window Objekt des Internet Explorer

Objekt eines Browserfenster das im Browser erzeugt wird
das in browserspezifischen Versionen erzeugt werden kann

Das window Objekt hat diverse Zeiger auf andere Objekte:

z.B. beim IE und NS:

.document	Zeiger auf das Objekt document im Fenster
.event	Zeiger auf das Objekt event im Fenster
.history	Zeiger auf das Objekt history (Verlauf)
.location	Zeiger auf das Objekt location
.navigator	Zeiger auf das Objekt navigator

z.B. beim IE:

.screen	Zeiger auf das Objekt screen

Diese Zeiger dienen **nur** der Zuordnung der Objekte zum Fenster. Solange es sich um das aktuelle Fenster handelt, kann also in der Punktnotation die Kodierung von window entfallen.

Das Objekt window beschreibt obige Objekt nicht: Aus Sicht des HTML-DOM sind Objekte, die im DOM hinterlegt sind, keine Kinder des Fensters, denn das ist im DOM nur **durch** das HTML-Dokument present..

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 Linzensierung 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 verusacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit
Kompression genutzt wird z.B. bei
onlick-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 kostenpflichte 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 fehlerhatem 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:
Scripfehleranzeige 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 Popupfenster geblockt. Sie können den Popupblocker deaktivieren
oder Popups 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 bemeckert 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 Fensters 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.visivility="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 (Tablular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformatvorliegen, 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)

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 kannauch 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 zuveröffentlichen. 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)

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/enus/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)

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. Umeine 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.mspx(http://www.microsoft.com/technet/security/bulletin/ms99-037.mspx)
http://www.microsoft.com/technet/security/bulletin/fq99-037.mspx (http://www.microsoft.com/technet/security/bulletin/fq99-037.mspx)
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 wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Runtime Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wirdabgeraten.

Media Player / ActiveMovie-Steuerelement{05589FA1-C356-11CE-BF01-00AA0055595A} Die Funktionalität dieses Steuerelements wird 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 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 wirdferner 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, undmuss bei Bedarf manuell hinzugefügt werden.

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

Popupmenüobjekt {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)

Aktive Inhalte im Internet Explorer

Ab IE 6.0 ist das Blockieren aktiver Inhalte möglich, z.B. als Standardeinstellung. Es wird also dem IE verboten, JScript zu nutzen.
Daher muss mit Start der Webseite auf das Blockieren von Inhalten der Webseite, die auf JScript basieren, aufmerksam gemacht werden.
Bleibt die Blockierung aktiv, so muss die Webseite ALLE Elemente, die per Script angesteuert werden, inaktiv machen: Am besten garnicht erst anzeigen. Oder es wird eine scriptfreie Version der Webseite per <NOSCRIPT> aktiviert, wobei dann Browser vorzuziehbar sind, die z.B. CSS exakter rendern als der IE (will man keine IE-spezifischen HTML-Elemente verwenden).
Wenn der IE 6.x aktive Inhalte blockiert, wird NOSCRIPT-Tag aktivert, Ausnahme: Frameset
FRAMESET ist ein aktiver Inhalt:
Da der Frameset anstelle <BODY> kodiert sein muss, gilt:
Alle Tags, die für BODY zulässig sind, werden ignoriert, auch NOSCRIPT.
Wird neben Frameset noch BODY kodiert, so wird Frameset ingoriert.
Die Freigabe der Scriptblockierung erzeugt Ausführung aller Script-Teile inklusive der Eventauslösungen

Bsp.: Folgendes funktioniert vom Dokument, das window.open() hat im geöffneten Dokument (Quelltext im Dokument das window.open() verwendet):

```
function Y_unload(X00){X85[X00].close();}

var X85=new Array();var X86=new Array();

X85[0]=window.open(...);

var X87='parent.Y_unload(0);'; X86[0]=new Function('',X87);

X85[0].document.body.onunload=X86[0];
```

Wird die Scriptblockierung im geööfneten Fenster abgeschaltet,
so wird das Fenster geschlossen, weil onunload ausgelöst wird.

Achtung: document.body.onunload funktioniert ev. nicht mehr
wenn z.B. mit attachevent() aktiviert wurde

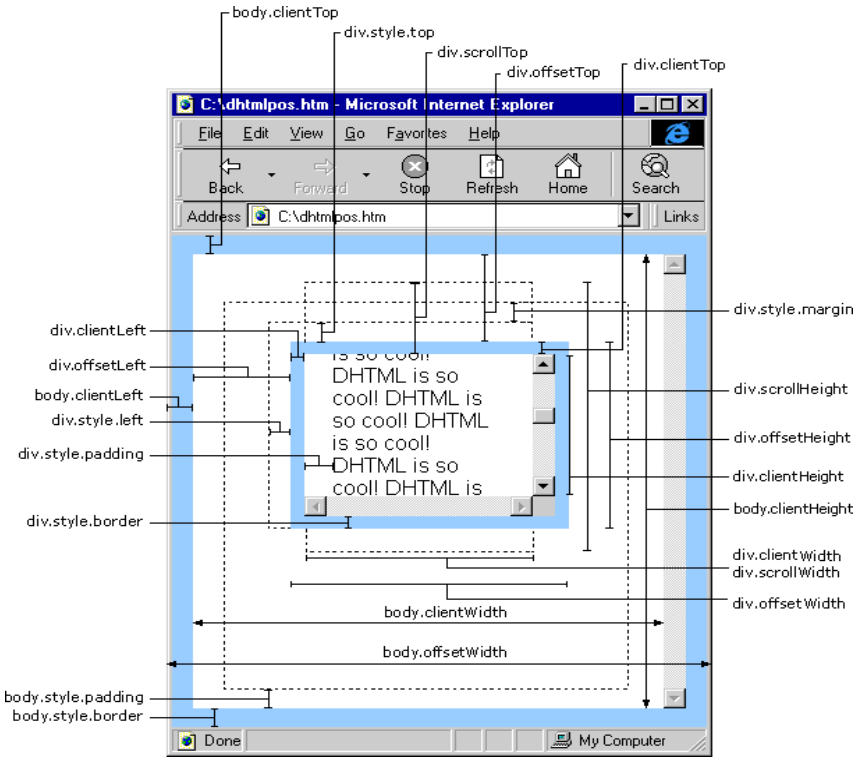
Folgende Metatags sind für den IE 6.x aktiver Inhalt:

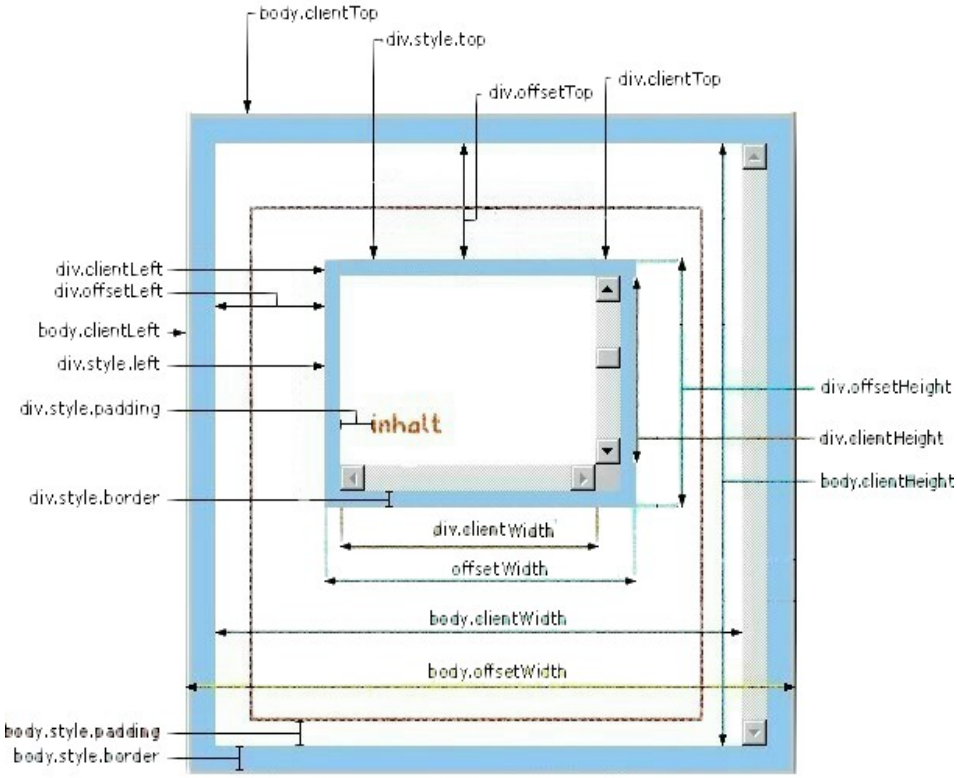
<META HTTP-EQUIV="imagetoolbar" CONTENT="no">
unterdrückt NICHT IE-Kontextmenü rechte Maus auf Bild

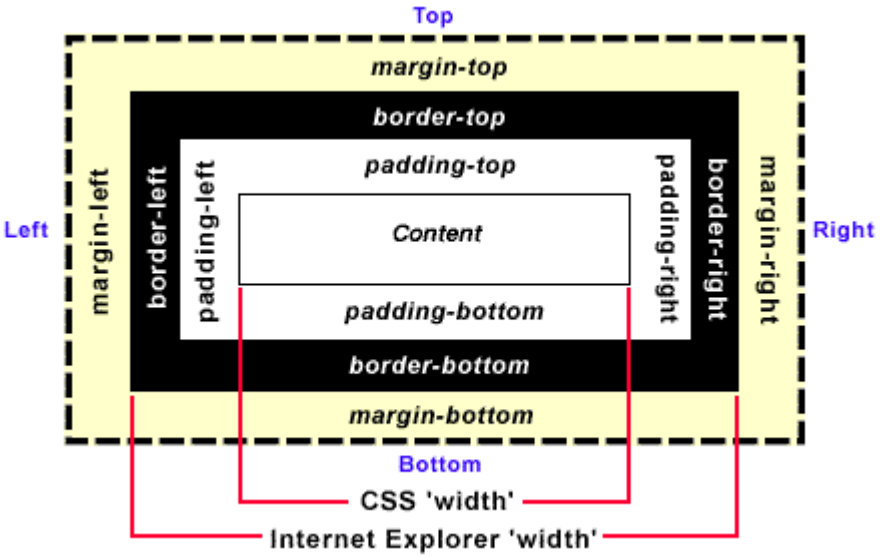
<META HTTP-EQUIV="site-enter" CONTENT="revealtrans(duration=0.3, transition=12) ">
<META HTTP-EQUIV="site-exit" CONTENT="revealtrans(duration=0.3, transition=12) ">

Achtung: Für das Hinzufügen von Elementen in den BODY (document.body) per DOM-Funktion createElement() MUSS der Body komplett geparst sein (document.body.readyState == 'complete').
Grund: Es wird standargemäß immer am Ende des BODY angefügt.
Für das Hinzufügen nicht an das Ende des BODY muss im HTML-Code ein Platzhalter z.B. DIV kodiert sein, innerhalb dessen dann die neuen HTML-Elemente erzeugt werden.

Eigenschaften:
Die Eigenschaften des IE als Grafiken:







Hinweis: Falls es sich um das aktuelle Fenster handelt, so kann die Notation window.eigenschaft auf eigenschaft abgekürzt werden.
Innerhalb von Eventhandlern ist immer **window**.eigenschaft zu kodieren, also die volle Referenzierung.
Anstelle von window kann auch self kodiert werden.
Theoretisch ist auch with (window) { .....} kodierbar.

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

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

.closed Zustand auf Geschlossensein eines Fensters
Syntax:
[var Wert =] logischer_window_name.closed

Wert false so ist Fenster offen

true so ist Fenster geschlossen

logischer_window_name Zeiger laut open()

nur lesen

.defaultStatus Standard-Text der Statuszeile (nicht der aktuelle Text)
siehe .status
Verwendung in Eventhandler-Funktion: Es muss **return true;** kodiert werden
Syntax:

```
logischer_window_name.defaultStatus = Kette
[ var Kette = ] logischer_window_name.defaultStatus
```

Kette String

logischer_window_name Zeiger laut open()

lesen und schreiben

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

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>
```

.dialogHeight — Höhe des Dialogfensters, das mit Methoden
.showModalDialog() bzw. .showModelessDialog()
erzeugt wurde
ab IE 5.x
Syntax:

```
logischer_window_name..dialogHeight [= Kette ]
[ var Kette  = ] logischer_window_name..dialogHeight
```

Kette	String als numerisches Literal mit kodierter Einheit "cm", "mm", "in", "pt", "pc" oder "px" wenn < 100 Pixel so 100 Pixel automatisch verwendet
logischer_window_name	Zeiger laut open()

lesen und schreiben

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>
```

.dialogLeft — X-Koordinate der linken oberen Ecke des Dialogfensters, das mit Methoden
.showModalDialog() bzw. .showModelessDialog()
erzeugt wurde
ab IE 5.x
Syntax:

```
logischer_window_name..dialogLeft [= Kette ]
[ var Kette  = ] logischer_window_name..dialogLeft
```

Kette	String als numerisches Literal mit kodierter Einheit "cm", "mm", "in", "pt", "pc" oder "px" wenn < 100 Pixel so 100 Pixel automatisch verwendet
logischer_window_name	Zeiger laut open()

lesen und schreiben

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
Syntax:

```
logischer_window_name..dialogTop [= Kette ]
[ var Kette  = ] logischer_window_name..dialogTop
```

Kette	String als numerisches Literal mit kodierter Einheit "cm", "mm", "in", "pt", "pc" oder "px"
	wenn < 100 Pixel so 100 Pixel automatisch verwendet
logischer_window_name	Zeiger laut open()

lesen und schreiben

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
Syntax:

```
logischer_window_name..dialogWidth [= Kette ]
[ var Kette  = ] logischer_window_name..dialogWidth
```

Kette	String als numerisches Literal mit kodierter Einheit "cm", "mm", "in", "pt", "pc" oder "px"
	wenn < 100 Pixel so 100 Pixel automatisch verwendet
logischer_window_name	Zeiger laut open()

lesen und schreiben

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>
```

.document — Zeiger auf das Objekt document im Fenster

.event — Zeiger auf das Objekt event im Fenster

.external — Zeiger auf das Objekt window.external im Fenster

.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

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

.history — Zeiger auf das Objekt history (Verlauf)

.length — Anzahl aller Frames bzw. IFrames im Fenster laut Collection document.frames
Syntax:

```
[ var Wert = ] logischer_window_name.length
```

Wert	Integer, ab 1
logischer_window_name	Zeiger laut open()

nur lesen

.location — Zeiger auf das Objekt location

.name — physischer Fenstertitel laut open()
entspricht Wert des Attributes TARGET eines Links
Text des Fenstertitels (Text in Titelzeile des Fensters) laut document.title
Syntax:

```
logischer_window_name.name  = Kette
[ var Kette = ] logischer_window_name.name
```

Kette	String
logischer_window_name	Zeiger laut open()

lesen und schreiben

Beispiel:

```
window.name="MyWindow";
window.open("file.htm","Frame1");
```

.navigator — Zeiger auf das Objekt navigator

.offscreenBuffering — alle Objekt in aktueller Seite im Offscreen zeichnen bevor sie sichtbar werden
Syntax:

```
logischer_window_name.offscreenBuffering [ = Kette ]
[ var Kette = ] logischer_window_name.offscreenBuffering
```

Kette	String	
	"auto"	Default, Browser entscheidet
	"true"	offscreen buffering ein
	"false"	offscreen buffering aus, also direkt sichtbar auf Bildschirm zeichnen

logischer_window_name Zeiger laut open()
lesen und schreiben

.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 Framset 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.
Syntax:
logischer_window_name.opener [= Zeiger]
[var Zeiger =] logischer_window_name.opener

logischer_window_name Zeiger laut open()
lesen und schreiben

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();
```

.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) ....
Syntax:
[var Zeiger =] logischer_window_name.parent

logischer_window_name Zeiger laut open()
nur lesen

.returnValue Returnwert des Dialog-Fensters, das mit Methoden .showModalDialog() bzw. .showModelessDialog() erzeugt wurde
wird gefüllt durch .showModalDialog() bzw. .showModelessDialog()
Syntax:
logischer_window_name.returnValue [= Wert]
[var Wert =] logischer_window_name.returnValue

Wert Ausdruck etc.

logischer_window_name Zeiger laut open()
lesen und schreiben

.screen Zeiger auf das Objekt screen

.screenLeft X-Koordinate der linken oberen Fensterecke (ohne Bar, Menü, Scrollbalken, Statuszeile etc)
bezüglich linke obere Ecke (0,0) vom Bildschirm
siehe Objekt window
Syntax:
[var Wert =] logischer_window_name.screenLeft

Wert	Integer, in Pixels 0 = linke obere Ecke des Bildschirmes
logischer_window_name	Zeiger laut open()

nur lesen

.screenTop Y-Koordinate der linken oberen Fensterecke (ohne Bar, Menü, Scrollbalken, Statuszeile etc)
bezüglich linke obere Ecke (0,0) vom Bildschirm
Syntax:
[var Wert =] logischer_window_name.screenTop

Wert	Integer, in Pixels 0 = linke obere Ecke des Bildschirmes
logischer_window_name	Zeiger laut open()

nur lesen

.self Zeiger auf das **aktuelle** Fenster
innerhalb der Fensterhierarchie
im FRAME
ist synonym zu window
Syntax:
self
nur lesen

.status enthält aktuellen Text der Statuszeile des Fensters (nicht den Standardtext)
siehe .defaultStatus
Verwendung in Eventhandler-Funktion: Es muss **return true;** kodiert werden.
Syntax:
logischer_window_name.status [= Kette]
[var Kette =] logischer_window_name.status

Kette	String
logischer_window_name	Zeiger laut open()

lesen und schreiben

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 Mauspfeil &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 initialsiert 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 Stautuszeile 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>
```

.top — Zeiger auf das oberste Fenster in der Fenster-Hierarchie
siehe Objekt window
Syntax:

```
[ var Zeiger = ] logischer_window_name.top

        logischer_window_name       Zeiger laut open()
```

nur lesen

Methoden

Hinweis: Falls es sich um das aktuelle Fenster handelt, so kann die Notation window.methode() auf methode() abgekürzt werden.
Innerhalb von Eventhandlern ist immer **window.**methode() zu kodieren, also die volle Referenzierung.
Anstelle von window kann auch self kodiert werden.
Theoretisch ist auch with (window) { .....} kodierbar.

.alert() — Dialogbox erzeugen

1. Anzeige von:
   Ausrufungszeichen-Symbol
   variablen String
   OK-Button und
2. warten auf Drücken des OK-Button

Syntax:

```
logischer_window_name alert([Kette])

Kette                       String oder Stringausdruck

logischer_window_name       Zeiger laut open()
```

liefert nichts

.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).

Syntax:

[var Wert =] logischer_window_name.attachEvent(Kette, Zeiger)

Kette	String Eventbezeichner
Zeiger	auf Eventhandler
Wert	true Einschalten erfolgreich false Einschalten nicht möglich gewesen
logischer_window_name	Zeiger laut open()

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;
```


.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
Syntax:

```
logischer_window_name.blur()
```

logischer_window_name	Zeiger laut open()

liefert nichts

.clearInterval() stoppt einen Timer, der mit .setInterval() gestartet wurde
Syntax:

```
logischer_window_name .clearInterval(timer_id)
```

timer_id	Integer laut Rückgabewert von .setInterval()
logischer_window_name	Zeiger laut open()

liefert nichts

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>
```

.clearTimeout() löscht ein Timeout, das mit .setTimeout() gesetzt wurde

Syntax:

```
logischer_window_name .clearTimeout(timeout_id)
```

timeout_id	Integer laut Rückgabewert von .setTimeout()
logischer_window_name	Zeiger laut open()

liefert nichts

.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
Syntax:
logischer_window_name.close()

logischer_window_name Zeiger laut open()
liefert nichts

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();
```

.confirm()

Dialogbox erzeugen mit
1. Anzeige Fragezeichen, String, OK/Abbrechen-Button
2. warten auf Drücken eines der beiden Button

Syntax:
[var Wert =] logischer_window_name.confirm([Kette])

Kette String oder Stringausdruck

Wert true für OK
false für Abbrechen

logischer_window_name Zeiger laut open()

.createPopup() Popupfenster 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 Popupfensters
ab IE 5.5
Syntax:

```
[ var Zeiger = ] logischer_window_name.createPopup()
```

Zeiger	Referenz auf das Popupfenster (Zeiger entspricht ID), wird für die Verwendung der Eigenschaften und Methoden benutzt per Zeiger.eigenschaft Zeiger.methode()
logischer_window_name	Zeiger laut open()

Beispiel

```
var PopUpID=window.createPopup();
PopUpID.document.body.style.backgroundColor="green";
PopUpID.document.body.innerHTML="TooltipText";
PopUpID.show(100,100,180,25,document.body);
```

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler, wobei Registrierung mit Methode .attachEvent() aktiviert wurde
Achtung: Vor dem Neubelegen des Standard-Eventhandler diesen vor .attachEcent() retten und den Standard-Eventhandler **nach** .detachEvent() wieder einbinden (siehe Beispiel).
Syntax:

```
logischer_window_name.detachEvent(Kette, Zeiger)
```

Kette	String mit Eventbezeichner
Zeiger	Zeiger auf **denselben** Eventhandler laut .attachEvent()
logischer_window_name	Zeiger laut open()

liefert nichts

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;
```

.execScript()

Pendant zur Methode .eval()
Script wird sofort ausgeführt
Script in diversen Sprachen möglich
alle vorhandenen-globalen Variablen ansprechbar
Syntax:

```
[ var Zeiger = ] logischer_window_name.execScript(Kette1 [, Kette2])
```

Kette1	String mit Scriptcode
Kette2	String mit Scriptsprache Standard ist "JScript"
Zeiger	immer null (nicht numerisch Null !!)
logischer_window_name	Zeiger laut open()

Beispiel.: window.execScript('alert("Hallo");' , 'JScript')

.focus()

Focus setzen und Event onfocus auslösen
nur nach dem kompletten Laden des Dokumentes
vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
Syntax:

```
logischer_window_name.focus()
```

logischer_window_name	Zeiger laut open()

liefert nichts

.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 Bildschirmes
nicht anwendbar bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop und dialogLeft verwenden.
Syntax:

```
logischer_window_name.moveBy(x, y)
```

x	X-Koordinatendifferenz Integer wenn positiv so Verschiebung nach rechts wenn negativ so Verschiebung nach links
y	Y-Koordinatendifferenz Integer wenn positiv so Verschiebung nach unten wenn negativ so Verschiebung nach oben
logischer_window_name	Zeiger laut open()

liefert nichts

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>
```

.moveTo() linke obere Fenster-Ecke laut Pixel-Position auf dem Bildschirm positionieren
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Bildschirmes
nicht anwendbar bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop und dialogLeft verwenden.
Syntax:

```
logischer_window_name.moveTo(x, y)
```

x	X-Koordinate Integer, >=0
y	Y-Koordinate Integer, >= 0
logischer_window_name	Zeiger laut open()

liefert nichts

.navigate() lädt neues HTML-Dokument laut Url in das Fenster
entspricht location.href = "...."
Syntax:

```
logischer_window_name.navigate(Kette)
```

Kette	String mit Url
logischer_window_name	Zeiger laut open()

liefert nichts

Beispiele:
```
navigate("index.html");
navigate ("file:///c:/index.html");
```

.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
Syntax:

```
[ var logischer_window_name_neues_fenster = ] logischer_window_name.open(
                                                [URL]
                                                [, physischer_fenster_name]
                                                [, Features]
                                                [, Replace]
                                                                        )
```

logischer_window_name	Zeiger laut open() muss Referenz per **window** oder **self** sein, wenn es sich um das erste Fenster seit dem Laden des HTML-Dokumentes handelt innerhalb von Eventhandler muss immer volle Referenzierung kodiert werden z.B. **window**.open() anstelle von .open()

logischer_window_name_neues_fenster Zeiger auf das neue Fenster

URL String mit Url des Dokumentes, das nach dem Öffnen in das Fenster geladen wird
Standard ist about:blank
kann Leerkette sein, also kein Dokument laden
Daten per '?' + escape() **nicht** übergebbar

physischer_fenster_name dient zur Referenzierung von einem HTML-Elementen mit dem Attribut TARGET (Wert des Attributes) auf das Fenster dienen
z.B. im Link

null null-Zeiger, also keine Referenzierung möglich

String Titel des Fensters oder vordefinierter Namen
Standard ist "_blank" also ohne Name

vordefinierte Namen:

"_blank"	Standard : ohne Name
"_media"	Dokument laut Url wird in den HTML-Content-Bereich der Media Bar geladen ab IE 6.x
"_parent"	Dokument laut Url wird in den Elternframe geladen wenn kein Frame so wirkunglos
"_search"	Dokument laut Url wird in das Browser-Search-Fenster geladen ab IE 5.x
"_self"	Dokument laut Url wird in das neue Fenster geladen
"_top"	Dokument laut Url wird in das oberste Fenster der Fensterhierarchie geladen

Features String als Parameterliste mit Kommatrennung (Liste der Fensterkomponenten)
gesamte Liste in " " bzw. ' ' setzen
z.B. "fullscreen=yes, toolbar=yes"
gesamte Liste ist 1 Zeichenkette,
die in 1 Quelltextzeile passen muss, oder in Teilketten zerlegt mit dem + Operator zusammengesetzt wird
Blanks in Liste **nicht** zulässig
Listenelement: option=wert
wenn mindestens 1 Element kodiert, so alle anderen nicht kodierten Elemente automatisch deaktiviert, also immer alle gewünschten Optionen kodieren !!!
Standardwert eines Merkmales, wenn
Liste kodiert wurde: no oder 0
Liste nicht kodiert wurde: yes oder 1
keine Standardwerte vorhanden für Pixelangaben da diese vom Browser automatisch belegbar sind

channelmode = { yes | no | 1 | 0 }
default ist no bzw. 0
yes oder 1 für Channelleiste anzeigen (Fenster im Channel-Mode anzeigen)

directories = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes oder 1 Directory Buttons anzeigen

fullscreen = { yes | no | 1 | 0 }
default ist no bzw. 0
yes bzw. 1 Vollbild anzeigen also ohne
Leisten etc., wobei damit fast alle
Steuerungselemente unsichtbar werden
Hinweis: ALT+F4 schliesst das Fenster

height = Pixelwert, Fensterhöhe
mindestens 100 (wenn < 100, so auf
100 automatisch gesetzt)

left = X-Koordinate in Pixels bezüglich linker
oberer Screen-Ecke (0,0)

location = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw 1 für URL-Eingabezeile anzeigen
(Adresszeile anzeigen)

menubar = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Menübar anzeigen
(Leiste der Pull-Down-Menüs anzeigen)

resizable = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Größenveränderung des
Fensters erlaubt per Mausziehen

scrollbars = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Scrollbalken anzeigen
sobald Fensterinhalt größer als
Anzeigebereich des Fensters

status = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Statuszeile anzeigen

titlebar = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Titelzeile anzeigen
Titel werden immer angezeigt bei Dialog-Box

toolbar = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Toolbar (Navigationsleiste)
anzeigen (Button Zurück etc.)

top = Y-Koordinate in Pixels bezüglich linker oberer
Screen-Ecke (0,0)

width = Pixelwert, Fensterbreite,
mindestens 100 (wenn < 100, so auf
100 automatisch gesetzt)

Replace true, so History-Eintrag des Dokumentes, das das neue
Fenster erzeugt, ersetzen durch den Eintrag
des neuen Fensters,
also Zurück zum Dokument, dass das Fenster öffnet,
nicht mehr möglich (**Achtung: Diese Einstellung ist absolut User-unfreundlich, da der User nicht das Button "Zurück" verwenden kann, sondern erst in der Verlauf -Leiste suchen muss, und somit den den Eindruck bekommt, als würde man ihn an die neue Webseite zwanghaft binden wollen ! Das gilt vorallem dann, wenn der User in den Browser-Einstellungen das Öffnen im selben Fenster gewählt hat.)**
false, so neuen Histrory-Eintrag zum neuen Fenster erzeugen,
also zurück zum alten Fenster möglich

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();
```

.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
Syntax:

```
logischer_window_name.print()
```

logischer_window_name	Zeiger laut open()

liefert nichts

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

Syntax:

```
[ var Wert = ] logischer_window_name ([Kette1 [,Kette2]])
```

Kette1	String freier Text der Meldung
Kette2	String, frei Text der Vorbelegung der Eingabezeile Standard ist "undefined"
Wert	String oder Integer Ergebnis der User-Eingabe
logischer_window_name	Zeiger laut open()

.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
Syntax:

```
logischer_window_name.resizeBy(x,y)
```


x	horizontale Spanne in Pixel, Integer wenn positiv so Veränderung nach rechts wenn negativ so Veränderung nach links	
y	vertikale Spanne in Pixel Integer wenn positiv so Veränderung nach unten wenn negativ so Veränderung nach oben	
logischer_window_name	Zeiger laut open()	

liefert nichts

.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
Syntax:

logischer_window_name.resizeTo(x, y)

x	Breite in Pixel, Integer >=100
y	Höhe in Pixel, Integer >=100
logischer_window_name	Zeiger laut open()

liefert nichts

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>
```

.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, Scrolleisten etc.
sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters
Syntax:

```
logischer_window_name.scroll(x, y)
```

x	Pixel-Abstand vom linken Fensterrand Integer, > 0
y	Pixel-Abstand vom oberen Fensterrand Integer, > 0
logischer_window_name	Zeiger laut open()

liefert nichts

.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, Scrolleisten etc.
sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters
Syntax:

```
logischer_window_name.scrollBy(x, y)
```

x	Pixel-Abstand vom linken Fensterrand Integer wenn positiv, so Verschiebung nach rechts wenn negativ, so Verschiebung nach links
y	Pixel-Abstand vom oberen Fensterrand Integer wenn positiv, so Verschiebung nach unten wenn negativ, so Verschiebung nach oben
logischer_window_name	Zeiger laut open()

liefert nichts

.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, Scrolleisten etc.
sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters
Syntax:

```
logischer_window_name.scrollTo(x, y)
```

x	X-Koordinate Integer, >= 0

y Y-Koordinate
Integer, >=0

logischer_window_name Zeiger laut open()

liefert nichts

.setActive() Fenster als aktiv setzen (also auch für die Eventdurchreichung aktivieren)
aber **ohne** es zu fokussieren
und **ohne** es scrollbar zu machen

Syntax:

logischer_window_name.setActive()

logischer_window_name Zeiger laut open()

liefert nichts

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>
```

.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.

Syntax:

[var TimerID =] logischer_window_name.setInterval(Code, Wartezeit [, Sprache])

Code — auszuführender Code bzw. Zeiger auf Codebaustein
vor IE 5.x: String aber kein Zeiger
ab IE 5.x: String oder Zeiger
Zeiger empfehlenswert, wenn Code in externer Datei liegt
String empfehlenswert, wenn Code im aktuellen Dokument liegt
Übergabe von Argumenten möglich

Wartezeit — Wartezeit nach deren Ablauf wird Code aktiviert wird
Integer
in Millisekunden
(1000 Millisekunden sind 1 Sekunde)

Sprache — Sprache des Codes
(analog zum LANGUAGE-Attribut)
String
z.B. "JScript", "VBScript", "JavaScript"

TimerID — ID für den Stop der periodischen Aufruffolge mit der Methode .clearInterval()

Integer

logischer_window_name Zeiger laut open()

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>
```

.setTimeout() einmaliger und somit **nicht periodischer** Aufruf eines Codes mit genau einem vorherigen Warten in Millisekunden (Timer)

Der mit setTimeout() aufgerufene Code wird **nicht zyklisch** 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.

Syntax:

```
[ var TimerID = ] logischer_window_name.setTimeout(Code, Wartezeit [, Sprache])
```

Code	auszuführender Code bzw. Zeiger auf Codebaustein vor IE 5.x: String aber kein Zeiger ab IE 5.x: String oder Zeiger Zeiger empfehlenswert, wenn Code in externer Datei liegt String empfehlenswert, wenn Code im aktuellen Dokument liegt Übergabe von Argumenten möglich
Wartezeit	Wartezeit nach deren Ablauf wird Code aktiviert wird Integer in Millisekunden (1000 Millisekunden sind 1 Sekunde)
Sprache	Sprache des Codes (analog zum LANGUAGE-Attribut) String z.B. "JScript", "VBScript", "JavaScript"
TimerID	ID für den Stopp der periodischen Aufruffolge mit der Methode .clearTimeout() Integer wird nur benötigt, wenn der Code auf eine rekursive Funktion weist, wobei der Zyklusabbruch durch.clearTimeout() erfolgen muss Hinweis: Ist keine Rekursion nötig, soll aber zyklisch aufgerufen werden, dann .setInterval() verwenden
logischer_window_name	Zeiger laut open()

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);
```

.showHelp()

Helpdatei anzeigen (*.chm und *.htm)
Syntax:

```
logischer_window_name.showHelp(URL [, ContextID])
```

URL	String, URL der Help-Datei
ContextID	String oder Integer Kontext-Identifizierung in der Help-Datei
logischer_window_name	Zeiger laut open()

liefert nichts

.showModalDialog()

Modales Dialog-Fenster erzeugen und anzeigen (modaler Dialog) sowie automatisch focussieren
Syntax:

```
[ var Wert = ] logischer_window_name.showModalDialog(  URL
                                                       [, Arguments]
                                                       [, Features]
                                                    )
```

URL	String als Url des zu ladendenen HTML-Dokumentes
Arguments	freie Parameterliste für Werteübergabe Dialog-Fenster Werteübergabe auch per Felder möglich Format wie Eigenschaft .dialogArguments wenn Liste als String, so max 4096 Zeichen in der Liste
Features	String als Parameterliste mit Kommatrennung gesamte Liste in " " setzen bzw. ' ' z.B. "center:yes, status:yes" Listenelement: option:wert dialogHeight:hoehe_in_pixel ab IE 5.x auch Gleitkomma cm, mm, in, pt, pc, oder px mindestens 100 Pixel (wenn < 100 so automatisch 100 verwendet) dialogLeft:X_koordinate_relativ_linke_obere_Ecke_Screen (0,0) dialogTop:Y_koordinate_relativ_linke_obere_Ecke_Screen (0,0) dialogWidth:breite_in_pixel ab IE 5.x auch Gleitkomma cm, mm, in, pt, pc, oder px center:{ yes | no | 1 | 0 | on | off } default ist yes bzw. 1 yes bzw. 1 Dialogbox im Screen zentrieren dialogHide:{ yes | no | 1 | 0 | on | off }

default ist no bzw. 0
yes bzw. 1 Dialogbox nicht druckbar bzw.
nicht angezeigt in
Druckvorschau

edge:{ sunken | raised }
default ist raised
Kantenstil sunken = versenkt
raised = nicht versenkt

help:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Anzeige des Fragezeichen in der
Titelzeile
(context-sensitive Help icon)

resizable:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Dialogboxgröße veränderbar

scroll:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Inhalt scrollbar

status:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Statuszeile anzeigen

unadorned:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Rahmen farbig anzeigen

Wert im Format laut Eigenschaft .returnValue

logischer_window_name Zeiger laut open()

Beispiel 1:

```
<SCRIPT>
    function ErmittleZufallsZahl(Faktor)
    {return parseInt(Math.random() * Faktor);}

    function BoxHoeheErmitteln()
    {
        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 BoxHoehe Features= BoxHoeheErmitteln();

        window.showModalDialog( "test.htm",
                                "",
                                BoxHoeheFeatures
                              );
    }
</SCRIPT>
<FORM NAME="ID_Formular">
    waehle 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

Syntax:

```
[ var Wert = ] logischer_window_name.showModelessDialog(URL
                                                        [, Arguments]
                                                        [, Features]
                                                       )
```

URL String als Url des zu ladendenen HTML-Dokumentes

Arguments freie Parameterliste für Werteübergabe an Dialogbox

Werteübergabe auch per Felder möglich
Format wie Eigenschaft .dialogArguments
wenn Liste als String, so max 4096 Zeichen in der Liste

Features String als Parameterliste mit Kommatrennung
gesamte Liste in " " setzen bzw. ' '
z.B. "center:yes, status:yes"
Listenelement: option:wert
dialogHeight:hoehe_in_pixel
ab IE 5.x auch Gleitkomma
cm, mm, in, pt, pc, oder px
mindestens 100 Pixel (wenn < 100 so
automatisch 100 verwendet)
dialogLeft:X_koordinate_relativ_linke_obere_Ecke_Screen
(0,0)
dialogTop:Y_koordinate_relativ_linke_obere_Ecke_Screen
(0,0)
dialogWidth:breite_in_pixel
ab IE 5.x auch Gleitkomma
cm, mm, in, pt, pc, oder px
center:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Dialogbox im Screen zentrieren
dialogHide:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Dialogbox nicht druckbar bzw.
nicht angezeigt in
Druckvorschau
edge:{ sunken | raised }
default ist raised
Kantenstil sunken = versenkt
raised = nicht versenkt
help:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Anzeige des Fragezeichen in der
Titelzeile
(context-sensitive Help icon)
resizable:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Dialogboxgröße veränderbar
scroll:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Inhalt scrollbar
status:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Statuszeile anzeigen
unadorned:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Rahmen farbig anzeigen

Wert im Format laut Eigenschaft .returnValue

logischer_window_name Zeiger laut open()

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 VornameAktualisieren()          // 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.VornameAktualisieren();
                                                        // Aufruf einer Funktion aus
                                                        // Elterndokument
        }

        function Init()
        {
                var DialogArgumente = dialogArguments;
                DialogArgumente.Vorname = "unbekannt";
                DialogArgumente.VornameAktualisieren();
                                                        // 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>
```

Events:

onactivate Fires when the object is set as the active element.
onafterprint Fires on the object immediately after its associated document prints or previews for printing.
onbeforedeactivate Fires immediately before the activeElement is changed from the current object to another object in the parent document.
onbeforeprint Fires on the object before its associated document prints or previews for printing.
onbeforeunload Fires prior to a page being unloaded.
onblur Fires when the object loses the input focus.
oncontrolselect Fires when the user is about to make a control selection of the object.
ondeactivate Fires when the activeElement is changed from the current object to another object in the parent document.
onerror Fires when an error occurs during object loading.
onfocus Fires when the object receives focus.
onhelp Fires when the user presses the F1 key while the browser is the active window.
onload Fires immediately after the browser loads the object.
onmove Fires when the object moves.

onmoveend Fires when the object stops moving.
onmovestart Fires when the object starts to move.
onresize Fires when the size of the object is about to change.
onresizeend Fires when the user finishes changing the dimensions of the object in a control selection.
onresizestart Fires when the user begins to change the dimensions of the object in a control selection.
onscroll Fires when the user repositions the scroll box in the scroll bar on the object.
onunload Fires immediately before the object is unloaded.

window.history Objekt

Dieses Objekt ist eigentlich eine Collection.
Erzeugung: keine, da vom Browser erzeugt

Zugriff: **logischer_window_name.history**.eigenschaft
logischer_window_name.history.methode()

logischer_window_name.history[index].eigenschaft
logischer_window_name.history[index].methode()

index: ab 0
Nummer des History-Eintrages im Dokument

ist eigentlich eine Collection:
beinhaltet die Urls, die der User auf dem Client besucht hat (entspricht einem Verlauf des Surfens des Users)
allerdings sortiert nach Urls und dort nach zugehörigen Unter-Urls (nicht sequnetiell)
Urls können seuqntiell-relativ zur aktuell besuchten Seite nur durch die Methoden des history-Objektes verwendet werden
wobei Sprünge möglich sind
anhand eines Indexs 0 = aktuelle Seite
< 0 zuvorliegende Seiten
> 0 nachliegende Seiten
nur nutzbar wenn aktuelle Seite (Index 0) durch Rückwärtsgehen
in der History eingestellt wurde
Alle Seiten werden immer aus dem Browser-Cache gelesen, wenn
im HTML-Dokument nichts anderes vereinbar wurde
in den Internet-Optionen des Browsert nichts anderes vereinbart wurde

Eigenschaften:

.length	Anzahl der Feldelemente also Feldlänge z.B. bei Collection ist der Index zum history-Objekt

Methoden:

.back()	Aktivierung einer zur aktuell besuchten Seite im Verlauf des Surfens davorliegenden Seite
.forward()	Aktivierung einer zur aktuell besuchten Seite im Verlauf des Surfens danachliegenden Seite
.go()	Aktivierung irgendeiner Url aus dem Verlauf entspricht beliebiger Selektion aus der Verlaufsliste

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>
```

window.location Objekt

Erzeugung: keine, da vom Browser erzeugt

Zugriff: innerhalb eines Eventhandler:
logischer_window_name.location.eigenschaft
logischer_window_name.location.methode()

sonst auch möglich
location.eigenschaft
location.methode()

Container der Informationen der aktuellen Url
Eigenschaft .href enthält die komplette Url-Information und ist die Standard-Eigenschaft (entspricht HREF-Attribut):
location='http://www.test.de' ist gleichwertig mit location.href='http://www.test.de'

Bsp.: Quelltext anzeigen: Öffnet lokalen Standard-Text-Editor z.B. Notepad
<INPUT TYPE="button" VALUE="Quelltext ansehen" onclick="window.location = '**view-source:**' + window.location.href">

Eigenschaften:

.hash	Teil des Wertes der Eigenschaft .href also Teil der Url als Anker Teil folgt direkt hinter dem Nummernkreuz # und ohne Nummernkreuz
.host	**Hostname** und **Port** einer Location oder Url in der Form "**hostname**:**port**"
.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
.pathname	Datei und Pfad eines Objektes
.port	**Port** einer Location oder Url in der Form "hostname:**port**" basierend auf dem aktuellen **Protokoll** laut Eigenschaft .protocol

z.B.

Standard-Ports	Protokoll
21	FTP
80	HTTP

.protocol	Protokoll-Teil einer Url inklusive http und ftp liefern
.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

Methoden:

.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.
.reload()	aktuelles Dokument neu laden
.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 isrt damit **nicht** mehr per Vorwärts- und Zurück-Button einstellbar.

window.navigator Objekt

Eigenschaften

.appCodeName	Codename des Browsers per navigator Objekt
.appMinorVersion	Update der Browserversion per navigator Objekt
.appName	Name des Browsers per navigator Objekt
.appVersion	Betriebsystem-Plattform und Browserversion per navigator Objekt
.browserLanguage	Sprache des Betriebssystemes und **nicht** des Browsers (Browsersprache kann andere sein als die des Betriebssystems z.B. französischer Browser auf deutschem Windows)
.cookieEnabled	Cookiesstatus lesen aber nicht verändern per navigator Objekt
.cpuClass	CPU-Klasse per navigator Objekt
.mimeTypes	Zeiger auf Collection mimeTypes
.onLine	Onlinestatus des Browsers ermitteln per navigator Objekt keine Überprüfung auf Netzwerkverbindung
.platform	Name des Betriebssystemes per navigator Objekt
.plugins	Zeiger auf Collection plugin
.systemLanguage	Standard-Sprache des Betriebssystemes (Sprache der Installation) per navigator Objekt
.userAgent	Browser-Codename **und** –Version per navigator Objekt
.userLanguage	vom User eingestellte Sprache des Betriebssystemes (nicht die Sprache der Installation, sondern die regionale Sprache des Betriebssystemes)
.userProfile	Zeiger auf Collection userProfile

Methoden

.javaEnabled()	prüfen auf generelle Ausführbarkeit von Javacode, also ob Java-Maschine im Browser aktiv ist per navigator Objekt
.taintEnabled()	Data-Tainting (Daten verwerfen) per navigator Objekt

navigator.mimeTypes Collection (auch bei IE 6.x)

Feld aller Mimetypen also Dateitypen, die Browser für Plugin und Dateiverarbeitung verarbeiten kann bzw. soll. Als Index dient der Mimetyp als String oder ein Integerwert ab 0.

Ein Plugin wird einem Mimetyp zugeordnet.

Ab dem IE 6.0 werden keine Plugins mehr unterstützt. Es muss ein Active-X-Control verwendet werden.

Die Eigenschaft .enablePlugin enthält den Zeiger auf das installierte Plugin (sonst null-Zeiger). Mit diesem Zeiger sind die Eigenschaften und Methoden des Plugins ansprechbar. Jedes Plugin hat eigene Eigenschaften und Methoden.
Das Speichern der Zeiger **aller** Plugin-Objekte erfolgt in der Collection navigator.plugins.

Feldelement:
Reihenfolge der Feldelemente ist browserspezifisch.
Es besteht die Möglichkeit, dass Mimetypen, die keine Plugins repräsentieren, ebenfalls mit in der Collection liegen
Bsp.: "image/jpeg".
Beispiel für einen Mimetyp für echten Plugin: "application/pdf" für Adobe Acrobat-Plugin

Beim Internet Explorer muss diese Collection nicht komplett gefüllt sein , kann aber (probieren !).
Ab dem IE 6.0 werden keine Plugins mehr unterstützt. Es muss ein Active-X-Control verwendet werden.

Es ist empfehlenswert, beim Internet Explorer **nur** mit der document.embeds Collection zu arbeiten. Die Collection navigator.mimeTypes kann zwar unter NS und IE mit dem gleichen Script-Code angesprochen werden, aber das ist spätestens ab IE 6.0 nicht mehr möglich.

Erzeugung:
keine, da vom Browser erzeugt

Zugriff:
navigator.mimeTypes[mime_typ].eigenschaft

mime_typ: String, der als Feldindex dient
z.B. "text/html"

Beispiele 1: Feld auslesen

```
for (var Index = 0; Index < navigator.mimeTypes.length; Index ++)
{ alert(    navigator.mimeTypes[i].type          + "\n"
          + navigator.mimeTypes[i].suffixes      + "\n"
          + navigator.mimeTypes[i].description
      );
}
```

Beispiel 2: Beschreibung zum Plugin anzeigen

```
if (navigator.mimeTypes["application/pdf"])
{alert(navigator.mimeTypes["application/pdf"].description);}
```

Beispiel 3: Daten an Plugin für VRML-Dateiverarbeitung übergeben

```
if (navigator.mimeTypes["x-world/x-vrml"])
{
    if(navigator.mimeTypes["x-world/x-vrml"].enabledPlugin != null)
    {
        document.write('<OBJECT DATA="yzeplin.wrl" WIDTH="400" HEIGHT="300"></OBJECT>');
    }
}
```

Beispiel 4: Suffix prüfen

```
if (navigator.mimeTypes["image/jpeg"])
{alert(navigator.mimeTypes["application/pdf "].suffixes);}
```

Eigenschaften:

.description	Zeichenkette mit der Kurzbeschreibung des Mimetyps
.enablePlugin	Zeiger auf ein Plugin null-Zeiger, wenn Plugin nicht installiert ist Mit diesem Zeiger sind die Eigenschaften und Methoden des Plugins ansprechbar. Jedes Plugin hat eigene Eigenschaften und Methoden.
.length	Anzahl der Feldelemente, ab 1
.suffixes	Liste aller erlaubten Dateisuffixe zum Mimetyp
.type	mime_typ z.B. "text/html"

Methoden:

keine

navigator.plugins Collection des Internet Explorer (auch bei IE 6.x)

Diese Collection dient nur für Kompatibilität mit anderen plugin-fähigen Browsern und stellt ein Alias der document.embeds Collection dar.

Die Collection document.embeds ist ein Alias für die plugins Collection nur bezüglich von Plugins (und nicht anderen einbettbaren Objekten), wobei letztere nur der Kompatibilität mit anderen plugin-fähigen Browsern dient.

Das Ansprechen von Plugins kann über die Collection navigator.mimeTypes per Eigenschaft .enabledPlugin erfolgen, die einen Zeiger auf das installierte Plugin enthält (wenn nicht installiert so null.Zeiger). Diese Collection muss aber beim Internt Explorer nicht komplett gefüllt sein, kann aber (ausprobieren). Wenn der Zeiger nicht null ist, dann sind die plugin-eigenen Methoden und Eigenschaften über Punktnotation ansprechbar. Beim Internet Explorer muss diese Collection nicht komplett gefüllt sein , kann aber (probieren !).

Es ist empfehlenswert, beim Internet Explorer **nur** mit der document.embeds Collection zu arbeiten. Die Collection navigator.mimeTypes kann zwar unter NS und IE mit dem gleichen Script-Code angesprochen werden, aber das ist spätestens ab IE 6.0 nicht mehr möglich.

Achtung: Ab IE 6.x werden keine Plugins mehr unterstützt, damit ist die plugins Collection hinfällig. Inwieweit diese Collectionen noch implementiert ist oder bleibt, ist durch den Programmierer zu prüfen. Ab dem IE 6.x muss jedes Plugin, das in das Dokument eingebunden werden soll, durch ein ActiveX-Control implementiert werden. Plugins, wie sie beim Netscape exisitieren, laufen unter dem IE ab 6.x nicht mehr.

Syntax:

```
[ var ZeigerAufFeld = ] navigator.plugins
[ var ZeigerAufFeldElement = ] navigator.plugins[Index]

    Index           Integer ab 0
        oder        String      Name oder ID des Elementes
                                    (analog zu NAME und ID-Attribut)
                                Name des Plugins
                    muss in [ ] kodiert sein

    ZeigerAufFeldElement
                    ist null, wenn Feldelement nicht vorhanden
```

Beispiel: Auf Adobe Acrobat-Reader-Plugin prüfen

```
<BODY>
    <SCRIPT LANGUAGE="JavaScript1.2">
    <!--
        if (navigator.plugins["Adobe Acrobat"] != null)
        {   document.write("<EMBED SRC='test.pdf' WIDTH=600 HEIGHT=800>");
            document.write("<NOEMBED> .....</NOEMBED>");
        }
        else
        {   document.write("Kein Adobe-Plugin !"); }
    //-->
    </SCRIPT>
</BODY>
```


Eigenschaften:

.length	Anzahl der Feldelemente also Feldlänge z.B. bei Collection

Methoden:

.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 !!!
.namedItem()	Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern
.tags()	Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern siehe tags Collection des DOM

navigator.userProfile Objekt des Internet Explorer

Objekt zur nur lesenden Verwaltung von User-Profile-Informationen (vCard-Daten)
ab IE 4.x
Die vCard-Daten werden beim Internet Explorer im Objekt document.form verwertet, wenn im Formular das AutoComplete aktiviert ist.

"vCard.Business.City Business"	Stadt für vCard.Business.City-Schema
"vCard.Business.Country Business"	Land/Region für vCard.Business.Country-Schema
"vCard.Business.Fax Business"	Faxnummer für vCard.Business.Fax-Schema
"vCard.Business.Phone Business"	Telefonnummer für vCard.Business.Phone-Schema
"vCard.Business.State Business"	Staat, Provinz, Gebiet für vCard.Business.State-Schema
"vCard.Business.StreetAddress Business"	Strassenname für vCard.Business.StreetAddress-Schema
"vCard.Business.URL Business"	Webadresse für vCard.Business.URL-Schema
"vCard.Business.Zipcode Business"	Postleitzahl für vCard.Business.Zipcode-Schema
"vCard.Cellular"	Einzelanschluss-Telefonnumer für vCard.Cellular-Schema
"vCard.Company"	Firmentitel (Firma) für vCard.Company-Schema
"vCard.Department"	Unternehmentitel oder –teiltitel für vCard.Department-Schema
"vCard.DisplayName"	nutzerspezifischer angezeigter Name für vCard.DisplayName-Schema
"vCard.Email"	E-mail Addresse für vCard.Email-Schema
"vCard.FirstName"	Vorname für vCard.FirstName-Schema
"vCard.Gender"	Geschlecht für vCard.Gender-Schema
"vCard.Home.City"	Name der Heimatstadt für vCard.Home.City-Schema.
"vCard.Home.Country"	Heimatland/- region für vCard.Home.Country-Schema
"vCard.Home.Fax"	Faxnummer zu Hause für vCard.Home.FAX-Schema
"vCard.Home.Phone"	Telefonnummer zu Hause für vCard.Home.Phone-Schema
"vCard.Home.State"	Heimatstaat/Provinz/Gebiet für vCard.Home.State-Schema
"vCard.Home.StreetAddress"	Heimat-Strassenname für vCard.Home.StreetAddress-Schema
"vCard.Home.Zipcode"	Heimat-Postleitzahl für vCard.Home.Zipcode-Schema
"vCard.Homepage"	private Webadresse für vCard.Homepage-Schema
"vCard.JobTitle"	Berufsbezeichnung für vCard.JobTitle-Schema
"vCard.LastName"	Nachname für vCard.LastName-Schema
"vCard.MiddleName"	zweiter Vorname für vCard.MiddleName-Schema
"vCard.Notes"	Bemerkungen für vCard.Notes-Schema
"vCard.Office"	Büroort für vCard.Office-Schema
"vCard.Pager"	Cittyrufnummer für vCard.Pager-Schema
"vCard.GenderXXX"	mit XXX laut oben z.B. Notes

Hinweis: AutoComplete betrifft auch Werte laut VALUE-Attribute von Textfeldern im Formular, aber NUR, wenn diese Felder auch das NAME-Attribut besitzen. VALUE-Werte werden automatisch für die automatische Wiederverwendung im Browser gespeichert (auch bei Password-Feldern !!!). Daher sollte die Nutzung von AutoComplete reiflich überlegt sein.
Das nachträgliche Löschen von per AutoComplete automatisch gespeicherte Werte ist in den Internet-Optionen des IE vollziehbar.

Beispiel 1:

```
// 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();
```

Beispiel 2:

```
// 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");
```

Eigenschaften:
keine
Methoden:

.addReadRequest()	Eintrag in Queue für Lesezugriff (Request Queue) erzeugen
.clearRequest()	Queue für Lesezugriff (Request Queue) leeren
.doReadRequest()	Lesezugriff auf alle vCard-Datenarten, die laut aktueller Request Queue vorgegeben sind Lesezugriff auf einzelnen vCard-Wert per .getAttribute()
.getAttribute()	einzelnen vCard-Wert lesen ohne Request Queue
.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

window.popup Objekt des Internet Explorer

Objekt für Popup-Fenster z.B. für Dialog, Meldungen, Tooltip
ab IE 5.5
Diese Fensterform ist eine stark reduzierte Instanz des window Objektes, wobei der Inhalt per Script kodiert werden muss (z.B. HTML-Tags).
Ein PopUp-Fenster
wird aktuell, wenn es angezeigt wird
wird automatisch geschlossen, sobald ein anderes Objekt den Fokus erhält, also aktiv wird
z.B. durch User-Klick außerhalb des PopUp-Fensters

Microsoft ändert fortlaufend die Active-X-Eigenschaften von Windows und somit des IE

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 Linzensierung 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 verusacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit Kompression genutzt wird z.B. bei
onlick-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 kostenpflichte 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 fehlerhatem 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:
Scripfehleranzeige 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 Popupfenster geblockt. Sie können den Popupblocker deaktivieren oder Popups 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 bemeckert 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 Fensters 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.visivility="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 (Tablular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformatvorliegen, 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)

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 kannauch 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 zuveröffentlichen. 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

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/enus/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)

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. Umeine 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.mspx(http://www.microsoft.com/technet/security/bulletin/ms99-037.mspx)

http://www.microsoft.com/technet/security/bulletin/fq99-037.mspx (http://www.microsoft.com/technet/security/bulletin/fq99-037.mspx)

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 wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Runtime Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wirdabgeraten.

Media Player / ActiveMovie-Steuerelement{05589FA1-C356-11CE-BF01-00AA0055595A} Die Funktionalität dieses Steuerelements wird 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 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 wirdferner 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, undmuss bei Bedarf manuell hinzugefügt werden.

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

Popupmenüobjekt {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)

Aktive Inhalte im Internet Explorer

Ab IE 6.0 ist das Blockieren aktiver Inhalte möglich, z.B. als Standardeinstellung. Es wird also dem IE verboten, JScript zu nutzen.
Daher muss mit Start der Webseite auf das Blockieren von Inhalten der Webseite, die auf JScript basieren, aufmerksam gemacht werden.
Bleibt die Blockierung aktiv, so muss die Webseite ALLE Elemente, die per Script angesteuert werden, inaktiv machen: Am besten garnicht erst anzeigen. Oder es wird eine scriptfreie Version der Webseite per <NOSCRIPT> aktiviert, wobei dann Browser vorzuziehbar sind, die z.B. CSS exakter rendern als der IE (will man keine IE-spezifischen HTML-Elemente verwenden).
Wenn der IE 6.x aktive Inhalte blockiert, wird NOSCRIPT-Tag aktivert, Ausnahme: Frameset
FRAMESET ist ein aktiver Inhalt:
Da der Frameset anstelle <BODY> kodiert sein muss, gilt:
Alle Tags, die für BODY zulässig sind, werden ignoriert, auch NOSCRIPT.
Wird neben Frameset noch BODY kodiert, so wird Frameset ingoriert.
Die Freigabe der Scriptblockierung erzeugt Ausführung aller Script-Teile inklusive der Eventauslösungen

Bsp.: Folgendes funktioniert vom Dokument, das window.open() hat im geöffneten Dokument (Quelltext im Dokument das window.open() verwendet):

```
function Y_unload(X00){X85[X00].close();}

var X85=new Array();var X86=new Array();

X85[0]=window.open(...);

var X87='parent.Y_unload(0);'; X86[0]=new Function('',X87);

X85[0].document.body.onunload=X86[0];
```

Wird die Scriptblockierung im geööfneten Fenster abgeschaltet,
so wird das Fenster geschlossen, weil onunload ausgelöst wird.

Achtung: document.body.onunload funktioniert ev. nicht mehr
wenn z.B. mit attachevent() aktiviert wurde

Folgende Metatags sind für den IE 6.x aktiver Inhalt:

```
<META HTTP-EQUIV="imagetoolbar" CONTENT="no">
```
unterdrückt NICHT IE-Kontextmenü rechte Maus auf Bild

```
<META HTTP-EQUIV="site-enter" CONTENT="revealtrans(duration=0.3, transition=12) ">
<META HTTP-EQUIV="site-exit" CONTENT="revealtrans(duration=0.3, transition=12) ">
```

Achtung: Für das Hinzufügen von Elementen in den BODY (document.body) per DOM-Funktion createElement() MUSS der Body komplett geparst sein (document.body.readyState == 'complete').
Grund: Es wird standargemäß immer am Ende des BODY angefügt.
Für das Hinzufügen nicht an das Ende des BODY muss im HTML-Code ein Platzhalter z.B. DIV kodiert sein, innerhalb dessen dann die neuen HTML-Elemente erzeugt werden.

Objektbeschreibung

Erzeugung:

.createPopup() Popupfenster ohne irgendwelche Fensterelemente öffnen z.B. für Anzeige von Tooltips
Fenster wird mit der Anzeige per Methode .show() zum aktuellen Fenster
erst geschlossen, wenn **anderes Fenster** aktuell wird
z.B. durch Klicken außerhalb des Popupfensters
ab IE 5.5
Syntax:

```
[ var Zeiger = ] logischer_window_name.createPopup()
```

Zeiger Referenz auf das Popupfenster (Zeiger entspricht ID),
wird für die Verwendung der Eigenschaften und
Methoden benutzt per
Zeiger.eigenschaft
Zeiger.methode()

logischer_window_name Zeiger laut open()

Zugriff:

```
zeiger_auf_popup_fenster.eigenschaft
zeiger_auf_popup_fenster.methode
```

zeiger_auf_popup_fenster laut Erzeugung

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
// erzeugen
```


```
var PopupFenster = window.createPopup(); // windows referenziert das aktuelle und instanzierte Fenster

// Körper erzeugen also body Objekt
var PopupFensterKoerper = PopupFenster.document.body;

// Körper füllen
PopupFensterKoerper.innerHTML = "Das ist ein Popup-Fenster";

// alles anzeigen
PopupFenster.show(100, 100, 200, 50, document.body);
</SCRIPT>
```

Beispiel 2 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 Feldelelemnte 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>
```

Hinweis: Es ist aufgrund mangelnder Eigenschaften nicht möglich, Daten von einem Popup-Fenster zu empfangen, die per Eventhandler übergeben werden sollen, der im Popupfenster aufgerufen wird, aber im Dokument kodiert ist, das das Popupfenster erzeugt. Leider existiert keine Eigenschaft .opener. Als Ersatz dient folgendes Beispiel, das ein normales Fenster benutzt:

```
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();
```

Eigenschaften:

.document — Zeiger auf das HTML-Dokument im Popup-Fenster, das per .show() instanziert wird
Es können damit alle Eigenschaften und Methoden des Objektes document referenziert werden
siehe Objekt window.popup
Syntax:

```
[ var Zeiger = ] zeiger_auf_popup_fenster.document

        zeiger_auf_popup_fenster    laut Erzeugung per .createPopup()
```

nur lesen

.isOpen — prüfen ob ein per .createPopup() instanziertes Popup-Fenster angezeigt wird
siehe Objekt window.popup
Syntax:

```
[ var Wert = ] zeiger_auf_popup_fenster.isOpen

        Wert                        true, so angezeigt
                                    false, so nicht angezeigt

        zeiger_auf_popup_fenster    laut Erzeugung per .createPopup()
```

nur lesen

Methoden:

.hide() — 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 .isOpen

siehe Objekt window.popup
Syntax:
zeiger_auf_popup_fenster.hide()

zeiger_auf_popup_fenster laut Erzeugung per .createPopup()
liefert nichts

.show() ein per .createPopup() instanziertes 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 .isOpen
siehe Objekt window.popup
Syntax:
zeiger_auf_popup_fenster. .show(Wert, Wert2, Wert3, Wert4 [, Zeiger])

Wert1	X-Koordinate der linken oberen Fenstecke bezüglich eines Objektes oder des Bildschirmes Koordinaten-Ursprung (0,0) des Bildschirmes liegt in der linken oberen Ecke Integer, in Pixel, >= 0
Wert2	Y-Koordinate der linken oberen Fenstecke bezüglich eines Objektes oder des Bildschirmes Koordinaten-Ursprung (0,0) des Bildschirmes liegt in der linken oberen Ecke Integer, in Pixel, >= 0
Wert3	Breite des Fensters in Pixel Integer, > 0
Wert4	Höhe des Fensters in Pixel Integer, > 0
Zeiger	auf Objekt zu dem Wert1 und Wert2 relativ sind Standard: Bildschirm
zeiger_auf_popup_fenster	laut Erzeugung per .createPopup()

liefert nichts

window.screen Objekt im Internet Explorer

Eigenschaften:

.availHeight	verfügbare Höhe des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
.availWidth	verfügbare Breite des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
.bufferDepth	Anzahl der Bits pro Pixel für eine Farbe UND Verwendung des off-screen bitmap buffer per screen Objekt
.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
.deviceXDPI	Aktuelle Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm per screen Objekt
.deviceYDPI	Aktuelle Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm per screen Objekt
.fontSmoothingEnabled	Fontglättung auf Bildschirm per screen Objekt
.height	Auflösung des Bildschirmes in Höhe, also Anzahl der vertikalen Pixel per screen Objekt
.logicalXDPI	Standard-Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm per screen Objekt
.logicalYDPI	Standard-Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm per screen Objekt
.updateInterval	Refesh-Intervall des Bildschirmes: aus dem Puffer neu schreiben per screen Objekt
.width	Auflösung des Bildschirmes in Breite, also Anzahl der horizontalen Pixel per screen Objekt

Methoden:

keine