if (wertebereich.indexOf(zeichen_aus_zeichenkette) == -1)
        {
            rueckgabewert = false;
            break; // ungültiges Zeichen gefunden, also abbrechen
        }
    }

    return rueckgabewert;
}
```



```

function pruefe_eingabe(zu_pruefende_zeichenkette)
{
    if (pruefe_zeichenkette(zu_pruefender_zeichenkette, "0123456789 -+/,()"))
    {alert("Eingabe ist korrekt !");}
    else
    {alert("Eingabe ist nicht korrekt !"); }
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
    Telefon:
    <INPUT  TYPE="text"
            NAME="Telefon"
            VALUE=""
        >
    <INPUT  TYPE="button"
            VALUE="Ueberpruefen"
            onclick="pruefe_eingabe(this.form.Telefon.value)">
</FORM>
</BODY>
</HTML>

```

**.charCodeAt()** Unicode des Zeichen liefern unter Nutzung des Indexes  
Unicode von 0 bis 65535, wobei  
0 bis 127 der ASCII-Zeichensatz ist  
0 bis 255 der ISO-Latin-1-Zeichensatz ist  
siehe Script-Objekt String

Beispiel:

```

var StringLiteral ="Text";
StringLiteral.charCodeAt(0)    liefert 84 für "T"
StringLiteral.charCodeAt(5)    liefert NaN
"Text".charCodeAt(0)           liefert 84 für "T"

```

**.clear()** Dokument löschen  
nur Objekt document

**.clear()** Selektion als Markierung aufheben per document.selection Objekt des Dokumentes per document.selection  
nicht Selektion löschen  
siehe Methode .empty()

**.clearAttributes()** alle HTML-Attribute eines Objektes entfernen  
außer ID, STYLE und per Script definierte Attribute  
Script-erzeugte Attribute nicht entfernbar  
DOM wird geändert

Beispiel:

```

<SCRIPT>
function Loeschen()
{
    ID_Span.children[0].clearAttributes();
}
</SCRIPT>
<SPAN ID="ID_Span">
    <DIV ID="ID_Div"
        ATTRIBUTE1="true"
        ATTRIBUTE2="true"
        onclick="alert('click');"
        onmouseover="this.style.color='#0000FF';"
        onmouseout="this.style.color='#000000';"
    >
        Test eines<B>Div</B>Elementes.
    </DIV>
</SPAN>
<INPUT TYPE="button" VALUE=" Loeschen" onclick="Loeschen()">

```

**.clearComponentRequest()** Puffer der per .addComponentRequest() angeforderten Downloads löschen  
Behavior .style.clientCaps

**.clearData()** Clipboardinhalt löschen  
Anwendung für Ereignisse ondragstart oder ondrop  
"Text" zu löschende Daten sind Text-Format  
"URL" zu löschende Daten sind im URL-Format.



"File" zu löschende Daten sind im File-Format.  
 "HTML" zu löschende Daten sind im HTML-Format  
 "Image" zu löschende Daten sind im Image-Format  
 Leerkette  
 wenn **nichts** kodiert, so werden **alle** Datenformat im Clipboard **gelöscht**

.clearInterval() stoppt einen Timer, der mit .setInterval() gestartet wurde  
 siehe Objekt window

Beispiel 1:

```
var timerID = null;

function timer()
{
    // tue was
}

function starttimer()
{
    if (timerID == null)
    {timerID = setInterval("timer()", 1000);}
}

function stoptimer()
{
    if (timerID != null)
    {
        clearInterval(timerID);
        timerID = null;
    }
}
```

Beispiel 2 für Sekundenbalken:

```
<HTML>
<HEAD>
<STYLE>
    BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
    var timerID = null;
    var Zahler = 0;

    function Init()
    {
        // DIV-Eigenschaften festlegen

        // DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
        ID_Div1.style.setExpression("width","Zahler *10");

        // DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
        ID_Div2.setExpression("innerText","Zahler.toString()");
    }

    function Uhr()
    {
        // Sekunden kumulieren
        Zahler ++;

        // und Anzeige neu berechnen
        document.recalc();
    }

    function Starten()
    {
        if (timerID == null)
        {
            // Start-Button nicht aktivierbar machen
            ID_Button1.disabled = true;

            // Stop-Button aktivierbar machen
            ID_Button2.disabled = false;

            // Uhr neu starten
            timerID = setInterval("Uhr()", 1000);
        }
    }
}
```





```

    }
}

function Stoppen()
{
    if (timerID != null)
    {
        clearInterval(timerID);
        timerID = null;
        ID_Button1.disabled = false;
        ID_Button2.disabled = true;
    }
}

function ZurueckSetzen()
{ Zahler = 0; }
</SCRIPT>
</HEAD>
<BODY onload="Init()">
    <DIV ID="ID_Div1" STYLE="background-color:lightblue" ></DIV>
    <DIV ID="ID_Div1" STYLE="color:hotpink;font-weight:bold"></DIV>
    <BR>
    <BUTTON ID="ID_Button1"
        onclick="Starten()"
    >
        Start
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button2"
        DISABLED="true"
        onclick="Stoppen()"
    >
        Stop
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button3"
        onclick="ZurueckSetzen()"
    >
        Reset
    </BUTTON>
    <BR>
    <P STYLE="width:200;color:white;background-color:gray">
        Sekundenbalken
    </P>
</BODY>
</HTML>

```

`.clearRequest()` Queue für Lesezugriff (Request Queue) leeren  
siehe Objekt `navigator.userProfile`

Beispiel :

```

// Request Queue füllen

// es soll der Wert zu "vcard.displayname" ermittelt werden
navigator.userProfile.addReadRequest("vcard.displayname");
// es soll der Wert zu " vcard.gender " ermittelt werden
navigator.userProfile.addReadRequest("vcard.gender");

// Datenermitteln per Queue
navigator.userProfile.doReadRequest(12, "Daten von unbekannt verwendet");

// Queue leeren
navigator.userProfile.clearRequest();

```

`.clearTimeout()` löscht ein Timeout, das mit `.setTimeout()` gesetzt wurde  
siehe Objekt `window`

`.click()` simuliert einen Klick auf das Element und löst onclick-Event aus  
manipuliert nicht den Focus

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function ClickMitFocus()
    {

```



```

        ID_CheckBox.focus();
        ClickOhneFocus();
    }

    function ClickOhneFocus ()
    { ID_CheckBox.click();}
</SCRIPT>
<SCRIPT FOR= ID_CheckBox EVENT=onfocus>
    alert("Check Box hat Focus");
</SCRIPT>
</HEAD>
<BODY>
    <INPUT Type="CHECKBOX" ID="ID_CheckBox"></INPUT>
    <BR>
    <BUTTON onclick="ClickMitFocus()">Klick mit Fokus</BUTTON>
    <BR>
    <BUTTON onclick="ClickOhneFocus()">Klick ohne Fokus</BUTTON>
</BODY>
</HTML>

```

`.cloneNode()` Objekt klonen und Referenz des erzeugten Klone liefern  
DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)

Beispiel:

```

<SCRIPT>
    function Klonen()
    {
        var ZeigerAufKlone = Liste.cloneNode(true); // DOM wird nicht geändert
        document.body.insertBefore(ZeigerAufKlone); // DOM wird geändert
    }
</SCRIPT>
<UL ID = "Liste" >
    <LI>Listenelement 1
    <LI>Listenelement 2
    <LI>Listenelement 3
    <LI>Listenelement 4
</UL>
<INPUT TYPE="button" VALUE=" Klonen" onclick=" Klonen()">

```

`.close()` Ausgabe-Datenstream schliessen und Anzeige des Dokumentes beenden  
schliessen einer mit `.open()` erzeugten Dokument-Instanz  
Objekt document

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function ErzeugeZurLaufZeit()
    {
        // Dokumentinstanz erzeugen
        var ID_Dokument = document.open("text/html", "replace");

        // Dokumentinhalt festlegen und anzeigen
        ID_Dokument.write("<HTML><BODY>Hallo</BODY></HTML>");

        // Dokumentinstanz schliessen
        ID_Dokument.close();
    }
</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE="button" onclick=" ErzeugeZurLaufZeit();">
</BODY>
</HTML>

```

Beispiel:

```

function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
    FensterZeiger.close();
}

....

// Fenster öffnen
var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");

```



```
var FensterDokumentZeiger = FensterZeiger.document;
```

```
var Kette= '<HTML>'
          + '<HEAD></HEAD>'
          + '<BODY>'
          + '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
          + '<BR>'
          + '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
          + ' onclick="opener.OnClickHandler();"
          + '>'
          + '</BODY>'
          + '</HTML>';
```

```
FensterDokumentZeiger.open("text/html");
FensterDokumentZeiger.write(Kette);
FensterDokumentZeiger.close();
```

.close() Fenster schliessen, das offen ist  
 Fenster muss nicht explizit mit Methode .open( )Methode erzeugt worden sein  
 Schliessen des letzten Browserfensters erzeugt immer Dialog-Box-Abfrage  
 Fenster des Dokumentes schliessen: document.close()  
 innerhalb von Eventhandler: Immer **window.close()** oder logischer\_window\_name.close() oder self.close()  
 kodieren  
 siehe Objekt window

Beispiel:

```
<BODY onclick="window.close();">
  Klick zum Schliessen des Fensters
</BODY>
```

Beispiel für Fenster schliesst sich selbst nach Wartezeit:

```
</HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function fenster_offnen_und_schliessen()
    {

        var fenster;
        fenster=window.open("", "Fenster", "width=180,height=100");
        fenster.document.write("<H1>Ich schließe mich nach 4 Sekunden</H1>");
        fenster.setTimeout('window.close()',4000);

    }

-->
</SCRIPT>
</HEAD>
<BODY onload="fenster_offnen_und_schliessen()">
</BODY>
</HTML>
```

Beispiel:

```
function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
    FensterZeiger.close();
}
```

....

```
// Fenster öffnen
var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");
var FensterDokumentZeiger = FensterZeiger.document;
```

```
var Kette= '<HTML>'
          + '<HEAD></HEAD>'
          + '<BODY>'
          + '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
          + '<BR>'
          + '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
          + ' onclick="opener.OnClickHandler();"
          + '>'
          + '</BODY>'
          + '</HTML>';
```



```
FensterDokumentZeiger.open("text/html");
FensterDokumentZeiger.write(Kette);
FensterDokumentZeiger.close();
```

<code>.Close()</code>	offene Datei schliessen nach Schliessen der Datei sind keine Methoden mehr verwendbar, ohne die Datei vorher neu zu öffnen
-----------------------	---

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
```

<code>.collapse()</code>	Textbereich-Zeichen-Zeiger auf Anfang oder Ende des Textbereiches setzen
--------------------------	--

Hinweis: Zeiger nur auf Plain-Text-Zeichen

Bsp.: <BODY><P><B>abc

Zeiger kann wandern von VOR a bis HINTER c

VOR a entspricht Start des Bereiches

mit Zeigerwert 0

HINTER c entspricht Ende des Bereiches

mit Zeigerwert 3

per textrange Objekt  
nur unter Windows 32-Bit

Beispiel:

```
<SCRIPT>
function Selektion()
{
    var ZeigerAufSelektionsQuelle = window.event.srcElement ;

    if (!ZeigerAufSelektionsQuelle.isTextEdit)
    { ZeigerAufSelektionsQuelle = ZeigerAufSelektionsQuelle.parentTextEdit;}

    if (ZeigerAufSelektionsQuelle != null)
    {
        var ZeigerAufTextBereich =
            ZeigerAufSelektionsQuelle.createTextRange();

        ZeigerAufTextBereich.moveToElementText(window.event.srcElement);
        ZeigerAufTextBereich.collapse();
        ZeigerAufTextBereich.expand("SelektionsText");
        ZeigerAufTextBereich.select();
    }
}
</SCRIPT>
```

<code>.compareEndpoints()</code>	Vergleich der Textbereich-Zeichen-Zeiger von 2 Textbereichen
----------------------------------	--

Hinweis: Zeiger nur auf Plain-Text-Zeichen

Bsp.: <BODY><P><B>abc

Zeiger kann wandern von VOR a bis HINTER c

VOR a entspricht Start des Bereiches

mit Zeigerwert 0

HINTER c entspricht Ende des Bereiches

mit Zeigerwert 3

per textrange Objekt  
nur unter Windows 32-Bit

<code>.compareVersions()</code>	2 Versionen einer Komponente vergleichen
---------------------------------	--

Bsp. für Version "5,0,18,1024"

Behavior .style.clientCaps

<code>.componentFromPoint()</code>	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout
------------------------------------	--

onmouseover-Event hat **nicht die Pixelgenauigkeit** wie die Angaben der Methode `.componentFromPoint()` also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben

Overbereich der Maus ist mehr als 1 Pixel gross

beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    function ComponentSnapshot()
    {
        var LayoutKomponente =
            document.body.componentFromPoint( event.clientX,
```



```

        event.clientY
    );

    if (LayoutKomponente == "")
    {alert("Keine Layout-Komponente");}
    else
    {alert("Layout-Komponenten = " + LayoutKomponente);}
}

function trackElement()
{
    var LayoutKomponente =
        document.body.componentFromPoint( event.clientX,
        event.clientY
        );

    window.status =
        "mousemove "
        + event.clientX
        + ", "
        + event.clientY
        + " Layout-Komponente = "
        + LayoutKomponente;
}
</SCRIPT>
</HEAD>
<BODY onmousemove="trackElement()"
onmousedown="ComponentSnapShot()"
onkeydown="ComponentSnapShot()"
oncontextmenu="ComponentSnapShot()"
>

    <TEXTAREA COLS=500 ROWS=500>
        Dieser Text erzeugt Scrollbalken.
    </TEXTAREA>
</BODY>
</HTML>

```

.concat()

Feld (Quellfeld1) kopieren in eine neue und automatisch erzeugte Instanz (Zielfeld) und optionales Anhängen von Werten (z.B. weiteren Quellfeldern2 ... ) an das Ende des Zielfeldes. Quellfeld1 kann leer sein

Verkettung erfolgt in der Folge der Kodierung der zu verkettenden Objekte, Ausdrücke etc.:

Erstes Objekt in der Verkettung ist immer Quellfeld1.

wenn mehrere Quellfelder zu verketteten sind, dann gilt

- wenn Feldelement **nicht** String **und nicht** Number ist, so
  - wird das Feldelemente als Zeiger kopiert bzw. verkettet
  - der Wert, auf den der Zeiger weist, wird nicht verwendet
  - bewirkt Änderung eines Wertes im Quell- bzw. Zielfeld auch die Änderung im Ziel- bzw. Quellfeld, da identische Zeigerbezüge vorliegen
- wenn Feldelement String oder Number ist, so
  - erfolgt Wertekopierung, also keine Zeigerkopierung
  - bewirkt Änderung eines Wertes im Quell- bzw. Zielfeld keine Änderung im Ziel- bzw. Quellfeld, da keine Zeigerbezüge vorliegen

Änderung der Anzahl der Feldelemente in den Quellfeldern hat keine Auswirkung auf die Anzahl der Feldelemente im Zielfeld

Verkettung von Feldern erfolgt

- feldweise in der Folge der Kodierung der Quellfelder
- und pro Quellfeld: elementweise in der Folge der Feldelemente

Felder haben die Objektklasse Script-Objekt Array

ab IE 5.5 und NS 6.x siehe auch .push()

Beispiel:

```

var Feld1      = new Array(0,1);
var Feld2_Quelle = new Array(2, 3);
var Wert_Quelle = 4;
var Feld3_Ziel = Feld1.concat(Feld2_Quelle, Wert_Quelle,);
alert(Feld3.join()); // "0,1,2,3,4"

```

Beispiel:

```

var QuellFeld1=new Array("a","b","c");
var QuellFeld2=new Array(1,2,3);
var ZielFeld=QuellFeld1.concat(QuellFeld2) // Zielfeld enthält Wertkopien also ["a","b","c",1,2,3]

```

Beispiel:

```

var Variable1="Hallo ";
var Variable2="Du";
var Variable3="!";

var QuellFeld1=new Array(Variable1,Variable2,Variable3);

```



```

var QuellFeld2=new Array("Halo ","Du","!");
var ZielFeld=QuellFeld1.concat(QuellFeld2);
// Zielfeld enthält Wertkopien also [
//                                     Zeiger_Auf_Variable1,
//                                     Zeiger_Auf_Variable2,
//                                     Zeiger_Auf_Variable3,
//                                     "Halo ","Du","!"
//                                     ]

```

**.concat()** Verkettung von String-Objekten bzw. String-Literalen zu einem neuen String-Objekt  
Alternative: + Operator der Stringverkettung  
siehe Script-Objekt String

Beispiel:

```

var StringLiteral1="Halo ";
var StringLiteral2="Du ";
var StringLiteral3="!";
var Kette=StringLiteral1.concat(StringLiteral2,StringLiteral3) // "Halo Du !"
var Kette="Halo ".concat("Du ","!") // "Halo Du !"

```

**.confirm()** Dialogbox erzeugen mit  
1. Anzeige Fragezeichen, String, OK/Abbrechen-Button  
2. warten auf Drücken eines der beiden Button  
siehe Objekt window

**.contains()** prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist  
DOM nicht geändert  
true Element liegt innerhalb eines Elementes  
false Element liegt nicht innerhalb eines Elementes

Beispiel:

```

<SCRIPT>
function TesteMaus(Zeiger)
{
    if( ! Zeiger.contains(event.fromElement) )
    {alert("Maus über Button erkannt");}
}
</SCRIPT>
<BUTTON onmouseover=" TesteMaus(this)">Ueberfahre mich mit der Maus</BUTTON>

```

**.Copy()** vorhandene Datei kopieren (sonst Fehler erzeugt)

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
Datei.Copy("c:\\windows\\desktop\\test2.txt");

```

**.Copy()** vorhandenen Ordner kopieren (sonst Fehler erzeugt)

Beispiel:

```

var OrdnerName      = "c:\\windows\\desktop\\";
var DateiSystem     = new ActiveXObject("Scripting.FileSystemObject");
var Ordner          = DateiSystem.GetFolder(OrdnerName);
Ordner.Copy("d:\\recycled\\");

```

**.CopyFile()** Datei(en) kopieren  
Wenn Copy abbricht, so bleiben bisher erfolgte Kopieraktionen leider erhalten

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CopyFile ("c:\\eigene dateien\\*.doc", "c:\\recycled\\")

```

**.CopyFolder()** Ordner kopieren  
Wenn Copy abbricht, so bleiben bisher erfolgte Kopieraktionen leider erhalten

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CopyFolder("c:\\*", "d:\\recycled\\")

```

**.cos()** Cosinus im Bogenmass  
liefert Wert von -1 bis +1  
siehe Script-Objekt Math

**.createAttribute()** ein Attribut im Dokument erzeugen und Referenz auf das erzeugte Attribut liefern  
Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!  
DOM nicht geändert

Beispiel:

```

<HTML>
<HEAD>

```



```

<SCRIPT>
    function Hinzufuegen()
    {
        // Attribut mit Wert erzeugen
        var ZeigerAufAttribut = document.createAttribute("title");
        ZeigerAufAttribut.value = "Tooltip-Text ";

        // Attribut als Feldelement anhängen
        var ZeigerAufFeld = ID_Div.attributes;
        ZeigerAufFeld.setNamedItem(ZeigerAufAttribut);
    }
</SCRIPT>
</HEAD>
<BODY onload=" Hinzufuegen();">
    <DIV ID="ID_Div">Es wird ein Tooltip-Text hinzugefuegt</DIV>
</BODY>
</HTML>

```

`.createCaption()` leeres CAPTION in einer Tabelle erzeugen **und** einbinden  
es darf nur 1 CAPTION zur Tabelle existieren  
siehe Objekt table

`.createComment()` Comment-Objekt (Kommentar-Objekt) im Dokument erzeugen und Referenz liefern  
verwendbar anstelle von direkt im HTML- bzw. Script-Quellcode kodiertem Kommentar  
DOM wird geändert, da das Dokument erweitert wird

Beispiel:

```
var Kommentar = document.createComment("Das ist ein Kommentar");
```

`.createControlRange()` Selektionsbereich erzeugen  
es kann nur genau 1 Range pro Zeitpunkt existieren: wenn einer vorhanden, so diesen überschreiben

Beispiel 1:

```
var Zeiger = document.body.createControlRange();
```

Beispiel 2:

```

function ControlRangeErzeugenUndSelektieren()
{
    var ControlRangeObjekt = document.body.createControlRange();
    ControlRangeObjekt.add(document.all.zeiger_auf_control_element);
    ControlRangeObjekt.select();
}

```

`.createDocumentFragment()` erzeugt neues Dokument  
Objekt document

`.createElement()` HTML-Objekt (Tags) im Dokument erzeugen und Referenz liefern  
Achtung: Erzeugtes Objekt muss in DOM noch per Methode `.insertBefore()` bzw. `.appendChild()` eingereiht werden.  
Hinweis: Attribute mit der Methode `.createAttribute()` erzeugen  
DOM wird geändert

Beispiel 1:

```

<SCRIPT>
    function Erzeugen()
    {
        ID_Span.innerHTML="";
        var FeldDerZeigerAufOption = ID_Select.options[ID_Select.selectedIndex];

        if(FeldDerZeigerAufOption.text.length>0)
        {
            var ZeigerAufElement=
                document.createElement(FeldDerZeigerAufOption.text);
            eval(
                " ZeigerAufElement."
                + FeldDerZeigerAufOption.value
                + "="
                + ID_Input.value
                + ""
            );

            if(FeldDerZeigerAufOption.text=="A")
            { ZeigerAufElement.href="javascript:alert('A link.');" }

            ID_Span.appendChild(ZeigerAufElement);
        }
    }
</SCRIPT>
<SELECT ID="ID_Select" onchange="Erzeugen()">

```



```

        <OPTION VALUE="innerText">A
        <OPTION VALUE="value">&lt;INPUT TYPE="button"&gt;
    </SELECT>
    <INPUT TYPE="text" ID="ID_Input" VALUE="Beispiel Text">
    <SPAN ID="ID_Span" ></SPAN>

```

Beispiel 2:

```

<HTML>
<HEAD>
<SCRIPT>
    function Erzeuge()
    {
        var Zeiger = document.createElement(
            "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Eins'>"
        );
        document.body.insertBefore(Zeiger);

        Zeiger = document.createElement(
            "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Zwei'>"
        );
        document.body.insertBefore(Zeiger);
    }
</SCRIPT>
</HEAD>
<BODY>
    <INPUT TYPE="BUTTON"
        ONCLICK=" Erzeuge()"
        VALUE="Zwei Radio Buttons erzeugen">

    <BR>
    <INPUT TYPE="BUTTON"
        ONCLICK="alert(document.body.outerHTML)"
        VALUE="Click um HTML zu sehen">

    <BODY>
</HTML>

```

.createEventObject()      Event-Objekt im Dokument erzeugen nur für Methode .fireEvent()

Beispiel:

```

var EventObjekt = document.createEventObject();
document.fireEvent("onclick", EventObjekt);

```

.createEventObject()      Event erzeugen in einem Eventhandler einer HTC-Komponente  
siehe Objekt element und HTC-Datei

Beispiel:

test.htc enthält:

```

<PUBLIC:COMPONENT NAME="HTC_Komponente">
    <PUBLIC:EVENT NAME="onResultChange" ID="ID_Event"/>

    <SCRIPT LANGUAGE="JScript">
        function Berechne()// Bezug über ID_Event
        {
            var EventObjekt    = createEventObject();
            EventObjekt.result  = "Wert";

            ID_Event.fire (EventObjekt);
        }
    </SCRIPT>
</PUBLIC:COMPONENT>

```

HTML-Dokument enthält:

```

<HTML XMLNS:privater_tag_namensraum>
<HEAD>
<STYLE>
    @media all { privater_tag_namensraum:\privater_tag {behavior:url(test.htc)} }
</STYLE>
</HEAD>
<BODY>
    <privater_tag_namensraum:privater_tag
        ID="ID_privater_tag"
        onResultChange="ID_Div.innerText=window.event.result"
    >

    <TABLE>

```





```

<TR>
  <DIV ID="ID_Div"
    ALIGN=RIGHT
    STYLE="border: '.025cm solid gray"
  >
    0.
  </DIV>
</TR>
<TR>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 7 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 8 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 9 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" / ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" C ">
  </TD>
</TR>
<TR>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 4 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 5 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 6 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" * ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" % " DISABLED>
  </TD>
</TR>
<TR>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 1 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 2 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 3 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" - ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 1/x " DISABLED>
  </TD>
</TR>
<TR>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" 0 ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" +/- ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" . ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" + ">
  </TD>
  <TD>
    <INPUT TYPE=BUTTON VALUE=" = ">

```



```

</TD>
</TR>
</TABLE>
</privater_tag_namensraum:privater_tag>
<BODY>
</HTML>

```

`.CreateFolder()` Ordner erzeugen, der nicht bereits existieren darf (sonst wird Fehler erzeugt)

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CreateFolder("c:\\test")

```

`.createPopup()` Pop-up-Fenster ohne irgendwelche Fensterelemente öffnen z.B. für Anzeige von Tooltips

siehe Objekt `window.popup`

Fenster wird mit der Anzeige per Methode `.show()` zum aktuellen Fenster erst geschlossen, wenn **anderes Fenster** aktuell wird  
z.B. durch Klicken außerhalb des Pop-up-Fensters

ab IE 5.5

siehe Objekt `window`

Beispiel

```

var PopUpID=window.createPopup();
PopUpID.document.body.style.backgroundColor="green";
PopUpID.document.body.innerHTML="TooltipText";
PopUpID.show(100,100,180,25,document.body);

```

`.createRange()` Zeiger auf einen Universal-Bereich erzeugen per `document.selection` Objekt des Dokumentes

Universal-Bereich kann enthalten:

Text (document.selection.textRange Collection  
textRange Objekt)

oder Control-Element(e) (document.selection.controlRange Collection)

siehe auch Methoden `.createControlRange()`, `.createTextRange()` und `.createRangeCollection()`  
Eigenschaft `.type`

`.createRangeCollection()` `document.selection.textRange` Collection erzeugen per `document.selection` Objekt des Dokumentes

nur wenn der Browser multiple Selektion unterstützt, so mehr als 1 Feld-Element `textRange` Objekt  
vorhanden bei Mehrfach-Selektion durch User

siehe auch `textRange` Objekt

Methoden `.createRange()`, `.createControlRange()` und `.createTextRange()`

`.createStyleSheet()` Style-Sheet-Objekt im Dokument erzeugen und Referenz liefern

DOM wird geändert

Beispiel:

```

document.createStyleSheet('styles.css');

```

`.CreateTextFile()` Textdatei anlegen und zum Schreiben als Textstream öffnen

Schreiben und Schliessen der Datei per Objekt `FileSystemObject.TextStream`

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.CreateTextFile("c:\\test.txt", true)
DateiOffen.WriteLine("Test");
DateiOffen.Close();

```

`.createTextNode()` Plain-Textelement im Dokument erzeugen und Referenz liefern

Plaintext kann keine HTML-Tags enthalten

DOM wird geändert, da Dokument erweitert wird

Beispiel:

```

<SCRIPT>
function TextElementAendern()
{
    var ZeigerAufTextElement = document.createTextNode("Neuer Text");
    var ZeigerAufSpanInhalt = ID_Span.childNodes(0);
    ZeigerAufSpanInhalt.replaceNode(ZeigerAufTextElement);
}
</SCRIPT>
<SPAN ID = "ID_Span" onclick="TextElementAendern()">
Original Text
</SPAN>

```

`.createTextRange()` Textbereich erzeugen

Beispiel 1:

```

<SCRIPT LANGUAGE="JScript">
var FeldAllerButtonElemente coll = document.all.tags("BUTTON");

```



```

        if ( (FeldAllerButtonElemente !=null )
            && (FeldAllerButtonElemente.length>0)
        )
        {
            var ZeigerAufRange = FeldAllerButtonElemente[0].createTextRange();
            ZeigerAufRange.text = "Clicked";
        }
    </SCRIPT>

```

Beispiel 2:

```

function ErzeugeUndSelektiereTextRange()
{
    var TextBereich = document.body.createTextRange();
    TextBereich.findText("Testtext");
    TextBereich.select();
}

```

Beispiel 3 Einbindung einer Suchmaschine:

```

var markierter_text=document.selection.createRange().text;
    // oder var markierter_text=prompt('Suchbegriff: ');
var suchmaschinen_url='http:// .....';
var suchmaschinen_parameter='.....';

if (markierter_text)
{location.HREF=suchmaschinen_url + suchmaschinen_parameter + escape(markierter_text);}
else
{location.HREF=suchmaschine_url; }

```

Beispiel für altavista: suchmaschinen\_url      'http://altavista.de/'  
suchmaschinen\_parameter      'cgi-bin/query?pg=q&what=web&q='

Hinweis: escape() setzt Umlaute und Leerzeichen in korrekte Darstellung um

.createTFOOT()	leeres TFOOT in einer Tabelle erzeugen <b>und</b> einbinden es darf nur 1 TFOOT zur Tabelle existieren siehe Objekt table
.createTHEAD()	leeres THEAD in einer Tabelle erzeugen <b>und</b> einbinden es darf nur 1 THEAD zur Tabelle existieren siehe Objekt table
decodeURI()	dekodiert einen Uniform Ressource Identifier (URI), der mit der Methode encodeURI() erzeugt wurde ersetzt die Methode unescape() welche deprecated ist ab IE 5.5
Beispiel:	<pre> alert(decodeURI("My%20phone%20 #%20is%20123-456-7890")); // erzeugt "My phone # is 123-456-7890" </pre>
decodeURIComponent()	dekodiert eine Komponente einer Uniform Ressource Identifier (URI), der mit der Methode encodeURI() erzeugt wurde ab IE 5.5
.Delete()	vorhandene Datei löschen (sonst Fehler erzeugt)
Beispiel:	<pre> var DateinameMitPfad               = "c:\\test.txt"; var Dateisystem                   = new ActiveXObject("Scripting.FileSystemObject"); var Datei                         = Dateisystem.GetFile(DateinameMitPfad); Datei.Delete(); </pre>
.Delete()	vorhandenen Ordner löschen (sonst Fehler erzeugt)
Beispiel:	<pre> var OrdnerName                   = "c:\\windows\\desktop\\"; var Dateisystem                   = new ActiveXObject("Scripting.FileSystemObject"); var Ordner                       = Dateisystem.GetFolder(OrdnerName); <b>Ordner.Delete();</b> </pre>
.deleteCaption()	CAPTION löschen aus Tabelle es darf nur 1 CAPTION zur Tabelle existieren siehe Objekt table
.deleteCell()	Zelle in die Zeile einer Tabelle löschen siehe Objekt table.tr



.deleteData() Teilkette aus einem Objekt entfernen

.DeleteFile() vorhandene Datei(en) löschen  
Wenn Delete abbricht, so bleiben bisher erfolgte Löschaktionen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.DeleteFile("c:\\*.txt", true);
```

.DeleteFolder() vorhandene Ordner löschen, wobei es egal ist, ob Daten und/oder Unterordner im jeweiligen Ordner liegen oder nicht  
Wenn Delete abbricht, so bleiben bisher erfolgte Löschaktionen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.DeleteFolder("c:\\*", true);
```

.deleteRow() Zeile löschen aus Tabelle  
siehe Objekt table  
siehe Objekt table.tBody  
siehe Objekt table.tHead  
siehe Objekt table.tFoot

.deleteTFoot() TFOOT der Tabelle löschen  
es darf nur 1 TFOOT zur Tabelle existieren  
siehe Objekt table  
siehe Objekt table.tBody

.deleteTHead() THEAD der Tabelle löschen  
es darf nur 1 THEAD zur Tabelle existieren  
siehe Objekt table  
siehe Objekt table.tBody

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler  
wobei Registrierung mit Methode .attachEvent() aktiviert wurde  
Abschalten = ordnet dem Window-Objekt das Event laut Parameter event\_ bezeichner zu, das **nicht** behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)

Beispiel:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
function CursorNeu()
{
    if (event.srcElement == element)
    {
        normalColor      = style.color;
        runtimeStyle.color = "red";
        runtimeStyle.cursor = "hand";
    }
}

function CursorNormal()
{
    if (event.srcElement == element)
    {
        runtimeStyle.color = normalColor;
        runtimeStyle.cursor = "";
    }
}

function EventEntfernen()
{
    detachEvent ('onmouseover', CursorNeu);           // nicht () kodieren !!!
    detachEvent ('onmouseout', CursorNormal); // nicht () kodieren !!
}

attachEvent ('onmouseover', CursorNeu);
attachEvent ('onmouseout', CursorNormal);
</SCRIPT>
```

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler, wobei Registrierung mit Methode .attachEvent() aktiviert wurde  
Achtung: Vor dem Neubelegen des Standard-Eventhandler diesen vor .attachEvent() retten und den Standard-Eventhandler **nach** .detachEvent() wieder einbinden (siehe Beispiel).



siehe Objekt window

Beispiel 1:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor      = style.color;
            runtimeStyle.color = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {
        if (event.srcElement == element)
        {
            runtimeStyle.color = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu); // nicht () kodieren !!!
        window.onmouseover = Rette_Standard_Eventhandler_OnMouseOver;

        detachEvent ('onmouseout', CursorNormal); // nicht () kodieren !!
        window.onmouseout = Rette_Standard_Eventhandler_OnMouseOut;
    }

    var Rette_Standard_Eventhandler_OnMouseOver = window.onmouseover;
    attachEvent ('onmouseover', CursorNeu);

    var Rette_Standard_Eventhandler_OnMouseOut = window.onmouseout;
    attachEvent ('onmouseout', CursorNormal);
</SCRIPT>
```

Beispiel 2:

```
function ResizeHandler()           // Homepage neu laden
{window.history.go(0);}

// Standardhandler retten
var RetteHandler_Resize = window.onresize;

// und neu belegen
window.onresize = ResizeHandler;    // Homepage neu laden
                                   // ohne () kodieren, damit nicht sofort ausgeführt wird

// und Start des Abfangens vom resize-Event
window.attachEvent('onresize', ResizeHandler);

.....

// irgendwann abschalten der Eventüberwachung
window.detachEvent('onresize', ResizeHandler);

// und Standardhandler wieder einbinden
window.onresize = RetteHandler_Resize;
```

.doComponentRequest()      Start des Download aller per .addComponentRequest() angeforderten Komponenten laut Puffer Behavior .style.clientCaps

.documentTimeToParentTime()      Wert der Timeline des Dokumentes in den korrespondierenden Wert der Timeline der Eltern des Elementes konvertieren  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
```



```

</STYLE>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onrepeat="innerText=parseInt(ID_Excl.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onrepeat="innerText=parseInt(ID_Excl.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <t:EXCL ID="ID_Excl"
        CLASS="time_line_klasse"
        REPEATCOUNT="3"
    >
        <DIV ID="ID_Div"
            CLASS="time_line_klasse"
            BEGIN="2s"
            DUR="5s"
        >
        </DIV>
    </t:EXCL>
    <BR>
    <SPAN ID="ID_Span">Punkt auf der Timeline der Eltern:</SPAN>
    <BR>
    <BUTTON ID="ID_Button"
        CLASS="time_line_klasse"
        DUR="1"
        REPEATCOUNT="indefinite"
        onclick="
            ID_Span.innerText=' Punkt auf der Timeline der Eltern: '
            + ID_Div.documentTimeToParentTime(
                ID_Excl.currTimeState.activeTime
            );"
        onrepeat="innerText=' documentTimeToParentTime = '"
    >
        documentTimeToParentTime ermitteln
    </BUTTON>
</BODY>
</HTML>

```

**.doImport()** Start des Import des Programmcodes aus einer \*.htc-Datei eines Behavior (Verhaltens) eines Elementes / Objektes  
 der Import bewirkt die Erzeugung eines Elementes in der Collection namespaces, welche Behavior dynamisch verwaltet  
 es kann binärer Code importiert werden  
 Hinweis: Standard-Behavior des Internet Explorer besitzen keine \*.htc-Datei und müssen nur importiert werden per #default#vordefinierter\_name\_des\_behavior

Beispiel 1:

```

<HTML XMLNS:Mein_NamensRaum>
<HEAD>
    <SCRIPT LANGUAGE="JScript">
        document.namespaces("Mein_NamensRaum").doImport("#default");
    </SCRIPT>
</HEAD>

```

Beispiel 2:

```

<HEAD>
<HTML XMLNS: Mein_NamensRaum>
</HEAD>
<BODY onload=StartImport(>
<SCRIPT>
    var NamensRaumObjekt;

    function StartImport()

```



```

{
    // HTML XMLNS füllt die Collection namespaces
    //      da nur ein Namensraum erzeugt wurde, ist also nur 1 Element
    //      in der Collection namespaces

    // Namensraum laut Collection referenzieren
    NamensRaumObjekt = document.namespaces[0];

    // Import des Programmcodes aus der Datei test.htc starten (Download)
    NamensRaumObjekt.doImport("test.htc");

    // prüfen ob Import (Download) beendet ist
    if (NamensRaumObjekt.readyState != "complete")
    {
        // Import läuft noch also Endeereignis abfangen per Eventhandler
        //      wenn Import beendet ist, soll die Einbindung des
        //      Tags in das body-Objekt erfolgen
        NamensRaumObjekt.attachEvent("onreadystatechange",
                                     TagInDasBodyObjektEinbinden
        ); // Funktion ohne () kodieren !!!
    }
    else
    {
        // Import erledigt, also den Tag in das BODY-Objekt einbinden
        addTagNamesToBody();
    }

    return true;
}

function TagInDasBodyObjektEinbinden()
{
    // prüfen ob Import (Download) beendet ist
    if (NamensRaumObjekt.readyState != "complete")
    {return;}
    else
    {
        // Eventabfangen abschalten
        NamensRaumObjekt.detachEvent( "onreadystatechange",
                                       TagInDasBodyObjektEinbinden
        );

        // Tagobjekt erzeugen
        //      Tag ist ein Textelement
        var TagObjekt = document.createElement("Mein_NamensRaum: MeinTag");

        // Textelement füllen zum Rendern
        TagObjekt.innerText = "ElementBehavior";

        // und in das body-Objekt einbinden
        document.body.appendChild(TagObjekt);
    }
}
}
</SCRIPT>
</BODY>
</HTML>

```

**Inhalt der test.htc**

```

<public:component tagName=MeinTag>
<public:attach event=ondocumentready onevent=TagAktion(/>
</public:component>
<script>
    function TagAktion()
    {
        element.document.bgColor = "red";
    }
</script>

```

Beispiel 3: Import des Standard-Behavior .style.time2

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }

```



```
</STYLE>
</HEAD>
```

**.doReadRequest()** Lesezugriff auf alle vCard-Datenarten, die laut aktueller Request Queue vorgegeben sind  
Lesezugriff auf einzelnen vCard-Wert per `.getAttribute()`  
siehe Objekt `navigator.userProfile` und Autovervollständigung im Internet Explorer

Beispiel :

```
// Request Queue füllen

// es soll der Wert zu "vcard.displayname" ermittelt werden
navigator.userProfile.addReadRequest("vcard.displayname");
// es soll der Wert zu "vcard.gender" ermittelt werden
navigator.userProfile.addReadRequest("vcard.gender");

// Datenermitteln per Queue
navigator.userProfile.doReadRequest(12, "Daten von unbekannt verwendet");

// Queue leeren
navigator.userProfile.clearRequest();
```

**.doScroll()** Klick auf Scrollbar simulieren  
Achtung: Scrollelemente müssen bereits vorhanden sein !

"scrollbarDown"	oder "down"	Down scroll Pfeil
"scrollbarHThumb"		horizontaler Scrollbalken
"scrollbarLeft"	oder "left"	Left scroll Pfeil
"scrollbarPageDown"	oder "pageDown"	Page-down Scrollbalken
"scrollbarPageLeft"	oder "pageLeft"	Page-left Scrollbalken
"scrollbarPageRight"	oder "pageRight"	Page-right Scrollbalken
"scrollbarPageUp"	oder "pageUp"	Page-up Scrollbalken
"scrollbarRight"	oder "right"	Right scroll Pfeil
"scrollbarUp"	oder "up"	Up scroll Pfeil
"scrollbarVThumb"		vertikaler Scrollbalken

Beispiel:

```
<HEAD>
<SCRIPT>
function ScrolleSeiteRechts()
{document.body.doScroll("scrollbarPageRight");}

function ScrolleDown()
{ID_Textarea.doScroll("scrollbarDown");}

function ScrolleSeiteDown()
{ID_Textarea.doScroll("scrollbarPageDown");}
</SCRIPT>
</HEAD>
<BODY>
<BUTTON onclick=" ScrolleSeiteRechts()">
    scrolle Seite rechts
</BUTTON>
<BR>
<BUTTON
    onclick=" ScrolleDown()"
    ondblclick=" ScrolleSeiteDown()"
>
    Texarea: Click = scrolle down Doppelklick = scrolle Seite down
</BUTTON>
<BR>
<BR>
<TEXTAREA ID="ID_Textarea" >
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
    1234567890123456789012345678901234567890
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
    1234567890123456789012345678901234567890
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
    1234567890123456789012345678901234567890
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
    1234567890123456789012345678901234567890
</TEXTAREA>
</BODY>
```

**.dragDrop()** prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element  
true Drag-Operation ist komplett erfolgreich durchgeführt  
Hinweis zum Event `ondragstart`: immer true, wenn Maustaste losgelassen wurde  
false Drag-Operation wurde nicht durchgeführt





<code>.DriveExists()</code>	prüfen auf Laufwerk im Dateisystem Laufwerk muss nicht bereits ein (z.B. Diskettenlaufwerk muss kein Medium enthalten)
Beispiel:	<pre>var DateiSystem = new ActiveXObject("Scripting.FileSystemObject"); alert(DateiSystem.DriveExists("A"));</pre>
<code>.duplicate()</code>	Zeiger auf eine Duplikat-Instanz eines Textbereiches liefern per textrange Objekt nur unter Windows 32-Bit Prüfung eines Textbereiches auf Kopie eines anderen Textbereiches per Methode <code>.inRange()</code> bzw. <code>.isEqual()</code>
<code>.elementFromPoint()</code>	liefert Referenz auf Objekt an Pixelpos relativ zum Fenster Fenster linke obere Ecke (0,0) Objekt muss Mausevents unterstützen
<code>.empty()</code>	Selektion löschen per <code>document.selection</code> Objekt des Dokumentes setzt zugleich Eigenschaft <div> <div><code>.item</code> auf null</div> <div>von <code>document.selection.textrange</code> Collection</div> </div> <div> <div>bzw.</div> <div>von <code>document.selection.controlRange</code> Collection</div> </div> <div> <div><code>.type</code> auf "none"</div> <div>von <code>document.selection</code></div> </div>
<code>encodeURIComponent()</code>	kodiert einen String als kompletten Uniform Ressource Identifier (URI): Ersatz von Zeichen durch Escape-Sequenzen (UTF-8 Zeichen) der Form %xx Es werden alle Zeichen kodiert, also durch Escape-Sequenzen ersetzt, außer folgende Zeichen <div> <div><code>., / ? : @ &amp; = + \$ - _ . ! ~ * ' ( ) #</code></div> <div>Buchstaben</div> <div>Ziffern</div> </div> ersetzt die Methode <code>escape()</code> welche deprecated ist ab IE 5.5 Beispiel: <pre>alert(encodeURIComponent("My phone # is 123-456-7890")); // erzeugt "My%20phone%20 #%20is%20123-456-7890"</pre>
<code>encodeURIComponentComponent()</code>	kodiert einen Teil-String aus einem kompletten Uniform Ressource Identifier (URI): Ersatz von Zeichen durch Escape-Sequenzen (UTF-8 Zeichen) der Form %xx Es werden alle Zeichen kodiert, also durch Escape-Sequenzen ersetzt, außer folgende Zeichen <div> <div><code>., / ? : @ &amp; = + \$ - _ . ! ~ * ' ( ) #</code></div> <div>Buchstaben</div> <div>Ziffern</div> </div> ab IE 5.5
<code>.endElement()</code>	aktives Element auf der Timeline stoppen identisch mit Wirkung, wenn Endezeit des Elementes erreicht wird auf der Timeline bzw. die Dauer (Duration) des Elementes abgearbeitet wurde Alle Kinder des Elementes erhalten Information über Start des Elternelementes und sind somit korrekt auf der Timeline des Elternelementes. per Eigenschaft <code>.isActive</code> das Element auf Aktivsein prüfen siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
Beispiel:	<pre>&lt;HTML XMLNS:t="urn:schemas-microsoft-com:time"&gt; &lt;HEAD&gt; &lt;?IMPORT namespace="t" implementation="#default#time2"&gt; &lt;STYLE&gt; .time_line_klasse { behavior: url(#default#time2) } &lt;/STYLE&gt; &lt;/HEAD&gt; &lt;BODY&gt; &lt;SPAN ID="ID_Span" CLASS="time_line_klasse" DUR=".01" REPEATCOUNT="indefinite" FILL="hold" onrepeat="innerText=parseInt(ID_Excl.currTimeState.activeTime);" &gt; 0 &lt;/SPAN&gt; &lt;BR&gt; &lt;t:EXCL ID="ID_Excl" REPEATDUR="indefinite"</pre>



```

>
    <DIV ID="ID_Div1"
        CLASS="time_line_klasse"
        BEGIN="0"
        DUR="3"
    >
        Erste Zeile
    </DIV>
    <DIV ID="ID_Div2"
        CLASS="time_line_klasse"
        BEGIN="3"
        DUR="3"
    >
        Zweite Zeile
    </DIV>
</t:EXCL>
<BR>
<BUTTON ID="ID_Button" onclick="ID_Excl.endElement()">
    Stopp
</BUTTON>
</BODY>
</HTML>

```

`.endElementAt()` aktives Element auf der Timeline nach einer Zeitspanne der Aktivität des Elementes  
 ab Beginn der Timeline stoppen  
 sinnvoll z.B. vor Erreichung des Ende der Dauer (Duration) des Elementes  
 wenn das Elternelement diesen Zeitpunkt schon erreicht hat, so endet das Kindelement sofort,  
 also ohne Zeitspanne  
 per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Stoppen()
    {
        // prüfen ob Elternobjekt, also body, den Endezeitpunkt schon erreicht hat
        // bzw. der Endewert falsch ist
        if ( (ID_Input.value <= document.body.currTimeState.activeTime)
            || (ID_Input.value==null)
        )
        {
            alert('Stoppwert falsch');
            ID_Input.value="";
            ID_Input.focus();
            return;
        }
        else
        { ID_Excl.endElementAt(ID_Input.value); }
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <SPAN ID="ID_Span2"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Excl.currTimeState.activeTime);"
    >

```



```

0
</SPAN>
<BR>
<t:EXCL ID="ID_Excl"
      BEGIN="0"
      CLASS="time_line_klasse"
      DUR="27"
>
  <DIV ID="ID_Div1"
      CLASS="time_line_klasse"
      BEGIN="0"
      DUR="5"
  >
    Erste Zeile
  </DIV>
  <DIV ID="ID_Div2"
      CLASS="time_line_klasse"
      BEGIN="5"
      DUR="5"
  >
    Zweite Zeile
  </DIV>
</t:EXCL>
<INPUT type="text" ID="ID_Input" SIZE="3" >
<BUTTON ID="ID_Button" onclick="Stoppen()">
  Start
</BUTTON>
</BODY>
</HTML>

```

**escape()**

kodiert einen String oder ein Literal in das Unicode-Format  
 ab Javascript 1.5 (Netscape 6.x) deprecated und zu ersetzen durch die Methoden  
 encodeURIComponent() und encodeURIComponent()  
 Ersatz von Zeichen durch Escape-Sequenzen (UTF-8 Zeichen) der Form %xx  
 Es werden alle Zeichen kodiert, also durch Escape-Sequenzen ersetzt, außer folgende Zeichen

. , / ? : @ & = + \$ - \_ ! ~ \* ' ( ) #  
 Buchstaben  
 Ziffern

URI (Uniform Resource Identifiers) werden nicht kodiert  
 UTF-8 Zeichen: Teil des Unicode (0 bis 255)

Beispiel für Einbindung einer Suchmaschine:

```

var markierter_text=document.selection.createRange().text;
// oder var markierter_text=prompt('Suchbegriff: ');
var suchmaschinen_url='http:// .....';
var suchmaschinen_parameter='.....';

if (markierter_text)
{location.HREF=suchmaschinen_url + suchmaschinen_parameter + escape(markierter_text);}
else
{location.HREF=suchmaschine_url; }

```

Beispiel für altavista: suchmaschinen\_url 'http://altavista.de/'  
 suchmaschinen\_parameter 'cgi-bin/query?pg=q&what=web&q='

**eval()**

parst einen String aus JavaScript-Anweisungen (kein HTML-Code) und führt diese sofort aus  
 JavaScript-Anweisungen  
 dürfen keine Referenz auf sich selbst haben (sonst endlose Rekursion !)  
 können Referenzen auf existierende Objekte (Variablen) und deren Objektmethoden wie  
 Objekteigenschaften enthalten z.B. für Wertzuweisung etc.  
 Objekte (Variablen) erzeugen und instanzieren  
 ab JavaScript 1.5 auch eval als Anweisung enthalten (Achtung: endlose Rekursionen  
 vermeiden !)

Beispiele:

```

var Kette = eval("new String('2+2')"); // liefert Zeiger auf den String '2+2'
eval("var Kette = new String('2+2')"); // liefert nichts
// Kette hat den Wert '2+2'
eval("var Kette = new String('2+2'); alert(Kette);"); // liefert nichts, da alert nichts liefert
// Kette hat den Wert '2+2'

```

```
var StringLiteral = "2 + 2";
```



```
eval(StringLiteral)           // liefert Zeiger auf numerische Variable
                               // mit Wert 4
var Wert = eval("2+2");       // liefert Zeiger auf numerische Variable
                               // mit Wert 4
var StringObjekt = new String("2 + 2");
eval(StringObjekt)           // liefert Zeiger auf Kette "2 + 2"
```

Beispiel:

```
var w = 2;
var x = 39;
var y = "42";
var z = "42";
eval("w + x + 1");           // liefert Zeiger auf numerische Variable
                               // mit Wert 42
eval(y);                     // liefert Zeiger auf numerische Variable
                               // mit Wert 42
                               // liefert nicht Zeiger auf y, da der String
                               // "42" ausgewertet wird
eval(z);                     // liefert Zeiger auf String-Variable
                               // mit Wert '42'
                               // liefert nicht Zeiger auf z, da der String
                               // "42" ausgewertet wird
```

Beispiel:

```
var Kette = "var x=5;"
+ "if (x == 5)"
+ "{"
+ "alert('Meldung aus eval\nx ist 5');" // \n ist Zeilenumbruch-Zeichen
+ "x = 42;"
+ "}"
+ "else"
+ "{"
+ "x = 0;"
+ "};";

var Kette1 = "alert('Meldung aus document.write\nx ist ' + eval(Kette));"
document.write(Kette1);
```

Beispiel für Ersatz von eval() durch die Eigenschaft .innerHTML (nur IE):

```
<SCRIPT>
var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild ">';

function Laden()
{
    ID_Div.innerHTML=Kette;           // wird sofort geparkt, also IMG-Tag ausgeführt
                                     // ( anstelle von eval()
                                     // bzw. document.write() )

    ID_IMG.src="";
    ID_IMG.style.display="block";

    ID_IMG.onerror= IMGAltTextAufFehlerMeldung; // ohne ()
}

function IMGAltTextAufFehlerMeldung ()
{
    ID_IMG.alt="Das Bild konnte nicht geladen werden.";
    return true;
}
</SCRIPT>
<INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
<DIV ID="ID_Div"></DIV>
```

Beispiel:

```
if (VarUserAgent != "")
{
    for ( var i=0; i < 10; i++)
    {
        eval( 'if (VarUserAgent.indexOf("'" + i + "'.") != -1)'
              + '{VarBrowser_Version_Haupt = ' + i + '};'
              );
    }
}
```



Beispiel:

```
function DatumHolen(Zeiger)
{
    var Kette = "Heute ist: ";
    Kette += Zeiger.getDate() + "/";
    Kette += (Zeiger.getMonth() + 1) + "/";
    Kette += Zeiger.getYear();

    return(Kette);
}

var Kette="Date";

eval(    "var Jetzt = new " + Kette + "();" // zur Laufzeit erzeugen
        "alert(DatumHolen(Jetzt));"      // Parameter ist zur Laufzeit bekannt
    );
```

.execCommand()

Kommando ausführen z.B. im aktuellen Dokument  
 in aktueller Selektion  
 im aktuellen Bereich  
 erst nach dem kompletten Laden des Dokumentes zulässig  
 Hinweis: Selektion = Markierung z.B. von Textbereich (Block)  
 Control = Element zur Steuerung analog zum HTML-Element (Tag)  
 Input-Control = Element mit Eingabeeigenschaft

Beispiel 1:

```
<HTML>
<BODY>
    <H1 UNSELECTABLE="on">Demo</H1>
    <SCRIPT>
        function AddLink()
        {
            var SelektierterText = document.selection.createRange();

            if (!SelektierterText == "")
            {
                // Link erzeugen
                document.execCommand("CreateLink");

                if (SelektierterText.parentElement().tagName == "A")
                {
                    // markierten Text mit Eltern-Url ersetzen
                    SelektierterText.parentElement().innerText=
                        SelektierterText.parentElement().href;

                    // Vordergrundfarbe setzen im Dokument
                    document.execCommand(
                        "ForeColor", "false", "#FF0033");
                }
            }
            else
            {alert("Bitte im blauen Text selektieren !");}
        }
    </SCRIPT>
    <P UNSELECTABLE="on">
        Selektiere (markiere) im nachfolgenden blauen Text die Stelle mit
        dem Text MARKIERE_MICH.<BR>
        Danach auf das Button klicken.<BR>
        Anstelle von MARKIERE_MICH im blauen Text erscheint dort
        nun eine Url.
    </P>
    <P STYLE="color=#3366CC">
        Meine beliebteste Webseite MARKIERE_MICH bitte besuchen !
    </P>
    <BUTTON onclick="AddLink()" UNSELECTABLE="on">Klick</BUTTON>
</BODY>
</HTML>
```

Beispiel 2:

```
<HTML>
<HEAD>
<SCRIPT>
    function HandlerFuerOnMoveStart()
    {
```



```

        // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
        var ZeigerAufObjektMitEvent = event.srcElement;

        ZeigerAufObjektMitEvent.style.backgroundColor = "green";
        ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
    }

    function HandlerFuerOnMove ()
    {
        ID_Span1.innerHTML = event.srcElement.offsetLeft;
        ID_Span2.innerHTML = event.srcElement.offsetTop;
    }

    function HandlerFuerOnMoveEnd()
    {
        // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
        var ZeigerAufObjektMitEvent = event.srcElement;

        ZeigerAufObjektMitEvent.style.backgroundColor = "red";
        ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
    }

    // 2-D Positionierung einschalten
    document.execCommand("2D-position",false,true);
</SCRIPT>
</HEAD>
<BODY onmovestart="HandlerFuerOnMoveStart();"
onmove="HandlerFuerOnMove();"
onmoveend="HandlerFuerOnMoveEnd();"
>
    offsetLeft = <SPAN ID="ID_Span1"></SPAN>
    <BR>
    offserTop = <SPAN ID="ID_Span2"></SPAN>
    <BR>
    <DIV CONTENTEDITABLE="true">
        <DIV STYLE=
            "position:absolute;width:300px;height:100px; background-color:red;"
        >
            bewegbarer DIV
        </DIV>
    </DIV>
</BODY>
</HTML>

```

.execScript()  
 Pendant zur Methode eval()  
 Script wird sofort ausgeführt  
 Script in diversen Sprachen möglich  
 alle vorhandenen-globalen Variablen ansprechbar  
 siehe Objekt window

Beispiel.: window.execScript('alert("Hallo");','JScript')

.Exists() prüfen auf Vorhandensein eines Datenschlüssels im Dictionary

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchlüsselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Meldung ob Date vorhanden ist
    alert(SchlüsselVorhanden(DatenSchluessel));
}

```

.exp()  
 liefert den Wert von E hoch zahl  
 siehe Script-Objekt Math



<code>.expand()</code>	Plain-Text als Teil eines Textbereiches derart ausdehnen, dass er komplett die Dimension des Elternobjektes (Container-Objektes) einnimmt per <code>textrange</code> Objekt nur unter Windows 32-Bit
<code>.expression()</code>	Wert einer Style-Eigenschaft per STYLE-Attribut-Wert in HTML als Ausdruck definieren für spätere Berechnung per Methode <code>getExpression()</code> Ausdruck nur als Script kodierbar DOM wird geändert siehe auch Methode <code>.setExpression()</code>

In nachfolgenden Beispielen wurde aus Platzgründen die Zeichenkette von STYLE umgebrochen, was eigentlich nicht zulässig ist, und es sind nicht alle SPAN kodiert.

Beispiel 1:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE ="JScript">
    function width_init()
    {
        ID_Span.style.setExpression( "width",
                                     " trueBlueSpan.style.pixelWidth
                                     + oldYellowSpan.style.pixelWidth
                                     ",
                                     "javascript"
                                   );
    }

    function Berechne()
    {alert(ID_Span.style.getExpression("width");}
</SCRIPT>
</HEAD>
<BODY onload= width_init();>
    <SPAN ID="ID_Span"
        STYLE="background-color:lightgreen;
              width:expression( trueBlueSpan.style.pixelWidth
                                + oldYellowSpan.style.pixelWidth
                                )
              "
    >
    </SPAN>
    <BUTTON onclick= Berechne();>
</BODY>
</HTML>
```

Beispiel 2:

```
ID_Span.style.setExpression("height","document.style.fontSize + 13");
ID_Span.style.setExpression("width","document.body.style.fontSize");
<SPAN ID="ID_Span"
    STYLE="background-color:lightgreen;
          width:expression( trueBlueSpan.style.pixelWidth
                            + oldYellowSpan.style.pixelWidth
                            )
          "
>
</SPAN>
```

`.FileExists()` prüfen auf Datei

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.FileExists("c:\\test.txt"));
```

`.findText()` Text im gesamten Textbereich suchen  
per `textrange` Objekt  
nur unter Windows 32-Bit  
für Nutzung einer Textmarke siehe Methoden `.getBookmark()` und `.moveToBookmark()`

`.fire()` ein in der HTC-Komponente definiertes Event erzeugen und an das HTML-Dokument weiterreichen  
siehe Objekt `element` und HTC-Datei

Beispiel:

test.htc enthält:

```
<PUBLIC:COMPONENT NAME="HTC_Komponente">
    <PUBLIC:EVENT NAME="onResultChange" ID="ID_Event"/>
```



```

<SCRIPT LANGUAGE="JScript">
    function Berechne()// Bezug über ID_Event
    {
        var EventObjekt    = createEventObject();
        EventObjekt.result = "Wert";

        ID_Event.fire (EventObjekt);
    }
</SCRIPT>
</PUBLIC:COMPONENT>

```

HTML-Dokument enthält:

```

<HTML XMLNS:privater_tag_namensraum>
<HEAD>
<STYLE>
    @media all { privater_tag_namensraum\:privater_tag {behavior:url(test.htc)} }
</STYLE>
</HEAD>
<BODY>
<privater_tag_namensraum:privater_tag
    ID="ID_privater_tag"
    onResultChange="ID_Div.innerText=window.event.result"
>
    <TABLE>
        <TR>
            <DIV
                ID="ID_Div"
                ALIGN=RIGHT
                STYLE="border: '.025cm solid gray'"
            >
                0.
            </DIV>
        </TR>
        <TR>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 7 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 8 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 9 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" / ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" C ">
            </TD>
        </TR>
        <TR>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 4 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 5 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 6 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" * ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" % " DISABLED>
            </TD>
        </TR>
        <TR>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 1 ">
            </TD>
            <TD>
                <INPUT TYPE=BUTTON VALUE=" 2 ">
            </TD>
        </TR>
    </TABLE>

```





```

        <TD>
        <INPUT TYPE=BUTTON VALUE=" 3 ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" - ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" 1/x " DISABLED>
    </TD>
</TR>
<TR>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" 0 ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" +/- ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" . ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" + ">
    </TD>
    <TD>
        <INPUT TYPE=BUTTON VALUE=" = ">
    </TD>
</TR>
</TABLE>
</privater_tag_namensraum:privater_tag>
</BODY>
</HTML>

```

`.fireChange()` erzeugt Event `onpropertychange` und gibt es an das HTML-Dokument, sobald der Wert der Eigenschaft per Aufruf der Funktion laut Attribut `PUT` verändert wird  
 Aufruf von `.fireChange()` muss programmiert werden  
 siehe Objekt `element` und HTC-Datei

Beispiel:

```

<PUBLIC:PROPERTY
    ID="ID_HTC_Komponente"
    NAME="HTC_Komponente"
    PUT="Eigenschaft_Schreiben"
    GET="Eigenschaft_Lesen"
/>

<SCRIPT LANGUAGE="JScript">
    var Eigenschaft_Wert =null;

    function Eigenschaft_Schreiben (Wert)
    {
        // Wert zuweisen
        Eigenschaft_Wert = Wert;

        // und Event erzeugen
        ID_HTC_Komponente.fireChange();
    }

    function Eigenschaft_Lesen ()
    {return Eigenschaft_Wert; }
</SCRIPT>

```

`.fireEvent()` ein Event auslösen und zugehöriges Objekt **object**.event erzeugen, wobei **object** das Event auslöst  
 mit Aufruf der Methode werden folgende Eigenschaften des even Objektes automatisch belegt:

<b>object</b> .event.cancelBubble	auf false
<b>object</b> .event.returnValue	auf true
<b>object</b> .event.srcElement	mit Zeiger des eventerzeugenden Objektes belegt
<b>object</b> .event.type	mit dem Eventtyp belegt

`object` kann entfallen wenn aktuelles Fenster gemeint ist, also `window`  
 zeiger\_auf\_fenster

siehe `.createEventObject()`

Beispiel 1:

```

<HTML>
<HEAD>
<SCRIPT>

```



```

function EventAusloesen()
{
    ID_Div.innerText = "onmouseover-Event wurde ausgelöst!";

    // Event onclick auslösen
    ID_Button.fireEvent("onclick");
}
</SCRIPT>
</HEAD>
<BODY>
<DIV ID="ID_Div" onmouseover=" EventAusloesen();">
    Bitte hier Mouse over!
</DIV>
<BUTTON ID="ID_Button" ONCLICK="this.innerText='onclick-Event wurde ausgelöst!' ">
    Klick mich nicht , denn ich werde durch den DIV geklickt!
</BUTTON>
</BODY>
</HTML>

```

Beispiel 2:

```

<SCRIPT LANGUAGE="JScript">
function DurchReichen()
{
    if (window.event.shiftKey)
    {window.event.cancelBubble = true;}
}

function Anzeigen()
{
    if (window.event.srcElement.tagName == "IMG")
    {alert(window.event.srcElement.src);}
}
</SCRIPT>
<BODY onclick="Anzeigen()">
<IMG SRC="test.gif" onclick="DurchReichen()">

```

.firstPage()

erste Seite des Datasets in Tabelle anzeigen  
Eigenschaft .dataPageSize muss belegt worden sein  
siehe Objekt table

Beispiel:

Datensätze liegen in der Datei adress.txt  
Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument  
hat folgenden Inhalt:

1. Satz	<b>vorname;name;telefon</b>
ab 2. Satz	Guericke;"Otto, von";0815 Willmann;Theo;1234 Xantippe;Isa;5678 Arktin;Richard;1055

```

<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
var SortierRichtung = true;

function Sortieren(FeldBezeichner)
{
    // Sortierrichtung festlegen
    var Kette = "-"; // Annahme
    if (SortierRichtung) {Kette = "+";}

    // nächste Sortierung umgekehrt
    SortierRichtung = !SortierRichtung;

    // sortieren
    ID_Datenbank.Sort= Kette + FeldBezeichner;

    // und das Ergebnis anzeigen
    ID_Datenbank.Reset();
}

function vorwaerts(AnzahlSaetze)
{
    // nächste Seite anzeigen

```



```

        document.all.ID_Tabelle.nextPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaezte; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition !=
                ID_Datenbank.recordset.RecordCount)
            {ID_Datenbank.recordset.MoveNext();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == ID_Datenbank.recordset.RecordCount)
        {alert("Letzter Datensatz erreicht!");}
    }

    function rueckwaerts(AnzahlSaezte)
    {
        // vorhergehende Seite anzeigen
        document.all.ID_Tabelle.previousPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaezte; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition > 1)
            {ID_Datenbank.recordset.MovePrevious();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == 1)
        {alert("Erster Datensatz erreicht!");}
    }

    // -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
    CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
    <PARAM NAME="DataURL" VALUE="adress.txt">
    <PARAM NAME="UseHeader" VALUE="True">
    <PARAM NAME="FieldDelim" VALUE=";">
</OBJECT>

<TABLE ID="ID_Tabelle"
    DATASRC=#ID_Datenbank
    DATAPAGESIZE=1
>
    <THEAD>
        <TR>
            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
            <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
        </TR>
    </THEAD>
    <TBODY>
        <TR>
            <TD><DIV DATAFLD="vorname"></DIV></TD>
            <TD><DIV DATAFLD="name"></DIV></TD>
            <TD><DIV DATAFLD="telefon"></DIV></TD>
        </TR>
    </TBODY>
</TABLE>
<BR>
<INPUT TYPE="button"
    VALUE="Zurueck"
    onclick=rueckwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->

<INPUT TYPE="button"
    VALUE="Vorwaerts"
    onclick=vorwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->
</BODY>
</HTML>

```

.fixed()

HTML-Tag &lt;TT&gt; erzeugen in der Form &lt;TT&gt;text&lt;/TT&gt;



siehe Script-Objekt String

Beispiel:

```
var StringLiteral    ="Text der TT wird"
var HTML_Kette      = StringLiteral.fixed();
HTML_Kette          = "Text der TT wird".fixed();
```

entspricht <TT>Text der TT wird</TT>

```
eval(HTML_Kette);          erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);
```

.floor()                    nächste ganze Zahl unterhalb zahl ermitteln  
                           Bsp:        floor(1.1) ergibt 1  
                                       floor(1)    ergibt 1  
                           siehe Script-Objekt Math

.focus()                    Focus setzen und Focus-Event auslösen  
                           nur nach dem kompletten Laden des Dokumentes  
                           vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen

Beispiel:

```
<BODY onload="document.body.focus();>
```

Beispiele für Abschaltung der automatischen Umrandung von angeklickten Objekten durch Fokussierung des BODY:

```
<BODY onclick="if (document.all){body.focus();}"> ..... </BODY>
```

```
<A HREF="...." onclick="if (document.all){body.focus();}"> ....
```

.focus()                    Focus setzen und Event onfocus auslösen  
                           nur nach dem kompletten Laden des Dokumentes  
                           vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen  
                           siehe Objekt window

.FolderExists()            prüfen auf Ordner

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.FolderExists("c:\\test\\"));
```

.fontcolor()                HTML-FONT als HTML-Kette erzeugen in der Form "<FONT COLOR='....'>text</FONT>  
                           und ohne weitere Attribute ID etc.  
                           siehe Script-Objekt String

Beispiel:

```
var StringLiteral    ="Sichtbarer Text"
var HTML_Kette      = StringLiteral.fontcolor("WertDesCOLORAttributes");
HTML_Kette          = "Sichtbarer Text".fontcolor("WertDesCOLORAttributes");
```

entspricht <FONT COLOR="WertDesCOLORAttributes">Sichtbarer Text</FONT>

```
eval(HTML_Kette);          erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);
```

.fontsize()                HTML-FONT als HTML-Kette erzeugen in der Form "<FONT SIZE='....'>text</FONT>  
                           und ohne weitere Attribute ID etc.  
                           siehe Script-Objekt String

Beispiel:

```
var StringLiteral    ="Sichtbarer Text"
var HTML_Kette      = StringLiteral.fontsize("WertDesSIZEAttributes");
HTML_Kette          = "Sichtbarer Text".fontsize("WertDesSIZEAttributes");
```

entspricht <FONT SIZE="WertDesSIZEAttributes">Sichtbarer Text</FONT>

```
eval(HTML_Kette);          erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);
```

.forward()                Aktivierung einer zur aktuell besuchten Seite im Verlauf des Surfens danachliegenden Seite  
                           leider ohne Sprünge  
                           entspricht Druck des Forward-Buttons

.fromCharCode()            String-Literal erzeugen aus Folge von Unicoden  
                           Unicode von 0 bis 65535, wobei  
                                       0 bis 127 der ASCII-Zeichensatz ist  
                                       0 bis 255 der ISO-Latin-1-Zeichensatz ist  
                           siehe Script-Objekt String

Beispiel 1:



```
String.fromCharCode(65,66,67) // liefert "ABC"
```

### Beispiel 2:

```
function HoleTaste(e)
{
    var Taste;
    if (document.layers)
    {Taste = String.fromCharCode(e.wich);}
    else
    {
        if (document.all)
        {Taste = String.fromCharCode(event.keyCode);}
    }

    Taste = Taste.toLowerCase();    // Kleinbuchstabe
}

if (document.layers)
{document.captureEvents(Event.KEYPRESS);}
else
{
    if (document.all)
    {document.attachEvent('onkeypress', HoleTaste);}
}

document.onkeypress = HoleTaste;
```

.GetAbsolutePathName()	absoluten Pfad ab Root mit Laufwerk ermitteln	
	Angabe von Wildcard "*" möglich	wird <b>leider nicht</b> aufgelöst
	Angabe von "\\." möglich	falls Anzahl von "\\." eine Logik hinter die
		Root erzeugen würde, so wird
		kein Fehler erzeugt, sondern korrekt
		ab Root aufgelöst
	Bsp.:	c:\test\ existiert
		Auflösung von c:\\..\\..\\test
		bringt keinen Fehler

Beispiel 1:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetAbsolutePathName("../test\\"));
```

Beispiel 2:

Es gilt: aktuelle Position im Dateisystem ist c:\test\texte  
 Ordner texte hat folgenden Unterordner jahr2002  
 Ordner jahr2002 hat folgenden Unterordner mai2002

<u>Kette1</u>	<u>Kette2</u>	<u>Bedeutung</u>
"c:"	"c:\test\texte"	aktuelle Position
"c:.."	"c:\test"	Ordner oberhalb der aktuellen Position
"c:\\"	"c:\"	Root und nicht aktuelle Position
"c:*.\\mai2002"	"c:\test\texte\*.\\mai2002"	Pfad von mai2002
"jahr2002"	"c:\test\texte\jahr2002"	keine Auflösung von "*"
"c:\\..\\..\\test"	"c:\test"	Pfad von jahr2002
		kein Fehler

<code>.getAdjacentText()</code>	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
---------------------------------	---

Beispiel:

```
<SCRIPT>
    function Anzeigen()
    {
        var Woher_Kette = ID_Selektion.options[ID_Selektion.selectedIndex].text;
        alert(ID_P.getAdjacentText(Woher_Kette));
    }
</SCRIPT>
Text vor dem Beginn des P-Tag (woher ist beforeBegin)
<P ID="ID_P">
    Text nach dem Beginn des P-Tag (woher ist afterBegin)
    <B>Irgend ein Text</B>
    Text vor dem Ende des P-Tag (woher ist beforeEnd)
</P>
Text nach dem Ende des P-Tag (woher ist afterEnd)

<SELECT ID="ID_Selektion">
    <OPTION SELECTED>woher ist beforeBegin
```



```

<OPTION>woher ist afterBegin
<OPTION>woher ist beforeEnd
<OPTION>woher ist afterEnd
</SELECT>
<INPUT TYPE="button" VALUE="Anzeigen" onclick=" Anzeigen()">

```

.getAttribute()                    Wert eines per HTML erzeugten Attributes liefern  
                                   DOM nicht geändert

Beispiel:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
//        es können diverse Cachenames definiert und somit Versionen von Cache
//        verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
//        es können diverse Attribute definiert und somit Versionen von Input-Daten
//        verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++++ Zeitstempel ++++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++++ Daten chachen ++++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache saveen
    InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
<FORM ID="ID_Formular">
<INPUT    ID="ID_Input"
          CLASS="user_data_speicher_klasse"
          TYPE="text"
>
<INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
<INPUT TYPE="button" VALUE="laden der Input-Daten"    onclick="InputLaden()">
</FORM>

```



```
</BODY>
</HTML>
```

`.getAttribute()` einzelnen vCard-Wert lesen ohne Request Queue  
siehe Objekt `navigator.userProfile` und Autovervollständigung im Internet Explorer

Beispiel:

```
// lesen vom Wert zu "vcard.displayName"
var Name = navigator.userProfile.getAttribute("vcard.displayName");

// lesen vom Wert zu "vcard.gender"
var Gender = navigator.userProfile.getAttribute("vcard.gender");
```

`.getAttributeName()` Name des mit dem MediaItem Objekt verbundenen Attributes liefern  
z.B. zur Feststellung, welche Attribute eine Advanced Stream Redirector (ASX)-Datei hat  
wie `abstract` oder `author` oder `copyright`

Beispiel für Aufbau eines Eintrages in einer ASX-Datei:

```
<ASX Version="1.0" PreviewMode="No" >
<entry>
  <title>Testitel</title>
  <author>Testautor</author>
  <copyright>Test 2002</copyright>
  <abstract>WAV Datei</abstract>
  <ref href=""></ref>
  <banner href = "Testbild.gif" >
    <moreinfo href = "Test.doc"></moreinfo>
    <abstract>besuche www.test.de</abstract>
  </banner>
</entry>
</ASX>
```

Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende Methoden nutzbar:

```
.getItemInfo()
.getAttributeName()
```

Eigenschaften nutzbar:  
`.attributeCount`

Es ist also vorher die Eigenschaft `.playState` auf Werte > 10 zu prüfen.  
siehe Behavior `.style.mediaBar`

`.getAttributeNode()` Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf `attribute.name` Eigenschaft.  
Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht  
Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt  
Wert des Attributes wird somit über die Referenz laut DOM erreichbar  
DOM nicht geändert

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
  function ToolTipKnotenErmitteln()
  {
    return (ID_Div.getAttributeNode("TITLE"));
  }
</SCRIPT>
</HEAD>
<BODY onload="ToolTipKnotenErmitteln();" >
  <DIV ID="ID_Div" TITLE="Tooltip-Text ">Es ist ein Tooltip-Text vorhanden</DIV>
</BODY>
</HTML>
```

`.GetBaseName()` absoluten Pfad einer Datei im Pfad liefern  
Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetBaseName("../test\\text.txt"));
```

`.getBookmark()` Textmarke (Bookmark) im Textbereich setzen  
Textmarke kann mit Methode `.moveToBookmark()` anpositioniert werden  
per `textrange` Objekt  
nur unter Windows 32-Bit

`.getBoundingClientRect()` Referenz auf `TextRectangle`-Objekt im Element holen



Beispiel:

```

<HEAD>
<SCRIPT>
    var ZeigerAufTextRectangleCollection;
    var Index =0;

    function Anzeigen(Zeiger) // Zeiger auf den einzigsten DIV mit Text also mit Textbereich
    {
        // aktuelle Collection der Rectangle von Div0 referenzieren:
        // Das ist nötig nach resize des Fensters, da resize leider nicht automatisch erkannt wird !
        ZeigerAufTextRectangleCollection = Zeiger.getClientRects();

        // Anzahl der Elemente ermitteln, also Anzahl der im Browser dargestellte Zeilen,
        // wobei es egal ist, ob diese per <BR> erzeugt wurden oder nicht
        // Hinweis: Zeilenlänge hängt auch von der Fensterbreite ab
        // (automatischer Umbruch)
        // Jede Zeile besitzt ihr eigenes Rectangle
        AnzahlTextRectangle = ZeigerAufTextRectangleCollection.length;

        // aktuellen Index prüfen ob letzte Zeile bereits erreicht wurde
        if (Index > AnzahlTextRectangle -1) // Index ab 1, Anzahl ab 1
        {
            // es wurde die letzte Zeile erreicht

            // Div2 unsichtbar machen also entfärben
            // Hinweis: Div2 liegt auf dem Rectangle von Div0
            ID_Div2.style.display="none";

            // rücksetzen des Index, also mit erster Zeile weitermachen
            Index = 0;
        }

        // Rechteck der aktuellen Zeile ermitteln unter Beachtung eines eventuellen Scrollens
        var PosRechts = ZeigerAufTextRectangleCollection[Index].right + ID_Body.scrollLeft;
        var PosLinks = ZeigerAufTextRectangleCollection[Index].left + ID_Body.scrollLeft;
        var PosOben1 = ZeigerAufTextRectangleCollection[Index].top + ID_Body.scrollTop;

        // und Div1 auf die Zeile positionieren, also Zeile einfärben
        ID_Div1.style.top = PosOben1;
        ID_Div1.style.width = (PosRechts - PosLinks) - 5;
        ID_Div1.style.display = 'inline';

        // aktuelle Position des Rechteckes von DIV0 ermitteln unter Beachtung eines eventuellen
        // Scrollens
        PosRechts = Zeiger.getBoundingClientRect().right + ID_Body.scrollLeft;
        PosLinks = Zeiger.getBoundingClientRect().left + ID_Body.scrollLeft;
        var PosOben2 = Zeiger.getBoundingClientRect().top + ID_Body.scrollTop;

        // und Div2 überlagern positionieren
        ID_Div2.style.top = PosOben2;
        ID_Div2.style.width = (PosRechts - PosLinks) - 5;
        ID_Div2.style.height = PosOben1 - PosOben2;

        // aber nur einfärben, wenn mindestens 1 Zeile bereits eingefärbt wurde
        if (Index > 0){ ID_Div2.style.display = 'inline';}

        // Rectangle der nächsten Zeile einstellen
        Index++;
    }
</SCRIPT>
</HEAD>
<BODY ID="ID_Body">
    <DIV ID="ID_Div0"
        onclick="Anzeigen(this)"
    >
        klicke
        <BR>
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
    </DIV>

```





```

ABCDEF GHIJKLMNOPQRSTUVWXYZ
1234567890123456789012345678901234567890
</DIV>
<DIV ID="ID_Div1"
STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:yellow; display:none"
>
</DIV>
<DIV ID="ID_Div2"
STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:beige; display:none"
>
</DIV>
</BODY>

```

.getClientRects() Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster  
 Feld mit Index als Integer ab 0  
 pro Eintrag ein Rectangle

Beispiel:

```

<HEAD>
<SCRIPT>
var ZeigerAufTextRectangleCollection;
var Index =0;

function Anzeigen(Zeiger) // Zeiger auf den einzigsten DIV mit Text also mit Textbereich
{
    // aktuelle Collection der Rectangle von Div0 referenzieren:
    // Das ist nötig nach resize des Fensters, da resize leider nicht automatisch erkannt wird !
    ZeigerAufTextRectangleCollection = Zeiger.getClientRects();

    // Anzahl der Elemente ermitteln, also Anzahl der im Browser dargestellte Zeilen,
    // wobei es egal ist, ob diese per <BR> erzeugt wurden oder nicht
    // Hinweis: Zeilenlänge hängt auch von der Fensterbreite ab
    // (automatischer Umbruch)
    // Jede Zeile besitzt ihr eigenes Rectangle
    AnzahlTextRectangle = ZeigerAufTextRectangleCollection.length;

    // aktuellen Index prüfen ob letzte Zeile bereits erreicht wurde
    if (Index > AnzahlTextRectangle -1) // Index ab 1, Anzahl ab 1
    {
        // es wurde die letzte Zeile erreicht

        // Div2 unsichtbar machen also entfärben
        // Hinweis: Div2 liegt auf dem Rectangle von Div0
        ID_Div2.style.display="none";

        // rücksetzen des Index, also mit erster Zeile weitermachen
        Index = 0;
    }

    // Rechteck der aktuellen Zeile ermitteln unter Beachtung eines eventuellen Scrollens
    var PosRechts = ZeigerAufTextRectangleCollection[Index].right + ID_Body.scrollLeft;
    var PosLinks = ZeigerAufTextRectangleCollection[Index].left + ID_Body.scrollLeft;
    var PosOben1 = ZeigerAufTextRectangleCollection[Index].top + ID_Body.scrollTop;

    // und Div1 auf die Zeile positionieren, also Zeile einfärben
    ID_Div1.style.top = PosOben1;
    ID_Div1.style.width = (PosRechts - PosLinks) - 5;
    ID_Div1.style.display = 'inline';

    // aktuelle Position des Rechteckes von DIV0 ermitteln unter Beachtung eines eventuellen
    // Scrollens
    PosRechts = Zeiger.getBoundingClientRect().right +
ID_Body.scrollLeft;

    PosLinks = Zeiger.getBoundingClientRect().left + ID_Body.scrollLeft;
    var PosOben2 = Zeiger.getBoundingClientRect().top + ID_Body.scrollTop;

    // und Div2 überlagern positionieren
    ID_Div2.style.top = PosOben2;
    ID_Div2.style.width = (PosRechts - PosLinks) - 5;
    ID_Div2.style.height = PosOben1 - PosOben2;

    // aber nur einfärben, wenn mindestens 1 Zeile bereits eingefärbt wurde
    if (Index > 0){ ID_Div2.style.display = 'inline';}

    // Rectangle der nächsten Zeile einstellen

```



```

        Index++;
    }
</SCRIPT>
</HEAD>
<BODY ID="ID_Body">
    <DIV ID="ID_Div0"
        onclick=" Anzeigen(this)"
    >
        klicke
        <BR>
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
    </DIV>
    <DIV ID="ID_Div1"
        STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:yellow; display:none"
    >
    </DIV>
    <DIV ID="ID_Div2"
        STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:beige; display:none"
    >
    </DIV>
</BODY>

```

`.getComponentVersion()`      Version einer Komponente  
                                  Behavior .style.clientCaps

`.getData()`                      Clipboard auslesen  
                                  Anwendung für Ereignisse oncopy oder oncut  
                                  Handler muss liefern window.event.returnValue=false;  
                                  "Text"      Daten im Text-Format auslesen  
                                  "URL"      Daten im Url-Format auslesen  
                                  "File"      Daten im File-Format auslesen  
                                  "HTML"      Daten im HTML-Format auslesen  
                                  "Image"      Daten im Image-Format auslesen: Es wird Pfad geliefert

`.getDate()`                      Tag des Monats in lokaler Zeit liefern  
                                  Script-Objekt Date  
                                  Integer 1 bis 31

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)      // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =      Name
        + "="
        + escape(Value)
        + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)      // Name und Wert sind Strings
{
    var document.cookie =      Name
        + "="
        + escape(Value)
        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    //      erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie

```



```

//      Aufbau   Cookie_Name = Cookie_Wert
//                      Separator ist Gleichheitszeichen
//      Index 0 für Cookie_Name
//      Index 1 für Cookie_Wert
var CookieNameUndWertAlsFeld;

// CookieFeld auslesen und auf Cookie laut Name prüfen
var Gefunden=false;
var Index = 0;
do
{
    // aktuelles Cookie lesen
    CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

    // und auf Name prüfen
    if (Name == CookieNameUndWertAlsFeld [0])
    {
        CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
        Gefunden=true;
    }
    else
    {Index++;}
}
while (      ( !Gefunden )
        && ( Index < AnzahlCookies)
        );

return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
//      es können diverse Cachenames definiert und somit Versionen von Cache
//      verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
//      es können diverse Attribute definiert und somit Versionen von Input-Daten
//      verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++++ Zeitstempel ++++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++++ Daten chachen ++++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache save

```



```

        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }
</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n.... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getDay()
    Tag der Woche in lokaler Zeit liefern
    Script-Objekt Date
    Integer 0 bis 6
    0 ist Sonntag
    6 ist Samstag

```

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

    function LoescheCookie (Name, Value)    // Name und Wert sind Strings
    {
        var document.cookie =    Name
                                + "="

```



```

+ escape(Value)
+ "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau Cookie_Name = Cookie_Wert
    // Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld[Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while ( ( !Gefunden )
        && ( Index < AnzahlCookies)
    );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;
}

```



```

// und als UTC-Format erzeugen
var ZeitstempelUTC = Zeitstempel.toUTCString();

// und Zeitstempel dem Input-Objekt verpassen
ID_Input.expires = ZeitstempelUTC;

// ++++++ Daten chachen ++++++
// aktuelle Daten holen laut Input-Objekt
var InputDaten = InputDatenObjekt.value;

// Attribut instanzieren und mit Daten füllen
InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

// und Cache saveen
InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```

.GetDrive()                      Objekt FileSystemObject.Drive anhand des Laufwerksbuchstaben eines existierenden Laufwerkes erzeugen (auch bei Netzlaufwerk) sonst wird Fehler erzeugt

Beispiel:

```

var DateiSystem                      = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk                         = DateiSystem.GetDrive("C");

```



```

var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_TotalerPlatz = Laufwerk.TotalSize;
alert(Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalerPlatz);

```

`.GetDriveName()` Laufwerksbuchstabe aus Pfad ermitteln in der Form "x:" mit x für Laufwerk  
 Pfad muss nicht existieren  
 muss Laufwerksbuchstaben enthalten  
 wird nicht auf Syntax geprüft

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetDriveName("c:\\..\\.*\\test{***}");

```

`.getElementById()` Referenz auf das im Dokument ZUERST gefundene Objekt laut ID (analog zum ID-Attribut) liefern  
 Achtung: Objekte, die kein ID besitzen, werden nicht erfasst !  
 Für Verwaltung per NAME (analog zum NAME-Attribut):  
     siehe Methode `getElementsByName()`  
 Für Verwaltung per Tag-Name  
     siehe Methode `getElementsByTagName()`  
 wenn mehrere Elemente mit ein und demselben ID, so das ERSTE Element von diesen referenziert  
**Eine Referenzierung einer Collection der Objekte mit gemeinsamen ID ist leider nicht möglich. Deswegen der strenge Hinweis:**  
 In der Regel werden ID vom Programmierer objektweise getrennt vergeben, es sei denn, man will bewusst eine Gruppe von Objekten (z.B. `RadioBox`) verwaltbar machen und kennt diese Objekte. Die maschinelle Analyse eines fremden Dokumentes mit der Methode `getElementById()` ist nicht möglich.  
 DOM nicht geändert  
`getElementById()` ist der Standard, um Referenz zu ermitteln, und den alle Browser können müssen  
 ID-Attribut ermöglicht zwar den bequemen Ersatz, aber nur, wenn das ID-Attribut überhaupt im Objekt implementiert ist.  
 Wird mit `createElement()` gearbeitet, so ist der Zeiger auf Das Objekt global speicherbar (in einem Feld), so dass `getElementByXXXX` nicht mehr benötigt wird.  
 Alternativ sind Collections von `document` etc. nutzbar.  
 Besonders faule Programmierer nutzen intensiv `getElementByXXXXXX` und blähen den Quellcode auf (Literaten als Programmierer)

Beispiel:

```

<SCRIPT>
function Referenziere()
{
    var Zeiger=document.getElementById("ID_Div");
}
</SCRIPT>
<DIV ID="ID_Div">Test</DIV>
<INPUT TYPE="button" VALUE=" Referenziere" onclick=" Referenziere()">

```

`.getElementsByName()` Referenz auf ein Feld (Collection) aller im Dokument befindlichen Objekte mit gemeinsamen NAME (analog zum Attribut NAME) liefern  
 Achtung: Objekte, die kein NAME besitzen, werden nicht erfasst !  
 Für Verwaltung per ID (analog zum ID-Attribut):  
     siehe Methode `getElementById()`  
 Für Verwaltung per Tag-Name  
     siehe Methode `getElementsByTagName()`  
 DOM nicht geändert  
 ID-Attribut ermöglicht zwar den bequemen Ersatz, aber nur, wenn das ID-Attribut überhaupt im Objekt implementiert ist.  
 Wird mit `createElement()` gearbeitet, so ist der Zeiger auf Das Objekt global speicherbar (in einem Feld), so dass `getElementByXXXX` nicht mehr benötigt wird.  
 Alternativ sind Collections von `document` etc. nutzbar.  
 Besonders faule Programmierer nutzen intensiv `getElementByXXXXXX` und blähen den Quellcode auf (Literaten als Programmierer)

Beispiel:

```

<SCRIPT>
function Referenziere()
{
    var FeldDerInput=document.getElementsByName("GemeinsamerName");
}
</SCRIPT>
<INPUT TYPE="text" NAME="GemeinsamerName">
<INPUT TYPE="text" NAME="GemeinsamerName">
<INPUT TYPE="button" NAME="Button" VALUE=" Referenziere" onclick=" Referenziere()">

```

`.getElementsByTagName()` Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc.  
 Hinweis: Natürlich kann auch das `document`-Objekt so verarbeitet werden (beachte dabei `document.all` Collection)



Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst !

Für Verwaltung per ID (analog zum ID-Attribut):

siehe Methode `getElementById()`

Für Verwaltung per NAME (analog zum NAME-Attribut):

siehe Methode `getElementByName()`

DOM nicht geändert

`getElementsByTagName()` ist der Standard, um Referenz zu ermitteln, und den alle Browser können müssen

ID-Attribut ermöglicht zwar den bequemeren Ersatz, aber nur, wenn das

ID-Attribut überhaupt im Objekt implementiert ist.

Wird mit `createElement()` gearbeitet, so ist der Zeiger auf Das Objekt global

speicherbar (in einem Feld), so dass `getElementByXXXX` nicht mehr benötigt wird.

Alternativ sind Collectionen von `document` etc. nutzbar.

Besonders faule Programmierer nutzen intensiv `getElementByXXXXXX` und blähen den

Quellcode auf (Literaten als Programmierer)

Beispiel 1:

```
var DIV_KinderZeigerFeld = document.body.getElementsByTagName("DIV");
```

Hinweis: entspricht `var DIV_KinderZeigerFeld = document.body.all.tags("DIV");`

Beispiel 2:

```
<SCRIPT>
var Feld_Span = ID_DivEltern.getElementsByTagName("SPAN");
// alle SPAN-Kinder referenzieren
</SCRIPT>
<DIV ID="ID_DivEltern">
  <SPAN>
    Span-Kind von ID_DivEltern
  </DIV>
    Div-Kind vonDivEltern-Span
    <SPAN>
      Span-Kind vonDivEltern-Span-Div
    </SPAN>
  </DIV>
</SPAN>
</DIV>
```

Beispiel 3:

```
<SCRIPT>
function Anzeige()
{
  var ZeigerAufOnClickEventQuelle=event.srcElement;
  var Feld =
    ZeigerAufOnClickEventQuelle.parentElement.getElementsByTagName("LI");
  alert(
    "Anzahl LI : "
    + Feld.length
    + "\nErster Eintrag: "
    + Feld [0].childNodes[0].nodeValue
  );
}
</SCRIPT>
<UL onclick="Anzeige()">
  <LI>Menuepunkt 1
  <UL>
    <LI> Menuepunkt 1.1
    <OL>
      <LI> Menuepunkt 1 1.1
      <LI> Menuepunkt 1 1.2
    </OL>
    <LI> Menuepunkt 1.2
    <LI> Menuepunkt 1.3
  </UL>
  <LI> Menuepunkt 2
  <UL>
    <LI> Menuepunkt 2.1
    <LI> Menuepunkt 2.3
  </UL>
  <LI> Menuepunkt 3
</UL>
```

`.getExpression()`

Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern

Style-Eigenschaft ist per Methoden

`expression()` oder `setExpression()`

zu definieren

DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout

(nach dem eventuellen expliziten Dokument-Refresh)





In nachfolgenden Beispielen wurde aus Platzgründen die Zeichenkette von STYLE umgebrochen, was eigentlich nicht zulässig ist, und es sind nicht alle SPAN kodiert.

Beispiel 1:

```
<SPAN ID="ID_Span"
      STYLE= "background-color:lightgreen;
              width:expression( trueBlueSpan.style.pixelWidth
                                + oldYellowSpan.style.pixelWidth
                                )
            "
>
</SPAN>
<BR>
<BUTTON onclick=alert(ID_Span.style.getExpression("width"));>
```

Beispiel 2:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE ="JScript">
    function width_init()
    {
        ID_Span.style.setExpression("width","
                                     trueBlueSpan.style.pixelWidth
                                     + oldYellowSpan.style.pixelWidth",
                                     "jscript"
                                     );
    }

    function Berechne()
    {alert(ID_Span.style.getExpression("width");}
</SCRIPT>
</HEAD>
<BODY onload= width_init();>
    <SPAN ID="ID_Span"
          STYLE="background-color:lightgreen;
                  width:expression( trueBlueSpan.style.pixelWidth
                                    + oldYellowSpan.style.pixelWidth
                                    )
                "
    >
    </SPAN>
    <BUTTON onclick= Berechne();>
</BODY>
</HTML>
```

.GetExtensionName()      Suffix einer Datei liefern ohne Trenner "."  
Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetExtensionName("../test\\text.txt"));
```

.GetFile()                Objekt FileSystemObject.File anhand Dateinamen einer existierenden Datei erzeugen  
(sonst wird Fehler erzeugt)

Beispiel:

```
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile("../test\\text.txt");
alert(Datei);
```

.GetFileName()            Name einer Datei mit Suffix und mit Trenner "."  
Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetFileName("../test\\text.txt"));
```

.GetFolder()              Objekt FileSystemObject.Folder anhand Ordernamen eines existierenden Ordners erzeugen  
sonst wird Fehler erzeugt

Beispiel:

```
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Ordner            = DateiSystem.GetFolder("../test\\");
alert(Ordner);
```

.getFullYear()            Jahreszahl vierstellig in lokaler Zeit liefern  
Script-Objekt Date



Integer, maximal **1970 + bzw. -285,616 Jahre**

Beispiel 1:

```
<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    //      erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    //      Aufbau    Cookie_Name = Cookie_Wert
    //                  Separator ist Gleichheitszeichen
    //      Index 0 für Cookieen_Name
    //      Index 1 für Cookieen_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (    ( !Gefunden )
        && ( Index < AnzahlCookies)
    );

    return CookieWert;
}
</SCRIPT>
```

Beispiel 2:

```
<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";
```



```

// Cache-Attribut festlegen
//      es können diverse Attribute definiert und somit Versionen von Input-Daten
//      verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitpunkt = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitpunktUTC = Zeitpunkt.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitpunktUTC;

    // ++++++ Daten chachen ++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache saveen
    InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;
    }

```



```

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " + parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getHours()           Stunden in lokaler Zeit liefern
                       Script-Objekt Date
                       Integer 0 bis 23

```

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    //      erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    //      Aufbau    Cookie_Name = Cookie_Wert
    //      Separator ist Gleichheitszeichen
    //      Index 0 für Cookie_Name
    //      Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (      ( !Gefunden )
              && ( Index < AnzahlCookies )
              );

    return CookieWert;
}
</SCRIPT>

```



Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = ZeitpunktJetzt.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

```



```

function fnInit()
{
    var AnzahlMillisekundenProTag = 86400000;

    var DatumDokumentErzeugung = new Date(document.fileCreatedDate);

    var DatumHeute = new Date();

    var TagesDifferenz = ( DatumHeute.getTime()
                        - DatumDokumentErzeugung.getTime()
                        )
                        / AnzahlMillisekundenProTag;

    alert( "Dokument erzeugt am " + DatumDokumentErzeugung
        + "\n..... also vor " + parseInt(TagesDifferenz) + " Tagen"
        );
}
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```

`.getItemInfo()` Wert des mit dem MediaItem Objekt verbundenen Attributes liefern  
 Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende Methoden nutzbar:

- `.getItemInfo()`
- `.getAttributeName()`

Eigenschaften nutzbar:

- `.attributeCount`

Es ist also vorher die Eigenschaft `.playState` auf Werte > 10 zu prüfen.  
 siehe Behavior `.style.mediaBar`

`.getMilliseconds()` Millisekunden in lokaler Zeit liefern  
 Script-Objekt Date  
 Integer 0 bis 999

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau    Cookie_Name = Cookie_Wert
    // Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen

```



```

        var Gefunden=false;
        var Index = 0;
        do
        {
            // aktuelles Cookie lesen
            CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

            // und auf Name prüfen
            if (Name == CookieNameUndWertAlsFeld [0])
            {
                CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
                Gefunden=true;
            }
            else
            {Index++;}
        }
        while (      ( !Gefunden )
            && ( Index < AnzahlCookies)
        );

        return CookieWert;
    }
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = ZeitpunktJetzt.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);
    }

```



```

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
            + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
        );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getMinutes()           Minuten in lokaler Zeit liefern
                        Script-Objekt Date
                        Integer 0 bis 59

Beispiel 1:
<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

    function LoescheCookie (Name, Value)    // Name und Wert sind Strings
    {
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
    }

    function LeseCookieWert(Name)
    // Name des zu lesenden Cookie ist String
    // liefert Cookiewert aus unescape() als String
    {
        var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert
    }

```





```

// Feld der Cookies referenzieren
// erzeugt durch Separation anhand Semikolon
var CookieFeld = document.cookie.split("; ");
var AnzahlCookies = CookieFeld.length;

// Feldelement umfasst Name und Wert des Cookie
// Aufbau Cookie_Name = Cookie_Wert
// Separator ist Gleichheitszeichen
// Index 0 für Cookie_Name
// Index 1 für Cookie_Wert
var CookieNameUndWertAlsFeld;

// CookieFeld auslesen und auf Cookie laut Name prüfen
var Gefunden=false;
var Index = 0;
do
{
    // aktuelles Cookie lesen
    CookieNameUndWertAlsFeld = CookieFeld[Index].split("=");

    // und auf Name prüfen
    if (Name == CookieNameUndWertAlsFeld [0])
    {
        CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
        Gefunden=true;
    }
    else
    {Index++;}
}
while ( ( !Gefunden )
    && ( Index < AnzahlCookies)
);

return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++ Daten chachen ++++++
    // aktuelle Daten holen laut Input-Objekt

```



```

        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache save
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }
</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
            + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
        );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getMonth()    Monat in lokaler Zeit liefern
                Script-Objekt Date
                Integer 0 bis 11

```

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

```



```

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =   Name
                        + "="
                        + escape(Value)
                        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau   Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (    ( !Gefunden )
            && ( Index < AnzahlCookies)
            );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();

```



```

        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n.... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```

.getNamedItem()

Zeiger auf Attribut liefern anhand des Attributnamen (analog zu ID oder NAME-Attribut)



ab IE 6.x

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Init()
    {
        var ZeigerAufFeld = ID_P.attributes;
        var ZeigerAufFeldElement = ZeigerAufFeld.getNamedItem("align");
        alert("ALIGN Attribut Wert = " + ZeigerAufFeldElement.value);
    }
</SCRIPT>
</HEAD>
<BODY ONLOAD="Init()">
    <P ID="ID_P" ALIGN="center">Test</P>
</BODY>
</HTML>

```

.GetParentFolderName()      Elternordner einer Datei liefern  
 Datei muss nicht existieren

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetParentFolderName("../test\\text.txt"));

```

.getSeconds()                Sekunden in lokaler Zeit liefern  
 Script-Objekt Date  
 Integer 0 bis 59

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)      // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                               + "="
                               + escape(Value)
                               + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

    function LoescheCookie (Name, Value)      // Name und Wert sind Strings
    {
        var document.cookie =    Name
                               + "="
                               + escape(Value)
                               + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
    }

    function LeseCookieWert(Name)
    // Name des zu lesenden Cookie ist String
    // liefert Cookiewert aus unescape() als String
    {
        var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

        // Feld der Cookies referenzieren
        // erzeugt durch Separation anhand Semikolon
        var CookieFeld = document.cookie.split("; ");
        var AnzahlCookies = CookieFeld.length;

        // Feldelement umfasst Name und Wert des Cookie
        // Aufbau    Cookie_Name = Cookie_Wert
        //            Separator ist Gleichheitszeichen
        // Index 0 für Cookie_Name
        // Index 1 für Cookie_Wert
        var CookieNameUndWertAlsFeld;

        // CookieFeld auslesen und auf Cookie laut Name prüfen
        var Gefunden=false;
        var Index = 0;
        do
        {
            // aktuelles Cookie lesen
            CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

            // und auf Name prüfen
            if (Name == CookieNameUndWertAlsFeld [0])
            {

```



```

        CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
        Gefunden=true;
    }
    else
    {Index++;}
}
while (      ( !Gefunden )
    && ( Index < AnzahlCookies)
);

return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = ZeitpunktJetzt.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"

```



```

        CLASS="user_data_speicher_klasse"
        TYPE="text"
    >
    <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
    <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```

.GetSpecialFolder()                    Objekt FileSystemObject.Folder von Ordnern des Betriebssytemes erzeugen

Beispiel:

```

var DateiSystem        = new ActiveXObject("Scripting.FileSystemObject");
var Ordner             = DateiSystem.GetSpecialFolder(0);
var DateiOffen         = Ordner.CreateTextFile("test.txt");
DateiOffen.writeline("Test");
DateiOffen.close();

```

.GetTempName()                    Bezeichner anhand Zufallszahl erzeugen  
Bezeichner verwendbar als Datei- oder Ordner-Bezeichner

Beispiel:

```

var DateiSystem        = new ActiveXObject("Scripting.FileSystemObject");
var Ordner             = DateiSystem.GetSpecialFolder(2);
var Dateiname          = DateiSystem.GetTempName();
var DateiOffen         = Ordner.CreateTextFile(Dateiname);
DateiOffen.writeline("Test");
DateiOffen.close();

```

.getTime()                    Zeitwert als Anzahl der Millisekunden relativ zum 01.01.1970 0 Uhr Weltzeit liefern  
positiv oder negativ möglich (wenn < 0 so vor obigen Datum)  
Anzahl der Millisekunden relativ zum 01.01.1970 0 Uhr Weltzeit  
**wenn < 0 so vor 01.01. 1970 0 Uhr Weltzeit**

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)     // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =     Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

    function LoescheCookie (Name, Value)     // Name und Wert sind Strings
    {
        var document.cookie =     Name
                                + "="
                                + escape(Value)

```



```

+ "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

```

```

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau Cookie_Name = Cookie_Wert
    // Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while ( ( !Gefunden )
        && ( Index < AnzahlCookies )
    );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

```





```

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getTimezoneOffset()    Minuten der lokalen Zeit als Differenz zur Weltzeit liefern, also das Offset der lokalen Zeitzone
                        Script-Objekt Date
                        Integer ab 0

```

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings

```



```

    {
        var date = new Date();
        var document.cookie =   Name
                               + "="
                               + escape(Value)
                               + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =   Name
                           + "="
                           + escape(Value)
                           + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau   Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while ( ( !Gefunden )
           && ( Index < AnzahlCookies )
           );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    // es können diverse Cachennamen definiert und somit Versionen von Cache
    // verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    // es können diverse Attribute definiert und somit Versionen von Input-Daten
    // verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

```



```
// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++ Daten chachen ++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache saveen
    InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>
```

Beispiel 3:

```
<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag = 86400000;

        var DatumDokumentErzeugung = new Date(document.fileCreatedDate);

        var DatumHeute = new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n.... also vor " + parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>
```



&lt;/SCRIPT&gt;

Beispiel 4:

```
var Jetzt = new Date();
alert(Jetzt.toLocaleString());
```

```
.getUTCDate()      Tag des Monats in Weltzeit liefern
                   Script-Objekt Date
                   Integer 1 bis 31
```

Beispiel 1:

```
<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
        + "="
        + escape(Value)
        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau    Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    //           Index 0 für Cookie_Name
    //           Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (    ( !Gefunden )
        && ( Index < AnzahlCookies )
    );

    return CookieWert;
}
</SCRIPT>
```

Beispiel 2:

```
<HTML>
<HEAD>
```



```

<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag = 86400000;
    }

```



```

var DatumDokumentErzeugung = new Date(document.fileCreatedDate);

var DatumHeute = new Date();

var TagesDifferenz = ( DatumHeute.getTime()
                      - DatumDokumentErzeugung.getTime()
                      )
                    / AnzahlMillisekundenProTag;

alert( "Dokument erzeugt am " + DatumDokumentErzeugung
      + "\n..... also vor " + parseInt(TagesDifferenz) + " Tagen"
      );
}
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getUTCDay()      Tag der Woche in Weltzeit liefern
                  Script-Objekt Date
                  Integer 0 bis 6
                   0 ist Sonntag
                   6 ist Samstag

```

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    //      erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    //      Aufbau    Cookie_Name = Cookie_Wert
    //                  Separator ist Gleichheitszeichen
    //      Index 0 für CookieName
    //      Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld[Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
    }
    while (Index < AnzahlCookies);
}

```



```

    }
    else
    {Index++;}
}
while ( ( !Gefunden )
    && ( Index < AnzahlCookies)
);

return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachname festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitpunkt = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitpunktUTC = Zeitpunkt.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitpunktUTC;

    // ++++++ Daten chachen ++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache saveen
    InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
<FORM ID="ID_Formular">
    <INPUT ID="ID_Input"
        CLASS="user_data_speicher_klasse"
        TYPE="text"

```



```

>
<INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
<INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
</FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
window.onload=fnInit;

function fnInit()
{
    var AnzahlMillisekundenProTag =86400000;

    var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

    var DatumHeute =new Date();

    var TagesDifferenz =    ( DatumHeute.getTime()
                            - DatumDokumentErzeugung.getTime()
                            )
                        / AnzahlMillisekundenProTag;

    alert( "Dokument erzeugt am " + DatumDokumentErzeugung
          + "\n.... also vor " +parseInt(TagesDifferenz) + " Tagen"
          );
}
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getUTCFullYear()    Jahreszahl vierstellig in Weltzeit liefern
                     Script-Objekt Date
                     Integer maximal 1970 + bzw. -285,616 Jahre

```

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                            + "="
                            + escape(Value)
                            + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                            + "="
                            + escape(Value)
                            + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau    Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;
}

```





```

        // CookieFeld auslesen und auf Cookie laut Name prüfen
        var Gefunden=false;
        var Index = 0;
        do
        {
            // aktuelles Cookie lesen
            CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

            // und auf Name prüfen
            if (Name == CookieNameUndWertAlsFeld [0])
            {
                CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
                Gefunden=true;
            }
            else
            {Index++;}
        }
        while ( ( !Gefunden )
            && ( Index < AnzahlCookies)
        );

        return CookieWert;
    }
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    // es können diverse Cachennamen definiert und somit Versionen von Cache
    // verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    // es können diverse Attribute definiert und somit Versionen von Input-Daten
    // verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitpunkt = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitpunktUTC = Zeitpunkt.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitpunktUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
    }

```



```

        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
            + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
        );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getUTCHours()    Stunden in Weltzeit liefern
                  Script-Objekt Date
                  Integer 0 bis 23

```

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

    function LoescheCookie (Name, Value)    // Name und Wert sind Strings
    {
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
    }

    function LeseCookieWert(Name)
    // Name des zu lesenden Cookie ist String
    // liefert Cookiewert aus unescape() als String
    {
        var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert
    }

```



```

// Feld der Cookies referenzieren
// erzeugt durch Separation anhand Semikolon
var CookieFeld = document.cookie.split("; ");
var AnzahlCookies = CookieFeld.length;

// Feldelement umfasst Name und Wert des Cookie
// Aufbau Cookie_Name = Cookie_Wert
// Separator ist Gleichheitszeichen
// Index 0 für Cookie_Name
// Index 1 für Cookie_Wert
var CookieNameUndWertAlsFeld;

// CookieFeld auslesen und auf Cookie laut Name prüfen
var Gefunden=false;
var Index = 0;
do
{
    // aktuelles Cookie lesen
    CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

    // und auf Name prüfen
    if (Name == CookieNameUndWertAlsFeld [0])
    {
        CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
        Gefunden=true;
    }
    else
    {Index++;}
}
while ( ( !Gefunden )
        && ( Index < AnzahlCookies)
        );

return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++ Daten chachen ++++++

```



```

        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }
</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                               - DatumDokumentErzeugung.getTime()
                             )
                               / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getUTCMilliseconds()    Millisekunden in Weltzeit liefern
                          Script-Objekt Date
                          Integer 0 bis 999

```

Beispiel 1:

```

<SCRIPT>
    function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
    {
        var date = new Date();
        var document.cookie =    Name
                                + "="
                                + escape(Value)
                                + "; expires=" + date.toGMTString(); // aktuelles Datum
    }

```



```

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =   Name
                        + "="
                        + escape(Value)
                        + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau   Cookie_Name = Cookie_Wert
    //          Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (      ( !Gefunden )
            && ( Index < AnzahlCookies)
            );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachname festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
// es können diverse Attribute definiert und somit Versionen von Input-Daten
// verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++++ Zeitstempel ++++++++

```



```

        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz =    ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
                                / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
            + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
        );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```



.getUTCMinutes()      Minuten in Weltzeit liefern  
 Script-Objekt Date  
 Integer 0 bis 59

Beispiel 1:

```
<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau    Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    // Index 0 für Cookieen_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (    ( !Gefunden )
            && ( Index < AnzahlCookies )
            );

    return CookieWert;
}
</SCRIPT>
```

Beispiel 2:

```
<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
// Cachename festlegen
// es können diverse Cachennamen definiert und somit Versionen von Cache
// verwaltet werden
```



```

var FreierCacheName = "InputCache";

// Cache-Attribut festlegen
//      es können diverse Attribute definiert und somit Versionen von Input-Daten
//      verwaltet werden
var FreiesCacheAttribut = "InputCacheAttribut";

// zu cachende Daten referenzieren
var InputDatenObjekt = ID_Formular.ID_Input;

function InputSichern()
{
    // ++++++ Zeitstempel ++++++
    var ZeitpunktJetzt = new Date();
    var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

    // Zeitstempel festlegen: Ab jetzt + 20 Minuten
    var Zeitstempel = ZeitpunktJetztInMinuten + 20;

    // und als UTC-Format erzeugen
    var ZeitstempelUTC = Zeitstempel.toUTCString();

    // und Zeitstempel dem Input-Objekt verpassen
    ID_Input.expires = ZeitstempelUTC;

    // ++++++ Daten chachen ++++++
    // aktuelle Daten holen laut Input-Objekt
    var InputDaten = InputDatenObjekt.value;

    // Attribut instanzieren und mit Daten füllen
    InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

    // und Cache saveen
    InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
    window.onload=fnInit;

    function fnInit()
    {
        var AnzahlMillisekundenProTag =86400000;

        var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

        var DatumHeute =new Date();

        var TagesDifferenz = ( DatumHeute.getTime()
                                - DatumDokumentErzeugung.getTime()
                                )
    }

```





```

        / AnzahlMillisekundenProTag;

        alert( "Dokument erzeugt am " + DatumDokumentErzeugung
              + "\n..... also vor " + parseInt(TagesDifferenz) + " Tagen"
              );
    }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getUTCMonth()      Monat in Weltzeit liefern
                    Script-Objekt Date
                    Integer 0 bis 11

```

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    //      erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    //      Aufbau    Cookie_Name = Cookie_Wert
    //      Separator ist Gleichheitszeichen
    //      Index 0 für Cookie_Name
    //      Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (    ( !Gefunden )
            && ( Index < AnzahlCookies )
            );

    return CookieWert;
}

```



```
    }
</SCRIPT>
```

Beispiel 2:

```
<HTML>
<HEAD>
<STYLE>
.user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    // es können diverse Cachenames definiert und somit Versionen von Cache
    // verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    // es können diverse Attribute definiert und somit Versionen von Input-Daten
    // verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>
```

Beispiel 3:

```
<SCRIPT>
```



```

window.onload=fnInit;

function fnInit()
{
    var AnzahlMillisekundenProTag =86400000;

    var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

    var DatumHeute =new Date();

    var TagesDifferenz =    ( DatumHeute.getTime()
                            - DatumDokumentErzeugung.getTime()
                            )
                          / AnzahlMillisekundenProTag;

    alert( "Dokument erzeugt am " + DatumDokumentErzeugung
          + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
          );
}
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

```

.getUTCSeconds()      Sekunden in Weltzeit liefern  
                          Script-Objekt Date  
                          Integer 0 bis 59

Beispiel 1:

```

<SCRIPT>
function ErzeugeCookie(Name, Value)    // Name und Wert sind Strings
{
    var date = new Date();
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=" + date.toGMTString(); // aktuelles Datum
}

function LoescheCookie (Name, Value)    // Name und Wert sind Strings
{
    var document.cookie =    Name
                          + "="
                          + escape(Value)
                          + "; expires=Fri, 31 Dec 1999 23:59:59GMT;"; // altes Datum
}

function LeseCookieWert(Name)
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
    var CookieWert=""; // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

    // Feld der Cookies referenzieren
    // erzeugt durch Separation anhand Semikolon
    var CookieFeld = document.cookie.split("; ");
    var AnzahlCookies = CookieFeld.length;

    // Feldelement umfasst Name und Wert des Cookie
    // Aufbau    Cookie_Name = Cookie_Wert
    //           Separator ist Gleichheitszeichen
    // Index 0 für Cookie_Name
    // Index 1 für Cookie_Wert
    var CookieNameUndWertAlsFeld;

    // CookieFeld auslesen und auf Cookie laut Name prüfen
    var Gefunden=false;
    var Index = 0;
    do
    {
        // aktuelles Cookie lesen
        CookieNameUndWertAlsFeld = CookieFeld [Index].split("=");

        // und auf Name prüfen
    }
}

```



```

        if (Name == CookieNameUndWertAlsFeld [0])
        {
            CookieWert = unescape(CookieNameUndWertAlsFeld [1]);
            Gefunden=true;
        }
        else
        {Index++;}
    }
    while (      ( !Gefunden )
        && ( Index < AnzahlCookies)
    );

    return CookieWert;
}
</SCRIPT>

```

Beispiel 2:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachenames definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++++ Zeitstempel ++++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = ZeitpunktJetzt.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++++ Daten chachen ++++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>

```



```

<FORM ID="ID_Formular">
  <INPUT ID="ID_Input"
    CLASS="user_data_speicher_klasse"
    TYPE="text"
  >
  <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
  <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
</FORM>
</BODY>
</HTML>

```

Beispiel 3:

```

<SCRIPT>
  window.onload=fnInit;

  function fnInit()
  {
    var AnzahlMillisekundenProTag =86400000;

    var DatumDokumentErzeugung =new Date(document.fileCreatedDate);

    var DatumHeute =new Date();

    var TagesDifferenz =  ( DatumHeute.getTime()
                          - DatumDokumentErzeugung.getTime()
                          )
                        / AnzahlMillisekundenProTag;

    alert( "Dokument erzeugt am " + DatumDokumentErzeugung
          + "\n..... also vor " +parseInt(TagesDifferenz) + " Tagen"
          );
  }
</SCRIPT>

```

Beispiel 4:

```

var Jetzt = new Date();
alert(Jetzt.toLocaleString());

.getYear()          deprecated

.go()               Aktivierung irgendeiner Url aus dem Verlauf
                   entspricht beliebiger Selektion aus der Verlaufsliste
< 0
-1                  entspricht wie wenn nicht kodiert
                   entspricht Druck des Back-Buttons
                   je kleiner umso weiter zuvorliegend
> 0                 +1      entspricht Druck des Forward-Buttons
                   je größer umso weiter nachliegend
0                   entspricht Reload des aktuellen Dokumentes

```

Beispiel: Reload des Dokumentes nach Resize des Browserfensters

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function neuLaden()
    {
        if (document.all)
        {history.go(0);}
    }
// -->
</SCRIPT>
</HEAD>
<BODY onResize=neuLaden();">
</BODY>

```

```

.hasChildNodes()    prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten
                   (Textelemente) in einem Objekt
                   DOM nicht geändert

```

Beispiel:

```

<HTML>
<BODY onload="alert(ID_Div.hasChildNodes());">

```



```
<DIV ID="ID_Div" TITLE="Tooltip-Text ">Es ist ein Tooltip-Text vorhanden</DIV>
</BODY>
</HTML>
```

<code>.hasFeature()</code>	Objektzugehörigkeit zum DOM (Document Object Model-Standard ) (HTML-DOM bzw. XML-DOM)
----------------------------	--

Beispiel:

```
var Wert = document.implementation.hasFeature("HTML","1.0");  
// Objekt document prüfen ob im HTML-DOM Version 1.0 enthalten ist
```

<code>.hasFocus()</code>	Objekt im Focus-Zustand
<code>true</code>	Objekt hat den Focus
<code>false</code>	Objekt hat keinen Focus

<code>.hasOwnProperty()</code>	<p>prüfen ob zum Objekt eine Eigenschaft vorhanden ist, jedoch leider <b>nicht</b> aus Prototyping einer per new erzeugten Instanz des JScript-Objektes</p> <p>Für private Objekte, die per new-Anweisung mit privatem Konstruktor erzeugt wurden, wird leider <b>nicht</b> die Eigenschaft <code>.prototype</code> erzeugt ! Prototyping kann also nur innerhalb des Konstruktors erfolgen und nicht nachträglich per Eigenschaft <code>.prototype</code> .</p> <p>siehe auch <code>.prototype</code></p>
--------------------------------	--

Beispiel:

```
function MaximumErmittleIn( )
{
    var max = this[0];    // this referenziert die Objekt-Instanz, in dem die Methode vorhanden ist,
                          // also Variable Feld

    for (var i = 1; i < this.length; i++)
    {
        if (max < this[i])
            {max = this[i];}
    }

    return max;
}
```

```
// neue Methode per Prototyping hinzufügen zum JScript-Objekt Array
Array.prototype.NeueArrayMethode = MaximumErmitteln; // ohne () kodieren !
```

```
// Feld vom Array-Typ erzeugen mit der Methode .NeueArrayMethode()
var Feld = new Array(1, 2, 3, 4, 5, 6);      // 6 numerische Elemente
```

```
alert( (Feld.prototype.hasOwnProperty("NeueArrayMethode")); // leider false
alert( (Array.prototype.hasOwnProperty("NeueArrayMethode")); // true
```

<code>.hide()</code>	ein angezeigtes Popup-Fenster schliessen Hinweis: Das Popup-Fenster wird automatisch geschlossen, wenn ein anderes Objekt den Focus erhält bzw. aktiv wird, z.B. durch Klick des Users außerhalb des Popupfensters. ändert Wert der Eigenschaft <code>.isOpen</code>
----------------------	--

Beispiel für diverse Meldungsfenster per **jeweiligem** PopUp-Fenster ( es kann immer nur genau 1 Popup-Fenster angezeigt werden):

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">

    var PopUpFensterFeld = new Array();

    // nachfolgende Felder beschreiben die PopUp-Fenster und müssen
    // identische Anzahl der Feldelemente haben
    // Feld-Index ist die Nummer des PopUp-Fensters

    var PopUpFenster_RahmenStyle_Feld = new Array
    (
        "solid black 4px",
        "solid blue 3px",
        "solid green 2px"
    );

    var PopUpFenster_HintergrundFarbe_Feld = new Array
    (
        "yellow",
        "gray",

```



```

        "orange"
    );

    var PopUpFenster_Inhalt_Feld = new Array
    (
        "<B>Inhalt PopUpFenster 0<B>",
        "Inhalt PopUpFenster 1",
        "<TT>Inhalt PopUpFenster 2<TT>"
    );

    var PopUpFenster_X_Koordinate_Feld = new Array
    (
        100,
        150,
        200
    );

    var PopUpFenster_Y_Koordinate_Feld = new Array
    (
        150,
        200,
        250
    );

    var PopUpFenster_Breite_Koordinate_Feld = new Array
    (
        80,
        140,
        200
    );

    var PopUpFenster_Hoehe_Koordinate_Feld = new Array
    (
        100,
        140,
        180
    );

    var AnzahlPopUpFenster = PopUpFenster_Hoehe_Koordinate_Feld.length;

    function PopupFensterInstanzieren(NummerDesPopUpFensters, ZeigerAufElternFenster)
    // NummerDesPopUpFensters ab 0
    {
        PopUpFensterFeld[NummerDesPopUpFensters] =
            ZeigerAufElternFenster.createPopup();
    }

    function PopupFensterFuellen(NummerDesPopUpFensters)
    {
        // Body des Popup-Fensters gestalten
        var PopupFenster_Body =
            PopUpFensterFeld[NummerDesPopUpFensters].document.body;

        PopupFenster_Body.style.backgroundColor =
            PopUpFenster_HintergrundFarbe_Feld[NummerDesPopUpFensters];

        PopupFenster_Body.style.border =
            PopUpFenster_RahmenStyle_Feld[NummerDesPopUpFensters];

        PopupFenster_Body.innerHTML =
            PopUpFenster_Inhalt_Feld[NummerDesPopUpFensters];
    }

    function PopupFensterAnzeigen(NummerDesPopUpFensters,
        ObjektZuDemPopUpFensterRelativPositioniertIst
    )
    {
        // Popup-Fenster anzeigen und damit öffnen
        PopUpFensterFeld[NummerDesPopUpFensters].show(
            PopUpFenster_X_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Y_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Breite_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Hoehe_Koordinate_Feld[NummerDesPopUpFensters],
            ObjektZuDemPopUpFensterRelativPositioniertIst
        )
    }

```



```

    }
    );

    function PopUpFensterSchliessen(NummerDesPopUpFensters)
    {
        var Zeiger = PopUpFensterFeld[NummerDesPopUpFensters];

        if (Zeiger.isOpen)
        { Zeiger.hide(); }
    }

    function Init()
    {
        // PopUpFenster instanzieren im aktuellen Fenster (window)
        for (var i = 0 ; i < AnzahlPopUpFenster; i++)
        {
            PopupFensterInstanzieren(i, window);
            PopupFensterFuellen(i);
        }
    }
</SCRIPT>
</HEAD>
<BODY onload="Init();">
    Durch Klick ausserhalb des Popup-Fensters wird dieses auch automatisch geschlossen.
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 anzeigen"
        onclick=" PopupFensterAnzeigen(0, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 anzeigen"
        onclick=" PopupFensterAnzeigen(1, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 anzeigen"
        onclick=" PopupFensterAnzeigen(2, document.body);"
    >
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 schliessen"
        onclick=" PopUpFensterSchliessen(0);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 schliessen"
        onclick=" PopUpFensterSchliessen(1);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 schliessen"
        onclick=" PopUpFensterSchliessen(2);"
    >
</BODY>
</HTML>

```

**.ImportExportFavorites()** Import und Export der Favoritenliste aus dem/ in das Netscape-Bookmark-Format  
 User muss Import/Export bestätigen per Dialogbox  
 es wird HTTP benutzt  
 Ziel-bzw. Quellort: Es wird durch Import immer hinzugefügt  
 Es wird nach Export nicht gelöscht.  
 siehe Objekt window.external

Beispiel:  
 window.external.ImportExportFavorites(true,"http://www.test.com");

**.indexOf()** Suche einer Teilkette in einem String bzw. Stringliteral und die Startposition der Teilkette als Index  
 im String bzw. Stringliteral liefern  
 es wird nur das ERSTE Auffinden der Teilkette geliefert  
 Suche im String erfolgt von links nach rechts  
 unterscheidet Gross und Klein  
 siehe Script-Objekt String

Beispiel 1:  
 var StringLiteral ="StringLiteral";  
 StringLiteral.indexOf("gLi", 2) liefert 5  
 StringLiteral.indexOf("gLi") liefert 5  
 StringLiteral.indexOf("gLi", 10) liefert -1





"StringLiteral".indexOf("gLi", 2) liefert 5

Beispiel 2 E-Mail-Adresse auf "@" prüfen:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function pruefe_zeichenkette(zeichenkette)
    {
        if (zeichenkette.indexOf('@') = -1)
        {alert("@ fehlt !");}
        else
        {alert("@ gefunden !");}
    }
//-->
</SCRIPT>
</HEAD>

<BODY>
<FORM>
    e-Mail:
    <INPUT TYPE="text"
        NAME="Mail"
        VALUE=""
    >
    <INPUT TYPE="button"
        VALUE="Ueberpruefen"
        onclick=" pruefe_zeichenkette (this.form.Mail.value)"
    >
</FORM>
</BODY>
</HTML>
```

Beispiel 3: Numerischen Wert als Zeichenkette liefern und dabei Dezimalpunkt zu Dezimalkomma umwandeln

Es wird angenommen, dass genau ein Dezimalpunkt vorkommt.

```
function punkt_zu_komma (numerischer_wert)
{
    var zeichenkette = numerischer_wert.toString();
    var funktionswert=zeichenkette;

    var pos_punkt = zeichenkette.indexOf(".");

    if(pos_punkt >=0)
    {
        funktionswert =      zeichenkette.substring(0, pos_punkt)
                           + ","
                           + zeichenkette.substring(pos_punkt + 1, zeichenkette.length);
    }

    return zeichenkette;
}
```

.inRange() prüfen ob 2 Textbereiche sich einschliessen z.B. bei verschachtelten DIV's  
per textrange Objekt  
nur unter Windows 32-Bit

Beispiel:

```
<HTML>
<SCRIPT>
    window.onload=fnCheck;

    function fnCheck()
    {
        var TextBereich = document.body.createTextRange();

        var TextBereichKopie = TextBereich.duplicate();

        alert(TextBereich.inRange(TextBereichKopie));
    }
</SCRIPT>
<BODY>
<DIV>
```



```

        Test
    </DIV>
</BODY>
</HTML>

```

**.insertAdjacentElement()** Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann  
wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM  
nur nach dem kompletten Laden des Dokumentes möglich  
DOM wird geändert

Beispiel:

```

<SCRIPT>
function Hinzufuegen()
{
    var ZeigerAufLI = document.createElement("LI");
    ID_Liste.children(0).insertAdjacentElement("beforeBegin", ZeigerAufLI);
    ZeigerAufLI.innerText = "Listeneintrag 0";
}
</SCRIPT>
<BODY>
<OL ID = "ID_Liste">
    <LI>Listeneintrag 1</LI>
    <LI>Listeneintrag 2</LI>
    <LI>Listeneintrag 3</LI>
</OL>
<INPUT TYPE = "button" VALUE = " Hinzufuegen" onclick=" Hinzufuegen()">
</BODY>

```

**.insertAdjacentHTML()** HTML-Code und/oder Script-Code in ein Element einfügen, wobei die  
Lage definiert sein kann  
nur nach dem kompletten Laden des Dokumentes möglich  
HTML- und Script-Code müssen syntaktisch korrekt sein  
wenn nicht, so wird das Einfügen **nicht** ausgeführt  
eingefügter Code wird **nur** dann sofort geparkt und ausgeführt, wenn syntaktisch  
korrekt ist  
bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden  
DOM wird geändert

Beispiel:

```

var HTML_Code =    "<INPUT TYPE =button"
                  +    " VALUE='Bitte klicken'"
                  +    " onclick='Anzeige()'"
                  + ">"
                  + "<BR>";

var JavaScript_Code =    "<SCRIPT DEFER>"                // DEFER muss kodiert sein
                        + "function Anzeige(){alert('Anzeige aktiv')} "
                        + "</SCRIPT>";

ID_Div.insertAdjacentHTML("afterBegin", HTML_Code + JavaScript_Code);

<DIV ID="ID_Div">
    Test
</DIV>

```

**.insertAdjacentText()** Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann  
nur nach dem kompletten Laden des Dokumentes  
DOM wird geändert

Beispiel:

```

var PlainText = " Hinzugefuegter Text ";
ID_Div.insertAdjacentText("afterBegin", PlainText);

<DIV ID="ID_Div">
    Test
</DIV>

```

**.insertBefore()** Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern  
einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein  
Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde  
DOM wird geändert

Beispiel:

```

<SCRIPT>
function Einfuegen()
{
    var ZeigerAufNeuesLI = document.createElement("LI");

```



```

        ID_UL.insertBefore(ZeigerAufNeuesLI, ID_LI);
        ZeigerAufNeuesLI.innerText="2";
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN onclick= Einfuegen()>Klick</SPAN>
    <UL ID="ID_UL">
        <LI >1</LI>
        <LI ID="ID_LI">3</LI>
        <LI >4</LI>
    </UL>
</BODY>

```

**.insertCell()**                      Zelle TD in die Zeile einer Tabelle einfügen  
 Zelle TH **nicht direkt** erzeugbar: Es muss **.innerHTML** mit **<B>**-Tag gefüllt werden  
 siehe Objekt **table.tr**

Beispiel :

```

var Zeile = zeiger_auf_tabelle.insertRow();
var Zelle = Zeile.insertCell();
Zelle.innerHTML = " <I>Test </I>";

```

**.insertData()**                      Teilkette in ein Objekt einfügen

**.insertRow()**                      Zeile in die Tabelle einfügen  
 siehe Objekt **table**  
 siehe Objekt **table.tBody**  
 siehe Objekt **table.tHead**  
 siehe Objekt **table.tFoot**

Beispiel :

```

var Zeile = zeiger_auf_tabelle.insertRow();
var Zelle = Zeile.insertCell();
Zelle.innerHTML = " <I>Test </I>";

```

**.isComponentInstalled()**           Verfügbarkeit der Komponente (gedownloadet UND installiert UND aktuelle Versionsnummer  
 ist mindestens die minimale Versionsnummer)  
 Behavior **.style.clientCaps**

**.isEqual()**                          Auf Identität zweier Textbereiche prüfen  
 per **textrange** Objekt

**isFinite()**                          ermittelt, ob Wert einer Instanz numerisch endlich ist

**.isHomePage()**                      Lage der Homepage auf einer Domain  
 true            ist eine Homepage UND Homepage liegt auf gleicher Domain  
 false           ist keine Homepage und/oder Homepage liegt auf anderen Domain

**isNaN()**                            ermittelt, ob Wert einer Instanz nicht numerisch ist

Beispiele:

```

var Kette ="10.23";
var Wert =10.23;

var Ergebnis1=parseFloat(Kette);
var Ergebnis2=floatValue=parseFloat(Wert);

alert(isNaN(Ergebnis1);
alert(isNaN(Ergebnis2);

```

Hinweis: Methoden **parseFloat** und **parseInt** liefern NaN, wenn Parameter nicht numerisch ist  
 Bsp:     var zahl=parseFloat("text");     ifNaN() liefert true

**.isPrototypeOf()**                   prüfen ob Objekt per **new** erzeugt wurde, also eine Instanz eines anderen JScript-Objektes ist  
 Für private Objekte, die per **new**-Anweisung mit privatem Konstruktor erzeugt wurden, wird leider **nicht**  
 die Eigenschaft **.prototype** erzeugt ! Prototyping kann also nur innerhalb des Konstruktors  
 erfolgen und nicht nachträglich per Eigenschaft **.prototype** .  
 siehe auch **.prototype**

Beispiel:

```

var Variable = new RegExp();
alert( (RegExp.prototype.isPrototypeOf(Variable));                      // true.

```

**.IsSubscribed()**                   auf Verfügbarkeit der Unterschrift zu einer Microsoft Active Channel-Datei prüfen  
 (Channel Definition Format-Datei mit Dateisuffix ist \*.cdf)  
 nur für Dateien in gemeinsamer Domain, also nicht domain-übergreifend  
 liefert immer Scriptfehler, wenn Domain-Wechsel durch Url der CDF-Datei erzeugt wird



siehe Objekt window.external

.italics() HTML-Tag <I> erzeugen in der Form <I>text</I>  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral    ="Text der I wird"
var HTML_Kette      = StringLiteral.italics();
HTML_Kette          = "Text der I wird".italics();
```

entspricht <I>Text der I wird</I>

eval(HTML\_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
document.write(HTML\_Kette);

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des  
Attributnamen (analog zu ID oder NAME-Attribut) liefern  
außer bei Formular mit <INPUT TYPE=image ...>  
da dafür die children-Collection verwendet werden muss !

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
var ZeigerAufCollectionDocumentAll = document.all;

if (ZeigerAufObjekt!=null)
{
    for (i=0; i< ZeigerAufCollectionDocumentAll.length; i++)
    {alert(ZeigerAufCollectionDocumentAll.item(i).tagName);}
}
</SCRIPT>
```

Beispiel 2:

```
<HTML>
<HEAD>
<SCRIPT>
function Init()
{
    var ZeigerAufFeld = ID_P.attributes;
    for (Index = 0; Index < ZeigerAufFeld.length; Index ++)
    {
        var ZeigerAufFeldElement = ZeigerAufFeld.item(Index);
        // hier numerischer Index
        var AttributWertSpezifiziert = ZeigerAufFeldElement.specified;
        // true oder false
        var KnotenName = ZeigerAufFeldElement.nodeName;
        // String
        var KnotenWert = ZeigerAufFeldElement.nodeValue;
        alert(
            "Knotenname = "
            + KnotenName
            + " mit spezifiziert = "
            + AttributWertSpezifiziert
            + " und Wert = "
            + KnotenWert
        );
    }
}
</SCRIPT>
</HEAD>
<BODY ONLOAD="Init()">
    <P ID="ID_P">Test</P>
</BODY>
</HTML>
```

.item() Element einer Collection liefern laut aktueller Position in der Collection  
zur Positionierung innerhalb der Collection wird ein interner Index verwendet  
siehe Enumerator JScript-Objekt

Beispiel Laufwerke auf dem PC des Users ermitteln:

```
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner
```

```
for ( ; ! DateiSystem_Laufwerke.atEnd(); DateiSystem_Laufwerke.moveNext())
```



```

{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]"; }
    }

    Kette += LaufwerkName + "\n";
}
alert (Kette);

```

.Item()                      Inhalt des Datenfeldes anhand Schlüssel liefern

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchlüsselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Date anzeigen
    alert(DatenSpeicher.Item(DatenSchluessel));
}

```

.Item()                      Eintrag in der Collection FileSystemObject.Drives liefern  
nur in Verbindung mit JScript-Objekt Enumerator

Beispiel Laufwerke auf dem PC des Users ermitteln:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netiname oder Volume-Bezeichner

for ( ; ! DateiSystem_Laufwerke.atEnd(); DateiSystem_Laufwerke.moveNext() )
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;

```



```
Kette = Kette + LaufwerkBuchstabe + " - ";

// Art des Laufwerkes ermitteln

if (Laufwerk.DriveType == 3)
{
    // ist Netzlaufwerk

    // öffentlicher Netz-Name des Laufwerkes
    LaufwerkName = Laufwerk.ShareName;
}
else
{
    // nicht Netzwerk-Laufwerk

    // prüfen ob Laufwerk bereit ist
    if (Laufwerk.IsReady)
    {
        // ist bereit, dann Volume-Bezeichner ermittelbar
        LaufwerkName = Laufwerk.VolumeName;
    }
    else
    {
        LaufwerkName = "[Drive not ready]";
    }
}

Kette += LaufwerkName + "\n";
}
alert (Kette);
```

.Item()	Eintrag in der Collection FileSystemObject.Files liefern nur in Verbindung mit JScript-Objekt Enumerator
---------	---

Beispiel:

```

var Ordner                = "c:\\windows\\desktop\\";
var DateiSystem           = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Ordner    = DateiSystem.GetFolder(Ordner);
var DateiSystem_Dateien   = DateiSystem_Ordner.Files;
var DateiSystem_FilesCollection = new Enumerator(DateiSystem_Dateien);
var Kette                = "";
for (; !DateiSystem_FilesCollection.atEnd(); DateiSystem_FilesCollection.moveNext())
{Kette = Kette + DateiSystem_FilesCollection.item() + "\n";}
alert(Kette);

```

.Item()	Eintrag in der Collection Collection.FileSystemObject.Folder.Folders liefern nur in Verbindung mit JScript-Objekt Enumerator
---------	--

Beispiel:

```
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner               = DateiSystem.GetFolder(OrdnerName);
var DateiSystem_FoldersCollection = new Enumerator(Ordner.SubFolders);
var Kette                = "";
for (; !DateiSystem_FoldersCollection.atEnd(); DateiSystem_FoldersCollection.moveNext())
{Kette = Kette + DateiSystem_FoldersCollection.item() + "\n",}
alert(Kette);
```

<code>.Items()</code>	Inhalte aller Datenfelder des Dictionary-Objektes als Referenz liefern, die als Zeiger für den Konstruktor <code>new VBAArray()</code> (VisualBasic-Anweisung) dient Datenfelder-Reihenfolge laut Folge im Dictionary
-----------------------	--

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}

// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date           = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // und Daten-Referenz bilden
}
```



```

var DatenReferenz = DatenSpeicher.Items();

// und Feld bilden
//      VisualBasic-Feld erzeugen
var DatenOhneSchluessel_Feld = new VBArray(DatenReferenz);
//      und nach JScript-Feld konvertieren
DatenOhneSchluessel_Feld = DatenOhneSchluessel_Feld.toArray();

// und Daten anzeigen
var DatenOhneSchluessel_FeldLaenge = DatenOhneSchluessel_Feld.length

if (DatenOhneSchluessel_FeldLaenge > 0)
{
    for (var i = 0 ; i < DatenOhneSchluessel_FeldLaenge; i++)
    {alert("Date " + i + " = " + DatenOhneSchluessel_Feld [i]);}
}
else
{alert("Dictionary ist leer!");}
}

```

`.javaEnabled()` prüfen auf generelle Ausführbarkeit von Javacode, also ob Java-Maschine im Browser aktiv ist  
per navigator Objekt  
true Javacode ist ausführbar  
false Javacode ist nicht ausführbar

Hinweis: Eigenschaft `.javaEnabled` des Behavior `clientCaps` zeigt die Verfügbarkeit der  
**Microsoft Virtuellen Maschine für Java** an

Achtung: Aufgrund eines Streites mit dem Unternehmen Sun entwickelt Microsoft ab IE 6.x die MS VM nicht mehr weiter, hält die VM aber noch per Download verfügbar. Für eine aktuellere Version muss der User selbständig beim Unternehmen Sun eine Lizenz für Java erwerben und die zugehörige Software installieren (inklusive Updates). Es reicht dabei die Run-Time-Version der Virtuellen Java Maschine von Sun. Ob allerdings Sun-Java-Standards in Microsoft-Produkten implementiert werden, ist nicht gesichert.

`.javaEnabled()` Java-Verfügbarkeit  
ab IE 6.x  
siehe Objekt `window.clientInformation`  
true Javacode ist ausführbar  
false Javacode ist nicht ausführbar

Hinweis: Eigenschaft `.javaEnabled` des Behavior `clientCaps` zeigt die Verfügbarkeit der  
**Microsoft Virtuellen Maschine für Java** an

Achtung: Aufgrund eines Streites mit dem Unternehmen Sun entwickelt Microsoft ab IE 6.x die MS VM nicht mehr weiter, hält die VM aber noch per Download verfügbar. Für eine aktuellere Version muss der User selbständig beim Unternehmen Sun eine Lizenz für Java erwerben und die zugehörige Software installieren (inklusive Updates). Es reicht dabei die Run-Time-Version der Virtuellen Java Maschine von Sun. Ob allerdings Sun-Java-Standards in Microsoft-Produkten implementiert werden, ist nicht gesichert.

`.join()` Feld auslesen und als String liefern, wobei jedes Feldelement mit einem freidefinierbaren  
Trenner im String getrennt wird, der mit eingebaut wird  
Feld elementweise nach String kopieren und als 1 gemeinsamen String liefern  
Feld hat die Objektklasse Script-Objekt Array

Beispiele:

```

var Feld = new Array(0,1,2,3,4);
alert (Feld.join("-")); // "0-1-2-3-4"
alert(Feld.join());    // "0,1,2,3,4"

var Feld = new Array("Wind","Rain","Fire");
var Kette = Feld.join(" + ");    // Kette enthält "Wind + Rain + Fire"

```

`.Key()` Datenschlüssel im Dictionary ändern

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date            = "test"

```



```
// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // dann Schlüssel ändern
    DatenSpeicher.Key(DatenSchlüssel) = "b";

    // und Meldung ob Date vorhanden ist
    alert(SchlüsselVorhanden("b"));
}
```

**.Keys()** Inhalt aller Schlüsselfelder des Dictionary-Objektes als Referenz liefern, die als Zeiger für den Konstruktor `new VBAArray ()` (VisualBasic-Anweisung) dient  
Schlüsselfelder-Reihenfolge laut Folge im Dictionary

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}

// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // und Daten-Referenz bilden
    var SchlüsselReferenz = DatenSpeicher.Keys();

    // und Feld bilden
    // VisualBasic-Feld erzeugen
    var SchlüsselOhneDaten_Feld = new VBAArray(SchlüsselReferenz);
    // und nach JScript-Feld konvertieren
    SchlüsselOhneDaten_Feld = SchlüsselOhneDaten_Feld.toArray();

    // und Daten anzeigen
    var SchlüsselOhneDaten_FeldLaenge = SchlüsselOhneDaten_Feld.length

    if (SchlüsselOhneDaten_FeldLaenge > 0)
    {
        for (var i = 0 ; i < SchlüsselOhneDaten_FeldLaenge; i++)
        {alert("Schlüssel " + i + " = " + SchlüsselOhneDaten_Feld [i]);}
    }
    else
    {alert("Dictionary ist leer!");}
}
```

**.lastIndexOf()** Suche einer Teilkette in einem String bzw. Stringliteral und die Startposition der Teilkette als Index im String bzw. Stringliteral liefern  
es wird nur das LETZTE Auffinden der Teilkette geliefert  
Suche im String erfolgt von links nach rechts  
unterscheidet Gross und Klein  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral ="StringLiteral";
StringLiteral.lastIndexOf("t", 2) liefert 8
StringLiteral.lastIndexOf("t") liefert 8
StringLiteral.lastIndexOf("t", 10) liefert -1
"StringLiteral".indexOf("t", 2) liefert 8
```

**.lastPage()** letzte Seite des Datasets in Tabelle anzeigen  
Eigenschaft `.dataPageSize` muss belegt worden sein  
siehe Objekt table

Beispiel:

```
Datensätze liegen in der Datei adress.txt
Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument
hat folgenden Inhalt:
```





1. Satz  
ab 2. Satz

**vorname;name;telefon**  
Guericke;"Otto, von";0815  
Willmann;Theo;1234  
Xantippe;Isa;5678  
Arktin;Richard;1055

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
    var SortierRichtung = true;

    function Sortieren(FeldBezeichner)
    {
        // Sortierrichtung festlegen
        var Kette = "-"; // Annahme
        if (SortierRichtung) {Kette = "+";}

        // nächste Sortierung umgekehrt
        SortierRichtung = !SortierRichtung;

        // sortieren
        ID_Datenbank.Sort= Kette + FeldBezeichner;

        // und das Ergebnis anzeigen
        ID_Datenbank.Reset();
    }

    function vorwaerts(AnzahlSaetze)
    {
        // nächste Seite anzeigen
        document.all.ID_Tabelle.nextPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaetze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition !=
                ID_Datenbank.recordset.RecordCount)
            {ID_Datenbank.recordset.MoveNext();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == ID_Datenbank.recordset.RecordCount)
        {alert("Letzter Datensatz erreicht!");}
    }

    function rueckwaerts(AnzahlSaetze)
    {
        // vorhergehende Seite anzeigen
        document.all.ID_Tabelle.previousPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaetze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition > 1)
            {ID_Datenbank.recordset.MovePrevious();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == 1)
        {alert("Erster Datensatz erreicht!");}
    }
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
    CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
    <PARAM NAME="DataURL" VALUE="adress.txt">
    <PARAM NAME="UseHeader" VALUE="True">
    <PARAM NAME="FieldDelim" VALUE=";">
</OBJECT>

<TABLE ID="ID_Tabelle"
    DATASRC=#ID_Datenbank
    DATAPAGESIZE=1
```



```

>
    <THEAD>
        <TR>
            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
            <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
        </TR>
    </THEAD>
    <TBODY>
        <TR>
            <TD><DIV DATAFLD="vorname"></DIV></TD>
            <TD><DIV DATAFLD="name"></DIV></TD>
            <TD><DIV DATAFLD="telefon"></DIV></TD>
        </TR>
    </TBODY>
</TABLE>
<BR>
<INPUT                TYPE="button"
                        VALUE="Zurueck"
                        onclick=rueckwaerts(1)"
>                                <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //-->

<INPUT                TYPE="button"
                        VALUE="Vorwaerts"
                        onclick=vorwaerts(1)"
>                                <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //-->
</BODY>
</HTML>

```

.link()                   HTML-Link <A> als HTML-Kette erzeugen in der Form "<A HREF='....'>text</A>  
                           und ohne weitere Attribute ID, HREF etc.  
                           siehe Script-Objekt String

Beispiel:

```

var StringLiteral      ="Sichtbarer Ankertext"
var HTML_Kette         = StringLiteral.link("WertDesHREFAttributes");
HTML_Kette             = "Sichtbarer Ankertext".link("WertDesHREFAttributes");

```

entspricht <A HREF="WertDesHREFAttributes">Sichtbarer Ankertext</A>

eval(HTML\_Kette);           erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
 document.write(HTML\_Kette); zeigt den Anker an und erzeugt Eintrag in der Collection document.anchors

.load()                   gespeicherten UserDataStore (User-Cache) auslesen per .style.userData Behavior  
                           Cache muss per Methode .save() zum Behavior gespeichert worden sein

Beispiel:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitpunkt = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen

```



```

        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

**.localeCompare()** Vergleich zweier Strings oder Stringlitterale bezüglich ihrer Sortierfolge laut aktuelle Vorgaben zur Sortierungsfolge auf dem PC des Users  
nur IE ab 5.5  
siehe Script-Objekt String

**.log()** liefert den Logarithmus zur Basis E der zahl (natürlicher Logarithmus)  
siehe Script-Objekt Math

**.match()** Suche in einem String oder String-literal per RegExp-Objekt (siehe dort Methode .exec())  
siehe Script-Objekt String

Beispiel 1:

```

var Kette           = "The rain in Spain falls mainly in the plain";
var RegExpressionAusdruck = /ain/i;
var Feld1           = Kette.match(RegExpressionAusdruck);
var Feld2           = "The rain in Spain falls mainly in the plain".match(RegExpressionAusdruck);

```

Beispiel 2:

```

var zu_durchsuchende_kette = "Otto";
var such_muster            = /(wOtto)/g; // vom Typ RegExp, also regulärer Ausdruck
var ergebnis_feld         = zu_durchsuchende_kette.match(such_muster);

```

**.max()** größte Zahl aller Zahlen liefern  
siehe Script-Objekt Math

**.mergeAttributes()** alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen  
Attribute sind: HTML  
Events  
Styles  
ab IE 5.01 auch ID, NAME  
Achtung: Diese Methode ist mir Vorsicht zu geniessen !!  
DOM wird geändert

Beispiel:

```

<SCRIPT>
function Mischen()
{ ID_SPAN.children[1].mergeAttributes(ID_SPAN.children[0]);}

```



```

</SCRIPT>
<SPAN ID=ID_SPAN>
  <DIV ID="ID_Div_Quelle"
    ATTRIBUTE1="true"
    ATTRIBUTE2="true"
    onclick="alert('click');"
    onmouseover="this.style.color='#0000FF';"
    onmouseout="this.style.color='#000000';"
  >
    Quell<B>Div</B>
  </DIV>
  <DIV ID="ID_Div_Ziel">
    Ziel-Div
  </DIV>
</SPAN>

<INPUT TYPE="button" VALUE=" Mischen" onclick="Mischen()">

```

**.min()**                      kleinste Zahl aller Zahlen liefern  
siehe Script-Objekt Math

**.move()**                      Textbereich-Zeichen-Zeiger bewegen bezüglich aktueller Position im Textbereich  
per textrange Objekt  
nur unter Windows 32-Bit

**.Move()**                      vorhandene Datei verschieben (sonst Fehler erzeugt)

Beispiel:

```

var DateinameMitPfad      = "c:\\test.txt";
var DateiSystem           = new ActiveXObject("Scripting.FileSystemObject");
var Datei                 = DateiSystem.GetFile(DateinameMitPfad);
Datei.Move("c:\\windows\\desktop\\");

```

**.Move()**                      vorhandenen Ordner verschieben (sonst Fehler erzeugt)

Beispiel:

```

var OrdnerName            = "c:\\windows\\desktop\\";
var DateiSystem           = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
Ordner.Move("d:\\recycled\\");

```

**.moveBy()**                      linke obere Fenster-Ecke um Pixeldifferenz auf dem Bildschirm verschieben  
Fenster ganz aus dem BS verschieben ist möglich  
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Bildschirms  
nicht anwendbar bei Dialog-Fenster:  
Dafür dialogHeight, dialogWidth, dialogTop und dialogLeft verwenden.  
siehe Objekt window

Beispiel für Fenster des Browser rausschieben und danach neu anzeigen:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
var BrowserFenster_AktuelleBreite=2000; // sollte größer als max. übliche Auflösung sein
var BrowserFenster_AktuelleHoehe=2000; // sollte größer als max. übliche Auflösung sein

function moveWin()
{
  for (var i = 1; i < BrowserFenster_AktuelleBreite; i++)
  {window.moveBy(1, 1);}

  window.moveBy((-1)* BrowserFenster_AktuelleBreite,(-1) * BrowserFenster_AktuelleHoehe);
}
//-->
</SCRIPT>
</HEAD>
<BODY onload="moveWin();">
</BODY>
</HTML>

```

**.moveEnd()**                      Textbereich-Ende neu setzen (für den Textbereich-Zeiger) bezüglich aktueller Position im Textbereich  
per textrange Objekt  
nur unter Windows 32-Bit

**.MoveFile(en)**                      Datei(en) verschieben



Wenn Move abbricht, so bleiben bisher erfolgte Verschiebungen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.MoveFile("c:\\eigene dateien\\*.doc", "c:\\recycled\\")
```

`.moveFirst()`                      aktuelle Position innerhalb der Collection auf das erste Element der Collection setzen (auch wenn Collection leer ist)  
zur Positionierung innerhalb der Collection wird ein interner Index verwendet  
Element an aktueller Position ermitteln per `.item()`  
siehe Enumerator JScript-Objekt

Beispiel Laufwerke auf dem PC des Users ermitteln:

```
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

// erstes Laufwerk im Dateisystem einstellen (erstes Element im Enumerator-Objekt)
DateiSystem_Laufwerke.moveFirst();
do
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]"; }
    }

    Kette += LaufwerkName + "\n";

    // nächstes Laufwerk positionieren (nächstes Element im Enumerator-Objekt)
    DateiSystem_Laufwerke.moveNext();
}
while (!DateiSystem_Laufwerke.atEnd());

alert(Kette);
```

`.MoveFolder()`                      Ordner verschieben  
Wenn Move abbricht, so bleiben bisher erfolgte Verschiebungen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.MoveFolder("c:\\eigene dateien\\", "c:\\recycled\\")
```

`.moveNext()`                      aktuelle Position innerhalb der Collection auf das nächste Element der Collection setzen (auch wenn Collection leer ist)  
zur Positionierung innerhalb der Collection wird ein interner Index verwendet  
Element an aktueller Position ermitteln per `.item()`  
siehe Enumerator JScript-Objekt



Beispiel Laufwerke auf dem PC des Users ermitteln:

```
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

// erstes Laufwerk im Dateisystem einstellen (erstes Element im Enumerator-Objekt)
DateiSystem_Laufwerke.moveFirst();
do
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]"; }
    }

    Kette += LaufwerkName + "\n";

    // nächstes Laufwerk positionieren (nächstes Element im Enumerator-Objekt)
    DateiSystem_Laufwerke.moveNext();
}
while (!DateiSystem_Laufwerke.atEnd());

alert(Kette);
```

.moveRow()                    2 Zeilen in der Tabelle austauschen  
                               siehe Objekt table  
                               siehe Objekt table.tBody  
                               siehe Objekt table.tHead  
                               siehe Objekt table.tFoot

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELSPACING=10>
  <TR>
    <TD>Body Zeile 1 Zelle 1</TD>
    <TD>Body Zeile 1 Zelle 2</TD>
  </TR>
  <TR>
    <TD>Body Zeile 2 Zelle 1</TD>
    <TD>Body Zeile 2 Zelle 2</TD>
  </TR>
  <TR>
    <TD>Body Zeile 3 Zelle 1</TD>
    <TD>Body Zeile 3 Zelle 2</TD>
  </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.moveRow(0,1);">verschieben Zeile 0 auf Zeile 1</BUTTON>
```



<code>.moveStart()</code>	Textbereich-Anfang neu setzen (für den Textbereich-Zeiger) bezüglich aktueller Position im Textbereich per <code>textrange</code> Objekt nur unter Windows 32-Bit
<code>.moveTo()</code>	linke obere Fenster-Ecke laut Pixel-Position auf dem Bildschirm positionieren Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Bildschirms nicht anwendbar bei Dialog-Fenster: Dafür <code>dialogHeight</code> , <code>dialogWidth</code> , <code>dialogTop</code> und <code>dialogLeft</code> verwenden. siehe Objekt <code>window</code>
<code>.moveToBookmark()</code>	Textbereich-Zeichen-Zeiger auf eine Textmarke (Bookmark) im Textbereich positionieren Textmarke wurde gesetzt per Methode <code>.getBookmark()</code> per <code>textrange</code> Objekt nur unter Windows 32-Bit
<code>.moveToElementText()</code>	Textbereich in ein Element/Objekt bewegen, das Text enthalten darf per <code>textrange</code> Objekt nur unter Windows 32-Bit
<code>.moveToPoint()</code>	Inhalt des Textbereiches um eine Pixelspanne verschieben (Offset) relativ zur linken oberen Fensterecke nach Verschiebung kann Textbereich leer sein per <code>textrange</code> Objekt nur unter Windows 32-Bit

Beispiel:

```
<SCRIPT LANGUAGE="JScript" FOR=document EVENT=onclick >
    var TextBereich = document.body.createTextRange();
    TextBereich.moveToPoint(window.event.x, window.event.y);
    TextBereich.expand("word");
    TextBereich.select();
</SCRIPT>
```

<code>.namedItem()</code>	Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern
---------------------------	--

Beispiel:

```
<DIV ID="ID_Div">Dieser Text wird geändert</DIV>
<BUTTON onclick="document.all.namedItem('ID_Div').innerText='Neuer Text!';">
    Klick mich
</BUTTON>
```

<code>.namedRecordset()</code>	Zeiger desjenigen Datenquellen-Record-Objektes mit Namen, das den Standard-Record-Set enthält Hinweis: Namen zeigt die Mitgliedschaft in einem Datasource-Objekt (DSO) an Eigenschaft <code>.recordset</code> liefert den Zeiger für den Standard-Record-Set in einem Datasource-Objekt (DSO)
--------------------------------	---

Beispiel 1:

```
<SCRIPT>
    // Fired when all the data is available
    function TabelleFuellen()
    {
        var EventObjekt = window.event;

        // prüfen ob nicht Standard-Recordset anliegt (hat keinen Namen)
        if (EventObjekt.qualifier != "")
        {
            // ersten Recordset hinter dem Standard-Recordset referenzieren
            var RecordSetObjektMitName=
                EventObjekt.srcElement.namedRecordset(EventObjekt.qualifier);

            // auf ersten Satz im Recordset positionieren
            RecordSetObjektMitName.MoveFirst();

            // alle Sätze abklappern im Recordset
            for (int i = 0; i < RecordSetObjektMitName.RecordCount; i++)
            {
                // erstes Feld des Satzes auslesen
                var Wert = RecordSetObjektMitName.Fields(0).value;

                // nächsten Satz im Recordset
                RecordSetObjektMitName.MoveNext();
            }
        }
    }
</SCRIPT>
```



```

<OBJECT CLASSID="clsid:00000000-0000-0000-0000-000000000000"
  ID="ID_Objekt"
  ondatasetcomplete="TabelleFuellen()"
>

<!--Zelle A1:A7 füllen per Recordset -->
<TABLE DATASRC="#ID_Objekt.A1:A7">
  <TR>
    <TD>
      <SPAN DATAFLD="A">
        </SPAN>
      </TD>
    </TR>
  </TABLE>

```

Beispiel 2 für XML:

```

<XML ID="ID_XML">
  <customers>
    <customer>
      < order ID="1">
        < item >
          <name>Butter</name>
          <quantity>12</quantity>
        </item>
        < item >
          <name>Kaese</name>
          <quantity>12</quantity>
        </item>
      </ order >
      < order ID="2">
        < item >
          <name>Wurst</name>
          <quantity>100</quantity>
        </item>
      </ order >
    </customer>
  </customers>
  <customer>
    < order ID="3">
      < item >
        <name>Kaese</name>
        <quantity>20</quantity>
      </item>
      < item >
        <name>Quark</name>
        <quantity>20</quantity>
      </item>
    </ order >
  </customer>
</XML>

<SCRIPT>
  // Pfad zu den Mengendaten ist:
  //      order item name quantity

  function Summieren(NameAlsKette)
  {
    var Summe = 0;

    // Standard-Recordset adressieren get the default data member
    var StandardRecordSet = ID_XML.recordset; // entspricht ID_XML.namedRecordset("");

    // ersten Satz positionieren
    StandardRecordSet.MoveFirst();

    for (var SatzZahler = 0; SatzZahler < StandardRecordSet.RecordCount; SatzZahler++)
    {
      var OrderRecordSet = ID_XML.namedRecordset("", "order");
      OrderRecordSet.MoveFirst(); // ersten Satz
    }
  }

```





```

        for ( var OrderZahler = 0;
              OrderZahler < OrderRecordSet.RecordCount;
              OrderZahler ++
            )
        {
            var ItemRecordSet = ID_XML.namedRecordset("", "order.item");
            ItemRecordSet.MoveFirst(); // ersten Satz

            for ( var ItemZahler = 0;
                  ItemZahler < ItemRecordSet.RecordCount;
                  ItemZahler ++
                )
            {
                // prüfen ob Funktionsargument NameAlsKette gefunden
                if ( ItemRecordSet.Fields("name").value == NameAlsKette )
                {
                    // ja also kumulieren aus dem Feld quantity
                    Summe +=
                        parseInt(ItemRecordSet.Fields("quantity").value);
                }

                // nächster Satz
                ItemRecordSet.MoveNext();
            }

            // nächster Satz
            OrderRecordSet.MoveNext();
        }

        // nächster Satz
        StandardRecordSet.MoveNext();
    }

    return Summe;
}

alert( Summieren ("Wurst");)
alert(Summieren ("Kaese");)
</SCRIPT>

```

.navigate() HTTP-Verzeichnis im aktuellen Fenster anzeigen

.navigate() lädt neues HTML-Dokument laut Url in das Fenster  
entspricht location.href = "...."  
siehe Objekt window

Beispiele: navigate("index.html");  
navigate ("file:///c:/index.html");

.NavigateAndFind() Webseite laden, dann Text dort suchen und wenn gefunden, so diesen in der Webseite markieren  
(analog zu CTRL-F in der Webseite für Suchen und Finden)  
auch Suche in Frames bzw. Unterseiten der Webseite  
siehe Objekt window.external

Beispiel

```

<HEAD>
<SCRIPT>
    function Suchen()
    {
        window.external.NavigateAndFind( "http://www.test.de/file.htm",
                                          ID_Seletc.options[ID_Seletc.selectedIndex].text,
                                          ""
        );
    }
</SCRIPT>
</HEAD>
<BODY>
    bitte selektieren
    <SELECT ID="ID_Seletc" onchange="Suchen()">
        <OPTION>Eintrag1
        <OPTION>Eintrag2
    </SELECT>
</BODY>

```



`.navigateFrame()` HTTP-Verzeichnis im Fenster nach Wahl anzeigen

`.navigateHomePage()` Homepage anspringen im Browser

`.nextElement()` nächstes Element aus aktiver t:SEQ animieren auf der Timeline  
wenn kein nächstes Element, so kein Fehler erzeugt  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:SEQ ID="ID_Seq"
        BEGIN="indefinite;"
    >

        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            DUR="10"
            style="position:relative;top:15px;left:25px;width:100px;height:100px;"
        >
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            DUR="10"
            style="position:relative;top:30px;left:50px;width:100px;height:100px;"
        >
        </DIV>
        <DIV ID="ID_Div3"
            CLASS="time_line_klasse"
            DUR="10"
            style="position:relative;top:35px;left:75px;width:100px;height:100px;"
        >
        </DIV>
        <DIV ID="ID_Div4"
            CLASS="time_line_klasse"
            DUR="10"
            style="position:relative;top:50px;left:100px;width:100px;height:100px;"
        >
        </DIV>
    </t:SEQ>
    <BR>
    <BUTTON ID="ID_Button1" onclick="ID_Seq.beginElement();"> start</BUTTON>
    <BUTTON ID="ID_Button2" onclick="ID_Seq.nextElement();">next</BUTTON>
    <BUTTON ID="ID_Button3" onclick="ID_Seq.prevElement();">previous</BUTTON>
    <BUTTON ID="ID_Button4" onclick="ID_Seq.endElement();">stop</BUTTON>
</BODY>
</HTML>
```

`.nextPage()` nächste Seite des Datasets in Tabelle anzeigen  
Eigenschaft `.dataPageSize` muss belegt worden sein  
siehe Objekt `table`

Beispiel:

Datensätze liegen in der Datei `adress.txt`  
Datei `adress.txt` liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument  
hat folgenden Inhalt:

1. Satz	<b>vorname;name;telefon</b>
ab 2. Satz	Guericke;"Otto, von";0815
	Willmann;Theo;1234
	Xantippe;Isa;5678
	Arktin;Richard;1055

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
    var SortierRichtung = true;

    function Sortieren(FeldBezeichner)
    {
```



```

        // Sortierrichtung festlegen
        var Kette = "-"; // Annahme
        if (SortierRichtung) {Kette = "+";}

        // nächste Sortierung umgekehrt
        SortierRichtung = !SortierRichtung;

        // sortieren
        ID_Datenbank.Sort= Kette + FeldBezeichner;

        // und das Ergebnis anzeigen
        ID_Datenbank.Reset();
    }

    function vorwaerts(AnzahlSaetze)
    {
        // nächste Seite anzeigen
        document.all.ID_Tabelle.nextPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaetze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition !=
                ID_Datenbank.recordset.RecordCount)
            {ID_Datenbank.recordset.MoveNext();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == ID_Datenbank.recordset.RecordCount)
        {alert("Letzter Datensatz erreicht!");}
    }

    function rueckwaerts(AnzahlSaetze)
    {
        // vorhergehende Seite anzeigen
        document.all.ID_Tabelle.previousPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaetze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition > 1)
            {ID_Datenbank.recordset.MovePrevious();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition ==1)
        {alert("Erster Datensatz erreicht!");}
    }
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
    CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
    <PARAM NAME ="DataURL" VALUE="adress.txt">
    <PARAM NAME ="UseHeader" VALUE="True">
    <PARAM NAME ="FieldDelim" VALUE=";">
</OBJECT>

<TABLE ID="ID_Tabelle"
    DATASRC=#ID_Datenbank
    DATAPAGESIZE=1
>
    <THEAD>
        <TR>
            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
            <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
        </TR>
    </THEAD>
    <TBODY>
        <TR>
            <TD><DIV DATAFLD="vorname"></DIV></TD>
            <TD><DIV DATAFLD="name"></DIV></TD>
            <TD><DIV DATAFLD="telefon"></DIV></TD>

```



```

        </TR>
    </TBODY>
</TABLE>
<BR>
<INPUT          TYPE="button"
                VALUE="Zurueck"
                onclick=rueckwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //-->

<INPUT          TYPE="button"
                VALUE="Vorwaerts"
                onclick=vorwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //-->
</BODY>
</HTML>

```

`.nextTrack()`                    nächstes `playItem` Objekt aktivieren laut Behavior-Collection `.style.time2.playList`  
                                       also Abspielen des Tracks starten  
 bzw. nächste Wiederholung starten wenn `.repeatCount > 0` und noch nicht  
       alle Wiederholungen abgearbeitet wurden  
 wenn letzter Track laut der Playliste aktiv ist und dann der nächste Track aktiviert werden soll,  
       so wird nicht der erste Track abgespielt, sondern der aktive Track in der  
       Wiedergabe gestoppt und danach keine weiteren Aktionen mehr  
 Track entspricht `playItem` Objekt  
 Trackliste liegt in der Behavior-Collection `.style.time2.playList` als Playliste:  
       Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline  
       aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
 Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei  
 aktiver Track in der Liste: wird abgespielt  
 Track in der Liste auf aktiv setzen: Abspielen des Tracks sofort starten  
 siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel 1:

```
object.playList.nextTrack()
```

Beispiel 2:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
    {
        ID_Span1.innerText = "Titel: " + ID_Media.playList.activeTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playList.activeTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playList.activeTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playList.activeTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playList.activeTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playList.activeTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playList.activeTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playList.activeTrack.BannerMoreInfo;
    }

    function AnzeigeLoeschen()

```



```

    {
        ID_Span1.innerText = "Titel: ";
        ID_Span2.innerText = "Autor: ";
        ID_Span3.innerText = "Abstract: ";
        ID_Span4.innerText = "Copyright: ";
        ID_Span5.innerText = "Filename: ";
        ID_Span6.innerText = "Banner: ";
        ID_Span7.innerText = "Banner Abstract: ";
        ID_Span8.innerText = "Banner MoreInfo: ";
    }
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

    <t:MEDIA          ID="ID_Media"
                      SRC="test.asx"
                      BEGIN="indefinite"
                      TIMEACTION="visibility"
                      onend="ButtonUpdate();"
                      ontrackchange="AnzeigeUpdate();"
                      onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN   ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN   ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN   ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN   ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN   ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN   ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN   ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN   ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON          ID="ID_Button1"
                    onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON          ID="ID_Button2"
                    onclick="ID_Media.playList.nextTrack();"
    >
        naechster Track
    </BUTTON>
    <BUTTON          ID="ID_Button3"
                    onclick="ID_Media.playList.prevTrack();"
    >
        vorhergehender Track
    </BUTTON>
    <BUTTON          ID="ID_Button4"
                    onclick="ID_Media.endElement(); AnzeigeLoeschen();"
    >
        Stop
    </BUTTON>
</BODY>
</HTML>

```

.normalize()                      Normalisierung des DOM zur Erreichung einer konsistenten Struktur  
 Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen

Beispiel:

```

<HTML>
<BODY onload="ID_Div.normalize();">
    <DIV ID="ID_Div" TITLE="Tooltip-Text">Es ist ein Tooltip-Text vorhanden</DIV>
</BODY>
</HTML>

```

Number()                      konvertiert eine Instanz zu einem numerischen Wert (Number-Objekt-Wert)



Beispiel:

```
var DatumJetzt = new Date();
alert (Number(DatumJetzt));
```

.open()

Dokument instanzieren und mit/ohne Fenster öffnen und Referenz liefern auf Dokument  
Ausgabe-Datenstream öffnen und Anzeige des Dokumentes  
Instanzieren ohne .open(): siehe .write() und .writeln()  
Hinweis: neues Fenster erzeugen als unterstes der aktuellen Fensterhierarchie  
und dann Fenster öffnen (anzeigen, rendern)

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    function ErzeugeZurLaufZeit()
    {
        // Dokumentinstanz erzeugen
        var ID_Dokument = document.open("text/html", "replace");

        // Dokumentinhalt festlegen und anzeigen
        ID_Dokument.write("<HTML><BODY>Hallo</BODY></HTML>");

        // Dokumentinstanz schliessen
        ID_Dokument.close();
    }
</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE="button" onclick=" ErzeugeZurLaufZeit();">
</BODY>
</HTML>
```

Beispiel:

```
function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
    FensterZeiger.close();
}

....

// Fenster öffnen
var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");
var FensterDokumentZeiger = FensterZeiger.document;

var Kette= '<HTML>'
+ '<HEAD></HEAD>'
+ '<BODY >'
+ '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
+ '<BR>'
+ '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
+ ' onclick="opener.OnClickHandler();"'
+ '>'
+ '</BODY>'
+ '</HTML>';

FensterDokumentZeiger.open("text/html");
FensterDokumentZeiger.write(Kette);
FensterDokumentZeiger.close();
```

.open()

neues Fenster als unterstes der aktuellen Fensterhierarchie erzeugen  
und dann oberstes sichtbare Fenster öffnen (anzeigen, rendern und als aktuell setzen)  
nicht möglich bei Dialog-Fenster oder Popup-Fenster  
Hinweis: Zeiger des Fensters, das .open() ausführt, liegt in der Eigenschaft .opener  
des neu erzeugten Fensters  
Daten per '?' + escape() **nicht** übergebbar

siehe Objekt window

Beispiel:

```
window.open("Sample.htm",null, "height=200,width=400,status=yes,toolbar=no,menubar=no,location=no");
```

Beispiel:

```
function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
```



```

        FensterZeiger.close();
    }

    ....

    // Fenster öffnen
    var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");
    var FensterDokumentZeiger = FensterZeiger.document;

    var Kette= '<HTML>'
        + '<HEAD></HEAD>'
        + '<BODY>'
        + '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
        + '<BR>'
        + '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
        + ' onclick="opener.OnClickHandler();"'
        + '>'
        + '</BODY>'
        + '</HTML>';

    FensterDokumentZeiger.open("text/html");
    FensterDokumentZeiger.write(Kette);
    FensterDokumentZeiger.close();

```

**.OpenAsTextStream()** vorhandene Datei als Text öffnen zum Lesen und Schreiben oder Append, wenn Dateiattribute es zulassen (sonst Fehler erzeugt)  
Schreiben und Schliessen der Datei per Objekt `FileSystemObject.TextStream`

Beispiel:

```

var DateinameMitPfad      = "c:\\test.txt";
var Dateisystem           = new ActiveXObject("Scripting.FileSystemObject");
var Datei                = Dateisystem.GetFile(DateinameMitPfad);
var DateiOffen            = Datei.OpenAsTextStream(2,0);
DateiOffen.Write( "neuer Text" );
DateiOffen.Close();
DateiOffen                = Datei.OpenAsTextStream(1,2);
var Kette                 = DateiOffen.ReadLine( );
DateiOffen.Close();
alert(Kette);

```

**.OpenTextFile()** Datei als Text öffnen zum Lesen, Schreiben oder Append (Schreiben durch Anhängen)  
Schreiben und Schliessen der Datei per Objekt `FileSystemObject.TextStream`

Beispiel:

```

var Dateisystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = Dateisystem.OpenTextFile("c:\\test.txt", 2,true,0)
DateiOffen.WriteLine("Test");
DateiOffen.Close();

```

**.parentElement()** Zeiger auf das Elternelement (Container) des Textbereiches liefern  
Hinweis bei Elementverschachtelung:  
es wird das direkt um den Textbereich liegende Element referenziert  
per `textrange` Objekt  
nur unter Windows 32-Bit

Beispiel:

```

<SCRIPT LANGUAGE="JScript">
    var SelectionObjekt = document.selection; // Eltern
    var TextBereichDerSelection = SelectionObjekt.createRange();
    var ZeigerAufSelection = TextBereichDerSelection.parentElement();
    alert(ZeigerAufSelection.tagName);
</SCRIPT>

```

**.parentTimeToActiveTime()** Wert der Timeline der Eltern zum korrespondierenden Wert der aktives Timeline des Elementes konvertieren  
per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:EXCL REPEATCOUNT="3"

```



```

>
    <DIV ID="ID_Div"
        CLASS="time_line_klasse"
        BEGIN="1s"
        DUR="5s"
    >
    </DIV>
</t:EXCL>
<BR>
<BUTTON ID="ID_Button"
    onclick=" alert( 'Punkt auf der aktiven Timeline: ' + ID_Div.parentTimeToActiveTime(3) ); "
>
    parentTimeToActiveTime(3);
</BUTTON>
</BODY>
</HTML>

```

.parentTimeToDocumentTime() Wert der Timeline der Eltern des Elementes in den korrespondierenden Wert der Timeline des Dokumentes konvertieren  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:PAR ID="ID_Par"
        CLASS="time_line_klasse"
        REPEATCOUNT="3"
    >
        <DIV ID="ID_Div"
            CLASS="time_line_klasse"
            BEGIN="2s"
            DUR="2s"
        >
        </DIV>
    </t:PAR>
    <SPAN ID="ID_Span">
        parentTimeToDocumentTime:
    </SPAN>
    <BR>
    <BUTTON ID="ID_Button"
        onclick="    ID_Span.innerText='parentTimeToDocumentTime: '
                    + ID_Div.parentTimeToDocumentTime(3);"
    >
        parentTimeToDocumentTime(3)
    </BUTTON>
</BODY>
</HTML>

```

.parse() Datum als String parsen und Anzahl der Millisekunden relativ zum 01.01.1970 0 Uhr Weltzeit liefern  
Datum als String in den Basis-Datentyp date konvertieren  
Script-Objekt Date

Beispiele:

```

"Jan 5, 1996 08:47:00"
"November 1, 1997 10:15:00 "
"November 1, 1997 10:15 AM"
"7/20/96 08:47:00"
"7-20-96 08:47:00"
"Tuesday November 9 1990"
"7-20-96 08:" es muss Doppelpunkt kodiert sein, damit 08 als Stundenangabe interpretiert wird
"7-20-96 08:47 GMT"

```

parseFloat() String oder Literal parsen und nach Floating-point umwandeln  
können enthalten  
Vorzeichen + und -  
Ziffern 0 bis 9  
Dezimalkomma als Punkt  
e oder E  
Blanks (werden ignoriert)





falls andere Zeichen enthalten sind, gilt:

String bzw. Literal bis vor die erste fehlerhaften  
Stelle geparkt und dann konvertiert

Erfolg der Konvertierung ist per Methode isNaN() zu prüfen

Beispiele:

gültiges Literal  
parseFloat("3.14")  
parseFloat("3.14e-2")  
parseFloat("0.0314E+2")

gültiger String  
var x = "3.14"  
parseFloat(x)

ungültiges Literal  
parseFloat("FF2")

alert(parseFloat("17.50 DM"));

Beispiel für Ziffern-Zeichenkettenwert nach numerisch und dabei Dezimalkomma zu Dezimalpunkt umwandeln:

Es wird angenommen, dass genau ein Komma vorkommt.

```
function punkt_zu_komma(zeichenkette)
{
    var pos_komma = zeichenkette.indexOf(",");
    var funktionswert;

    if(pos_komma == -1)
    {
        if(zeichenkette.indexOf(".") == -1)
        { funktionswert = parseInt(zeichenkette); }
        else
        { funktionswert = parseFloat(zeichenkette); }
    }
    else
    {
        funktionswert = parseFloat(
            zeichenkette.substring(0, pos_komma)
            + "."
            + zeichenkette.substring(pos_komma + 1, zeichenkette.length)
        );
    }

    return funktionswert;
}
```

parseInt()      String oder Literal      parsen und nach Integer umwandeln  
können enthalten  
Vorzeichen + und -  
Ziffern 0 bis 9, aber keine Vornull(en)  
Blanks (werden ignoriert)  
eventuelle Buchstaben A bis F

falls andere Zeichen enthalten sind, gilt:

String bzw. Literal bis vor die erste fehlerhaften Stelle geparkt und dann  
konvertiert

Erfolg der Konvertierung ist per Methode isNaN() zu prüfen

Beispiel:

gültige Literale:  
parseInt("F", 16)  
parseInt("17", 8)  
parseInt("15", 10)  
parseInt(15.99, 10)  
parseInt("FXX123", 16)  
parseInt("1111", 2)  
parseInt("15\*3", 10)  
parseInt("17")  
parseInt("0x7", 16)  
parseInt("0x7")  
parseInt("0")  
parseInt("0x11", 16)  
parseInt("0x11", 0)  
parseInt("0x11")



ungültige Literale:

```
parseInt("F")
parseInt("Hello", 8)
parseInt("0x7", 10)
parseInt("FFF", 10)
parseInt('002')
```

.pasteHTML()

Textbereich-Inhalt durch Text ersetzen oder leeren Textbereich füllen  
Text kann auch HTML enthalten  
Tabelle nur mit kompletten HTML-Code einfügbar, also z.B. keine einzelne Zelle  
Achtung: Wenn HTML-Code im Text enthalten ist, muss das Elternobjekt  
auch diesen ansich unterstützen  
Bsp.: textArea erlaubt kein HTML-Code  
HTML-Code wird geparkt  
per textrange Objekt  
nur unter Windows 32-Bit

Beispiel:

```
<SCRIPT LANGUAGE="JScript">
// Selektion erzeugen
var SelectionObjekt = document.selection;

// prüfen ob erzeugt wurde
if (SelectionObjekt!=null)
{
    // wurde erzeugt
    var TextBereichDerSelektion rng = SelectionObjekt.createRange();

    // prüfen ob erzeugt wurde
    if (TextBereichDerSelektion!=null)
    {
        // wurde erzeugt, also Selektion füllen
        TextBereichDerSelektion.pasteHTML("<P><B>Text der Selektion</B></P>");
    }
}
</SCRIPT>
```

.pause()

deprecated  
auf Timeline pausieren

.pauseElement()

aktives Element auf Timeline pausieren lassen  
ersetzt Methode pause(), die deprecated ist und nicht mehr verwendet werden darf !  
per Eigenschaft .isActive das Element auf Aktivsein prüfen  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
.TextAufHighLight{ color:#CCCCC; font-weight:bold;}
</STYLE>
<SCRIPT>
function PauseAufheben()
{
    ID_Table.resumeElement();
    ID_Button1.disabled = false;
    ID_Button2.disabled = true;
}

function Pausieren()
{
    ID_Table.pauseElement();
    ID_Button1.disabled = true;
    ID_Button2.disabled = false;
}
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<TABLE ID="ID_Table"
BORDER="1"
CLASS="time_line_klasse"
DUR="8s"
REPEATCOUNT="indefinite"
```



```

        TIMECONTAINER="SEQ"
        TIMEACTION="none"
    >
    <TBODY>
        <TR>
            <TH WIDTH="120">Eins</TH>
            <TH WIDTH="50">Zwei</TH>
            <TH WIDTH="40">Drei</TH>
        </TR>
        <TR
            ID="ID_TR1"
            CLASS="time_line_klasse"
            TIMEACTION="class: TextAufHighLight "
            DUR="2s"
        >
            <TD>EinsA</TD>
            <TD>ZweiA</TD>
            <TD>DreiA</TD>
        </TR>
        <TR
            ID="ID_TR1"
            CLASS="time_line_klasse"
            TIMEACTION="class: TextAufHighLight "
            DUR="2s"
        >
            <TD>EinsB</TD>
            <TD>ZweiB</TD>
            <TD>DreiB</TD>
        </TR>
    </TBODY>
</TABLE>
<BR>
<BUTTON
    ID="ID_Button1"
    onclick="Pausieren();"
>
    pausieren
</BUTTON>
<BUTTON
    ID="ID_Button2"
    DISABLED="true"
    onclick="PauseAufheben();"
>
    Pause aufheben
</BUTTON>
</BODY>
</HTML>

```

**.playNext()** Wiedergabe des nächsten Eintrages in der Playliste (Objekt PlaylistInfo) starten  
verändert Eigenschaft **.playState**  
Es kann zu jedem Zeitpunkt nur genau 1 Media-Datei wiedergegeben werden, es sei denn, man öffnet mehrere Browser-Instanzen.  
**aktuelle** Die Wiedergabe ist nicht möglich, wenn der User in der Media Bar per Links navigiert, da damit die

Url in der Media Bar geändert wird gegenüber der Url der gerade wiedergegebenen Media-Datei.  
Eine aktive Wiedergabe wird durch HTML-Navigation sofort gestoppt.  
Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende

Methoden nutzbar:  
**.getItemInfo()**  
**.getAttributeName()**  
 Eigenschaften nutzbar:  
**.attributeCount**

Es ist also vorher die Eigenschaft **.playState** auf Werte > 10 zu prüfen.  
siehe Methoden **.playURL()** **.stop()** und Eigenschaft **.playState**  
siehe Behavior **.style.mediaBar**

Beispiel:

```

<DIV
    ID="ID_Div"
    STYLE="behavior:url(#default#mediaBar)"
    onopenstatechange="alert(ID_Div.openState);"
>
</DIV>
<INPUT
    TYPE=button
    ID="ID_Input"
    VALUE='abspielen des nächsten Eintrages in der Playliste'
    onclick= "ID_Div.playNext();
            ID_Div.disabledUI = true;
            "
>

```



**.playURL()** Laden einer Media-Datei in die Media Bar und wenn Laden erfolgreich, so Wiedergabe der Media-Daten  
 Achtung: Wird die Media-Datei nicht gefunden oder ist der Mime-Typ der Media-Datei unbekannt, dann wird eine Standard-Seite in das Medien-Fenster geladen.  
 verändert Eigenschaft **.playState**  
 Es kann zu jedem Zeitpunkt nur genau 1 Media-Datei wiedergegeben werden, es sei denn, man öffnet mehrere Browser-Instanzen.  
 Die Wiedergabe ist nicht möglich, wenn der User in der Media Bar per Links navigiert, da damit die aktuelle  
 Url in der Media Bar geändert wird gegenüber der Url der gerade wiedergegebenen Media-Datei.  
 Eine aktive Wiedergabe wird durch HTML-Navigation sofort gestoppt.  
 Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende Methoden nutzbar:  
     **.getItemInfo()**  
     **.getAttributeName()**  
 Eigenschaften nutzbar:  
     **.attributeCount**  
 Es ist also vorher die Eigenschaft **.playState** auf Werte > 10 zu prüfen.  
 siehe Methoden **.playNext()** **.stop()** und Eigenschaft **.playState**  
 siehe Behavior **.style.mediaBar**

Beispiel:

```
<DIV ID="ID_Div"
STYLE="behavior:url(#default#mediaBar)"
onopenstatechange="alert(ID_Div.openState);">
>
</DIV>
<INPUT TYPE=button
ID="ID_Input"
VALUE='abspielen von test.asx'
onclick= "ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
ID_Div.disabledUI = true;
">
```

**.pop()** letztes Feldelement liefern und danach aus dem Feld entfernen  
 Feld hat die Objektklasse JScript-Objekt Array bzw. Script-Objekt Array  
 ab IE 5.5 und NS 6.x  
 siehe auch **.shift()**

Beispiel:

```
var Feld = new Array(0,1,2,3,4);
alert (Feld.pop()); // "4"
alert(Feld.join()); // "0,1,2,3"
```

**.pow()** liefert die Potenz von Basis hoch Exponent  
 siehe Script-Objekt **Math**

**.prevElement()** vorhergehendes Element aus aktiver t:SEQ animieren auf der Timeline  
 wenn kein vorhergehendes Element, so wird Timeline von t:SEQ auf 0 gesetzt  
 siehe Objekt **currTimeState** und Behavior **.style.time2**

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
<t:SEQ ID="ID_Seq"
BEGIN="indefinite;"
>

<DIV ID="ID_Div1"
CLASS="time_line_klasse"
DUR="10"
style="position:relative;top:15px;left:25px;width:100px;height:100px;"
>
</DIV>
<DIV ID="ID_Div2"
CLASS="time_line_klasse"
DUR="10"
style="position:relative;top:30px;left:50px;width:100px;height:100px;"
>
```



```

</DIV>
<DIV ID="ID_Div3"
      CLASS="time_line_klasse"
      DUR="10"
      style="position:relative;top:35px;left:75px;width:100px;height:100px;"
>
</DIV>
<DIV ID="ID_Div4"
      CLASS="time_line_klasse"
      DUR="10"
      style="position:relative;top:50px;left:100px;width:100px;height:100px;"
>
</DIV>
</t:SEQ>
<BR>
<BUTTON ID="ID_Button1" onclick="ID_Seq.beginElement();"> start</BUTTON>
<BUTTON ID="ID_Button2" onclick="ID_Seq.nextElement();">next </BUTTON>
<BUTTON ID="ID_Button3" onclick="ID_Seq.prevElement();">previous</BUTTON>
<BUTTON ID="ID_Button4" onclick="ID_Seq.endElement();">stop</BUTTON>
</BODY>
</HTML>

```

.previousPage()                    vorhergehende Seite des Datasets in Tabelle anzeigen  
 Eigenschaft .dataPageSize muss belegt worden sein  
 siehe Objekt table

Beispiel:

Datensätze liegen in der Datei adress.txt  
 Datei adress.txt                liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument  
 hat folgenden Inhalt:

1. Satz	<b>vorname;name;telefon</b>
ab 2. Satz	Guericke;"Otto, von";0815
	Willmann;Theo;1234
	Xantippe;Isa;5678
	Arktin;Richard;1055

```

<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
    var SortierRichtung = true;

    function Sortieren(FeldBezeichner)
    {
        // Sortierrichtung festlegen
        var Kette = "-"; // Annahme
        if (SortierRichtung) {Kette = "+";}

        // nächste Sortierung umgekehrt
        SortierRichtung = !SortierRichtung;

        // sortieren
        ID_Datenbank.Sort= Kette + FeldBezeichner;

        // und das Ergebnis anzeigen
        ID_Datenbank.Reset();
    }

    function vorwaerts(AnzahlSaetze)
    {
        // nächste Seite anzeigen
        document.all.ID_Tabelle.nextPage();

        // und Satzzeiger korrigieren
        for (var i = 0; i < AnzahlSaetze; i++)
        {
            if (ID_Datenbank.recordset.AbsolutePosition !=
                ID_Datenbank.recordset.RecordCount)
            {ID_Datenbank.recordset.MoveNext();}
        }

        if (ID_Datenbank.recordset.AbsolutePosition == ID_Datenbank.recordset.RecordCount)
        {alert("Letzter Datensatz erreicht!");}
    }

```



```

function rueckwaerts(AnzahlSaetze)
{
    // vorhergehende Seite anzeigen
    document.all.ID_Tabelle.previousPage();

    // und Satzzeiger korrigieren
    for (var i = 0; i < AnzahlSaetze; i++)
    {
        if (ID_Datenbank.recordset.AbsolutePosition > 1)
        {ID_Datenbank.recordset.MovePrevious();}
    }

    if (ID_Datenbank.recordset.AbsolutePosition == 1)
    {alert("Erster Datensatz erreicht!");}
}

// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
    CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
    <PARAM NAME="DataURL" VALUE="adress.txt">
    <PARAM NAME="UseHeader" VALUE="True">
    <PARAM NAME="FieldDelim" VALUE=";">
</OBJECT>

<TABLE ID="ID_Tabelle"
    DATASRC=#ID_Datenbank
    DATAPAGESIZE=1
>
    <THEAD>
        <TR>
            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
            <TH><A HREF="JavaScript:Sortieren('name')">Name </A></TH>
            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon </A></TH>
        </TR>
    </THEAD>
    <TBODY>
        <TR>
            <TD><DIV DATAFLD="vorname"></DIV></TD>
            <TD><DIV DATAFLD="name"></DIV></TD>
            <TD><DIV DATAFLD="telefon"></DIV></TD>
        </TR>
    </TBODY>
</TABLE>
<BR>
<INPUT TYPE="button"
    VALUE="Zurueck"
    onclick=rueckwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->

<INPUT TYPE="button"
    VALUE="Vorwaerts"
    onclick=vorwaerts(1)"
>
<!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 -->
</BODY>
</HTML>

```

.prevTrack()      vorhergehendes playItem Objekt aktivieren laut Behavior-Collection .style.time2.playList  
 also Abspielen des Tracks starten  
 aber **nicht** vorhergehende Wiederholung starten wenn .repeatCount > 0 und noch nicht  
 alle Wiederholungen abgearbeitet wurden  
 wenn erster Track laut der Playliste aktiv ist und dann der vorhergehende Track aktiviert werden soll,  
 so wird der erste Track nochmal abgespielt  
 Track entspricht playItem Objekt  
 Trackliste liegt in der Behavior-Collection .style.time2.playList als Playliste:  
 Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline  
 aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
 Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei  
 aktiver Track in der Liste: wird abgespielt  
 Track in der Liste auf aktiv setzen: Abspielen des Tracks sofort starten  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel 1:



```
object.playlist.prevTrack()
```

Beispiel 2:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
function ButtonUpdate()
{
    if (ID_Media.currTimeState.isActive)
    {
        ID_Button1.disabled=true;
        ID_Button2.disabled=false;
        ID_Button3.disabled=false;
        ID_Button4.disabled=false;
    }
    else
    {
        ID_Button1.disabled=false;
        ID_Button2.disabled=true;
        ID_Button3.disabled=true;
        ID_Button4.disabled=true;
    }
}

function AnzeigeUpdate()
{
    ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
    ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
    ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
    ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
    ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
    ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
    ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
    ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
}

function AnzeigeLoeschen()
{
    ID_Span1.innerText = "Titel: ";
    ID_Span2.innerText = "Autor: ";
    ID_Span3.innerText = "Abstract: ";
    ID_Span4.innerText = "Copyright: ";
    ID_Span5.innerText = "Filename: ";
    ID_Span6.innerText = "Banner: ";
    ID_Span7.innerText = "Banner Abstract: ";
    ID_Span8.innerText = "Banner MoreInfo: ";
}
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate()">

<t:MEDIA ID="ID_Media"
SRC="test.asx"
BEGIN="indefinite"
TIMEACTION="visibility"
onend="ButtonUpdate();"
ontrackchange="AnzeigeUpdate();"
onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
>
</t:MEDIA>

<SPAN ID="ID_Span1">Titel:</SPAN>
<BR>
<SPAN ID="ID_Span2">Autor:</SPAN>
<BR>
<SPAN ID="ID_Span3">Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span4">Copyright:</SPAN>
<BR>
```



```

<SPAN ID="ID_Span5">Filename:</SPAN>
<BR>
<SPAN ID="ID_Span6">Banner:</SPAN>
<BR>
<SPAN ID="ID_Span7">Banner Abstract:</SPAN>
<BR>
<SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

<BR>
<BUTTON ID="ID_Button1"
        onclick="ID_Media.beginElement(); ButtonUpdate();"
>
    Start
</BUTTON>
<BUTTON ID="ID_Button2"
        onclick="ID_Media.playList.nextTrack();"
>
    naechster Track
</BUTTON>
<BUTTON ID="ID_Button3"
        onclick="ID_Media.playList.prevTrack();"
>
    vorhergehender Track
</BUTTON>
<BUTTON ID="ID_Button4"
        onclick="ID_Media.endElement(); AnzeigeLoeschen();"
>
    Stop
</BUTTON>
</BODY>
</HTML>

```

**.print()** ruft Dialog-Box-Druckfenster auf zum Druck des Dokumentes im aktuellen Fenster  
entspricht Druckbutton oder Datei-Menü-Drucken  
löst folgende Ereignisse aus: onbeforeprint und onafterprint  
siehe Objekt window

Beispiel für Verwendung der Ereignisse per Handler

```

onbeforeprint    Handler blendet Teile der Seite ein/aus, die gedruckt werden sollen
onafterprint     Handler hebt Veränderungen von onbeforeprint wieder auf

```

Beispiel:

```
<INPUT TYPE="button" VALUE="Drucken"onclick="javascript:self.print()">
```

**.prompt()** Eingabefenster erzeugen:

1. Meldungstext anzeigen,  
Eingabezeile vorbelegen und anzeigen,  
OK- bzw. CANCEL-Button anzeigen
2. auf User.Eingabe in die Zeile und anschliessendem Druck auf OK warten  
siehe Objekt window

**.push()** optionales Anhängen von Werten an das Ende des Feldes (auch wenn das Feld leer ist)  
Anzahl der Feld-Elemente liefern (auch nach optionalem Anhängen)  
Feld hat die Objektklasse JScript-Objekt Array bzw. Script-Objekt Array  
ab IE 5.5 und NS 6.x  
siehe auch .concat()

Beispiel:

```

var Feld1      = new Array(0,1);
var Feld2_Quelle = new Array(2,3);
var Wert_Quelle = 4;
alert(Feld1.push(Wert_Quelle , Feld2_Quelle));    // 4 und nicht 5 !
alert(Feld1.join());    // "0,1,2,3"              // 4 wird nicht angehangen

```

**.queryCommandEnabled()** prüfen ob Kommando ausführbar ist

**.queryCommandIndeterm()** prüfen ob Kommando-Status bestimmbar ist oder nicht

**.queryCommandState()** Status des aktuellen Kommando ermitteln: ob ausgeführt wurde oder nicht

**.queryCommandSupported()** prüfen ob Kommando im aktuellen Bereich unterstützt wird

**.queryCommandValue()** Wert eines Kommandos liefern

**.random()** Zufallszahl liefern





0<= zufallszahl <1  
siehe Script-Objekt Math

**.Read()** in der offenen Datei die nächsten Zeichen lesen (ab Position laut Satzzeiger)  
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.Read(6));
DateiOffen.Close();
```

**.ReadAll()** offene Datei komplett auslesen (auf einen Schlag)  
nur direkt nach Dateiöffnen möglich

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.ReadAll());
DateiOffen.Close();
```

**.ReadLine()** in der offene Datei die nächste Zeile lesen (ab Position laut Satzzeiger)  
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.ReadLine());
DateiOffen.Close();
```

**.recalc()** dynamischen Eigenschaften des Dokumentes neu berechnen  
Hinweis: andere Objekte, die Eigenschaften des Dokumentes nutzen, werden auch neu berechnet,  
wenn Eigenschaften nicht in einem Berechnungsausdruck vorliegen

Beispiel 1:

```
<HTML>
<HEAD>
<STYLE>
    BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
    var timerID = null;
    var seconds = 0;

    function init()
    {
        ID_Div1.style.setExpression("width","seconds*10");
        ID_Div2.setExpression("innerText","seconds.toString()");
    }

    function timer()
    {
        seconds++;
        document.recalc();
    }

    function starttimer()
    {
        if (timerID == null)
        {
            timerID = setInterval("timer()", 1000);
            ID_Button1.disabled = true;
            ID_Button2.disabled = false;
        }
    }

    function stoptimer()
    {
        if (timerID != null)
```



```

        {
            clearInterval(timerID);
            timerID = null;
            ID_Button1.disabled = false;
            ID_Button2.disabled = true;
        }
    }

    function resettimer()
    {seconds = 0;}
</SCRIPT>
</HEAD>
<BODY onload="init()">
    <DIV ID="ID_Div1" STYLE="background-color:lightblue" ></DIV>
    <DIV ID="ID_Div2" STYLE="color:hotpink;font-weight:bold"></DIV>
    <BR>
    <BUTTON ID="ID_Button1"                onclick="starttimer()">Start Timer</BUTTON></BR>
    <BUTTON ID="ID_Button2" DISABLED="true" onclick="stoptimer()">Stop Timer</BUTTON><BR>
    <BUTTON                                onclick="resettimer()">Reset Timer</BUTTON><BR>
</BODY>
</HTML>

```

Beispiel 2 für Sekundenbalken:

```

<HTML>
<HEAD>
<STYLE>
    BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
    var timerID = null;
    var Zahler = 0;

    function Init()
    {
        // DIV-Eigenschaften festlegen

        // DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
        ID_Div1.style.setExpression("width","Zahler *10");

        // DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
        ID_Div2.setExpression("innerText","Zahler.toString()");
    }

    function Uhr()
    {
        // Sekunden kumulieren
        Zahler ++;

        // und Anzeige neu berechnen
        document.recalc();
    }

    function Starten()
    {
        if (timerID == null)
        {
            // Start-Button nicht aktivierbar machen
            ID_Button1.disabled = true;

            // Stop-Button aktivierbar machen
            ID_Button2.disabled = false;

            // Uhr neu starten
            timerID = setInterval("Uhr()", 1000);
        }
    }

    function Stoppen()
    {
        if (timerID != null)
        {
            clearInterval(timerID);
            timerID = null;

```



```

        ID_Button1.disabled = false;
        ID_Button2.disabled = true;
    }

    function ZurueckSetzen()
    { Zahler = 0; }
</SCRIPT>
</HEAD>
<BODY onload="Init()">
    <DIV ID="ID_Div1" STYLE="background-color:lightblue" ></DIV>
    <DIV ID="ID_Div1" STYLE="color:hotpink;font-weight:bold"></DIV>
    <BR>
    <BUTTON ID="ID_Button1"
        onclick="Starten()"
    >
        Start
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button2"
        DISABLED="true"
        onclick="Stoppen()"
    >
        Stop
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button3"
        onclick="ZurueckSetzen()"
    >
        Reset
    </BUTTON>
    <BR>
    <P STYLE="width:200;color:white;background-color:gray">
        Sekundenbalken
    </P>
</BODY>
</HTML>

```

Beispiel 3 für Sound mit Sekundenanzeige:

```

<HTML>
<HEAD>
<SCRIPT>
    // ++++++ globale Variablen, die verändert werden können
    var SoundUrl = "56sec.mid";
    var SoundDauerInSekunden = 56; // Dauer muss exakt stimmen !

    var PixelBreiteProBalkenErweiterung = 10;

    // ++++++ Browser-Typ ermitteln
    // Dieser Quellcode muss VOR allen anderen Routinen codiert sein, damit zuerst abgearbeitet
    var ns = document.layers ? true : false;
    var ie = document.all ? true : false;

    // ++++++ Routinen der Sekundenzählung
    var SekundenZahler = 0;
    var SekundenZahlerTimeoutID = null;

    function SekundenZaehlen() // wird durch Rekursion alle Sekunde neu gestartet
    {
        // Zähler erhöhen
        SekundenZahler++;

        // visuelle Anzeige im Dokument auffrischen, also alle Audrucke der Style-Werte
        // neu berechnen und damit alle DIV's neu visualisieren
        document.recalc();
    }

    function SekundenZaehlen_Start()
    {
        // prüfen ob Sekundenzählen nicht bereits läuft
        if (SekundenZahlerTimeoutID == null)
        {
            // nicht aktiv

```



```

        // Rekursion starten: SekundenZaehlen() wird permanent alle Sekunde aktiviert
        SekundenZahlerTimeoutID = setInterval("SekundenZaehlen()", 1000);
    }
}

function SekundenZaehlen_Stop()
{
    // prüfen ob Sekunden zählen aktiv ist
    if (SekundenZahlerTimeoutID != null)
    {
        // aktiv, also stoppen
        clearInterval(SekundenZahlerTimeoutID);
        SekundenZahlerTimeoutID = null;
    }
}

// ++++++++ Routine zur Erzeugung Sound-Objekt
function SoundObjektErzeugen(SoundFileUrl, SoundFileDauerInSekunden)
{
    this.SoundFileUrl = SoundFileUrl;
    this.SoundFileDauerInMillisekundeSekunden = SoundFileDauerInSekunden * 1000;
    // Timerzeit für Rekursion
    this.SoundFileBeendet = true; // kein Sound aktiv
}

// ++++++++ Routinen zur Wiedergabe Sound-Objekt
var SoundTimeoutID=0;

function SoundAbspielen()
{
    // prüfen ob Sound nicht bereits aktiv ist
    if (SoundObjekt.SoundFileBeendet)
    {
        // Anzeige initialisieren
        SekundenAnzeigeInit();

        // Sekundenzähler starten, wobei das Zählen eigenständig und parallel erfolgt
        SekundenZaehlen_Start();

        // Sound erzeugen und sofort starten durch Url-Zuweisung
        ID_BGSound.src=SoundObjekt.SoundFileUrl ;
        SoundObjekt.SoundFileBeendet=false;

        // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen
        SoundTimeoutID = setTimeout(
            "SoundAbspielen()",
            SoundObjekt.SoundFileDauerInMillisekundeSekunden
        );
    }
    else
    {
        // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

        // und Meldung
        var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;
        var TimerUngenauigkeit2 = TimerUngenauigkeit1 / SoundDauerInSekunden;
        alert(
            "Wiedergabe beendet\nUngenauigkeit des Timers = "
            + TimerUngenauigkeit1.toString()+ " Sekunden\n"
            + "also " + TimerUngenauigkeit2.toString()+ " Ticks pro Sekunde"
        );
    }
}

// ++++++++ Sekunden-Anzeige initialisieren
function SekundenAnzeigeInit()
{
    // ---- Variablen init

```



```

        SekundenZahler          = 0;
        SekundenZahlerTimeoutID = null;

// ---- visuelle Anzeige erzeugen
// - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
//      Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
//      Der Ausdruck liefert den Wert , welcher sofort das Layout der
//      DIV's beeinflusst.
//      Jeder Ausdruck besitzt den SekundenZahler als Komponente.
//      Damit ändert sich der Wert des Ausdrucks.
//      Für die Neuberechnung des Ausdrucks ist der Aufruf von
//      document.recal()
//      nötig.
//      Dieser Aufruf erfolgt in SekundenZahlen(), also permanent pro Sekunde.
//      Damit wird der Style-Wert permanent neu berechnet.
//      Damit visualisieren sich die DIV's permanent neu.

// Sekundenbalken in der Style-Eigenschaft width (Breite) mit Ausdruck belegen,
//      also dynamisch anzeigen
ID_DIV_Balken.style.setExpression( "width",
                                   "SekundenZahler * PixelBreiteProBalkenErweiterung"
                                   );

// Sekundenzähler in der Eigenschaft .innerText mit Ausdruck belegen,
//      also dynamisch anzeigen
ID_DIV_SekundenZahler.setExpression("innerText", "SekundenZahler.toString()");

// - - - Messlatte statisch anzeigen
ID_DIV_MessLatte.style.width = SoundDauerInSekunden * PixelBreiteProBalkenErweiterung;
ID_DIV_MessLatte.innerText  = "Der Sound dauert "
                               + SoundDauerInSekunden.toString()
                               + " Sekunden";
}

// ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
if (ie)
{
    document.write('<BODY></BODY>');

    document.write( ' <BGSOUND ID= "ID_BGSound" LOOP="0">'

    document.write( ' <DIV ID="ID_DIV_Balken"'
                    + ' STYLE="background-color:lightblue"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_SekundenZahler"'
                    + ' STYLE="color:hotpink;font-weight:bold"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_MessLatte"'
                    + ' STYLE="color:white;background-color:gray"'
                    + '>'
                    + '</DIV>'

                    );

// ++++++ Sound initialisieren und starten mit Laden des Dokumentes

// ---- Sound-Objekt erzeugen anhand globaler Variablen
SoundObjekt = new SoundObjektErzeugen(SoundUrl, SoundDauerInSekunden);

// ---- Sound-Objekt wiedergeben
SoundAbspielen(); // meldet wenn Wiedergabe beendet ist
}
</SCRIPT>
</HEAD>
<BODY>
<!-- BODY-Teil muss leer bleiben!-->
</BODY>

```



&lt;/HTML&gt;

**.refresh()** Anzeige der Tabelle neu erzeugen  
 Änderungen an der Tabelle sichtbar machen z.B. wenn Tabelle in Anzahl Zeilen bzw. Spalten manipuliert wurde  
 siehe Objekt table

**.releaseCapture()** Maus-Überwachung ausschalten für ein Objekt  
 Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.  
 Hinweis: einschalten per Methode .setCapture()

Beispiel:

```
<BODY onload="ID_Div.setCapture();"
onclick="document.releaseCapture();"
>
<DIV ID="ID_Div"
onmousemove="ID_Textarea.value = event.clientX + event.clientY;"
onlosecapture="alert(event.srcElement.id + ' hat keine Mausüberwachung mehr');"
>
<TEXTAREA ID="ID_Textarea" COLS=2>Test</TEXTAREA>
</DIV>
</BODY>
```

**.reload()** aktuelles Dokument neu laden  
 false Default  
 Dokument aus dem Browsercache laden  
 true Dokument vom Server laden  
 Hinweis: Server kann eigenen Cache haben

Beispiel:

```
<HEAD>
<SCRIPT>
function Entfernen()
{
    // versuche den Text zu entfernen
    try
    {
        var KindZeigerAufTextImDiv = ID_Div.children(0);
        ID_Div.removeChild(KindZeigerAufTextImDiv);
        // Achtung: Der Text ist noch sichtbar !!!!
    }
    // oder fange das Ereignis des bereits entfernten Textes ein
    // und behandle das Ereignis
    catch(x)
    {
        alert( "Text wurde entfernt !\n"
            + "Das Dokument muss neu geladen werden !
        );
        document.location.reload();
    }
}
</SCRIPT>
</HEAD>
<BODY>
<DIV ID="ID_Div" onclick="Entfernen()">
    Klick, um diesen Text zu entfernen !
</DIV>
</BODY>
```

**.remove()** ein Element aus einer Collection entfernen

**.Remove()** genau eine Date aus dem Dictionary Objekt entfernen (Schlüssel- und Datenfeld entfernen)

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden(DatenSchluessel)))
{
```



```

// Schlüssel nicht vorhanden, also Date hinzufügen
DatenSpeicher.add (DatenSchluessel, Date);

// und Date anzeigen
alert(DatenSpeicher.Item(DatenSchluessel));

// und löschen
DatenSpeicher.remove(DatenSchluessel);

// und Meldung
alert(SchluesselVorhanden(DatenSchluessel));
}

```

**.RemoveAll()**                    alle Daten aus dem Dictionary entfernen (alle Schlüssel- und Datenfelder), so dass das Dictionary Objekt leer aber weiterhin instanziiert ist

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel  = "a";
var Date              = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden(DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Date anzeigen
    alert(DatenSpeicher.Item(DatenSchluessel));

    // und löschen
    DatenSpeicher.removeAll();

    // und Meldung
    alert(SchluesselVorhanden(DatenSchluessel));
}

```

**.removeAttribute()**            entfernen eines per HTML erzeugten Attributes  
Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!  
per Methode **.createAttribute()** erzeugte Attribute werden nicht erfasst  
DOM wird geändert

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Entfernen()
    {
        ID_Div.removeAttribute("TITLE");
    }
</SCRIPT>
</HEAD>
<BODY onload="Entfernen();">
    <DIV ID="ID_Div" TITLE="Tooltip-Text ">Es ist ein Tooltip-Text vorhanden</DIV>
</BODY>
</HTML>

```

**.removeAttributeNode()**        entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern  
DOM wird geändert

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function ToolTipKnotenEntfernen()
    {
        var Knoten = ID_Div.getAttributeNode("TITLE");
        ID_Div removeAttributeNode(Knoten);
    }
</SCRIPT>
</HEAD>

```



```
<BODY onload="ToolTipKnotenEntfernen();">
  <DIV ID="ID_Div" TITLE="Tooltip-Text ">Es ist ein Tooltip-Text vorhanden</DIV>
</BODY>
</HTML>
```

`.removeBehavior()` per Methode `.addBehavior()` einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert

Beispiel:

```
<SCRIPT>
var FeldDerEigenschaftenID      = new Array(); // für removeBehavior
var FeldDerTagsLlimDokument     = new Array ();
var FeldDerTagsLlimDokument_Laenge = 0;

function EigenschaftHinzufuegen()
{
    FeldDerTagsLlimDokument      = document.all.tags ("LI");
    FeldDerTagsLlimDokument_Laenge = FeldDerTagsLlimDokument.length;
    for (i=0; i < FeldDerTagsLlimDokument_Laenge; i++)
    {
        var EigenschaftenID // immer neu anlegen wegen Zeigerprüfung
        = FeldDerTagsLlimDokument [i].addBehavior ("hilite.htc");

        if (iEigenschaftenID)
        {FeldDerEigenschaftenID[i] = EigenschaftenID;}
    }
}

function EigenschaftEntfernen()
{
    for (i=0; i < FeldDerTagsLlimDokument_Laenge; i++)
    {FeldDerEigenschaftenID[i].removeBehavior (FeldDerEigenschaftenID[i]); }
}
</SCRIPT>
<A HREF="javascript:EigenschaftHinzufuegen()">Eigenschaft hinzufuegen</A>
<A HREF="javascript:EigenschaftEntfernen()">Eigenschaft entfernen</A>
```

`.removeChild()` Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert

Beispiel:

```
<HEAD>
<SCRIPT>
function Entfernen()
{
    // versuche den Text zu entfernen
    try
    {
        var KindZeigerAufTextImDiv = ID_Div.children(0);
        ID_Div.removeChild(KindZeigerAufTextImDiv);
        // Achtung: Der Text ist noch sichtbar !!!!
    }
    // oder fange das Ereignis des bereits entfernten Textes ein
    // und behandle das Ereignis
    catch(x)
    {
        alert(      "Text wurde entfernt !\n"
                    + "Das Dokument muss neu geladen werden !
                );
        document.location.reload();
    }
}
</SCRIPT>
</HEAD>
<BODY>
  <DIV ID="ID_Div" onclick=" Entfernen()">
    Klick, um diesen Text zu entfernen !
  </DIV>
</BODY>
```

`.removeExpression()` Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form `objekt.style.eigenschaft.dient.`





Ausdruck muss mit der Methode .setExpression() gesetzt worden sein  
DOM wird nicht geändert

Beispiel: aus Platzgründen die Zeichenkette von STYLE umgebrochen,  
was eigentlich nicht zulässig ist, und es sind nicht alle SPAN kodiert.

```

ID_Span.style.setExpression("width","document.body.style.fontSize");
<SPAN ID="ID_Span"
      STYLE="background-color:lightgreen;
            width:expression( trueBlueSpan.style.pixelWidth
                              + oldYellowSpan.style.pixelWidth
                              )
">
</SPAN>
ID_Span.style.removeExpression("width");

```

.removeNamedItem() Attribut entfernen anhand Attributname (analog zu ID oder NAME-Attribut),  
wobei danach der Standard-Attributwert automatisch weiterverwendet wird  
(falls Standard vorhanden ist),  
und Zeiger auf gelöscht Attribut liefern  
ab IE 6

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
function Entfernen()
{
    var ZeigerAufFeld = ID_Div.attributes;
    ZeigerAufFeld.removeNamedItem("TITLE");
}
</SCRIPT>
</HEAD>
<BODY>
<DIV onclick="Entfernen();" ID="ID_Div" TITLE="Tooltip-Text ">
    Klick um den Tooltip-Text zu entfernen
</DIV>
</BODY>
</HTML>

```

.removeNode() Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde  
DOM wird geändert

Beispiel:

```

<SCRIPT>
function Entfernen()
{
    Tabelle.removeNode(true);
}
</SCRIPT>
<TABLE ID = "Tabelle" >
<TR>
<TD>Zelle 1</TD>
<TD>Zelle 2</TD>
</TR>
</TABLE>
<INPUT TYPE = button VALUE = " Entfernen" onclick = " Entfernen()">

```

.removeRule() StyleSheet aus der Collection styleSheet.rules entfernen  
Achtung: Um Style auch sichtbar zu entfernen, muss  
Dokument muss neu geladen werden  
oder alle betroffene Style-Elemente neu mit Wert belegen  
durch Zuweisung auf sich selbst (siehe Beispiel)  
siehe Objekt styleSheet und Collection styleSheet.rules

Beispiel:

```

<STYLE>
P {color:green}
</STYLE>
<SCRIPT>
function StyleEntfernen()
{
    var StyleSheetsObjekt = document.styleSheets;
    var AnzahlStyleSheets = StyleSheetsObjekt.length;

    if (AnzahlStyleSheets > 0)
    {

```



```

// StyleSheet mit Index 0 bearbeiten also P {color:green}
var StyleSheetAnIndex0 = StyleSheetsObjekt[0];

// wobei P {color:green} eine Regel ist, also Collection rules verwenden
var StyleSheetsRegelCollection = StyleSheetAnIndex0.rules;

var AnzahlRegeln = StyleSheetsRegelCollection.length;

if (AnzahlRegeln > 0)
{
    // Regel P {color:green} entfernen
    StyleSheetAnIndex0.removeRule(0);

    // visuell auch die Farbe entfernen durch Zuweisung auf sich selbst also
    // nun ohne die Regel P {color:green}
    ID_P.innerHTML= ID_P.innerHTML;
}
}
}
</SCRIPT>

<P ID="ID_P" >Test</P>
<BUTTON onclick="StyleEntfernen()">Text im P-Tag entfaerben.</BUTTON>

```

.replace() neues Dokument zuweisen und laden  
im Verlauf (History) wird der Eintrag des alten, vorherigen Dokumentes durch  
den Eintrag des neuen zu ladenden Dokumentes ersetzt.  
Altes Dokument ist damit **nicht** mehr per Vorwärts- und Zurück-Button einstellbar.

Beispiel:  
window.location.replace("test.html");

.replace() Suche in einem String oder String-literal per RegExp Objekt (siehe dort Methode .exec())  
und gefundenen String ersetzen  
siehe Script-Objekt String

Beispiel 1:

```

var Kette =
    "The man hit the ball with the bat.\nwhile the fielder caught the ball with the glove.";
var RegExpressionAusdruck = /The/g;
var Feld1 = Kette.replace(RegExpressionAusdruck, "A"); // "A" ersetzt "The"
var Feld2 =
    "The man hit the ball with the bat.while the fielder caught the ball with the glove.".replace(
        RegExpressionAusdruck, "A"); // "A" ersetzt "The"

```

Beispiel 2:

```

function F_ahrenheitTauschenGegen_C_eslius(Kette)
{
    var RegExpressionAusdruck = /(\\d+(\\.d*)?)F\\b/g;

    return ( Kette.replace(
        RegExpressionAusdruck,
        function($0,$1,$2) {return(((($1-32) * 5/9) + "C");}
    )
    );
}
document.write(F_ahrenheitTauschenGegen_C_eslius ("Wasser kristallisiert bei 32F und siedet bei 212F.));

```

Beispiel 3:

```

var zu_durchsuchende_kette = "Otto Waalkes"; // oder RegExp.input="Otto Waalkes"
var such_muster = /(\\wOtto)/g; // such_muster ist ein regulärer Ausdruck
// (Objekt RegExp)
// Suchmuster ist Otto, wobei Otto gefunden
// werden muss für eine erfolgreiche Suche
// anstelle von " ist / zu kodieren
// alternativ nicht möglich
// RegExp.input="Otto"
// dann kein detailliertes Suchmuster
// Option g alternativ kodierbar per
// RegExp.multiline=true;

var ergebnis_kette = zu_durchsuchende_kette.replace(such_muster,"Heinrich");

```

.replaceAdjacentText() Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern



DOM wird nicht geändert

Beispiel:

```
var PlainText = " Neuer Text ";
ID_Div.replaceAdjacentText("afterBegin", PlainText);
```

```
<DIV ID="ID_Div">
  Test
</DIV>
```

.replaceChild()

Kind-Objekt ersetzen durch ein Objekt  
 ersetzende Objekt muss per Methode .createElement() erzeugt worden sein  
 Sichtbarkeit erst wenn Ende-Tag geparkt wurde  
 DOM wird geändert

Beispiel:

```
<HEAD>
<SCRIPT>
  function Ersetze()
  {
    var KindZeigerAufDivText = ID_Div.children(0);
    var RetteInnerHTML = KindZeigerAufDivText.innerHTML;

    // prüfen auf Tag im Div-Text
    if (KindZeigerAufDivText.tagName=="B")
    {
      // Bold-Tag <B>gefunden, also I-Tag erzeugen
      var ZeigerAufNeuenSchriftStilTag =document.createElement("I");

      // komplettes ersetzen von Div-Text,
      ID_Div.replaceChild(ZeigerAufNeuenSchriftStilTag, KindZeigerAufDivText);
      ZeigerAufNeuenSchriftStilTag.innerHTML= RetteInnerHTML;
    }
    else
    {
      // keinen Bold-Tag <B>gefunden
      var ZeigerAufNeuenSchriftStilTag =document.createElement("B");

      // komplettes ersetzen von Div-Text,
      ID_Div.replaceChild(ZeigerAufNeuenSchriftStilTag, KindZeigerAufDivText);
      ZeigerAufNeuenSchriftStilTag.innerHTML= RetteInnerHTML;
    }
  }
</SCRIPT>
</HEAD>
<BODY>
  <DIV ID="ID_Div" onclick=" Ersetze()">
    Klicke für den Wechseln des <B>Schriftstils<B>
  </DIV>
</BODY>
```

.replaceData()

Teilkette in einem Objekt ersetzen

.replaceNode()

Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern  
 sichtbar erst mit parsen des Endetags  
 DOM wird geändert

Beispiel:

```
<SCRIPT>
  function Ersetze()
  {
    var RetteInnerHTML = Liste.innerHTML;
    var ZeigerAufNeuenKnoten = document.createElement("OL");
    Liste.replaceNode(ZeigerAufNeuenKnoten);
    ZeigerAufNeuenKnoten.innerHTML = RetteInnerHTML;
  }
</SCRIPT>
<UL ID = "Liste" >
  <LI>Listeneintrag 1
  <LI>Listeneintrag 2
  <LI>Listeneintrag 3
  <LI>Listeneintrag 4
</UL>
<INPUT TYPE = button VALUE = "Ersetze" onclick = "Ersetze()">
```

.reset()

löst für Formular das Event onreset aus, dessen Eventhandler die Reset-Aktion beinhaltet, die gestartet wird  
 VOR dem Reset der Elemente (Browser setzt zurück)



Eventhandler muss liefern:      return true,      für Ausführung des Reset durch Browser  
    return false;      für Nicht-Ausführung des Reset durch Browser

Beispiel: Man beachte, dass NAME-Attribut für alle zu sendenden bzw. rückzusetzenden Elemente **Pflicht** ist !!

```
<HTML>
<HEAD>
<SCRIPT>
    function EventHandler_AktionVorReset()
    {
        // ein Hinweis im Textarea anzeigen
        ID_Textarea.value += "Formular zuruecksetzen ???";

        // wirklich rücksetzen ???
        return( confirm("Wirklich rücksetzen ?"));
        // true so Reset durch Browser ausführen lassen
        // false so kein Reset durch Browser
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <FORM NAME="Formular" onreset="EventHandler_AktionVorReset();">
            <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
            <BR>
            <BUTTON onclick="form.reset();">OnReset ausloesen</BUTTON>
            <BR><BR>
            <INPUT TYPE="reset" VALUE="oder hiermit zuruecksetzen"
                onclick="form.reset()"
            >
        </FORM>
    </DIV>
    <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>
```

.resetElement()      alle Timeline-Einstellungen zum Element aufheben (egal ob Element aktiv oder nicht)  
                          siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:EXCL ID="ID_Excl"
        REPEATDUR="indefinite"
        onbegin="ID_Button1.disabled=false; ID_Button2.disabled=true;"
    >
        <DIV ID="ID_Div1"
            CLASS="time_line_klasse"
            BEGIN="0"
            DUR="5"
        >
            Erste Zeile
        </DIV>
        <DIV ID="ID_Div2"
            CLASS="time_line_klasse"
            BEGIN="5"
            DUR="5"
        >
            Zweite Zeile
        </DIV>
    </t:EXCL>
    <BR>
    <BUTTON ID="ID_Button1"
        onclick=" ID_Excl.endElementAt(5);
            alert('Time-Container endet in 5 Sekunden');
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
        "
    >
```



```

        Ende der Timeline in 5 Sekunden
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Excl.resetElement();
                alert('Alle Einstellungen zuruecksetzen');
                ID_Button1.disabled=false;
                ID_Button2.disabled=true;
        "
    >
        Reset
    </BUTTON>
</BODY>
</HTML>

```

**.resizeBy()** Fenstergröße um Pixeldifferenz verändern  
 Fenstergröße auf weniger als 100 x 100 Pixel ist nicht möglich  
 nicht bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop, dialogLeft benutzen  
 funktioniert nur, wenn open-Merkmal resizable=yes kodiert wurde  
 siehe Objekt window

**.resizeTo()** Fenstergröße neu dimensionieren in Pixel  
 Fenstergröße auf weniger als 100 x 100 Pixel ist nicht möglich  
 nicht bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop, dialogLeft benutzen  
 funktioniert nur, wenn open-Merkmal resizable=yes kodiert wurde  
 siehe Objekt window

Beispiel für Fensterauflösung ändern:

```
javascript:window.resizeTo(640,480); javascript:window.resizeTo(800,600); javascript:window.resizeTo(1024,768)
```

Beispiel für sich aufblasendes Fenster von 100x100 bis auf 640x480

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
    var start_hoehe=100;
    var start_breite=100;
    var max_hoehe=480;
    var max_breite=640;
    var aktuelle_hoehe=start_hoehe;
    var aktuelle_breite=start_breite;
    var y=5;
    var TimerID=null;
    var fenster;

    function start()
    {
        fenster=window.open("", "", "scrollbars");

        if ( document.layers || document.all)
        {
            fenster.resizeTo(start_breite,start_hoehe);
            fenster.moveTo(0,0);
            blasen();
        }
        else
        {alert("Weder Netscape noch Microsoft erkannt !");}
    }

    function blasen()
    {
        if (aktuelle_hoehe>=max_hoehe)
        {x=0;}

        fenster.resizeBy(5,y);
        aktuelle_hoehe+=5;
        aktuelle_breite+=5;

        if (aktuelle_breite>=max_breite)
        {
            alert("Maximal aufgeblasen !");
            fenster.close();
            x=5;
        }
    }

```



```

        TimerID=null;
    }
    else
    {TimerID=setTimeout("blasen()",50);}
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<A      HREF="javascript:start()"
        onMouseOver="javascript:window.status='Oeffne Fenster';return true;" // Text nach Statuszeile
        onMouseout="javascript:window.status=";" // Statuszeile löschen
>Oeffne NEUES Fenster mit 100x100 Pixel und blase es auf 640x480 Pixel !
</A>
</BODY>
</HTML>

```

**.resume()** deprecated  
Pause auf der Timeline beenden

**.resumeElement()** Pause eines aktiven Elementes, das auf der Timeline pausiert, aufheben  
falls Element nicht pausiert, passiert nichts  
ersetzt die Methode resume(), da sie deprecated ist und nicht mehr verwendet werden darf  
per Eigenschaft .isActive das Element auf Aktivsein prüfen  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
.TextAufHighLight{ color:#CCCCC; font-weight:bold;}
</STYLE>
<SCRIPT>
function PauseAufheben()
{
    ID_Table.resumeElement();
    ID_Button1.disabled = false;
    ID_Button2.disabled = true;
}

function Pausieren()
{
    ID_Table.pauseElement();
    ID_Button1.disabled = true;
    ID_Button2.disabled = false;
}
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<TABLE ID="ID_Table"
      BORDER="1"
      CLASS="time_line_klasse"
      DUR="8s"
      REPEATCOUNT="indefinite"
      TIMECONTAINER="SEQ"
      TIMEACTION="none"
>
<TBODY>
<TR>
<TH WIDTH="120">Eins</TH>
<TH WIDTH="50">Zwei</TH>
<TH WIDTH="40">Drei</TH>
</TR>
<TR
      ID="ID_TR1"
      CLASS="time_line_klasse"
      TIMEACTION="class: TextAufHighLight "
      DUR="2s"
>
<TD>EinsA</TD>
<TD>ZweiA</TD>
<TD>DreiA</TD>
</TR>
<TR
      ID="ID_TR2"

```



```

        CLASS="time_line_klasse"
        TIMEACTION="class: TextAufHighLight "
        DUR="2s"
    >
        <TD>EinsB</TD>
        <TD>ZweiB</TD>
        <TD>DreiB</TD>
    </TR>
</TBODY>
</TABLE>
<BR>
<BUTTON        ID="ID_Button1"
                onclick="Pausieren();"
>
    pausieren
</BUTTON>
<BUTTON        ID="ID_Button2"
                DISABLED="true"
                onclick="PauseAufheben();"
>
    Pause aufheben
</BUTTON>
</BODY>
</HTML>

```

**.reverse()** Reihenfolge der Feldelemente physisch umkehren, also

alt	neu
1. Element	letztes Element
letztes Element	1. Element

Feld hat die Objektklasse Script-Objekt Array

Beispiel:

```

var Feld1 = new Array(0,1,2,3,4);    // Feld 1 ist Zeiger vor reverse
var Feld2 = Feld1.reverse();          // Feld 2 ist Zeiger nach reverse, Feld1 ist ungültig
alert(Feld2.join());    // "4,3,2,1,0"

```

**.round()** zahl auf ganz runden; ab >=5 aufwärts  
Bsp: 0,5 ergibt 1  
siehe Script-Objekt Math

**.save()** UserDataStore (User-Cache) speichern per .style.userData Behavior

Beispiel:

```

<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //      es können diverse Cachennamen definiert und somit Versionen von Cache
    //      verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //      es können diverse Attribute definiert und somit Versionen von Input-Daten
    //      verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;
    }

```



```

// ++++++ Daten chachen ++++++
// aktuelle Daten holen laut Input-Objekt
var InputDaten = InputDatenObjekt.value;

// Attribut instanzieren und mit Daten füllen
InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

// und Cache saveen
InputDatenObjekt.save(FreierCacheName);
}

function InputLaden()
{
    // Cache laden
    InputDatenObjekt.load(FreierCacheName);

    // und Daten zum Attribut laut Sicherung lesen
    var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
}

</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

.ScriptEngine()      Sprache der gerade benutzten Scriptmaschine im Internet Explorer

"JScript"	Microsoft JScript
"VBA"	Microsoft Visual Basic for Applications
"VBScript"	Microsoft Visual Basic Scripting Edition

Beispiel:

```

function Anzeigen()
{
    var Kette = "";

    Kette += "Aktuelle Maschine = " + ScriptEngine();
    Kette += " mit Hauptversion " + ScriptEngineMajorVersion();
    Kette += " mit Unterversion " + ScriptEngineMinorVersion();
    Kette += " mit Buildnummer " + ScriptEngineBuildVersion();

    alert(Kette);
}

```

.ScriptEngineBuildVersion()      Buildnummer der gerade benutzten Scriptmaschine im Internet Explorer

Beispiel:

```

function Anzeigen()
{
    var Kette = "";

    Kette += "Aktuelle Maschine = " + ScriptEngine();
    Kette += " mit Hauptversion " + ScriptEngineMajorVersion();
    Kette += " mit Unterversion " + ScriptEngineMinorVersion();
    Kette += " mit Buildnummer " + ScriptEngineBuildVersion();

    alert(Kette);
}

```

.ScriptEngineMajorVersion()      Hauptversion der gerade benutzten Scriptmaschine im Internet Explorer

Beispiel:

```

function Anzeigen()
{
    var Kette = "";

    Kette += "Aktuelle Maschine = " + ScriptEngine();
    Kette += " mit Hauptversion " + ScriptEngineMajorVersion();
}

```





```

        Kette += " mit Unterversion " + ScriptEngineMinorVersion();
        Kette += " mit Buildnummer " + ScriptEngineBuildVersion();

        alert(Kette);
    }

```

`.ScriptEngineMinorVersion()` Unterversion der gerade benutzten Scriptmaschine im Internet Explorer

Beispiel:

```

function Anzeigen()
{
    var Kette = "";

    Kette += "Aktuelle Maschine = " + ScriptEngine();
    Kette += " mit Hauptversion " + ScriptEngineMajorVersion();
    Kette += " mit Unterversion " + ScriptEngineMinorVersion();
    Kette += " mit Buildnummer " + ScriptEngineBuildVersion();

    alert(Kette);
}

```

`.scroll()` deprecated und zu ersetzen durch `.scrollBy()` bzw. `.scrollTo()`  
 linke obere Ecke des Dokumentes im Fenster verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters  
 Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisen etc.  
 sinnvoll, wenn automatische Anzeige von Scrolleisen verhindert wurde  
 Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters  
 siehe Objekt window

`.scrollBy()` ersetzt `.scroll()`  
 linke obere Ecke des Dokumentes im Fenster verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters  
 Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisen etc.  
 sinnvoll, wenn automatische Anzeige von Scrolleisen verhindert wurde  
 Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters  
 siehe Objekt window

`.scrollIntoView()` Objekt derart scrollen, dass es im Fenster für User sichtbar wird  
 Objekt muss an sich schon renderbar sein  
 true Default  
 Obere Kante des Objekt bis an den oberen Fensterrand scrollen  
 false Untere Kante des Objektes an den unteren Fensterrand scrollen

Beispiel:

```

var FeldAllerPTags = document.all.tags("P");
FeldAllerPTags[4].scrollIntoView(true);

```

`.scrollTo()` ersetzt `.scroll()`  
 linke obere Ecke des Dokumentes im Fenster auf Pixelposition bezüglich linker oberer Ecke des Anzeigebereiches des Fensters setzen  
 Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisen etc.  
 sinnvoll, wenn automatische Anzeige von Scrolleisen verhindert wurde  
 Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters  
 siehe Objekt window

Beispiel für Dokument vertikales Scrollen:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function scrollen(pixel_anzahl)
    {
        for (i=1; i<pixel_anzahl; i++)
        {window.scrollTo(0,i);}
    }
-->
</SCRIPT>
</HEAD>

<BODY>
....
<INPUT TYPE=button

```



```

        VALUE="Start scrollen!"
        onclick="scrollen(100);"
    >
</BODY>
</HTML>

```

`.search()` Suche in einem String oder String-literal per RegExp-Objekt (siehe dort Methode `.exec()`)  
siehe Script-Objekt String

Beispiel 1:

```

var Kette          = "The rain in Spain falls mainly in the plain.";
var RegExpAusdruck = /falls/i;
var Wert           = Kette.search(RegExpAusdruck);

```

Beispiel 2:

```

var zu_durchsuchende_kette = "Otto";
var such_muster            = /(\wOtto)/g; // vom Typ RegExp, also regulärer Ausdruck
var ergebnis              = zu_durchsuchende_kette.search(such_muster);

```

`.seekActiveTime()` aktives Element animieren ab einem Zeitpunkt auf der Timeline  
wenn Element nicht aktiv, so Fehler erzeugt  
per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
alle Media-Typen für Element zulässig  
siehe Eigenschaft `.canSeek`  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    function Suchen()
    {
        // prüfen ob Element nicht aktiv ist
        if (!ID_Video.currTimeState.isActive)
        {
            // nicht aktiv, also starten
            ID_Video.beginElement();
        }
        else
        {
            // prüfen des Eingabewertes
            if( ( isFinite(ID_Input.value) )
                && (ID_Input.value <= ID_Video.mediaDur)
                && (ID_Input.value > 0)
            )
            {
                // ist okay, also ab Zeitpunkt animieren
                ID_Video.seekActiveTime(ID_Input.value);
            }
            else
            {
                // fehlerhaft
                alert( "Fehler ! Sekunden-Wert muss > 0 und < Wert laut mediaDur = "
                    + ID_Video.mediaDur
                    + " sein"
                );
                ID_Input.focus();
            }
        }
    }
}
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Video.currTimeState.activeTime);"
    >
        0
    </SPAN>

```



```

<BR>
<t:VIDEO ID="ID_Video"
SRC"test.avi"
STYLE="width:175px; height:150px;"
onmediacomplete="ID_Span2.innerText= ID_Video.mediaDur;"
>
</t:VIDEO>
<BR>
Dauer des AVI:
<SPAN ID="ID_Span2"></SPAN>
&nbsp;Sekunden
<BR>
setze seekActiveTime:
<INPUT NAME="ID_Input"
TYPE="text"
VALUE=""
SIZE="3"
>&nbsp;Sekunden
<BR>
<BUTTON ID="ID_Button1"
onclick="Suchen();"
>
Klick fuer Seek
</BUTTON>
<BUTTON ID="ID_Button2"
onclick="ID_Video.beginElement()"
>
Restart
</BUTTON>
</BODY>
</HTML>

```

.seekSegmentTime() aktives Element animieren ab einem Zeitpunkt auf der Segment-Timeline  
ohne Wiederholung der Animation  
wenn Element nicht aktiv, so Fehler erzeugt  
per Eigenschaft .isActive das Element auf Aktivsein prüfen  
Software-Player zum Mediatyp des Elementes wird nicht davon beeinflusst  
nicht alle Media-Typen für Element zulässig  
wenn unzulässig, so kein Fehler  
siehe Eigenschaft .canSeek  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
function Suchen()
{
    // prüfen ob Element nicht aktiv ist
    if (!ID_Video.currTimeState.isActive)
    {
        // nicht aktiv, also starten
        ID_Video.beginElement();
    }
    else
    {
        // prüfen des Eingabewertes
        if( ( isFinite(ID_Input.value) )
        && (ID_Input.value <= ID_Video.segmentDur )
        && (ID_Input.value > 0)
        )
        {
            // ist okay, also ab Zeitpunkt animieren
            ID_Video.seekSegmentTime(ID_Input.value);
        }
        else
        {
            // fehlerhaft
            alert( "Fehler ! Sekunden-Wert muss > 0 und < Wert laut segmentDur = "
            + ID_Video.segmentDur
            + " sein"
            );
        }
    }
}

```



```

    );
    ID_Input.focus();
  }
}
</SCRIPT>
</HEAD>
<BODY>
  <SPAN ID="ID_Span1"
    CLASS="time_line_klasse"
    DUR=".01"
    REPEATCOUNT="indefinite"
    FILL="hold"
    onrepeat="innerText=parseInt(ID_Video.currTimeState.activeTime);"
  >
    0
  </SPAN>
  <BR>
  <t:VIDEO ID="ID_Video"
    SRC="test.avi"
    STYLE="width:175px; height:150px;"
    onmediacomplete="ID_Span2.innerText= ID_Video.segmentDur;"
  >
  </t:VIDEO>
  <BR>
  Dauer des AVI:
  <SPAN ID="ID_Span2"></SPAN>
  &nbsp;Sekunden
  <BR>
  setze seekSegmentTime:
  <INPUT NAME="ID_Input"
    TYPE="text"
    VALUE=""
    SIZE="3"
  >&nbsp;Sekunden
  <BR>
  <BUTTON ID="ID_Button1"
    onclick="Suchen();"
  >
    Klick fuer Seek
  </BUTTON>
  <BUTTON ID="ID_Button2"
    onclick=" ID_Video.beginElement()"
  >
    Restart
  </BUTTON>
</BODY>
</HTML>

```

.seekTo()      aktives Element animieren ab einem Zeitpunkt auf der Segment-Timeline  
                  einschliesslich möglicher Wiederholungen der Animation  
 wenn Element nicht aktiv, so Fehler erzeugt  
                  per Eigenschaft .isActive das Element auf Aktivsein prüfen  
 nicht alle Media-Typen für Element zulässig  
                  wenn unzulässig, so kein Fehler  
 siehe Eigenschaft .canSeek  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
  .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
  function Suchen()
  {
    // prüfen ob Element nicht aktiv ist
    if (!ID_Video.currTimeState.isActive)
    {
      // nicht aktiv, also starten
      ID_Video.beginElement();
    }
    else
  }

```



```

        {
            // prüfen des Eingabewertes
            if(
                ( isFinite(ID_Input.value) )
                && (ID_Input.value <= ID_Video.segmentDur )
                && (ID_Input.value > 0)
            )
            {
                // ist okay, also ab Zeitpunkt animieren
                ID_Video.seekTo(1,ID_Input.value);
            }
            else
            {
                // fehlerhaft
                alert( "Fehler ! Sekunden-Wert muss > 0 und < Wert laut segmentDur = "
                    + ID_Video.segmentDur
                    + " sein"
                );
                ID_Input.focus();
            }
        }
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Video.currTimeState.activeTime);"
    >
        0
    </SPAN>
    <BR>
    <t:VIDEO ID="ID_Video"
        SRC="test.avi"
        STYLE="width:175px; height:150px;"
        onmediacomplete="ID_Span2.innerText= ID_Video.segmentDur;"
    >
    </t:VIDEO>
    <BR>
    Dauer des AVI:
    <SPAN ID="ID_Span2"></SPAN>
    &nbsp;&nbsp;&nbsp;Sekunden
    <BR>
    setze seekTo:
    <INPUT TYPE="text"
        VALUE=""
        NAME="ID_Input"
        SIZE="3"
    >&nbsp;&nbsp;&nbsp;Sekunden
    <BR>
    <BUTTON ID="ID_Button1" onclick="Suchen();">
        Klick fuer Seek
    </BUTTON>
    <BUTTON ID="ID_Button2" onclick=" ID_Video.beginElement();">
        Restart
    </BUTTON>
</BODY>
</HTML>

```

`.seekToFrame()` Frame eines aktiven Elementes auf der Timeline anwählen  
wenn Element nicht aktiv, so Fehler erzeugt  
per Eigenschaft `.isActive` das Element auf Aktivsein prüfen  
nicht alle Media-Typen für Element zulässig  
wenn unzulässig, so kein Fehler  
siehe Eigenschaft `.canSeek`  
siehe Objekt `currTimeState` und Behavior `.style.time2`

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }

```



```

</STYLE>
<SCRIPT>
    var FrameAnzahl=600;

    function Suchen()
    {
        // prüfen ob Element nicht aktiv ist
        if (!ID_Video.currTimeState.isActive)
        {
            // nicht aktiv, also starten
            ID_Video.beginElement();
        }
        else
        {
            // prüfen des Eingabewertes
            if( ( isFinite(ID_Input.value) )
                && (ID_Input.value < FrameAnzahl )
                && (ID_Input.value > 0)
            )
            {
                // ist okay, also ab Zeitpunkt animieren
                ID_Video.seekToFrame(ID_Input.value);
            }
            else
            {
                // fehlerhaft
                alert( "Fehler ! Frame-Wert muss > 0 und < "
                    + FrameAnzahl
                    + " sein"
                );
                ID_Input.focus();
            }
        }
    }
}
</SCRIPT>
</HEAD>
<BODY>
    <SPAN ID="ID_Span1"
        CLASS="time_line_klasse"
        DUR=".01"
        REPEATCOUNT="indefinite"
        FILL="hold"
        onrepeat="innerText=parseInt(ID_Video.currentFrame);"
    >
        0
    </SPAN>
    <BR>
    <t:VIDEO ID="ID_Video"
        SRC="test.avi"
        STYLE="width:175px; height:150px;"
        onmediacomplete="ID_Span2.innerText= ID_Video.mediaDur;"
    >
    </t:VIDEO>
    <BR>
    Dauer des AVI:
    <SPAN ID="ID_Span2"></SPAN>
    &nbsp;Sekunden
    <BR>
    setze seekToFrame:
    <INPUT TYPE="text"
        VALUE=""
        NAME="ID_Input"
        SIZE="3"
    >&nbsp;Sekunden
    <BR>
    <BUTTON ID="ID_Button1" onclick="Suchen();">
        Klick fuer Seek
    </BUTTON>
    <BUTTON ID="ID_Button2" onclick=" ID_Video.beginElement();">
        Restart
    </BUTTON>
</BODY>
</HTML>

```



.segmentTimeToActiveTime() Wert der Segment-Timeline des Elements in den korrespondierenden Wert der aktiven Timeline des Elements konvertieren  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
window.onload = Rekursion; // ohne () kodieren

function Rekursion()
{ window.setInterval(Anzeige, 100);}

function Anzeige()
{
    ID_Span1.innerHTML = "segmentTimeToActiveTime: "
                        + (ID_Animation.segmentTimeToActiveTime(
                            ID_div.currTimeState.activeTime
                        ));
    ID_Span1.innerHTML = "activeTime: "
                        + (ID_Animation.currTimeState.activeTime);
}
</SCRIPT>
</HEAD>
<BODY>
<DIV ID="ID_Div"
CLASS="time_line_klasse"
STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<t:ANIMATEMOTION ID="ID_Animation"
TARGETELEMENT="ID_div"
TO="250,0"
DUR="3"
AUTOREVERSE="true"
>
</t:ANIMATEMOTION>
<SPAN ID="ID_Span1">
segmentTimeToActiveTime:
</SPAN>
<BR>
<SPAN ID="ID_Span2">
activeTime:
</SPAN>
</BODY>
</HTML>
```

.segmentTimeToSimpleTime() Wert der Segment-Timeline eines Elements in den korrespondierenden Wert der Simple-Timeline des Elements konvertieren  
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
.time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
window.onload = Rekursion; // ohne () kodieren

function Rekursion()
{ window.setInterval(Anzeige, 100);}

function Anzeige()
{
    ID_Span1.innerHTML = "segmentTimeToSimpleTime: "
                        + (ID_Animation.segmentTimeToSimpleTime(
                            ID_div.currTimeState.activeTime
                        ));
}
```



```

        ID_Span1.innerHTML = "simpleTime: "
        + (ID_Animation.currTimeState.simpleTime);
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV ID="ID_Div"
        CLASS="time_line_klasse"
        STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
    >
    </DIV>
    <t:ANIMATEMOTION ID="ID_Animation"
        TARGETELEMENT="ID_div"
        TO="250,0"
        DUR="3"
        AUTOREVERSE="true"
    >
    </t:ANIMATEMOTION>
    <SPAN ID="ID_Span1">
        segmentTimeToSimpleTime:
    </SPAN>
    <BR>
    <SPAN ID="ID_Span2">
        simpleTime:
    </SPAN>
</BODY>
</HTML>

```

.select() Bereich des Input-Objektes im Formular markieren  
setzt **nicht** den Focus (dafür Methode .focus() verwenden)

.select() Objekt selektieren  
z.B. Textbereich: wird hervorgehoben  
ControlRange: es wird Rechteck-Rahmen erzeugt  
nur unter Windows 32-Bit

Beispiel:

```

<SCRIPT LANGUAGE="JScript" FOR=document EVENT=onclick >
    var TextBereich = document.body.createTextRange();
    TextBereich.moveToPoint(window.event.x, window.event.y);
    TextBereich.expand("word");
    TextBereich.select();
</SCRIPT>

```

.select() Selektion eines Textranges (Textbereich, Objekt textrange) oder ControlRange (Control-Elemente)  
nur unter Windows 32-Bit  
siehe Objekt document.selection  
Methoden .createTextRange() .createControlRange() und .createRange()

Beispiel 1 Textrange (Textbereich) erzeugen und den Inhalt markieren, also selektieren

```

function ErzeugeUndSelektiereTextRange()
{
    var TextBereich = document.body.createTextRange();
    TextBereich.findText("Testtext");
    TextBereich.select();
}

```

Beispiel 2 Controlrange erzeugen und den Inhalt markieren, also selektieren

```

function ControlRangeErzeugenUndSelektieren()
{
    var ControlRangeObjekt = document.body.createControlRange();
    ControlRangeObjekt.add(document.all.zeiger_auf_control_element);
    ControlRangeObjekt.select();
}

```

Hinweis: Elternobjekte mit Textrange sind

body Objekt  
button Objekt  
textarea Objekt  
input text Objekt  
selection Objekt  
(nur wenn ein Text selektiert wurde (mit oder ohne HTML))





können weitere HTML-Elemente enthalten, die ebenfalls Textbereiche besitzen können

`.setActive()` Objekt für die Eventdurchreichung aktivieren  
aber ohne es zu fokussieren  
und ohne es scrollbar zu machen

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    var ID_Fenster;

    function FensterErzeugen()
    {
        ID_Fenster = window.open( "/test /test.htm",
                                " ID_Fenster",
                                "top=10px,left=480px,height=375px,width=200px,resizable=1"
                                );
        this.focus();
    }

    function ButtonAktivieren()
    {window.parent.ID_Fenster.ID_Button.setActive();}
</SCRIPT>
</HEAD>
<BODY onload="FensterErzeugen();">
    <BUTTON ID="ID_Button" onclick="ButtonAktivieren();">
        Button aktivieren
    </BUTTON>
</BODY>
</HTML>
```

`.setActive()` playItem Objekt als aktiven Track setzen und als solchen in der Behavior-Collection `.style.time2.playList` registrieren  
verändert `.isActive`  
Track entspricht playItem Objekt (Playlisten-Eintrag)  
Trackliste liegt in der Behavior-Collection `.style.time2.playList` als Playliste:  
Diese ist nur referenzierbar, solange das Elternobjekt des Elementes auf der Eltern-Timeline aktiv ist, also das Elternobjekt noch nicht auf seiner Timeline geendet ist.  
Advanced Stream Redirector (ASX) –Datei: Playlisten-Datei  
aktiver Track in der Liste: wird abgespielt  
Track in der Liste auf aktiv setzen: Abspielen des Tracks sofort starten

Beispiel 1:

`object.playList.item(index).setActive()`

Index Integer, ab 0

Beispiel 2:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
<SCRIPT LANGUAGE="JScript">
    function ButtonUpdate()
    {
        if (ID_Media.currTimeState.isActive)
        {
            ID_Button1.disabled=true;
            ID_Button2.disabled=false;
            ID_Button3.disabled=false;
            ID_Button4.disabled=false;
        }
        else
        {
            ID_Button1.disabled=false;
            ID_Button2.disabled=true;
            ID_Button3.disabled=true;
            ID_Button4.disabled=true;
        }
    }

    function AnzeigeUpdate()
```



```

    {
        ID_Span1.innerText = "Titel: " + ID_Media.playlist.activeTrack.title;
        ID_Span3.innerText = "Autor: " + ID_Media.playlist.activeTrack.author;
        ID_Span3.innerText = "Abstract: " + ID_Media.playlist.activeTrack.abstract;
        ID_Span4.innerText = "Copyright: " + ID_Media.playlist.activeTrack.copyright;
        ID_Span5.innerText = "Filename: " + ID_Media.playlist.activeTrack.src;
        ID_Span6.innerText = "Banner: " + ID_Media.playlist.activeTrack.Banner;
        ID_Span7.innerText = "Banner Abstract: " + ID_Media.playlist.activeTrack.BannerAbstract;
        ID_Span8.innerText = "Banner MoreInfo: " + ID_Media.playlist.activeTrack.BannerMoreInfo;
    }

function AnzeigeLoeschen()
{
    ID_Span1.innerText = "Titel: ";
    ID_Span2.innerText = "Autor: ";
    ID_Span3.innerText = "Abstract: ";
    ID_Span4.innerText = "Copyright: ";
    ID_Span5.innerText = "Filename: ";
    ID_Span6.innerText = "Banner: ";
    ID_Span7.innerText = "Banner Abstract: ";
    ID_Span8.innerText = "Banner MoreInfo: ";
}
</SCRIPT>
</HEAD>
<BODY onload="ButtonUpdate(">

    <t:MEDIA ID="ID_Media"
        SRC="test.asx"
        BEGIN="indefinite"
        TIMEACTION="visibility"
        onend="ButtonUpdate();"
        ontrackchange="AnzeigeUpdate();"
        onmediacomplete="ButtonUpdate();AnzeigeUpdate();"
    >
    </t:MEDIA>

    <SPAN ID="ID_Span1">Titel:</SPAN>
    <BR>
    <SPAN ID="ID_Span2">Autor:</SPAN>
    <BR>
    <SPAN ID="ID_Span3">Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span4">Copyright:</SPAN>
    <BR>
    <SPAN ID="ID_Span5">Filename:</SPAN>
    <BR>
    <SPAN ID="ID_Span6">Banner:</SPAN>
    <BR>
    <SPAN ID="ID_Span7">Banner Abstract:</SPAN>
    <BR>
    <SPAN ID="ID_Span8">Banner MoreInfo:</SPAN>

    <BR>
    <BUTTON ID="ID_Button1"
        onclick="ID_Media.beginElement(); ButtonUpdate();"
    >
        Start
    </BUTTON>
    <BUTTON ID="ID_Button2"
        onclick="ID_Media.playlist.nextTrack();"
    >
        naechster Track
    </BUTTON>
    <BUTTON ID="ID_Button3"
        onclick="ID_Media.playlist.prevTrack();"
    >
        vorhergehender Track
    </BUTTON>
    <BUTTON ID="ID_Button4"
        onclick="ID_Media.endElement(); AnzeigeLoeschen();"
    >
        Stop
    </BUTTON>
</BODY>

```



&lt;/HTML&gt;

.setActive() Fenster als aktiv setzen (also auch für die Eventdurchreichung aktivieren)  
aber **ohne** es zu fokussieren  
und **ohne** es scrollbar zu machen  
siehe Objekt window

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    var ID_Fenster;

    function FensterErzeugen()
    {
        ID_Fenster = window.open( "/test /test.htm",
                                   " ID_Fenster",
                                   "top=10px,left=480px,height=375px,width=200px,resizable=1"
                                   );
        this.focus();
    }

    function ButtonAktivieren()
    {window.parent.ID_Fenster.ID_Button.setActive();}
</SCRIPT>
</HEAD>
<BODY onload="FensterErzeugen();">
    <BUTTON ID="ID_Button" onclick="ButtonAktivieren();">
        Button aktivieren
    </BUTTON>
</BODY>
</HTML>
```

.setAttribute() Wert von vorhandenem Attribut setzen  
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt  
DOM wird nur bei Erzeugung geändert

Beispiel:

```
<HTML>
<HEAD>
<STYLE>
    .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
    // Cachename festlegen
    //     es können diverse Cachenames definiert und somit Versionen von Cache
    //     verwaltet werden
    var FreierCacheName = "InputCache";

    // Cache-Attribut festlegen
    //     es können diverse Attribute definiert und somit Versionen von Input-Daten
    //     verwaltet werden
    var FreiesCacheAttribut = "InputCacheAttribut";

    // zu cachende Daten referenzieren
    var InputDatenObjekt = ID_Formular.ID_Input;

    function InputSichern()
    {
        // ++++++ Zeitstempel ++++++
        var ZeitpunktJetzt = new Date();
        var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

        // Zeitstempel festlegen: Ab jetzt + 20 Minuten
        var Zeitstempel = ZeitpunktJetztInMinuten + 20;

        // und als UTC-Format erzeugen
        var ZeitstempelUTC = Zeitstempel.toUTCString();

        // und Zeitstempel dem Input-Objekt verpassen
        ID_Input.expires = ZeitstempelUTC;

        // ++++++ Daten chachen ++++++
        // aktuelle Daten holen laut Input-Objekt
        var InputDaten = InputDatenObjekt.value;
```



```

        // Attribut instanzieren und mit Daten füllen
        InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

        // und Cache saveen
        InputDatenObjekt.save(FreierCacheName);
    }

    function InputLaden()
    {
        // Cache laden
        InputDatenObjekt.load(FreierCacheName);

        // und Daten zum Attribut laut Sicherung lesen
        var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
    }
</SCRIPT>
</HEAD>
<BODY>
    <FORM ID="ID_Formular">
        <INPUT ID="ID_Input"
            CLASS="user_data_speicher_klasse"
            TYPE="text"
        >
        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

**.setAttributeNode()** Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
    function Hinzufuegen()
    {
        // Attribut mit Wert erzeugen
        var ZeigerAufAttribut = document.createAttribute("title");
        ZeigerAufAttribut.value = "Tooltip-Text ";

        // Attribut als Knoten erzeugen
        var AttributKnoten = ID_Div.setAttributeNode(ZeigerAufAttribut);
    }
</SCRIPT>
</HEAD>
<BODY onload=" Hinzufuegen();">
    <DIV ID="ID_Div">Es wird ein Tooltip-Text hinzugefuegt</DIV>
</BODY>
</HTML>

```

**.setCapture()** Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.

ab IE 5.5

Hinweis: ausschalten per Methode .releaseCapture()  
true Default. Überwachung durch Kinder nicht möglich  
false Überwachung durch Kinder möglich

Beispiel:

```

<BODY onload="ID_Div.setCapture();"
    onclick="document.releaseCapture();"
>
    <DIV ID="ID_Div"
        onmousemove="ID_Textarea.value = event.clientX + event.clientY;"
        onlosecapture="alert(event.srcElement.id + ' hat keine Mausueberwachung mehr');"
    >
        <TEXTAREA ID="ID_Textarea" COLS=2>Test</TEXTAREA>
    </DIV>
</BODY>

```

**.setData()** Clipboard füllen, also Daten dort ablegen  
wenn Clipboard nicht leer so immer anhängen



<code>.setDate()</code>	Tag des Monats in lokaler Zeit setzen wenn Tag > Anzahl Tage im Monat, so erfolgt automatisch Monatswechsel und Korrektur aller Angaben (inklusive Jahr ) Script-Objekt Date Integer 1 bis 31
<code>.setEndPoint()</code>	Textbereichanfang bzw. -ende von 2 Textbereichen synchronisieren per textrange Objekt nur unter Windows 32-Bit
<code>.setExpression()</code>	Wert definieren, der als Ausdruck für die Methode <code>.getExpression()</code> zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert

In nachfolgenden Beispielen wurde aus Platzgründen die Zeichenkette von STYLE umgebrochen, was eigentlich nicht zulässig ist, und es sind nicht alle SPAN kodiert.

Beispiel 1:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE ="JScript">
    function width_init()
    {
        ID_Span.style.setExpression("width",
                                    " trueBlueSpan.style.pixelWidth
                                    + oldYellowSpan.style.pixelWidth
                                    ",
                                    "jscript"
                                    );
    }

    function Berechne()
    {alert(ID_Span.style.getExpression("width");}
</SCRIPT>
</HEAD>
<BODY onload= width_init();>
    <SPAN ID="ID_Span"
        STYLE="background-color:lightgreen;
            width:expression( trueBlueSpan.style.pixelWidth
                            + oldYellowSpan.style.pixelWidth
                            )
            "
    >
    </SPAN>
    <BUTTON onclick= Berechne();>
</BODY>
</HTML>
```

Beispiel 2:

```
ID_Span.style.setExpression("height","document.style.fontSize + 13");
ID_Span.style.setExpression("width","document.body.style.fontSize");
<SPAN ID="ID_Span"
    STYLE="background-color:lightgreen;
        width:expression( trueBlueSpan.style.pixelWidth
                        + oldYellowSpan.style.pixelWidth
                        )
        "
    >
    </SPAN>
```

Beispiel 3 für Sekundenbalken:

```
<HTML>
<HEAD>
<STYLE>
    BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
    var timerID = null;
    var Zahler = 0;
```



```

function Init()
{
    // DIV-Eigenschaften festlegen

    // DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
    ID_Div1.style.setExpression("width","Zahler*10");

    // DIV-Inhalt als Sekudentext, der permanent aktualisiert wird
    ID_Div2.setExpression("innerText","Zahler.toString()");
}

function Uhr()
{
    // Sekunden kumulieren
    Zahler++;

    // und Anzeige neu berechnen
    document.recalc();
}

function Starten()
{
    if (timerID == null)
    {
        // Start-Button nicht aktivierbar machen
        ID_Button1.disabled = true;

        // Stop-Button aktivierbar machen
        ID_Button2.disabled = false;

        // Uhr neu starten
        timerID = setInterval("Uhr()", 1000);
    }
}

function Stoppen()
{
    if (timerID != null)
    {
        clearInterval(timerID);
        timerID = null;
        ID_Button1.disabled = false;
        ID_Button2.disabled = true;
    }
}

function ZurueckSetzen()
{ Zahler = 0; }
</SCRIPT>
</HEAD>
<BODY onload="Init()">
    <DIV ID="ID_Div1" STYLE="background-color:lightblue"></DIV>
    <DIV ID="ID_Div1" STYLE="color:hotpink;font-weight:bold"></DIV>
    <BR>
    <BUTTON ID="ID_Button1"
        onclick="Starten()"
    >
        Start
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button2"
        DISABLED="true"
        onclick="Stoppen()"
    >
        Stop
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button3"
        onclick="ZurueckSetzen()"
    >
        Reset
    </BUTTON>
    <BR>

```



```

        <P          STYLE="width:200;color:white;background-color:gray">
            Sekundenbalken
        </P>
    </BODY>
</HTML>

```

Beispiel 4 für Sound mit Sekundenanzeige:

```

<HTML>
<HEAD>
<SCRIPT>
    // ++++++ globale Variablen, die verändert werden können
    var SoundUrl          = "56sec.mid";
    var SoundDauerInSekunden = 56; // Dauer muss exakt stimmen !

    var PixelBreiteProBalkenErweiterung = 10;

    // ++++++ Browser-Typ ermitteln
    // Dieser Quellcode muss VOR allen anderen Routinen codiert sein, damit zuerst abgearbeitet
    var ns = document.layers ? true : false;
    var ie = document.all ? true : false;

    // ++++++ Routinen der Sekundenzählung
    var SekundenZahler          = 0;
    var SekundenZahlerTimeoutID = null;

    function SekundenZaehlen()    // wird durch Rekursion alle Sekunde neu gestartet
    {
        // Zähler erhöhen
        SekundenZahler++;

        // visuelle Anzeige im Dokument auffrischen, also alle Ausdrücke der Style-Werte
        // neu berechnen und damit alle DIV's neu visualisieren
        document.recalc();
    }

    function SekundenZaehlen_Start()
    {
        // prüfen ob Sekunden zählen nicht bereits läuft
        if (SekundenZahlerTimeoutID == null)
        {
            // nicht aktiv

            // Rekursion starten: SekundenZaehlen() wird permanent alle Sekunde aktiviert
            SekundenZahlerTimeoutID = setInterval("SekundenZaehlen()", 1000);
        }
    }

    function SekundenZaehlen_Stop()
    {
        // prüfen ob Sekunden zählen aktiv ist
        if (SekundenZahlerTimeoutID != null)
        {
            // aktiv, also stoppen
            clearInterval(SekundenZahlerTimeoutID);
            SekundenZahlerTimeoutID = null;
        }
    }

    // ++++++ Routine zur Erzeugung Sound-Objekt
    function SoundObjektErzeugen(SoundFileUrl,SoundFileDauerInSekunden)
    {
        this.SoundFileUrl          = SoundFileUrl;
        this.SoundFileDauerInMillisekundenSekunden = SoundFileDauerInSekunden * 1000;
                                                // Timerzeit für Rekursion
        this.SoundFileBeendet      = true;    // kein Sound aktiv
    }

    // ++++++ Routinen zur Wiedergabe Sound-Objekt
    var SoundTimeoutID=0;

    function SoundAbspielen()
    {
        // prüfen ob Sound nicht bereits aktiv ist

```



```

        if (SoundObjekt.SoundFileBeendet)
        {
            // Anzeige initialisieren
            SekundenAnzeigeInit();

            // Sekundenzähler starten, wobei das Zählen eigenständig und parallel erfolgt
            SekundenZaehlen_Start();

            // Sound erzeugen und sofort starten durch Url-Zuweisung
            ID_BGSound.src=SoundObjekt.SoundFileUrl;
            SoundObjekt.SoundFileBeendet=false;

            // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen
            SoundTimeoutID = setTimeout(
                "SoundAbspielen()",
                SoundObjekt.SoundFileDauerInMillisekundeSekunden
            );
        }
        else
        {
            // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

            // Sound zu Ende
            SoundObjekt.SoundFileBeendet=true;

            // Sekundenzähler stoppen
            SekundenZaehlen_Stop();

            // und Meldung
            var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;
            var TimerUngenauigkeit2 = TimerUngenauigkeit1 / SoundDauerInSekunden;
            alert(
                "Wiedergabe beendet\nUngenauigkeit des Timers = "
                + TimerUngenauigkeit1.toString() + " Sekunden\n"
                + "also " + TimerUngenauigkeit2.toString() + " Ticks pro Sekunde"
            );
        }
    }

// ++++++ Sekunden-Anzeige initialisieren
function SekundenAnzeigeInit()
{
    // ---- Variablen init
    SekundenZahler          = 0;
    SekundenZahlerTimeoutID = null;

    // ---- visuelle Anzeige erzeugen
    // - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
    //      Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
    //      Der Ausdruck liefert den Wert , welcher sofort das Layout der
    //      DIV's beeinflusst.
    //      Jeder Ausdruck besitzt den SekundenZahler als Komponente.
    //      Damit ändert sich der Wert des Ausdruckes.
    //      Für die Neuberechnung des Ausdruckes ist der Aufruf von
    //      document.recal()
    //      nötig.
    //      Dieser Aufruf erfolgt in SekundenZaehlen(), also permanent pro Sekunde.
    //      Damit wird der Style-Wert permanent neu berechnet.
    //      Damit visualisieren sich die DIV's permanent neu.

    // Sekundenbalken in der Style-Eigenschaft width (Breite) mit Ausdruck belegen,
    //      also dynamisch anzeigen
    ID_DIV_Balken.style.setExpression( "width",
                                        "SekundenZahler * PixelBreiteProBalkenErweiterung"
    );

    // Sekundenzähler in der Eigenschaft .innerText mit Ausdruck belegen,
    //      also dynamisch anzeigen
    ID_DIV_SekundenZahler.setExpression("innerText","SekundenZahler.toString()");

    // - - - Messlatte statisch anzeigen
    ID_DIV_MessLatte.style.width = SoundDauerInSekunden * PixelBreiteProBalkenErweiterung;
    ID_DIV_MessLatte.innerText  = "Der Sound dauert "
    + SoundDauerInSekunden.toString()
    + " Sekunden";
}

```





```

    }

    // ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
    if (ie)
    {
        document.write('<BODY></BODY>');

        document.write(    '<BGSOUND ID= "ID_BGSound" LOOP="0">'

        document.write(    '<DIV    ID="ID_DIV_Balken"'
                            +    'STYLE="background-color:lightblue"'
                            + '>'
                            + '</DIV>'
                            + '<BR>'

                                );

        document.write(    '<DIV    ID="ID_DIV_SekundenZahler"'
                            +    'STYLE="color:hotpink;font-weight:bold"'
                            + '>'
                            + '</DIV>'
                            + '<BR>'

                                );

        document.write(    '<DIV    ID="ID_DIV_MessLatte"'
                            +    'STYLE="color:white;background-color:gray"'
                            + '>'
                            + '</DIV>'

                                );

        // ++++++ Sound initialisieren und starten mit Laden des Dokumentes

        // ---- Sound-Objekt erzeugen anhand globaler Variablen
        SoundObjekt = new SoundObjektErzeugen(SoundUrl, SoundDauerInSekunden);

        // ---- Sound-Objekt wiedergeben
        SoundAbspielen(); // meldet wenn Wiedergabe beendet ist
    }
</SCRIPT>
</HEAD>
<BODY>
    <!-- BODY-Teil muss leer bleiben /-->
</BODY>
</HTML>

```

**.setFullYear()**      Jahreszahl vierstellig in lokaler Zeit setzen und optional Monat wie Tag im Monat sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert  
Script-Objekt Date

**.setHomePage()**      Homepage-Lade-Dialogbox aufrufen

**.setHours()**      Stunden in lokaler Zeit setzen und optional Minuten, Sekunden, Millisekunden sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert  
Script-Objekt Date

**.setInterval()**      endlos-periodischer Aufruf eines Codes mit jeweiligem vorherigen Warten in Millisekunden (Timer)  
(getimte Rekursion)  
Der mit setInterval() aufgerufene Code wird zyklisch aktiviert, wobei nach dem ERSTEN Aufruf von setInterval() mit der Folgeanweisung **hinter** setInterval() sofort weitergemacht wird, während die Rekursion parallel läuft. Damit gilt: setInterval() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da sonst diese Anweisungen während der parallelen rekursiven Ereignisüberwachung bereits abgearbeitet werden.

siehe Objekt window

Beispiel 1

String

window.setInterval("EineFunktion()", 5000);

Zeiger

window.setInterval(EineFunktion, 5000); // ohne () kodieren

Beispiel 2

function callback1(){alert("callback1");}

function callback2(){alert("callback2");}



```

function chooseCallback(Nummer)
{
    switch (Nummer)
    {
        case 0: return callback1;
        case 1: return callback2;
        default: return "";
    }
}
var Nummer = 1;
window.setInterval(chooseCallback(Nummer), 5000);

```

## Beispiel 3

```

var SekundenZahler=0;
var timer_id=null;

function ZyklischeAktion()
{
    SekundenZahler++;
    window.status= SekundenZahler + " Sekunden ";

    if ( ( SekundenZahler >= 60 )
        && ( timer_id != null )
    )
    {window.clearInterval(timer_id);}
}

timer_id=window.setInterval(ZyklischeAktion,1000);

```

## Beispiel 4

```

var SekundenZahler=0;
var timer_id=window.setInterval("window.status= SekundenZahler++",1000);

```

## Beispiel 5:

```

var timerID = null;

function timer()
{
    // tue was
}

function starttimer()
{
    if (timerID == null)
    {timerID = setInterval("timer()", 1000);}
}

function stoptimer()
{
    if (timerID != null)
    {
        clearInterval(timerID);
        timerID = null;
    }
}

```

## Beispiel 6 für Sekundenbalken:

```

<HTML>
<HEAD>
<STYLE>
    BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
    var timerID = null;
    var Zahler = 0;

    function Init()
    {
        // DIV-Eigenschaften festlegen

        // DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
        ID_Div1.style.setExpression("width","Zahler *10");
    }

```



```

        // DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
        ID_Div2.setExpression("innerText","Zahler.toString()");
    }

    function Uhr()
    {
        // Sekunden kumulieren
        Zahler++;

        // und Anzeige neu berechnen
        document.recalc();
    }

    function Starten()
    {
        if (timerID == null)
        {
            // Start-Button nicht aktivierbar machen
            ID_Button1.disabled = true;

            // Stop-Button aktivierbar machen
            ID_Button2.disabled = false;

            // Uhr neu starten
            timerID = setInterval("Uhr()", 1000);
        }
    }

    function Stoppen()
    {
        if (timerID != null)
        {
            clearInterval(timerID);
            timerID = null;
            ID_Button1.disabled = false;
            ID_Button2.disabled = true;
        }
    }

    function ZurueckSetzen()
    { Zahler = 0; }
</SCRIPT>
</HEAD>
<BODY onload="Init()">
    <DIV ID="ID_Div1" STYLE="background-color:lightblue" ></DIV>
    <DIV ID="ID_Div1" STYLE="color:hotpink;font-weight:bold"></DIV>
    <BR>
    <BUTTON ID="ID_Button1"
        onclick="Starten()"
    >
        Start
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button2"
        DISABLED="true"
        onclick="Stoppen()"
    >
        Stop
    </BUTTON>
    <BR>
    <BUTTON ID="ID_Button3"
        onclick="ZurueckSetzen()"
    >
        Reset
    </BUTTON>
    <BR>
    <P STYLE="width:200;color:white;background-color:gray">
        Sekundenbalken
    </P>
</BODY>
</HTML>

```

.setMilliseconds()      Millisekunden in lokaler Zeit setzen  
 Script-Objekt Date



.setMinutes()	Minuten in lokaler Zeit setzen und optional Sekunden, Millisekunden sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
.setMonth()	Monat in lokaler Zeit setzen und optional Tag im Monat sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
.setNamedItem()	Attribut hinzufügen anhand Zeiger auf Attribut wenn noch nicht im Feld vorhanden, so Anhängen an das Feldende wenn schon im Feld vorhanden, so überschreiben und Referenz auf das überschriebene Attribut (vor dem Überschreiben) liefern Bsp: Attribut erzeugen document.createAttribute("title"); Hinweis: in HTML können Attributnamen groß oder klein geschrieben werden ab IE 6
Beispiel:	<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;SCRIPT&gt;     function Hinzufuegen()     {         // Attribut mit Wert erzeugen         var ZeigerAufAttribut = document.createAttribute("title");         ZeigerAufAttribut.value = "Tooltip-Text ";          // Attribut als Feldelement anhängen         var ZeigerAufFeld = ID_Div.attributes;         ZeigerAufFeld.setNamedItem(ZeigerAufAttribut);     } &lt;/SCRIPT&gt; &lt;/HEAD&gt; &lt;BODY onload="Hinzufuegen();"&gt;     &lt;DIV ID="ID_Div"&gt;Es wird ein Tooltip-Text hinzugefuegt&lt;/DIV&gt; &lt;/BODY&gt; &lt;/HTML&gt; </pre>
.setSeconds()	Sekunden in lokaler Zeit setzen und optional Millisekunden sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
.setTime()	Zeitwert in Weltzeit setzen, auch vor 1970 Script-Objekt Date Anzahl der Millisekunden relativ zum 01.01.1970 0 Uhr Weltzeit <b>wenn &lt; 0 so vor obigem Datum</b>
.setTimeout()	einmaliger und somit <b>nicht periodischer</b> Aufruf eines Codes mit genau einem vorherigen Warten in Millisekunden (Timer) Der mit setTimeout() aufgerufene Code wird <b>nicht zyklisch</b> aktiviert, wobei nach dem Aufruf von setTimeout() mit der Folgeanweisung hinter setTimeout() sofort weitergemacht wird. Damit gilt: setTimeout() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da diese Anweisungen bereits während der Ereignisüberwachung abgearbeitet werden. siehe Objekt window

Beispiel 1  
window.setTimeout("alert('Hallo')", 1000);

Beispiel 2  
var Text = "Hallo ";  
window.setTimeout( "alert(" + Text  
+ ")",  
1000  
);

Beispiel 3  

```

<SCRIPT>
function Start(ZeigerAufObjekt)
{window.setTimeout("Kode(" + ZeigerAufObjekt.id + ")", 3000);}

function Kode(ID)
{
    var Zeiger = eval(ID);
    Zeiger.style.display="none";
}

```



```
</SCRIPT>
<INPUT TYPE=button VALUE="Unsichtbar in 3 Sekunden"
      ID="ID_Button" onclick=" Start(this)"
```

## Beispiel 4

```
var SekundenZahler=0;
var timeout_id=null;

function ZyklischeAktion()
{
    SekundenZahler++;
    window.status= SekundenZahler + " Sekunden ";

    if (      ( SekundenZahler >= 60 )
        && ( timeou_id != null)
        )
    {window.clearTimeout(timeout_id);}
}

timeout_id=window.setTimeout(ZyklischeAktion,1000);
```

## Beispiel 5

```
var timeout_id=window.setTimeout("window.status= 'Es ist 1 Sekunde vergangen !'",1000);
```

## Beispiel 6 für Sound mit Sekundenanzeige:

```
<HTML>
<HEAD>
<SCRIPT>

// ++++++ globale Variablen, die verändert werden können
var SoundUrl          = "56sec.mid";
var SoundDauerInSekunden = 56; // Dauer muss exakt stimmen !

var PixelBreiteProBalkenErweiterung = 10;

// ++++++ Browser-Typ ermitteln
// Dieser Quellcode muss VOR allen anderen Routinen codiert sein, damit zuerst abgearbeitet
var ns = document.layers ? true : false;
var ie = document.all ? true : false;

// ++++++ Routinen der Sekundenzählung
var SekundenZahler          = 0;
var SekundenZahlerTimeoutID = null;

function SekundenZaehlen()    // wird durch Rekursion alle Sekunde neu gestartet
{
    // Zähler erhöhen
    SekundenZahler++;

    // visuelle Anzeige im Dokument auffrischen, also alle Audrucke der Style-Werte
    // neu berechnen und damit alle DIV's neu visualisieren
    document.recalc();
}

function SekundenZaehlen_Start()
{
    // prüfen ob Sekundenzählen nicht bereits läuft
    if (SekundenZahlerTimeoutID == null)
    {
        // nicht aktiv

        // Rekursion starten: SekundenZaehlen() wird permanent alle Sekunde aktiviert
        SekundenZahlerTimeoutID = setInterval("SekundenZaehlen()", 1000);
    }
}

function SekundenZaehlen_Stop()
{
    // prüfen ob Sekundenzählen aktiv ist
    if (SekundenZahlerTimeoutID != null)
    {
        // aktiv, also stoppen
        clearInterval(SekundenZahlerTimeoutID);
        SekundenZahlerTimeoutID = null;
    }
}
```



```

    }
}

// ++++++ Routine zur Erzeugung Sound-Objekt
function SoundObjektErzeugen(SoundFileUrl,SoundFileDauerInSekunden)
{
    this.SoundFileUrl = SoundFileUrl;
    this.SoundFileDauerInMillisekundeSekunden = SoundFileDauerInSekunden * 1000;
    // Timerzeit für Rekursion
    this.SoundFileBeendet = true; // kein Sound aktiv
}

// ++++++ Routinen zur Wiedergabe Sound-Objekt
var SoundTimeoutID=0;

function SoundAbspielen()
{
    // prüfen ob Sound nicht bereits aktiv ist
    if (SoundObjekt.SoundFileBeendet)
    {
        // Anzeige initialisieren
        SekundenAnzeigeInit();

        // Sekundenzähler starten, wobei das Zählen eigenständig und parallel erfolgt
        SekundenZaehlen_Start();

        // Sound erzeugen und sofort starten durch Url-Zuweisung
        ID_BGSound.src=SoundObjekt.SoundFileUrl ;
        SoundObjekt.SoundFileBeendet=false;

        // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen
        SoundTimeoutID = setTimeout(
            "SoundAbspielen()",
            SoundObjekt.SoundFileDauerInMillisekundeSekunden
        );
    }
    else
    {
        // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

        // und Meldung
        var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;
        var TimerUngenauigkeit2 = TimerUngenauigkeit1 / SoundDauerInSekunden;
        alert(
            "Wiedergabe beendet\nUngenauigkeit des Timers = "
            + TimerUngenauigkeit1.toString()+ " Sekunden\n"
            + "also " + TimerUngenauigkeit2.toString()+ " Ticks pro Sekunde"
        );
    }
}

// ++++++ Sekunden-Anzeige initialisieren
function SekundenAnzeigeInit()
{
    // ---- Variablen init
    SekundenZahler = 0;
    SekundenZahlerTimeoutID = null;

    // ---- visuelle Anzeige erzeugen
    // - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
    // Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
    // Der Ausdruck liefert den Wert , welcher sofort das Layout der
    // DIV's beeinflusst.
    // Jeder Ausdruck besitzt den SekundenZahler als Komponente.
    // Damit ändert sich der Wert des Ausdruckes.
    // Für die Neuberechnung des Ausdruckes ist der Aufruf von
    // document.recal()
    // nötig.
    // Dieser Aufruf erfolgt in SekundenZaehlen(), also permanent pro Sekunde.

```



```

//          Damit wird der Style-Wert permanent neu berechnet.
//          Damit visualisieren sich die DIV's permanent neu.

// Sekundenbalken in der Style-Eigenschaft width (Breite) mit Ausdruck belegen,
//          also dynamisch anzeigen
ID_DIV_Balken.style.setExpression( "width",
                                   "SekundenZahler * PixelBreiteProBalkenErweiterung"
                                   );

// Sekundenzähler in der Eigenschaft .innerText mit Ausdruck belegen,
//          also dynamisch anzeigen
ID_DIV_SekundenZahler.setExpression("innerText","SekundenZahler.toString()");

// - - - Messlatte statisch anzeigen
ID_DIV_MessLatte.style.width = SoundDauerInSekunden * PixelBreiteProBalkenErweiterung;
ID_DIV_MessLatte.innerText  = "Der Sound dauert "
                               + SoundDauerInSekunden.toString()
                               + " Sekunden";
}

// ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
if (ie)
{
    document.write('<BODY></BODY>');

    document.write( ' <BGSOUND ID= "ID_BGSound" LOOP="0">'

    document.write( ' <DIV ID="ID_DIV_Balken"'
                    + 'STYLE="background-color:lightblue"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_SekundenZahler"'
                    + 'STYLE="color:hotpink;font-weight:bold"'
                    + '>'
                    + '</DIV>'
                    + '<BR>'

                    );

    document.write( ' <DIV ID="ID_DIV_MessLatte"'
                    + 'STYLE="color:white;background-color:gray"'
                    + '>'
                    + '</DIV>'

                    );

    // ++++++ Sound initialisieren und starten mit Laden des Dokumentes

    // ----- Sound-Objekt erzeugen anhand globaler Variablen
    SoundObjekt = new SoundObjektErzeugen(SoundUrl, SoundDauerInSekunden);

    // ----- Sound-Objekt wiedergeben
    SoundAbspielen(); // meldet wenn Wiedergabe beendet ist
}
</SCRIPT>
</HEAD>
<BODY>
    <!-- BODY-Teil muss leer bleiben!-->
</BODY>
</HTML>

```

.setUTCDate()      Tag des Monats in Weltzeit setzen  
wenn Tag > Anzahl Tage im Monat, so erfolgt automatisch Monatswechsel und Korrektur aller  
Angaben (inklusive Jahr )  
Script-Objekt Date

.setUTCFullYear()      Jahreszahl vierstellig in Weltzeit setzen und optional Monat wie Tag im Monat  
sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert  
Script-Objekt Date

.setUTCHours()      Stunden in Weltzeit setzen und optional Minuten, Sekunden, Millisekunden  
sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert  
Script-Objekt Date



<code>.setUTCMilliseconds()</code>	Millisekunden in Weltzeit setzen Script-Objekt Date
<code>.setUTCMinutes()</code>	Minuten in Weltzeit setzen und optional Sekunden, Millisekunden sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
<code>.setUTCMonth()</code>	Monat in Weltzeit setzen und optional Tag im Monat sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
<code>.setUTCSeconds()</code>	Sekunden in Weltzeit setzen und optional Millisekunden sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert Script-Objekt Date
<code>.setYear()</code>	deprecated
<code>.shift()</code>	erstes Feldelement liefern und danach aus dem Feld entfernen Feld hat die Objektklasse JScript-Objekt Array bzw. Script-Objekt Array ab IE 5.5 und NS 6.x siehe auch <code>.pop()</code>
Beispiel:	<pre>var Feld = new Array(0,1,2,3,4); alert (Feld.shift()); // 0 alert(Feld.join()); // "1,2,3,4"</pre>
<code>.show()</code>	ein per <code>.createPopup()</code> instanziiertes Popup-Fenster anzeigen Hinweis: Es kann immer nur genau 1 Popup-Fenster angezeigt werden. Das Popup-Fenster wird automatisch geschlossen, wenn ein anderes Objekt den Focus erhält bzw. aktiv wird, z.B. durch Klick des Users außerhalb des PopUpfensters ändert Wert der Eigenschaft <code>.isOpen</code> siehe Objekt <code>window.popup</code>

Beispiel für diverse Meldungsfenster per **jeweiligem** PopUp-Fenster ( es kann immer nur genau 1 PopUp-Fenster angezeigt werden):

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">

    var PopUpFensterFeld = new Array();

    // nachfolgende Felder beschreiben die PopUp-Fenster und müssen
    // identische Anzahl der Feldelemente haben
    // Feld-Index ist die Nummer des PopUp-Fensters

    var PopUpFenster_RahmenStyle_Feld = new Array
    (
        "solid black 4px",
        "solid blue 3px",
        "solid green 2px"
    );

    var PopUpFenster_HintergrundFarbe_Feld = new Array
    (
        "yellow",
        "gray",
        "orange"
    );

    var PopUpFenster_Inhalt_Feld = new Array
    (
        "<B>Inhalt PopUpFenster 0<B>",
        "Inhalt PopUpFenster 1",
        "<TT>Inhalt PopUpFenster 2<TT>"
    );

    var PopUpFenster_X_Koordinate_Feld = new Array
    (
        100,
        150,
        200
```





```

    );

    var PopUpFenster_Y_Koordinate_Feld = new Array
    (
        150,
        200,
        250
    );

    var PopUpFenster_Breite_Koordinate_Feld = new Array
    (
        80,
        140,
        200
    );

    var PopUpFenster_Hoehe_Koordinate_Feld = new Array
    (
        100,
        140,
        180
    );

    var AnzahlPopUpFenster = PopUpFenster_Hoehe_Koordinate_Feld.length;

    function PopupFensterInstanzieren(NummerDesPopUpFensters, ZeigerAufElternFenster)
        // NummerDesPopUpFensters ab 0
    {
        PopUpFensterFeld[NummerDesPopUpFensters] =
            ZeigerAufElternFenster.createPopup();
    }

    function PopupFensterFuellen(NummerDesPopUpFensters)
    {
        // Body des Popup-Fensters gestalten
        var PopupFenster_Body =
            PopUpFensterFeld[NummerDesPopUpFensters].document.body;

        PopupFenster_Body.style.backgroundColor =
            PopUpFenster_HintergrundFarbe_Feld[NummerDesPopUpFensters];

        PopupFenster_Body.style.border =
            PopUpFenster_RahmenStyle_Feld[NummerDesPopUpFensters];

        PopupFenster_Body.innerHTML =
            PopUpFenster_Inhalt_Feld[NummerDesPopUpFensters];
    }

    function PopupFensterAnzeigen(NummerDesPopUpFensters,
        ObjektZuDemPopUpFensterRelativPositioniertIst
    )
    {
        // Popup-Fenster anzeigen und damit öffnen
        PopUpFensterFeld[NummerDesPopUpFensters].show(
            PopUpFenster_X_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Y_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Breite_Koordinate_Feld[NummerDesPopUpFensters],
            PopUpFenster_Hoehe_Koordinate_Feld[NummerDesPopUpFensters],
            ObjektZuDemPopUpFensterRelativPositioniertIst
        );
    }

    function PopupFensterSchliessen(NummerDesPopUpFensters)
    {
        var Zeiger = PopUpFensterFeld[NummerDesPopUpFensters];

        if (Zeiger.isOpen)
        { Zeiger.hide(); }
    }

    function Init()
    {
        // PopUpFenster instanzieren im aktuellen Fenster (window)

```



```

        for (var i = 0 ; i < AnzahlPopUpFenster; i++)
        {
            PopupFensterInstanzieren(i, window);
            PopupFensterFuellen(i);
        }
    }
</SCRIPT>
</HEAD>
<BODY onload="Init();">
    Durch Klick ausserhalb des Popup-Fensters wird dieses auch automatisch geschlossen.
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 anzeigen"
        onclick=" PopupFensterAnzeigen(0, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 anzeigen"
        onclick=" PopupFensterAnzeigen(1, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 anzeigen"
        onclick=" PopupFensterAnzeigen(2, document.body);"
    >
    <BR>
    <INPUT TYPE=button
        VALUE="PopUpFenster 0 schliessen"
        onclick=" PopupFensterSchliessen(0);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 1 schliessen"
        onclick=" PopupFensterSchliessen(1);"
    >
    <INPUT TYPE=button
        VALUE="PopUpFenster 2 schliessen"
        onclick=" PopupFensterSchliessen(2);"
    >
</BODY>
</HTML>

```

.ShowBrowserUI()      öffnen einer IE-Dialogbox (UI = User-Interface) bezüglich  
                               Spracheinstellungen  
                               oder      Favoritenverwaltung  
                               oder      Sicherheitseinstellungen  
 Inwieweit eine **geöffnete** Dialog-Box dann per Windows Script Host z.B. per VBScript programmierbar ist,  
 wird hier nicht betrachtet.  
 Ein User, dem diese Boxen durch eine Webseite geöffnet werden, sollte stets misstrauisch ein.  
 siehe Objekt window.external

Beispiel:

```
<BUTTON onclick="window.external.ShowBrowserUI('LanguageDialog', null)">Spracheinstellungen</BUTTON>
```

.showHelp()      Helpdatei anzeigen (\*.chm und \*.htm)  
 siehe Objekt window

.showModalDialog()      Modales Dialog-Fenster erzeugen und anzeigen (modaler Dialog) sowie automatisch focussieren  
 siehe Objekt window

Beispiel 1:

```

<SCRIPT>
function ErmittleZufallsZahl(Faktor)
{return parseInt(Math.random() * Faktor);}

function BoxHoeheErmittleIn()
{
    var OptionsFeld = ID_Formular.ID_Select.options;
    var OptionsFeldIndex = ID_Formular.ID_Select.selectedIndex;
    var OptionsWert = OptionsFeld [OptionsFeldIndex].text;

    // auf Auswahl " Zufallshoehe" prüfen
    if (OptionsWert.indexOf("Zufallshoehe") > -1 )
    {OptionsWert = ErmittleZufallsZahl(document.body.clientHeight);}

    // Features für Boxöffnen ermitteln
    var BoxHoeheFeatures ="dialogHeight:" + OptionsWert + "px;";

    // und liefern

```



```

        return BoxHoeheFeatures;
    }

    function BoxOeffnen()
    {
        var BoxHoeheFeatures= BoxHoeheErmitteln();

        window.showModalDialog( "test.htm",
                                "",
                                BoxHoeheFeatures
                                );
    }
</SCRIPT>
<FORM NAME="ID_Formular">
    wachle Boxhoehe in Pixel
    <SELECT NAME="ID_Select">
        <OPTION>Zufallshoehe
        <OPTION>150
        <OPTION>200
        <OPTION>250
        <OPTION>300
    </SELECT>
    oeffne Modal Dialog Box
    <INPUT TYPE="button" VALUE="Box oeffnen" onclick="BoxOeffnen()">
</FORM>

```

Beispiel 2:

```

<HTML>
<HEAD>
<SCRIPT>
    function Anzeige()
    {
        // Zeiger auf die Formularelemente holen
        var FormularElemente = ID_Formular.elements;

        // Formulardaten in einem privaten Objekt kapseln als Argument für modalen Dialog
        var FormularDaten = new Object();
        FormularDaten.firstName = FormularElemente.ID_Input1.value;
        FormularDaten.lastName = FormularElemente.ID_Input2.value;

        // Fenster als modalen Dialog anzeigen und dabei Eigenschaft .dialogArguments füllen
        // Mit der Übergabe der Daten erfolgt das öffnen des neuen Fenster, das
        // sich auf die namensgleichen Eigenschaften von FormularDaten
        // beruft, deren Bezeichner mit in den Argumenten übergeben werden
        window.showModalDialog( "test.htm",
                                FormularDaten,
                                "dialogHeight:300px; dialogLeft:200px;"
                                );
    }
</SCRIPT>
</HEAD>
<BODY>
    <FORM ID= "ID_Formular">
        Vorname:
        <INPUT TYPE="text" NAME="ID_Input1" VALUE="Vorname">
        <BR>
        Nachname:
        <INPUT TYPE="text" NAME="ID_Input2" VALUE="Nachname">
    </FORM>
    <BR>
    <BUTTON onclick="Anzeige();" >Formulardaten im modalen Dialog anzeigen</BUTTON>
</BODY>
</HTML>

```

test.htm enthält:

```

<HTML>
<HEAD>
<SCRIPT>
    function Anzeigen()
    {
        var DialogArgumente = window.dialogArguments;

        // Es müssen namensidentische Bezeichner der Eigenschaften von

```



```

//      DialogArgumente verwendet werden:
//      firstName und lastName werden in der
//      aufrufenden Webseite definiert
//      und müssen hier ebenfalls verwendet werden
document.writeln("Vorname = " + DialogArgumente.firstName);
document.write("Nachname = " + DialogArgumente.lastName);
}
</SCRIPT>
</HEAD>
<BODY onload="Anzeigen();">
</BODY>
</HTML>

```

.showModelessDialog() nicht-modales Fenster erzeugen, aber anzeigen nur, wenn Focus geändert wird  
 verwendbar für Desing von Menüs  
 Tooltips  
 siehe Objekt window

Beispiel

```

<HTML>
<HEAD>
<SCRIPT>
  var Vorname=""; // muss global sein da in Test.htm benutzt

  function VornameAnzeigen()
  {
    showModelessDialog( "Test.htm", // Url der Dialog-Box
                        window,
                        "status:false;dialogWidth:300px;dialogHeight:300px"
                      );
  }

  function Vorname.Aktualisieren() // Funktion wird in Test.htm aufgerufen
  { ID_Span.innerText = Vorname;}

</SCRIPT>
</HEAD>
<BODY>
  <P> aktueller Vorname ist :
    <SPAN ID="ID_Span"
      STYLE="color:red;font-size:24">
      unbekannt
    </SPAN>
  </P>
  <INPUT TYPE="button"
    VALUE="öffnen der Modeless Dialog Box"
    onclick="VornameAnzeigen()">

</BODY>
</HTML>

```

**Kode für *Test.htm*, also dem Inhalt der Box**

```

<HTML>
<HEAD>
<TITLE>Test.htm</TITLE>
<SCRIPT>
  function VornameAktuellAnzeigen()
  {
    var DialogArgumente = dialogArguments;
    DialogArgumente.Vorname = ID_Input.value;
    DialogArgumente.Vorname.Aktualisieren();
    // Aufruf einer Funktion aus
    // Elterndokument
  }

  function Init()
  {
    var DialogArgumente = dialogArguments;
    DialogArgumente.Vorname = "unbekannt";
    DialogArgumente.Vorname.Aktualisieren();
    // Aufruf einer Funktion aus
    // Elterndokument
  }

</SCRIPT>
</HEAD>
<BODY>

```



```

<LABEL FOR="ID_Input" ACCESSKEY="v">
    Gib den
    <SPAN STYLE="text-decoration:underline">
        V
    </SPAN>ornamen ein :
</LABEL>
<INPUT ID= "ID_Input">
<BR><BR>
<INPUT VALUE="Anzeige aktueller Vorname und Box offen lassen"
    TYPE=button
    onclick=" VornameAktuellAnzeigen();">

<INPUT VALUE=" Anzeige aktueller Vorname und Box schliessen"
    TYPE=button
    onclick=" VornameAktuellAnzeigen();window.close();">

<INPUT VALUE="Init"
    TYPE=button
    onclick=" Init();window.close();"
>
</BODY>
</HTML>

```

.simpleTimeToSegmentTime() Wert der Simple-Timeline eines Elementes in den korrespondierenden Wert der Segment-Timeline des Elementes konvertieren  
 siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```

<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    window.onload = Rekursion;    // ohne () kodieren

    function Rekursion()
    { window.setInterval(Anzeige, 100);}

    function Anzeige()
    {
        ID_Span2.innerHTML = " simpleTimeToSegmentTime: "
            + (ID_Animation. simpleTimeToSegmentTime (
                ID_div.currTimeState.activeTime
            )
        );
        ID_Span3.innerHTML = " segmentTime: "
            + (ID_Animation.currTimeState.segmentTime);
    }
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="ID_Span1"
    CLASS="time_line_klasse"
    DUR="1"
    REPEATCOUNT="indefinite"
    onrepeat="innerText=parseInt(document.body.currTimeState.activeTime);"
>
    0
</SPAN>
<DIV ID="ID_Div"
    CLASS="time_line_klasse"
    STYLE="position:absolute; left:50px; top:250px; width:100px; height:100px;"
>
</DIV>
<t:ANIMATEMOTION ID="ID_Animation"
    TARGETELEMENT="ID_div"
    TO="250,0"
    DUR="3"
    AUTOREVERSE="true"
>
</t:ANIMATEMOTION>
<SPAN ID="ID_Span2">
    simpleTimeToSegmentTime:

```



```

</SPAN>
<BR>
<SPAN ID="ID_Span3">
    segmentTime:
</SPAN>
</BODY>
</HTML>

```

`.sin()` Sinus im Bogenmass  
liefert Wert von -1 bis +1  
siehe Script-Objekt Math

`.Skip()` in der offenen Datei die nächsten Zeichen überlesen (egal ob newline dabei ist oder nicht)  
verändert den Satzzeiger

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.Skip(1);
alert(DateiOffen.Read(1));
DateiOffen.Close();

```

`.SkipLine()` in der offenen Datei die nächsten Zeilen überlesen  
verändert den Satzzeiger  
sinnvoll nur verwendbar, wenn mindestens 1 newline-Zeichen in der Datei auftaucht

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("Test");
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.SkipLine(1);
alert(DateiOffen.ReadLine());
DateiOffen.Close();

```

`.slice()` Elementefolge aus Quellfeld als neues Feld (Zielfeld) liefern  
nachträgliche Änderung der Elementanzahl im Quellfelde wirken sich nicht auf Zielfeld aus  
(keine dynamische Verkettung)  
wenn Quellfeld **nicht** String **und nicht** Number enthält:  
neues Feld hat Feldelemente als Zeiger auf die Elemente also nicht als Wertkopie  
Werte-Änderung im Quellfeld bewirkt sofortige Änderung im Zielfeld, da identische Zeiger  
vorliegen  
wenn Quellfeld Strings oder Number enthält:  
neues Feld hat Feldelemente als Wertkopien aus dem Quellfeld  
Werte-Änderung im Quellfeld bewirkt keine Änderung im Zielfeld, da nicht identische Zeiger  
vorliegen  
Felder haben die Objektklasse Script-Objekt Array

Beispiel:

```

var Feld1 = new Array(0,1,2,3,4);
var Feld2 = Feld1.slice(0,1);
alert(Feld2.join()); // "0"
var Feld3 = Feld1.slice(1);
alert(Feld3.join()); // "1,2,3,4"

```

Beispiel:

```

var QuellFeld=new Array("a","b","c")
var ZielFeld=QuellFeld.slice(0,2) // Zielfeld enthält Wertkopien also ["a","b"]

```

Beispiel:

```

var Variable1="Hallo ";
var Variable2="Du";
var Variable3="!";

var QuellFeld=new Array(Variable1,Variable2,Variable3);
var ZielFeld=QuellFeld.slice(0,-1) // 2 verwendet, denn Länge 3 minus 1 ist 2
// Zielfeld enthält Wertkopien also [ Zeiger_Auf_Variable1,
// Zeiger_Auf_Variable2
// ]

```

`.slice()` Teilkette aus String oder Stringliteral liefern als neuen String  
siehe Script-Objekt String



Beispiel:

```
var StringLiteral = "StringLiteral";
StringLiteral.length      liefert 13
StringLiteral.slice(3,-5) liefert "ingLit" // 13 + (-5) ergibt 8
StringLiteral.slice(3,5)  liefert "ing"
"StringLiteral".slice(3,5) liefert "ing"
```

`.small()` HTML-Tag <SMALL> erzeugen in der Form <SMALL>text</SMALL>  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral = "Text der SMALL wird"
var HTML_Kette = StringLiteral.small();
HTML_Kette = "Text der SMALL wird".small();
```

entspricht <SMALL>Text der SMALL wird</SMALL>

```
eval(HTML_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);
```

`.sort()` Feldelemente physisch sortieren  
sind zwei direkt zusammenliegende Feldelemente wertmäßig identisch, so werden deren Positionen im Feld nicht geändert  
ein Feldelement mit Wert undefined landet am Feldende:  
Liegen mehrere Feldelemente mit Wert undefined vor, so landen diese in der Reihenfolge vor der Sortierung am Ende des Feldes nach der Sortierung  
Standardsortierung : laut ASCII-Zeichensatz und aufsteigend  
Feld hat die Objektklasse Script-Objekt Array

Beispiel:

```
var Feld1 = new Array("4","3","2","1","0");
Feld1.sort(); // Feld1 ist ungültig
alert(Feld1.join()); // "0,1,2,3,4"
```

Beispiel für numerische Feldelemente:

```
function Sortiere(Arg1, Arg2)
{return Arg1 - Arg2;} // Arg1 > Arg2 so positiver Wert
// Arg1 < Arg2 so negativer Wert
// Arg1 == Arg2 so 0
zeiger_auf_feld.sort(Sortiere);
```

`.splice()` Feldelemente-Folge aus dem Feld entfernen ab Indexposition  
optional neue Objekte in das Feld einfügen ab Indexposition  
alle entfernte Feldelemente in einem neuen Feld liefern in der Reihenfolge des Entfernen  
Felder haben die Objektklasse Script-Objekt Array

Beispiel:

```
var Feld1 = new Array(0,0,1,2,3,4);
var Feld2 = Feld1.splice(0,1);
alert(Feld2.join()); // "0,1,2,3,4"
var Feld3 = Feld1.splice(0,2,-1,0);
alert(Feld3.join()); // "-1,0,1,2,3,4"
```

`.split()` String bzw. Stringliteral zerlegen und als Feld der Teilketten liefern  
Zerlegung von links nach rechts  
Feldelemente-Folge wie Zerlegungsfolge  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral = "StringLiteral";
var Feld = StringLiteral.split("r", 2); // Feld enthält 2 Elemente mit "St" und "ingLiteral"
Feld = StringLiteral.split("r", 1); // Feld enthält 1 Elemente mit "St"
Feld = StringLiteral.split("r"); // Feld enthält 3 Elemente mit "St" und "ingLite" und "al"
Feld = "StringLiteral".split("r"); // Feld enthält 3 Elemente mit "St" und "ingLite" und "al"
```

`.sqrt()` Quadratwurzel ziehen  
siehe Script-Objekt Math

`.start()` Start der Scrollaktion eines marquee Objektes  
Anzahl der Scrollaktionen laut Eigenschaft .loop  
erzeugt **nicht** das Ereignis onstart  
Hinweis: Eigenschaften .scrollLeft und .scrollTop sind **nicht** beschreibbar, solange eine Scrollaktion aktiv ist.

`.startDownload()` Start des Download

Beispiel für Funktion:



```
function Anzeige( Kette )
{
    var Feld = Kette.split("\n");
    var FeldLaenge = Feld.length;

    if (FeldLaenge > 0)
    {
        for ( var Index=0; Index < FeldLaenge; Index++)
        {alert("Index = " + Index + ' Teil = ' + Feld[Index]);}
    }
}
```

Beispiel:

```
<HTML XMLNS:IE>
<HEAD>
<SCRIPT>
    function NachDemDownload(DownLoadedFileContent)
    {alert (DownLoadedFileContent);}
</SCRIPT>
</HEAD>
<BODY>
<IE:Download ID="ID_IETag" STYLE="behavior:url(#default#download)" >
<P>
Click
<A HREF="javascript:ID_IETag.startDownload('download.htm', NachDemDownload)">
hier
</A>
zum Start des Downloads dieser Seite.
</BODY>
</HTML>
```

- .stop()** Stopp der aktuellen und laut Eigenschaft .loop noch offener Scrollaktionen eines marquee Objektes erzeugt **nicht** das Ereignis onstop  
Hinweis: Eigenschaften .scrollLeft und .scrollTop sind **nicht** beschreibbar, solange eine Scrollaktion aktiv ist.
- .stop()** aktive Wiedergabe stoppen, also beenden  
verändert Eigenschaft .playState  
Es kann zu jedem Zeitpunkt nur genau 1 Media-Datei wiedergegeben werden, es sei denn, man öffnet mehrere Browser-Instanzen.  
Die Wiedergabe ist nicht möglich, wenn der User in der Media Bar per Links navigiert, da damit die
- aktuelle** Url in der Media Bar geändert wird gegenüber der Url der gerade wiedergegebenen Media-Datei.  
Eine aktive Wiedergabe wird durch HTML-Navigation sofort gestoppt.  
Achtung: Nur wenn eine Media-Datei gerade wiedergegeben wird, dann sind per Script folgende Methoden nutzbar:  
    .getItemInfo()  
    .getAttributeName()  
Eigenschaften nutzbar:  
    .attributeCount  
Es ist also vorher die Eigenschaft .playState auf Werte > 10 zu prüfen.  
Achtung: Wird die Methode .stop() innerhalb von 10 Sekunden nach dem ERSTEN Aufruf erneut aufgerufen, dann wird eine Standard-Seite in das Medien-Fenster geladen.  
siehe Methoden .playURL() .playNext() und Eigenschaft .playState  
siehe Behavior .style.mediaBar
- .strike()** HTML-Tag <STRIKE> erzeugen in der Form <STRIKE>text</STRIKE>  
siehe Script-Objekt String
- Beispiel:
- ```
var StringLiteral    ="Text der STRIKE wird"
var HTML_Kette      = StringLiteral.strike();
HTML_Kette           = "Text der STRIKE wird".strike();

entspricht <STRIKE>Text der STRIKE wird</STRIKE>
```
- eval(HTML\_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
document.write(HTML\_Kette);
- String()** konvertiert eine Instanz zu String  
ist identisch mit Methode .toString()
- .sub()** HTML-Tag <SUB> erzeugen in der Form <SUB>text</SUB>  
siehe Script-Objekt String





Beispiel:

```
var StringLiteral    ="Text der SUB wird"  
var HTML_Kette      = StringLiteral.sub();  
HTML_Kette          ="Text der SUB wird".sub();
```

entspricht <SUB>Text der SUB wird</SUB>

```
eval(HTML_Kette);           erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann
document.write(HTML_Kette);
```

|                            |                                                                                                                                                             |                                               |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <code>.submit()</code>     | löst für Formular das Event onsubmit aus, dessen Eventhandler die Submit-Aktion beinhaltet, die gestartet wird VOR dem Senden der Elemente (Browser sendet) |                                               |
| Eventhandler muss liefern: | <code>return true;</code>                                                                                                                                   | für Ausführung des Submit durch Browser       |
|                            | <code>return false;</code>                                                                                                                                  | für Nicht-Ausführung des Submit durch Browser |

Beispiel: Man beachte, dass NAME-Attribut für alle zu sendenden bzw. rückzusetzenden Elemente **Pflicht** ist !!

```
<HTML>
<HEAD>
<SCRIPT>
    function EventHandler_AktionVorSubmit()
    {
        // ein Hinweis im Textarea anzeigen
        ID_Textarea.value += "Formular senden ???";}

        // wirklich senden ???
        return( confirm("Wirklich senden ?"));
        // true so Submit durch Browser ausführen lassen
        // false so kein Submit durch Browser
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <FORM NAME="Formular" ACTION=" ..." METHOD=" "
            onsubmit="EventHandler_AktionVorSubmit();">
                <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
                <BR>
                <BUTTON onclick="form.submit();">OnSubmit auslösen</BUTTON>
                <BR><BR>
                <INPUT TYPE="submit" VALUE="oder hiermit senden"
                    onclick="form.submit()"
                >
            </FORM>
        </DIV>
        <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
    </BODY>
</HTML>
```

<code>.substr()</code>	Teilkette aus String oder Stringliteral liefern als neuen String siehe Script-Objekt String
------------------------	------------------------------------------------------------------------------------------------

<code>.substring()</code>	Teilkette aus String oder Stringliteral liefern als neuen String siehe Script-Objekt String
---------------------------	------------------------------------------------------------------------------------------------

Beispiel:

```
var StringLiteral="StringLiteral";
StringLiteral.substring(3,3) liefert ""
StringLiteral.substring(3,5) liefert "in"
"StringLiteral".substring(3,5) liefert "in"
```

Beispiel für Erzeugung von Vornull mit Vorzeichen bei Ziffern-Zeichenkette, die numerisch  $<1$  bzw.  $>-1$  ist

```
function vornull(zeichenkette)
{
    var funktionswert=zeichenkette;

    if (      (zeichenkette.substring(0, 1) == ",")
        ||    (zeichenkette.substring(0, 1) == ".")
        )
    {funktionswert = "0" + zeichenkette }

    if (      (zeichenkette.substring(0, 2) == "-.")
        ||    (zeichenkette.substring(0, 2) == "-.")
        )
    {funktionswert = "0" + zeichenkette }
```



```

    { funktionswert = "-0" + zeichenkette.substring(1)}

    return funktionswert;
}

```

**.substringData()** Teilkette aus einem Objekt lesen

**.sup()** HTML-Tag <SUP> erzeugen in der Form <SUP>text</SUP>  
siehe Script-Objekt String

Beispiel:

```

var StringLiteral    ="Text der SUP wird"
var HTML_Kette       = StringLiteral.sup();
HTML_Kette           = "Text der SUP wird".sup();

```

entspricht <SUP>Text der SUB wird</SUP>

eval(HTML\_Kette); erzeugt Fehler, da HTML-Code von eval nicht ausgeführt werden kann  
document.write(HTML\_Kette);

**.swapNode()** Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

Beispiel:

```

<SCRIPT>
    function Tauschen()
    {Liste.children(0).swapNode(Liste.children(1)); }
</SCRIPT>
<UL ID = Liste>
    <LI>Listeneintrag 1
    <LI>Listeneintrag 2
    <LI>Listeneintrag 3
    <LI>Listeneintrag 4
</UL>
<INPUT TYPE = button VALUE = "Tauschen" onclick = "Tauschen()">

```

**.tags()** Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern  
siehe tags Collection des DOM

Beispiel:

```

<SCRIPT LANGUAGE="JScript">
    var ZeigerAufFeldAllerPTag = document.all.tags("P");
    if (ZeigerAufFeldAllerPTag!=null)
    {
        for (i=0; i< ZeigerAufFeldAllerPTag.length; i++)
        { ZeigerAufFeldAllerPTag [i].style.textDecoration="underline";}
    }
</SCRIPT>

```

**.taintEnabled()** Data-Tainting (Daten verwerfen) per navigator Objekt  
ab IE 6.x

**.taintEnabled()** Data Tainting-Verfügbarkeit  
ab IE 6.x  
siehe Objekt window.clientInformation

**.tan()** Tangens im Bogenmass  
liefert Wert von -1 bis +1  
siehe Script-Objekt Math

**.toDateString()** kompletten Inhalt des Date-Objektes als String liefern  
Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
Script-Objekt Date

**.toExponential()** String liefern, der den Wert in Exponential-Darstellung enthält  
auch bei numerischem Literal z.B. "10.2345678e+13"  
IE ab 5.5, NS ab 6.x  
siehe Script-Objekt Number

Beispiel:

```

var num=77.1234
num.toExponential()    liefert "7.71234e+1"
num.toExponential(4)) liefert "7.7123e+1"
num.toExponential(2)) liefert "7.71e+1"
77.1234.toExponential() liefert "7.71234e+1"
77 .toExponential()    liefert "7.7e+1"

```

**.toFixed()** String liefern, der den Wert in Festkomma-Darstellung enthält



auch bei numerischem Literal z.B. "10.2345678"  
 IE ab 5.5, NS ab 6.x  
 siehe Script-Objekt Number

Beispiele:

```
var num=10.1234
num.toFixed()           liefert "10"
num.toFixed(4)          liefert "10.1234"
num.toFixed(2)          liefert "10.12"
0.124.toFixed(2)        liefert "0.12"
0.125.toFixed(2)        liefert "0.13"
0.126.toFixed(2)        liefert "0.13"
0.045.toFixed(2)        liefert "0.05"
1234.56789.toFixed(4)   liefert "1234.5679"
```

**.toGMTString()** deprecated

**.toLocaleDateString()** kompletten Inhalt des Date-Objektes als String in lokaler Einstellung des Betriebssystems auf User-PC liefern  
 immer verwenden für Jahre von 1601 bis 1999 --> siehe **.toLocaleString()**  
 Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
 Script-Objekt Date

**.toLocaleLowerCase()** String bzw. Stringliteral nach neuen String kopieren und dort nach Kleinbuchstaben umwandeln  
 laut aktuelle Sprach-Einstellungen der Umgebung auf PC des Users  
 nur IE ab 5.5  
 siehe Script-Objekt String

**.toLocaleString()** Wert eines Objektes in System-lokale Einstellungen umwandeln  
 System-lokale Einstellung z.B. Ländereinstellung, Uhrzeitformat  
 Konvertierung: Analog wie z.B. bei Excel, das die lokalen Einstellungen ausliest.  
 nur anwenden für Anzeige des konvertierten Wertes:  
 Berechnungen mit dem Wert immer unkonvertiert vollziehen, da sämtliche  
 Berechnungsfunktionen **nur** das interne, also unkonvertierte  
 Format kennen (**nicht** analog zu Excel)  
 Wenn das Objekt ein Script-Objekt Array ist, also Feldelemente in lokale Einstellungen konvertiert werden  
 sollen, dann wird ein String geliefert, der die Feldelemente in der Reihenfolge im Feld  
 und diese getrennt durch **dasjenige** Trenner-Zeichen laut **lokale** Einstellungen enthält.  
 Wenn das Objekt eine Script-Objekt Date ist, so wird der Wert von dem Objekt in den lokalen  
 Datumeinstellungen geliefert. Dabei sind nur Jahreszahlen von 1601 bis 9999 zulässig.  
 Andere Jahresangaben werden nicht konvertiert.  
 Wenn das Objekt ein Script-Objekt Number ist, so werden die lokalen Einstellungen zum Zahlenformat  
 geliefert.  
 Wenn das Objekt ein Script-Objekt Object ist, so werden nur dann lokale Einstellungen berücksichtigt,  
 insoweit das Objekt diese berücksichtigen kann. Ein String wird immer geliefert.

Beispiel für Konvertierung eines Feld mit Gleitkomma-Werten:

```
var Feld = new Array(6);
var Wert = 3,201,300.20; // in JScript ist das Dezimalkomma der Punkt
                        // der Tausendertrenner das Komma

// Feld füllen
for( var i = 0; i < 7; i++)
{
    Wert +1 = 10;
    Feld [i] = Wert;
}

alert(Feld.toLocaleString());
```

Beispiel für Konvertierung eines Datums:

```
var Jetzt = new Date();
alert(Jetzt.toLocaleString());
```

**.toLocaleString()** kompletten Inhalt des Date-Objektes als String in lokaler Einstellung auf User-PC liefern  
 nur für Jahre ab 2000  
 aber für Jahre von 1601 bis 1999 immer laut lokale Einstellungen des Betriebssystems auf User-PC  
 siehe **.toLocaleDateString()**  
 Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
 Script-Objekt Date

**.toLocaleTimeString()** Zeitwert des Date-Objektes als String in lokaler Einstellung des Betriebssystems auf User-PC liefern



Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
Script-Objekt Date

`.toLocaleUpperCase()` String bzw. Stringliteral nach neuen String kopieren und dort nach Grossbuchstaben umwandeln  
laut aktuelle Sprach-Einstellungen der Umgebung auf PC des Users  
nur IE ab 5.5  
siehe Script-Objekt String

`.toLowerCase()` String bzw. Stringliteral nach neuen String kopieren und dort nach Kleinbuchstaben umwandeln  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral ="StringLiteral";
StringLiteral.toLowerCase()      liefert "stringliteral"
"StringLiteral".toLowerCase()    liefert "stringliteral"
```

`.toFixed()` liefert String, der die Nachkommastellen enthält  
auch bei numerischem Literal z.B. "10.2345678"  
IE ab 5.5, NS ab 6.x  
siehe Script-Objekt Number

Beispiele:

```
var num=5.123456
num.toFixed()           liefert "5.123456"
num.toFixed(4)          liefert "5.123"
num.toFixed(2)          liefert "5.1"
num.toFixed(1)          liefert "5"

1250 .toFixed(2)        liefert "1.3e+3"
1250 .toFixed(5)        liefert "1250.0"
1234.56789.toFixed(2)   liefert "1.2e+3"
1234.56789.toFixed(9)   liefert "1.234.56789" entspricht also e+0
```

`.toString()` Wert eines Objektes in einen String umwandeln  
wenn Objekt ein Script-Objekt Array ist, dann Elemente in der Feldreihenfolge und mit Komma getrennt geliefert  
wenn Objekt ein Script-Objekt Boolean ist, dann "true" bzw "false" geliefert (Kleinschreibung !)  
wenn Objekt ein Script-Objekt Error ist, dann die Fehlermeldung geliefert (Objekt-Wert ist der Fehlercode)  
wenn Objekt ein Script-Objekt Function ist, dann Quellcode der Funktion geliefert (inklusive Funktionskopf)  
wenn Objekt ein Script-Objekt Object ist, so wird geliefert:  
"**[object** objectname]"  
mit objectname als konkreter Bezeichner der Objektklasse bzw. des Objektes fettgedrucktes wird so geliefert wie angegeben  
wenn der Bezug vor `.toString()` ein Wert laut ID-Attribut oder NAME-Attribut ist, dann wird die Objektklasse geliefert

Beispiel 1:

```
var Wert = 33,33;
alert(Wert.toString(2) + ' ' + Wert.toString(16) + ' ' + Wert.toString(10));
```

Beispiel 2:

```
<BUTTON ID="ID_Button" .... > .... </BUTTON>
ID_Button.toString() liefert "button"
```

Beispiel 3: Numerischen Wert als Zeichenkette liefern und dabei Dezimalpunkt zu Dezimalkomma umwandeln

Es wird angenommen, dass genau ein Dezimalpunkt vorkommt.

```
function punkt_zu_komma (numerischer_wert)
{
    var zeichenkette = numerischer_wert.toString();
    var funktionswert=zeichenkette;

    var pos_punkt = zeichenkette.indexOf(".");

    if(pos_punkt >=0)
    {
        funktionswert =    zeichenkette.substring(0, pos_punkt)
                        + ","
                        + zeichenkette.substring(pos_punkt + 1, zeichenkette.length);
    }

    return zeichenkette;
}
```

`.getTimeString()` Zeitwert des Date-Objektes als String liefern



Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
Script-Objekt Date

.toUpperCase() String bzw. Stringliteral nach neuen String kopieren und dort nach Grossbuchstaben umwandeln  
siehe Script-Objekt String

Beispiel:

```
var StringLiteral ="StringLiteral";
StringLiteral.toUpperCase() liefert "STRINGLITERAL"
"StringLiteral".toUpperCase() liefert "STRINGLITERAL"
```

.toUTCString() Zeitwert des Date-Objektes als String in Weltzeit liefern  
Achtung: nur für Anzeige etc. verwenden und nicht für Berechnungen !  
Script-Objekt Date

unescape ab Javascript 1.5 (Netscape 6.x) deprecated und zu ersetzen durch die Methoden  
decodeURI() und decodeURIComponent()  
dekodiert einen mit der Methode escape() kodierten String (siehe escape())

.unshift() Werte dem Feld voransetzen (auch wenn Feld leer ist)  
Feld hat die Objektklasse JScript-Objekt Array bzw. Script-Objekt Array  
ab IE 5.5 und NS 6.x

Beispiel:

```
var Feld = new Array(0,1,2,3,4);
alert (Feld.unshift(-1));
alert(Feld.join()); // "-1,0,1,2,3,4"
```

url() Diese Methode ist nicht direkt als Methode des Style-Objektes implementiert und darf daher  
**nicht** mit Punktnotation kodiert werden.  
Diese Methode wird nur in Verbindung mit dem Style Objekt verwendet.

Beispiele für Bild:

```
<STYLE>
.style1 {background:beige url(sphere.jpg) no-repeat top center}
.style2 {background:ivory url(sphere.jpeg) no-repeat bottom right}
</STYLE>

<SPAN onclick="this.style.background='beige url(sphere.jpeg) no-repeat top center'"></SPAN>

<STYLE>
.setUrl { background-image: url(sphere.jpg) }
.loseUrl { background-image: url(none) }
</STYLE>
<SPAN STYLE="font-size:14"
onmouseover="this.className='setUrl'"
onmouseout="this.className='loseUrl'"
>
</SPAN>
```

Beispiele für Behavior:

```
url(#objID) mit objID als ID des OBJECT-Tags
url(#default#behaviorName) eines Standard-Behaviors des IE

STYLE="behavior:url(a1.htc) url(a2.htc)"

<STYLE>
.Klasse { behavior:url(#myObject) }
</STYLE>

<STYLE>
DIV { behavior:url(fly.htc) url (zoom.htc) url (fade.htc)}
</STYLE>
```

Beispiel für Cursorformen aus **Datei** und **nicht** für im Browser implementierte Standard-Cursorformen:  
url('mycursor.cur')

Hinweise: alert (zeiger\_auf\_objekt.style.cursor); liefert beim IE den String 'url(cursor\_form)'  
mit cursor\_form für den aktuellen **und** gesetzten Cursor.

style.cursor ist standardgemäß mit einer Leerkette belegt, solange **kein** Cursor gesetzt wird  
kann nicht mit url(cursor\_form) belegt werden, wenn es sich um eine **standardgemäß im Browser implementierte** Cursorform handelt wie z.B. 'hand' oder 'normal'.  
Grund: Diese Cursorformen sind keine Dateien, obwohl alert den String



'url(cursor\_form)' anzeigt.

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

Beispiel:

```
<SCRIPT LANGUAGE="JScript">
  var ZeigerAufFeldAllerURN1 coll = document.all.urns("URN1");
  var Text = "";

  if (ZeigerAufFeldAllerURN1 != null)
  {
    for (i=0; i< ZeigerAufFeldAllerURN1.length; i++)
    {Text += ZeigerAufFeldAllerURN1.item(i).id + ' ';}
    alert (Text);
  }
</SCRIPT>
```

.UTC() Inhalt des Date-Objektes setzen nach Weltzeit  
(auch wenn Objekt bereits initialisiert wurde per new Anweisung)  
sollte ein Wert außerhalb des Wertebereiches sein, so wird automatisch korrigiert  
Script-Objekt Date

.valueOf() Wert eines JScript-Objektes bzw. Instanz eines JScript-Objektes ermitteln  
nicht bei Script-Objekt Math und Script-Objekt Error (auch nicht bei Instanzen dieser Objekte)  
Wert ist im Datentyp des Objektes, aber:  
wenn Objekt ein Script-Objekt Array ist, dann Elemente in der Feldreihenfolge und mit Komma  
getrennt geliefert (identisch in der Wirkung mit .toString() und der Array-Methode  
.join())  
wenn Objekt ein Script-Objekt Boolean ist, dann true bzw false geliefert (kein String !)  
wenn Objekt ein Script-Objekt Date ist, dann Anzahl der Millisekunden seit dem 1.1.1970 0 Uhr  
geliefert  
wenn Objekt ein Script-Objekt Function ist, dann Zeiger auf Funktion geliefert  
wenn Objekt ein Script-Objekt Number ist, dann numerischen Wert geliefert  
wenn Objekt ein Script-Objekt Object ist, so Zeiger geliefert  
siehe auch .toLocaleString() und .toString()

.write() Text bzw. HTML-Ausdruck in das Dokument ausgeben und anzeigen bzw. sofort parsen

#### Hinweis:

Die **ERSTE** Erzeugung eines HTML-Tags bewirkt das **automatische Öffnen eines neuen** HTML-Dokumentes, wenn das aktuelle Dokument **bereits komplett geladen**, also das Ereignis onload **bereits** ausgelöst wurde. Letzteres ist immer dann der Fall, wenn der BODY-Teil des Dokumentes komplett geparkt wurde. Grund: Ein komplett geparktes Dokument kann per write() bzw. writeln() nicht um HTML-Elemente verändert werden, da diese Methoden keine des HTML-DOM sind. Mit anderen Worten: Nur die Verwendung von Methoden des HTML-DOM lassen eine **nachträgliche** HTML-Elemente-Veränderung des Dokumentes zu.

Plain-Text  
HTML-Text  
Script

Beispiel für Ersatz von document.write() durch die Eigenschaft .innerHTML (nur IE):

```
<SCRIPT>
  var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild ">';

  function Laden()
  {
    ID_Div.innerHTML=Kette;           // wird sofort geparkt, also IMG-Tag ausgeführt
                                     // ( anstelle von eval()
                                     // bzw. document.write() )

    ID_IMG.SRC="";
    ID_IMG.style.display="block";

    ID_IMG.onerror= IMGAltTextAufFehlerMeldung; // ohne ()
  }

  function IMGAltTextAufFehlerMeldung ()
  {
    ID_IMG.alt="Das Bild konnte nicht geladen werden.";
    return true;
  }
</SCRIPT>
<INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
<DIV ID="ID_Div"></DIV>
```

.Write() in die offenen Datei zeichenweise schreiben (ab Position laut Satzzeiger)



verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
```

.WriteBlankLines() in die offenen Datei newline-Zeichen schreiben (ab Position laut Satzzeiger)  
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.WriteBlankLines(1);
DateiOffen.Close();
```

.WriteLine() in die offenen Datei genau 1 Zeile schreiben (ab Position laut Satzzeiger)  
wobei am Ende automatisch genau ein newline-Zeichen geschrieben wird  
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("123456789");
DateiOffen.Close();
```

.writeln() Text bzw. HTML-Ausdruck in das Dokument ausgeben und anzeigen bzw. sofort parsen

**Hinweis:**

Die **ERSTE** Erzeugung eines HTML-Tags bewirkt das **automatische Öffnen eines neuen** HTML-Dokumentes, wenn das aktuelle Dokument **bereits komplett geladen**, also das Ereignis onload **bereits** ausgelöst wurde. Letzteres ist immer dann der Fall, wenn der BODY-Teil des Dokumentes komplett geparkt wurde. Grund: Ein komplett geparktes Dokument kann per write() bzw. writeln() nicht um HTML-Elemente verändert werden, da diese Methoden keine des HTML-DOM sind. Mit anderen Worten: Nur die Verwendung von Methoden des HTML-DOM lassen eine **nachträgliche** HTML-Elemente-Veränderung des Dokumentes zu.

nach der Anzeige einen Zeilenvorschub anzeigen

Plain-Text

HTML-Text

Script

Beispiel für Ersatz von document.write() durch die Eigenschaft .innerHTML (nur IE):

```
<SCRIPT>
var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild">';

function Laden()
{
    ID_Div.innerHTML=Kette;                // wird sofort geparkt, also IMG-Tag ausgeführt
   // ( anstelle von eval()
   // bzw. document.write() )

    ID_IMG.SRC="";
    ID_IMG.style.display="block";

    ID_IMG.onerror= IMGAltTextAufFehlerMeldung; // ohne ()
}

function IMGAltTextAufFehlerMeldung ()
{
    ID_IMG.alt="Das Bild konnte nicht geladen werden.";
    return true;
}
</SCRIPT>
<INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
<DIV ID="ID_Div"></DIV>
```

