

FlipV	Light	Shadow
Glow	MaskFilter	Slide
Gradient	Matrix	Spiral
GradientWipe	MotionBlur	Stretch
Gray	Pixelate	Strips
ICMFilter	RadialWipe	Wave
Inset	RandomBars	Wheel
Invert	RandomDissolve	Xray
Iris	RevealTrans	Zigzag

**window Objekt**

keine Filter

**4.3.2.2.4.3. HTML-Elemente im HTML-Dokument des Internet Explorer (Auswahl)****Voraussetzungen für die programmtechnische Nutzung von HTML-Elementen als Objekte in JScript:**

Kenntnisse der Programmierers zu:

DOM  
 Zeigerverwendung  
 Vererbung  
 Browserversionen/Script-Maschinen-Versionen  
 JScript-Sprache  
 HTML-Syntax zum HTML-Element (Attribute etc.)

Hinweise: JScript von Microsoft ist programmtechnisch nicht netzwerkfähig, es sei denn, es werden Komponenten des Betriebssystems Windows

in JScript aktiviert (falls überhaupt möglich). JScript ist nur client-orientiert und arbeitet direkt auf dem PC des Internet-Users, der eine Webseite mit JScript-Teilen besucht. Aus Sicht des Webseitenbetreibers (Hoster) und dessen Server ist der User der "Endverbraucher" des Internets, also der Klient ("Kunde") des Webseitenangebotes. Wer netzwerkfähige Produkte erzeugen will, sollte sich mit der NET-Software zu Windows XP von Microsoft beschäftigen, die JScript und XML auf Basis einer **installierten und aktiven** Server-Software netzwerkfähig machen.

Bezüglich des implementierten HTML-DOM vom Internet Explorer ist festzustellen, dass große Ähnlichkeiten bzw. teilweise Identität mit anderen DOM, z.B. von XML, vorliegen. Für den Programmierer, der nicht nur JScript nutzen will, ist es für den Einstieg in XML vorteilhaft, die Prinzipien des HTML-DOM und dessen Umsetzung in JScript verinnerlicht zu haben. Microsoft setzt ganz bewusst auf prinzipielle Gemeinsamkeiten und Integration der Script-Programmierung per JScript mit bzw. in XML. So gesehen wird XML das HTML problemlos ablösen.

JScript kann nur dann auf dem User-PC aktiv werden, wenn der User Javascript zugelassen hat. Die Zulassung von Javascript hängt von der Browser-Software und optional eingesetzter Firewall-Software ab.

JScript kann Komponenten des Betriebssystems Windows wie DirectX oder ActiveX nutzen. Auch hier gilt: Der User muss die Nutzung von DirectX und ActiveX zugelassen haben. Es gibt Webseitenanbieter, z.B. Hoster für kostenlose Homepages, deren Host-Angebote (Webseiten der Kunden des Hosters) **nur** unter Nutzung von ActiveX funktionieren, wenn die Homepage mit Komponenten des Hosters erstellt wurden. Dem Besucher solcher Webseiten ist in der Regel nicht bekannt, was die ActiveX-Komponenten bewirken. Solche Webseitenangebote sind dann aus Sicht des Schutzes der Privatsphäre des Users **wertlos**, wenn der User ActiveX gesperrt hat (weil kein Risiko eingehen will) und in der Webseite keine Alternativen zu ActiveX bzw. Hinweise zu Art und Zweck von eingesetzten ActiveX-Komponenten programmiert wurden.

Der JScript-Programmierer muss daher **regelmäßig** mit dem Einsatz von Firewall-Software auf dem User-PC rechnen und daher den Einsatz von Windowskomponenten für den User **transparent** halten, wenn der Programmierer möchte, dass **sicherheitsbewusste** User die erstellte Webseite nutzen können und der User **keine Risiko** tragen soll.

JScript ist im Gegensatz zu JavaScript von Netscape nur schwer mit JAVA verbindbar, da die Browserhersteller divergierende Interessen zu JAVA besitzen und somit Bezüge auf JAVA mehr oder weniger in die Scriptmaschine implementieren.

JScript ist wie jede JavaScript-Sprache eine rein objektorientierte Sprache, die mit Zeigern arbeitet., denn HTML-Elemente können **nur als Objekte**, also nur objektorientiert verarbeitet werden. Die Zeigernutzung erfolgt intern, aber im Regelfall auch durch den Programmierer per Script-Code, vorallem dann sehr intensiv, wenn der Programmierer einen dynamischen Inhalt der Webseite erstellen will.

Die Pflege und Dokumentation der Scriptsprache und deren Implementationen im Browser hängt vom Willen des Herstellers ab, der mehr oder weniger in der Lage sein möchte, für denjenigen Transparenz zu gewähren, der Script programmtechnisch nutzen will. Damit können Objektbeschreibungen kurz- oder langelig sein, letzteres nur dann, wenn Browser-Versionen des Herstellers kompatibel sind bzw. Unterschiede regelmäßig qualifiziert und leicht zugänglich publiziert werden.

**Abbildung des HTML-Elementes als Objekt:**

HTML-Elemente werden als Objekte im Rahmen des HTML-DOM erzeugt.

Attribute des HTML-Elementes laut HTML-Syntax sind im Objekt als Eigenschaften abgebildet, leider - je nach Art des HTML-Elementes - **nicht komplett (nicht alle Attribute)**. Und: Es existieren z.T. Eigenschaften, denen der jeweilige Pedant in HTML-Attribut-Form fehlt - besonders beim Internet Explorer, der damit wesentlich mehr als HTML bietet, das allerdings Microsoft-spezifisch für das Betriebssystem Windows. Diese Abbildungs-Varianten sind der Ausdruck des Grades einer HTML-Unterstützung im Browser, denn HTML-Elemente können **nur als Objekte**, also nur objektorientiert verarbeitet werden. JScript ist daher , wie jede JavaScript-Sprache, eine rein objektorientierte Sprache, die grundsätzlich mit Zeigern auf Instanzen eines Objektes arbeitet.



Methoden des Objektes, das das HTML-Element abbildet,

dienen nur z.T. dem Aktions-Zweck des Elementes im HTML-Design. Beispiel MARQUEE-Element (scrollender Text):

Während in HTML das Element scheinbar selbständig mit vorher fest definierten Attribut-Werten startet, ist **dasselbe** Element als Objekt in JScript nach dessen Start dynamisch steuerbar - u.a. auch anhand der Methoden zum Objekt. (Selbst Werte von in HTML fest definierten Attributen sind per Objekt-Eigenschaften dynamisch zur Aktionszeit des HTML-Elementes (zur Laufzeit des Objektes im Dokument) abänderbar.),

lassen z.B. auch steuerbare Kommunikation zwischen anderen HTML-Elementen per Ereignisse (Events) zu, ermöglichen z.B. Manipulation von Vererbungs-Hierarchien, können z.B. Kind-Objekte erzeugen und das Layout des HTML-Elementes verändern.

Das HTML-Objekt wird nicht 1:1 abgebildet. Der Programmierer kann aber den Zusammenhang zwischen HTML-Element und Objekt erkennen (z.B. häufige Bezeichneridentität von Attribut im HTML-Element und Eigenschaft im abbildenden Objekt).

JScript kann HTML-Code verwenden und erzeugen, muss es aber je nach Wunsch des Programmierers nicht: Wenn das HTML-DOM es zulässt, sind HTML-Elemente per JScript ohne irgend einen HTML-Code im JScript-Code komplett erzeugbar.

Per Methoden des Objektes `window.document.write()` oder `window.document.writeln()`

programmtechnisch erzeugte HTML-Elemente werden browserintern immer mit den Methoden des DOM erzeugt. Alternativ zu `write()` und `writeln()` kann der Programmierer auch gleich die DOM-Methoden in JScript kodieren.

### Zeigerbezüge (Objekt-Referenzen) auf HTML-Elemente:

#### *im Dokument*

HTML-Elemente sind Objekte des Dokumentes, also Kinder des Objektes `document`. Die Referenzierung der HTML-Elemente im Dokument erfolgt in der Regel per

`Collection document.all`  
 oder per Zeiger laut ID-Attribut bzw. NAME-Attribut.  
 oder per spezieller Collectionen (je nach HTML-Elemente-Art bzw. DOM)  
 oder per DOM (Document Object Model) des HTML

Collectionen sind Felder aus Zeigern auf Instanzen je nach Art der Collection (siehe Beschreibungen der Collectionen). existieren z.T. nur spezifisch zu einer HTML-Elemente-Art oder sind Komponenten des DOM (Document Object Model)

sind immer in der Scriptmaschine zum Browser implementiert (kurz: "im Browser implementiert")  
 haben immer genau definierte Eigenschaften und Methoden.

Hinweis: Programmtechnisches Analogon ist das Feld (array Objekt).

Alle Zeigerbezüge nutzen ein DOM, das im Browser (je nach Version der Scriptmaschine, also je nach Browser-Version) fest implementiert ist. Hinweis: Es gibt mehrere DOM. Nachfolgende Betrachtungen nutzen das DOM zu HTML und z.T. zu XML.

#### *im Fenster*

Da das Dokument in einem Fenster eingebettet ist, sind HTML-Elemente auch Kinder des dem Objekt `document` zugehörigen Eltern-Objektes `window`.

#### *im aktuellen Fenster und dessen Dokument (kurz "im aktuellen Dokument")*

Der komplette Objektpfad lautet

`window.document.all.object_zeiger.eigenschaft`  
`window.document.all.object_zeiger.methode()`

`window.objekt_zeiger_laut_id_attribut_bzw_name_attribut.eigenschaft`  
`window.objekt_zeiger_laut_id_attribut_bzw_name_attribut.methode()`

Achtung: Die Pfad-Komponente `window.document.` ohne `.all` wird nur dann ohne Fehler vom Interpretierer akzeptiert, wenn nicht explizit die Collection `document.all` benötigt wird.

Zur Vereinheitlichung der Zeigerbezüge ist die Verwendung des Zeigers laut im HTML-Element kodiertem ID-Attribut wärmstens zu empfehlen, da dieser Zeiger immer im korrekten Kontext erzeugt wird und die Pfadkomponente `window.document.all` nicht zusätzlich kodiert werden muss, solange kein Bezug auf `document.all` explizit nötig ist (nur bestimmte Objekte oder Collectionen müssen per `document.all` referenziert werden).

Üblicherweise wird der Suffix (Pfadkomponente) `window.`  
 oder `window.document.all.`

nicht kodiert, wenn es sich um das Dokument im aktuellen Fenster handelt (ansonsten ist der Zeiger auf das konkrete Fenster und dessen Dokument zu kodieren). Dabei ist die Browser-Performance zu beachten (siehe tiefer).

Achtung: Zur Browserunterscheidung ist es wichtig, dass der Programmierer weiss, welche Kind-Objekte vom Objekt `document` unterstützt werden.

Es ist Fakt, dass neben dem Internet Explorer auch andere Browser Objekte zum Objekt `document` implementiert haben. Damit ist die Referenz `document.objekt_zeiger`



nicht **nur** für den Internet Explorer gültig.

Soll aber nur der Internet Explorer beachtet werden, dann ist document.all zu kodieren, vorallem dann, wenn ein Kind-Objekt des Objektes document in mehreren Browsern implementiert ist, aber dort mit **verschiedenen** Eigenschaften oder Methoden.

Damit sind IE-spezifische Eigenschaften zum Objekt per document.all.objekt\_zeiger.eigenschaft zu kodieren (analog zu Methoden).

Nachfolgende Objekt- und Collectionen-Beschreibungen

lassen die Referenz per window.document.all oder document.all weg und gehen von der Nutzung des ID-Attributes aus, dessen Wert als Objekt-Zeiger dient verwendet im Bezeichner des Objekte nur dann den Suffix document., wenn deutlich gemacht werden soll, dass das Objekt **nur ein Kind vom Objekt document sein kann**, also nicht als Kind eines anderen Objektes erzeugt werden darf. Wichtig ist dieser Umstand auch bei einigen Collectionen.

Bsp. document.body Objekt Body kann nur Kind von document sein  
Bsp. a Objekt a kann Kind eines anderen Kindes von document sein.

verzichten auf eine Baumhierarchie der Objekte, da wie in HTML mehr **frei gestaltbare** Verschachtelungsvarianten zulässig sind, als im DOM fest vorgeschriebene. Im Prinzip gilt: Jedes HTML-Dokument hat als Ergebnis des Webdesign **seine eigene, spezifische Hierarchie**.

stellen Objekte und Collectionen als Systematik in alphabetischer Reihenfolge bzw. im Sinnzusammenhang dar. Es gibt natürlich auch andere Gesichtspunkte der Objektbeschreibungen, die bereits weiter oben in dieser Dokumentation dargestellt wurden, z.B. Objekte und Collectionen zur Verwaltung aller bzw. spezieller HTML-Elemente oder die Beschreibung des DOM. Die Trennung dieser Beschreibungen von denen der nachfolgend beschriebenen Objekte bewirkt, dass z.T. Objekte nicht nachfolgend, sondern an anderer Stelle beschrieben werden bzw. wurden. Das bedingt sich aufgrund der Systematik dieser Dokumentation. Als Empfehlung zum Nachschlagen wird zusätzlich das Inhaltsverzeichnis bzw. der Index nahegelegt.

sind in ihrer Objekt-Hierarchie am besten aufzuarbeiten, wenn mit dem Lesen der Beschreibung zum Body-Objekt begonnen wird. Ein weiteres und sehr oft in der JScript-Programmierung verwendetes Objekt ist das div Objekt. Beide genannten Objekte sind wichtige Container für HTML-Elemente innerhalb des Dokumentes, also wichtige Kinder des Objektes document.

zum Layout (Style) vom Dokument und dessen HTML-Elemente liegen bezüglich **aller** zugehörigen Objekte und Collectionen **nur** innerhalb der Beschreibung des **style Objektes** (STYLE-Elementes) und **nicht in alphabetischer Folge**. Style ist aufgrund seiner Komplexität und als weiteres wichtiges Objekt im Webdesign nur als Einheit beschreibbar.

zu den zusätzlichen Standard-Verhaltensweisen (Standard-Behavior) von HTML-Elementen liegen als Komponenten des style Objektes bezüglich **aller** zugehörigen Objekte und Collectionen zu den Verhaltensweisen **nur** innerhalb der Beschreibung des **style Objektes** (STYLE-Elementes) und **nicht in alphabetischer Folge**. Style ist aufgrund seiner Komplexität und als weiteres wichtiges Objekt im Webdesign nur als Einheit beschreibbar.

Hinweis: Aus Gründen der Browser-Performance sind:

sämtliche Objekte, die das ID- bzw. NAME-Attribut unterstützen, mit diesem Attribut zu versehen, wenn mindestens 1 im JScript-Quellcode kodierter Zeiger-Bezug auf das abbildende Objekt existiert: Der Wert des Attributes ist der Zeigerbezeichner auf das Objekt. (Aus Gewohnheitsgründen einfach immer das ID-Attribut kodieren z.B. für im Design sich erst später als notwendig erweisende Bezüge auf das Objekt). Wenn ID **und** NAME unterstützt werden, so ID kodieren, da NAME nur im Spezialfall kodiert werden muss, z.B. im Formular-Element.

Bsp. ohne ID-Attribut-Kodierung:

```
document.write('<ELEMENT></ELEMENT>');
document.all[index_auf_element].style.visibility='hidden';

// der Index auf die Collection document.all muss
// dem Programmierer bekannt sein und direkt in den Quellcode
// kodiert werden
// oder vorher als Variable programmtechnisch ermittelt worden sein
// var index_auf_element= .....;
// ELEMENT ist Platzhalter für ein konkretes HTML-Element-Tag
```

Bsp.: mit ID-Attribut-Kodierung:

```
document.write('<ELEMENT ID="ZeigerAufDasElement"></ELEMENT>');
ZeigerAufDasElement.style.visibility='hidden'; // keine var-Anweisung !!!

// der Index auf die Collection document.all ist automatisch ermittelt, so dass
// die Kodierung von document.all entfallen kann
// ELEMENT ist Platzhalter für ein konkretes HTML-Element-Tag
```

Punktnotationen, die gemeinsame Komponenten besitzen, sich aber ansonsten unterscheiden, zu zerlegen:

Die gemeinsamen Komponenten sind aufzulösen und je als 1 eigenständiger Zeiger zwischenspeichern.

Bsp.: window\_zeiger.document\_zeiger.object\_zeiger



zerlegen in

```

var ZeigerAufAktuellesDokument = window_zeiger.document_zeiger;
var ZeigerAufObjektImAktuellenDokument =
    ZeigerAufAktuellesDokument.object_zeiger;
.....
var ZeigerAufObjektImAktuellenDokument_AufStyleEigenschaft=
    ZeigerAufObjektImAktuellenDokument.style; // 1x berechnet

ZeigerAufObjektImAktuellenDokument_AufStyleEigenschaft.visibility='hidden';
ZeigerAufObjektImAktuellenDokument_AufStyleEigenschaft.cursor='hand';
....
ZeigerAufObjektImAktuellenDokument.stop();
.....
ZeigerAufObjektImAktuellenDokument=null;

```

Ziele der Zerlegung, also der Auflösung der Punktnotation auf einen Minimalumfang, sind:  
 Vermeidung von Mehrfachkodierungen von identischen Punktnotationen im Quellcode  
 Vermeidung der pro erkannten und unaufgelösten Punktnotation durchgeführten  
 Referenzberechnung (Zeigerberechnung), so dass nur für den zwischengespeicherten  
 Zeiger die Referenz genau 1x berechnet werden muss und danach der bereits  
**berechnete**  
 Zeiger mehrfach verwendet wird.

Hinweis zur Referenzierung von Eigenschaften und Methoden:

Diverse Objekte und Collectionen haben bezeichner-identische Eigenschaften und Methoden. Daher sind die exakte Referenzierung der Objekte und Collection und das Wissen, welche Eigenschaften und Methoden zu den Objekten und Collectionen implementiert sind, wichtig.

#### 4.3.2.2.4.3.1. a Objekt des Internet Explorer

Hypertext-Link (Linkt mit Textangabe für User) also Anker:

Es kann auch ein Image im Text angegeben werden (zusätzlich oder anstelle des Textes).

Bei Image: Border unsichtbar durch Eigenschaft .border auf 0 setzen.

Achtung: keine Tabelle einbindbar (kein Objekt table) !!

Script einbindbar

benötigt **immer** die kodierte Eigenschaft .href oder .name

nicht verschachtelbar

Objekt mit Timeline nur wirksam, wenn diese aktiv ist.

siehe auch link Objekt

#### Erzeugung:

Beispiele:

```

<A ID="ID_A" HREF="http://www.test.de.com">gehe zu test.de</A>

<A ID="ID_A" HREF="home.htm">home.htm</A>

<A ID="ID_A" TARGET="name_des_fensters" HREF="test.htm">im Fenster oeffnen</A>

<A ID="ID_A" HREF="http://www.test.de "><IMG SRC="test.gif">Link mit Image und Text</A>

<A ID="ID_A" NAME="anchor_name">Eine Anker als Sprungziel</A>

<A ID="ID_A" HREF="#anchor_name">Anker ausführen </A>

<A ID="ID_A" HREF="javascript:window.open()">Javascript</A>

<A HREF="text.doc">Den Text als Word-Dokument herunterladen</A>

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

```

#### Zugriff:

```

ID_A.eigenschaft
ID_A.methode()

```



Beispiel 1:

```

<HTML XMLNS:IE>
<HEAD>
<SCRIPT>
    function NachDemDownload(DownLoadedFileContent)
    {alert (DownLoadedFileContent);}
</SCRIPT>
</HEAD>
<BODY>
<IE:Download ID="ID_IETag" STYLE="behavior:url(#default#download)" >
<P>
Click
<A HREF="javascript:ID_IETag.startDownload('download.htm', NachDemDownload)">
hier
</A>
zum Start des Downloads dieser Seite.
</BODY>
</HTML>

```

Beispiel 2 für Link ohne Linkrahmen bei IE NS ab 6.x

Achtung: Eventuell wird Navigation per TAB-Taste nicht mehr möglich sein !

Dies ist ein <A HREF="#" onfocus="if(this.blur){this.blur();}">Link ohne Rahmen bei Focuserhalt</A>

Dies ist ein <A HREF="seite.html" onclick="this.blur()">Link ohne Rahmen nach klick</A>

Beispiel 3 für Link ohne Unterstreichung:

```

<HTML>
<HEAD>
    <STYLE TYPE="text/css">
        A { text-decoration: none; }
    </STYLE>
</HEAD>
<BODY>
    <A HREF="http://www.test.de/">Ein Link ohne Unterstreichung</A>
</BODY>
</HTML>

```

**Alternative:**

Beispiel:

```

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

```

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird fokussiert die Sprungquelle defokussiert das Focus-Ereignis ausgelöst
ATOMICSELECTION	vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.begin	Selektierbarkeit des Objektes einstellen Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.charset	Zeichensatz zum Encoden eines Objektes im Dokument
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.coords	Koordinaten eines Objektes in Abhängigkeit zur Eigenschaft .shape
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen



.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hash	Teil des Wertes der Eigenschaft .href also Teil der Url als Anker Teil folgt direkt hinter dem Nummernkreuz # und ohne Nummernkreuz
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Fokussierbarkeit
.host	<b>Hostname</b> und <b>Port</b> einer Location oder Url in der Form " <b>hostname:port</b> "
.hostname	<b>Hostname</b> einer Location oder Url in der Form " <b>hostname:port</b> " kann Domain oder IP sein
.href	Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag) siehe auch Eigenschaften .rel und .rev
.hreflang	Sprachcode des Objektes laut RFC1766
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.Methods	Liste der vom Objekt unterstützten HTTP-Methoden
.name	Name des Objektes (nicht ID !!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes Collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie



.parentTextEdit	Textbereich des Elternobjektes referenzieren						
.pathname	Datei und Pfad eines Objektes						
.port	<b>Port</b> einer Location oder Url in der Form "hostname: <b>port</b> " basierend auf dem aktuellen <b>Protokoll</b> laut Eigenschaft .protocol z.B. <table border="1"> <thead> <tr> <th>Standard-Ports</th> <th>Protokoll</th> </tr> </thead> <tbody> <tr> <td>21</td> <td>FTP</td> </tr> <tr> <td>80</td> <td>HTTP</td> </tr> </tbody> </table>	Standard-Ports	Protokoll	21	FTP	80	HTTP
Standard-Ports	Protokoll						
21	FTP						
80	HTTP						
.previousSibling	Referenz auf das Vorgängerkind						
.protocol	Protokoll-Teil einer Url inklusive http und ftp liefern						
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten						
.recordNumber	Datenquelle-Satznummer eines Datenfeldes						
.rel	Beziehung zwischen Objekt in Quellseite und Nachfolger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen beim durchforsten der Seiten und Nachfolgerseiten vorrangig kodiert in <A> oder <LINK> es <b>mus</b> zugleich die Eigenschaft .href kodiert und mit <b>gültigen</b> Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)						
.rev	Beziehung zwischen Objekt in Quellseite und Vorgänger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen beim durchforsten der Seiten und Vorgängerseiten vorrangig kodiert in <A> oder <LINK> es <b>mus</b> zugleich die Eigenschaft .href kodiert und mit <b>gültigen</b> Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)						
.scopeName	Namensraum laut XMLNS-Attribut						
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes						
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes						
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes						
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes						
.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 " ...?....." Zielseite ruft Zielseite mit diesem HREF auf wurde von Quellseite aufgerufen liest den Teil von HREF hinter dem ? als Zeichenkettendaten						
.shape	Form/Gestalt eines Objektes siehe auch Eigenschaft .coords						
.sourceIndex	Index des Objektes in der Collection document.all						
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen						
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior						
.systemBitrate	wird hier nicht erklärt						
.systemCaptions	wird hier nicht erklärt						
.systemLanguage	Sprache festlegen für das Objekt						
.systemOverdubOrSubtitle	wird hier nicht erklärt						
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD						
.tagName	Tag-Bezeichner des Objektes						



.tagUrn	Uniform Ressource Name (URN) laut Namensraum laut XMLNS-Attribut
.target	Name des Ziel-Fenster bzw. Ziel-Frame
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	MIME-Typ des Objektes (Multipurpose Internet Mail Extension) wenn die Eigenschaft .classid nicht kodiert wird, dann immer .type kodieren bzw. die im Browserstandard enthaltenen MIME verwenden Hinweis: MIME wird von Simple Mail Transfer Protocol (SMTP) benutzt z.B. bei Audio, Images, Video, Texten
.uniqueID	durch den Browser automatisch generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.urn	Uniform Ressource Name (URN) des Dokumentes
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar



	DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
.hasChildNodes()	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh) prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
.insertAdjacentElement()	DOM nicht geändert Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich
.insertAdjacentHTML()	DOM wird geändert HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
.insertAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes
.mergeAttributes()	DOM wird geändert alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
.normalize()	DOM wird geändert Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmouseover, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
.removeAttributeNode()	DOM wird geändert entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
.removeBehavior()	DOM wird geändert per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
.replaceAdjacentText()	DOM wird nicht geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.scrollIntoView()	DOM wird nicht geändert Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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

.setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
 DOM wird geändert

.setCapture() Maus-Überwachung einschalten für ein Objekt  
 Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
 onmouseover und onmouseout.  
 ab IE 5.5  
 Hinweis: ausschalten per Methode .releaseCapture()

.setExpression() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
 Eigenschaft als Objektreferenz der Form  
 objekt.style.eigenschaft.  
 dient  
 Ausdruck nur als Script kodierbar  
 DOM wird nicht geändert

.swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
 nur sichtbar wenn Endetag geparkt  
 DOM wird geändert

**4.3.2.2.4.3.2. document.anchors Collection des Internet Explorer (HTML-Element A (Anker))**

Feld der Zeiger aller Anker im Dokument

Feldelementefolge laut HTML-Coding

**Erzeugung:**

durch den Browser

Beispiel für Anker:

```
<A ID="ID_Anker"
  HREF="url"
  NAME="logischer_anker_name"
  TARGET="logischer_window_name"
  >
  anker_text
</A>
```

Hinweis zu HREF= :

kann leer sein	per	Leerkette ""
	oder	"javascript:void(0);"

zu Anspringen eines Ankers:

den Wert laut NAME ablegen in	
window.location.hash	ohne vorgesetztes #
window.location.href	mit vorgesetztem #

**Syntax:**

```
[ var ZeigerAufFeld = ] document.anchors
[ var ZeigerAufFeldElement = ] document.anchors[Index, [ SubIndex]]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement.eigenschaft  
 ZeigerAufFeldElement.methode()

document.all.ID\_Anker.eigenschaft  
 document.all.ID\_Anker.methode()

Beispiel: Alle Textmarken (Anker) eines Dokumentes für deren Anwahl in einem Extra-Fenster anzeigen

```
<HTML>
<HEAD>
  <SCRIPT>
  <!--
    function textmarken_fenster_anzeigen()
    {
      var textmarken_fenster = open( "",
        "Textmarken-Fenster",
        "toolbar=0,location=0,scrollbars=0,menu=0,"
        + "dependent,resizable,status,width=150,height=300"
      );
      with(textmarken_fenster.document)
```



```

    {
        // HTML-Fenster formatieren
        window.name = "logischer_fenster_name"
        open("text/html")
        writeln("<HTML><HEAD>")
        writeln(    "<TITLE>Steuerung f&uuml;r &quot;"
                    + document.title
                    + "&quot;</TITLE>"
                )
        writeln(    "<BASE HREF="
                    + location.href
                    + "' TARGET='logischer_fenster_name'"
                )
        writeln("</HEAD><BODY>")

        // Textmarken (Anker im Fenster anzeigen
        for(var i = 0; i < document.anchors.length; i++)
        {
            writeln(    "<P><A HREF='#"
                        + document.anchors[i].name
                        + "'>"
                        + document.anchors[i].name
                        + "</A></P>"
                    );
        }

        writeln("</BODY></HTML>");
        close();
        focus();
    }
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
<!--
    document.write(    "<P><A HREF='javascript: textmarken_fenster_anzeigen ();'"
                    + "Textmarken-Fenster anzeigen</A></P>"
                    );
// -->
</BODY>
</HTML>

```

**Eigenschaften:**

.length Anzahl der Feldelemente also Feldlänge

**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

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.3. applet Objekt des Internet Explorer**

Objekt zum Einbinden ausführbaren Codes auf Basis einer Javamaschine (Virtuelle Javamaschine).

Applet muss alle Eigenschaften und Methoden, die im HTML-Dokument benutzt werden sollen, als **public** deklarieren.

**Java-Applet (\*.class) in HTML einbinden:**

Beispiel

```

<APPLET
    ID="ID_Applet"
    CODE="class_datei"
    CODEBASE="quellverzeichnis"
    HEIGHT=applet_fenster_hoehe_in_pixel
    WIDTH=applet_fenster_breite_in_pixel
    MAYSCRIPT=true oder false
    ALT="alternativer_text"
    ALIGN=ausrichtung
    HSPACE=fenster_abstand_links_und_rechts_von_umgebung
    VSPACE=fenster_abstand_loben_und_unten_von_umgebung
>
    <PARAM Name="parameter_name"
        VALUE=parameter_wert

```



```

>
.....
</APPLET>

```

**Zugriff:**

**ID\_Applet**.eigenschaft  
**ID\_Applet**.methode()

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst
.align	vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt Ausrichtung
.alt	Alternativer Text als Tooltip
.altHTML	HTML-Script, der ausgeführt wird, wenn Objekt nicht geladen werden kann
.archive	Zeichenkette für Archive-Funktionalität eines Objektes
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.code	Url der *.class-Datei (kompilierter Javacode)
.codeBase	Url der Komponente für Dowload der Komponente vom Server auf den Client optionale Versionsprüfung möglich (nicht bei Applet-Komponente) um unnötigen Download zu ersparen: vorheriger Ableich der Version der Komponente auf Server und Client auch wenn der Versionsabgleich keinen Download ergab, wird immer HTTP Header-Transaktion ausgelöst
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hideFocus	Focussierbarkeit
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.object	Zeiger auf das Objekt laut Applet bzw. laut Objekt object
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag



	wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten
.sourceIndex	Index des Objektes in der Collection document.all
.src	Url der Daten z.B. vom Image in normaler Auflösung
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.vspace	vertikaler Abstand in Pixel zum Elternobjekt
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt



	auch für CSS-Layout
	onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode <code>.componentFromPoint()</code> also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben
	Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
<code>.contains()</code>	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
	DOM nicht geändert
<code>.detachEvent()</code>	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode <code>.attachEvent()</code> aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter <code>event_bezeichner</code> zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
<code>.fireEvent()</code>	ein Event auslösen
<code>.focus()</code>	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
<code>.getAdjacentText()</code>	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht
	DOM nicht geändert
<code>.getAttribute()</code>	Wert eines per HTML erzeugten Attributes liefern
	DOM nicht geändert
<code>.getAttributeNode()</code>	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf <code>attribute.name</code> Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar
	DOM nicht geändert
<code>.getBoundingClientRect()</code>	Referenz auf TextRectangle-Objekt im Element holen
<code>.getClientRects()</code>	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
<code>.getElementsByTagName()</code>	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei <code>document.all</code> Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode <code>getElementsById()</code> Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode <code>getElementsByName()</code>
	DOM nicht geändert
<code>.hasChildNodes()</code>	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
	DOM nicht geändert
<code>.insertAdjacentElement()</code>	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
<code>.mergeAttributes()</code>	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
<code>.namedRecordset()</code>	Zeiger desjenigen Datenquellen-Record-Objektes mit Namen, das den Standard-Record-Set enthält Hinweis: Namen zeigt die Mitgliedschaft in einem Datasource-Objekt (DSO) an Eigenschaft <code>.recordset</code> liefert den Zeiger für den Standard-Record-Set in einem Datasource-Objekt (DSO)
<code>.normalize()</code>	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
<code>.releaseCapture()</code>	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : <code>onmousedown</code> , <code>onmouseup</code> , <code>onmousemove</code> , <code>onclick</code> , <code>ondblclick</code> , <code>onmouseover</code> und <code>onmouseout</code> . Hinweis: einschalten per Methode <code>.setCapture()</code>
<code>.removeAttribute()</code>	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode <code>.createAttribute()</code> erzeugte Attribute werden nicht erfasst DOM wird geändert
<code>.removeAttributeNode()</code>	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
<code>.removeBehavior()</code>	per Methode <code>.addBehavior()</code> einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)



.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient. Ausdruck muss mit der Methode <code>.setExpression()</code> gesetzt worden sein
.replaceAdjacentText()	DOM wird nicht geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.scrollIntoView()	DOM wird nicht geändert Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : <code>onmousedown</code> , <code>onmouseup</code> , <code>onmousemove</code> , <code>onclick</code> , <code>ondblclick</code> , <code>onmouseover</code> und <code>onmouseout</code> . ab IE 5.5 Hinweis: ausschalten per Methode <code>.releaseCapture()</code>
.setExpression()	Wert definieren, der als Ausdruck für die Methode <code>.getExpression()</code> zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

**4.3.2.2.4.3.4. document.applets Collection des Internet Explorer**

Feld der Zeiger aller Applet-Objekte im Dokument  
Elementefolge laut HTML-Coding

Applet muss alle Eigenschaften und Methoden, die im HTML-Dokument benutzt werden sollen, als **public** deklarieren

**Erzeugung:**

durch den Browser

**Java-Applet (\*.class) in HTML einbinden:**

```

<APPLET ID="ID_Applet"
  CODE="class_datei"
  CODEBASE="quellverzeichnis"
  NAME="Name_Applet"
  HEIGHT=applet_fenster_hoehe_in_pixel
  WIDTH=applet_fenster_breite_in_pixel
  MAYSCRIPT=true oder false
  ALT="alternativer_text"
  ALIGN=ausrichtung
  HSPACE=fenster_abstand_links_und_rechts_von_umgebung
  VSPACE=fenster_abstand_loben_und_unten_von_umgebung
>
  <PARAM Name="parameter_name"
    VALUE=parameter_wert
  >
  .....
</APPLET>

```

**Syntax:**

```

[ var ZeigerAufFeld = ] document.applets
[ var ZeigerAufFeldElement = ] document.applets[Index [, SubIndex]]

```

Index	Integer ab 0 muss in [ ] kodiert sein
SubIndex	optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

```

ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()

```

```

document.all.ID_Applet.eigenschaft
document.all.ID_Applet.methode()

```



document.all.Name\_Applet.eigenschaft  
document.all.Name\_Applet.methode()

**Eigenschaften:**

.length Anzahl der Feldelemente also Feldlänge

**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

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.5. area Objekt des Internet Explorer**

Area-Tag  
Scripting ab 4.x  
siehe auch Objekt map

**Erzeugung:**

Beispiel 1:

```
<IMG ID="ID_Img"
SRC="test.gif"
USEMAP="#MapName"
>
<MAP ID="ID_Map"
NAME="MapName">
<AREA SHAPE="rect" COORDS="0,0,33,33" HREF="test1.gif">
<AREA SHAPE="circle" COORDS="90,33,3" HREF="test2.gif">
</MAP>
```

Beispiel 2:

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

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

**Zugriff:**

ID\_Map.eigenschaft  
ID\_Map.methode()

**Eigenschaften:**

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt  
.alt Alternativer Text als Tooltip  
ATOMICSELECTION Selektierbarkeit des Objektes einstellen  
.begin Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2  
.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf  
.className Klassenreferenz, Klassenname  
.coords Koordinaten eines Objektes in Abhängigkeit zur Eigenschaft .shape  
.dir Umflussrichtung



.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich						
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2						
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes						
.hash	Teil des Wertes der Eigenschaft .href also Teil der Url als Anker Teil folgt direkt hinter dem Nummernkreuz # und ohne Nummernkreuz						
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2						
.hideFocus	Focussierbarkeit						
.host	<b>Hostname</b> und <b>Port</b> einer Location oder Url in der Form " <b>hostname:port</b> "						
.hostname	<b>Hostname</b> einer Location oder Url in der Form " <b>hostname:port</b> " kann Domain oder IP sein						
.href	Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag) siehe auch Eigenschaften .rel und .rev						
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);						
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.						
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich						
.isMultiLine	Mehrzeiligkeit des Objektinhaltes						
.isTextEdit	Erzeugbarkeit eines Textbereiches						
.lang	Sprache für Anzeige von Sonderzeichen etc.						
.language	Sprache für Script festlegen						
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes						
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes						
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker						
.nodeType	Knotentyp laut attributes Collection						
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)						
.noHref	HREF-Wirkung (Eigenschaft .href) ein/ aus bei Click auf den Link						
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)						
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)						
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth						
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)						
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)						
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar						
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar						
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde						
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM						
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie						
.parentTextEdit	Textbereich des Elternobjektes referenzieren						
.pathname	Datei und Pfad eines Objektes						
.port	<b>Port</b> einer Location oder Url in der Form " <b>hostname:port</b> " basierend auf dem aktuellen <b>Protokoll</b> laut Eigenschaft .protocol z.B. <table> <thead> <tr> <th>Standard-Ports</th> <th>Protokoll</th> </tr> </thead> <tbody> <tr> <td>21</td> <td>FTP</td> </tr> <tr> <td>80</td> <td>HTTP</td> </tr> </tbody> </table>	Standard-Ports	Protokoll	21	FTP	80	HTTP
Standard-Ports	Protokoll						
21	FTP						
80	HTTP						
.previousSibling	Referenz auf das Vorgängerkind						
.protocol	Protokoll-Teil einer Url inklusive http und ftp liefern						
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten						
.scopeName	Namensraum laut XMLNS-Attribut						
.search	Teil des Wertes der 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:						



	<p>Quellseite besitzt HREF mit "...?....."          ruft Zielseite mit diesem HREF auf</p> <p>Zielseite wurde von Quellseite aufgerufen          liest den Teil von HREF hinter dem ? als Zeichenkettendaten</p>
.shape	Form/Gestalt eines Objektes siehe auch Eigenschaft .coords
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.target	Name des Ziel-Fenster bzw. Ziel-Frame
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernen DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt



	auch für CSS-Layout
	onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben
	Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
	DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
	DOM nicht geändert
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur



.releaseCapture()	Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5
.setExpression()	Hinweis: ausschalten per Methode .releaseCapture() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.6. bgsound Objekt des Internet Explorer

Hintergrundmusik (wave, midi, mp3 etc. ) oder Videoclip-Sound	
per HTML-Element	ab IE 3.x
per Script ohne ID-Verwendung	ab IE 4.x
mit ID-Verwendung	ab IE 5.x

Sound ist solange aktiv laut Anzahl der Wiederholungen der Sounderzeugung, bis zum Entladen des des Dokumentes bis zur Stummschaltung (siehe unten)

Alternativen zum bgsound Objekt sind: .style.time2 Behavior  
Windows Media Player 7.1

Hinweis: Wenn ein Fenster mit dem Unterstrichsymbol rechts oben in der Fensterecke minimiert wurde, dann ist es durch Windows in die Hintergrund-Prioritätenfolge eingeordnet worden und arbeitet im Hintergrund bzw. garnicht. Mit anschließendem Maximieren (Symbol in der Fensterleiste rechts oben) wird das Fenster wieder angezeigt und das geladene Dokument **erneut** geöffnet. Konsequenz daraus ist, dass dieses Maximieren einem Neustart des Dokumentes entspricht, auch wenn der Programmierer diesen Zustand nicht erwünscht. Sound, der z.B. mit Start des Dokumentes erzeugt wird, erklingt erneut mit Maximierung nach einer Fensterminimierung. Analogon ist die Fenstereneingung im Browser durch z.B. Ein- und Ausblenden der Favoritenleiste. Dieses Verhalten des Fensters mit seinem Dokument ist aber **nicht identisch** mit dem Verhalten nach einer Fenstergrößenänderung z.B. durch Rahmenverschiebung (resize). Die Aktionen des Fensters und seines Dokumentes bezüglich Resize müssen programmiert werden, sind also nicht standardmäßig vorhanden - im Gegensatz zum Verhalten mit/nach Fensterminimierung/-maximierung per Symbole in der rechten oberen Ecke der Fensterleiste. Sollte ein Frameset minimiert werden, dann wird das für den gesamten Frameset getan (Frameset hat 1 Fenster für alle Frames gesamt), allerdings mit Maximierung wird auch das Frameset-Dokument neu geladen. Wichtig dabei sind für Zeiger-Bezüge der Frames auf den Frameset per parent-Zeiger, dass die Daten im Frameset-Dokument mit Maximierung initialisiert werden und somit ebenfalls Daten der Frames, die im Frameset-Dokument abgelegt wurden, also z.B. Daten, die Verhaltensweisen der Frames vor einer Änderung der Frames-Inhalte gespeichert haben.



Erzeugung:

```

.....
    <BGSOUND
        ID="freies_id"
        SRC="url_oder_dateiname"
        LOOP="infinite" oder Anzahl der Wiederholungen ab 1
        BALANCE =Wert
        VOLUME=Wert
    >
.....

```

Hinweise: Es sind **mehrere** BGSOUND-Objekte (mit verschiedenen ID's) in einem Dokument instanzierbar, die auch **parallel** arbeiten. Man beachte dabei unbedingt die **intensive** Timer-Ressourcennutzung der BGSOUND-Objekte. Es ist dringend anzuraten, das Objekt, dessen Eigenschaft `.loop` auf unendlich belegt wurde, per Urlzuweisung an die Eigenschaft `.src` stumm zuschalten, sobald der Sound nicht mehr benötigt wird: Es wird eine leere Url (Leerkette) zugewiesen. Eine Anwendung der Stummschaltung ist auch in der Realisierung eine Playliste aus Sounds anhand des BGSOUND-Objektes möglich (Folge von Urlzuweisungen, deren zeitlicher Abstand die jeweilige bekannte Sounddauer ist (per Anweisung `window.setTimeout()` lässt sich die Wiedergabezeit der Sounds verwalten)). Die Nutzung von Playlisten-Alternativen zum BGSOUND-Objekt ist nur dann nötig, wenn Steuerungselemente zur Media-Datei oder zur Lautstärkeinstellung, die über den Rahmen der windowseigenen gehen, genutzt werden sollen.

Jede Urlzuweisung an die Eigenschaft `.src` bewirkt den automatischen Neustart der Medium-Wiedergabe per BGSOUND-Objekt mit den aktuellen Einstellungen der Objekt-Eigenschaften sowie den Einstellungen innerhalb der Media-Datei. Wurde `.src` mit einer Leerkette belegt, so wird kein Sound wiedergegeben (Stummschaltung ohne Veränderung der windowseigenen Lautstärkeregelung).

Für **jeden** Medientyp ist ein **eigenständiges** BGSOUND-Objekt zu kodieren. Werden verschiedene Medientypen wie z.B. Midi und mp3 per gemeinsamen BGSOUND-Objekt verwaltet, so **kann** es passieren, dass die aktuelle Reglereinstellung zum Medientyp (z.B. mp3/wave) der windowseigenen Lautstärkeinstellung für die Soundwiedergabe nicht korrekt so belassen wird, wie sie der User dort permanent eingestellt hat, sondern für die Wiedergabe des Medientyps unerwünscht verändert wird, egal ob die Eigenschaft `.volume` des BGSOUND-Objektes durch den Programmierer verändert wurde oder nicht.

Das Vorladen eines Sounds per Script geht nur per BGSOUND selbst, da es keine Funktion analog zu `new Image()` gibt.

BGSOUND muss mit VOLUME auf -10 erzeugt werden (also stumm), wobei die Zuweisung zu `.src` das Laden in den Browser-Cache bewirkt,

Zugriff:

z.T. erst ab IE 5.x

```
<BGSOUND ID="freies_id" .....>
```

in Script dann `freies_id` verwenden per `freies_id.eigenschaft` etc.

Beispiele für LOOP-Varianten:

<code>&lt;BGSOUND SRC="file:///c:\test\wav\test.wav"&gt;</code>	genau 1 mal
<code>&lt;BGSOUND SRC="file:///c:\test\wav\test.wav" LOOP&gt;</code>	genau 1 mal
<code>&lt;BGSOUND SCR="file:///c:\test\wav\test.wav" LOOP=&gt;</code>	genau 1 mal
<code>&lt;BGSOUND SCR="file:///c:\test\wav\test.wav" LOOP=0&gt;</code>	genau 1 mal
<code>&lt;BGSOUND SCR="file:///c:\test\wav\test.wav" LOOP=1&gt;</code>	genau 1 mal
<code>&lt;BGSOUND SCR="file:///c:\test\wav\test.wav" LOOP=10&gt;</code>	genau 10 mal
<code>&lt;BGSOUND SCR="file:///c:\test\wav\test.wav" LOOP=-1&gt;</code>	endlos

```

<HEAD>
<SCRIPT>
    function KanalVerteilung(Wert)
    { ID_bgsound.balance = Wert;}

    function GenauEinmal()
    {
        ID_bgsound.loop = 1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }

    function Endlos()
    {
        ID_bgsound.loop = -1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>

```



```

<BUTTON onclick="Endlos()"></BUTTON>
<BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
<BUTTON onclick="KanalVerteilung(10)"></BUTTON>
<BUTTON onclick="KanalVerteilung (0)"></BUTTON>
<B>Volume control:</B>&nbsp;
<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-10;"
>Mute

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-7;"
>25% Volume

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe" CHECKED
onpropertychange="ID_bgsound.volume=-5;"
>50% Volume

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-2;"
>75% Volume

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

```

Beispiel für Sound mit Sekundenanzeige:

```

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

var PixelBreiteProBalkenErweiterung = 10;

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

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

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

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

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

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

function SekundenZaehlen_Stop()
{
    // prüfen ob Sekundenzählen aktiv ist
    if (SekundenZahlerTimeoutID != null)
    {
        // aktiv, also stoppen

```



```

clearInterval(SekundenZahlerTimeoutID);
SekundenZahlerTimeoutID = null;
}
}

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

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

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

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

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

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

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

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

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

    // ---- visuelle Anzeige erzeugen
    // - - - Sekundenbalken und Sekundenzähler dynamisch visualisieren
    // Es wird jedem DIV als Style-Wert ein Ausdruck hinterlegt, also kein Wert.
    // Der Ausdruck liefert den Wert , welcher sofort das Layout der
    // DIV's beeinflusst.
    // Jeder Ausdruck besitzt den SekundenZahler als Komponente.
    // Damit ändert sich der Wert des Ausdrucks.
    // Für die Neuberechnung des Ausdruckes ist der Aufruf von
    // document.recal()
}

```



```

//          nötig.
//          Dieser Aufruf erfolgt in SekundenZaehlen(), also permanent pro Sekunde.
//          Damit wird der Style-Wert permanent neu berechnet.
//          Damit visualisieren sich die DIV's permanent neu.

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

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

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

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

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

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

                    );

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

                    );

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

                    );

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

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

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

```

Beispiel für Start und Stop des Sounds:

Das Objekt bgsound besitzt **keine** Methoden und auch nicht für Start und Stop des Sounds. Folgendes Beispiel schafft aber Abhilfe:

```

function SoundObjektErzeugen(Kette,Wert)
{
    this.SoundFileUrl = Kette;
    this.SoundFileDauerInSekunden = Wert;
}

```



```

function SoundStummInstanzieren()
{
    document.write( '<BGSOUND ID="ID_BGSound"'
        + ' SRC="' + StummSoundObjekt.SoundFileUrl + '"'
        + ' LOOP="1"'
        + '>'
    ); // alternativ auch SRC-Attribut weglassen
}

var SoundEndlosAbspielenAktiv=false;

function SoundEndlosAbspielen()
{
    ID_BGSound.src= SoundObjekt.SoundFileUrl;
    ID_BGSound.loop='infinite';
}

function SoundStoppen()
{
    ID_BGSound.src= StummSoundObjekt.SoundFileUrl; // alternativ Leerkette zuweisen
    ID_BGSound.loop='1';
}

SoundDauerInSekunden = 88;
SoundUrl = "sound/test88.mid";
StummSoundUrl = "sound/stumm.mid"; // leere MIDI-Datei ohne Daten, also ohne Tonwiedergabe

// +++++ Sound vorladen durch Erzeugung von Objekten
SoundObjekt = new SoundObjektErzeugen(SoundUrl,SoundDauerInSekunden);
StummSoundObjekt = new SoundObjektErzeugen(StummSoundUrl,0);
SoundStummInstanzieren();

// durch Aufruf von SoundEndlosAbspielen() wird der Sound erneut bzw. überhaupt gestartet
// durch Aufruf von SoundStoppen() wird der Sound gestoppt

```

**Alternative zu BGSOUND:**

Beispiel: Windows Media Player verwenden:

Achtung: Der Windows-Media-Player benutzt **sehr intensiv** die Timer von Windows, so dass parallele Rekursionen z.B. per setTimeout() nicht kontinuierlich ablaufen können !

```

function Y_SoundEndlosAbspielen()
{
    document.write( '<EMBED NAME="ID_WindowsMediaPlayer"'
        + ' SRC="' + SoundObjekt.SoundFileUrl + '"'
        + ' LOOP="infinite"'
        + ' STYLE="width=0px;height=0px"'
        + '>'
    );
}

SoundDauerInSekunden = 88;
X_SoundUrl = "sound/test88.mid";

// +++++ Sound vorladen durch Erzeugung von Objekten
SoundObjekt = new SoundObjektErzeugen(SoundUrl,SoundDauerInSekunden);

```

Hinweis: Durch Veränderung von Style-Angaben width und height erscheint das Fenster des MediaPlayer inklusive der Bedienungsknöpfe, die je nach Medium aktivierbar sind.  
Werden die Style-Angaben width bzw. height **nicht** angegeben, so erscheint der Windows Media Player im Standardfensterumfang.

Per **ID\_WindowsMediaPlayer** sind Eigenschaften und Methoden des Player nutzbar.

Beispiel: Sound anstelle bgsound Objekt mit Timeline erzeugen:

```

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

```





```

    {
        ID_bgsound.loop = 1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }

function Endlos()
{
    ID_bgsound.loop = -1;
    ID_bgsound.src = ID_bgsound.src; // restart
}
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>
    <BUTTON onclick="Endlos()"></BUTTON>
    <BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung(10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung (0)"></BUTTON>
    <B>Volume control:</B>&nbsp;
    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-10;"
    >Mute

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-7;"
    >25% Volume

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED
        onpropertychange="ID_bgsound.volume=-5;"
    >50% Volume

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-2;"
    >75% Volume

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

```

.disabled

Interaktionsfähigkeit

.id

nur wenn sichtbar so User-Interaktion möglich

.isDisabled

Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)

Interaktionsfähigkeit

.loop

nur wenn sichtbar so User-Interaktion möglich

Anzahl der Wiederholungen

nur Objekt BGSOUND bzw. IMG mit Sound bzw. INPUT-Element mit Sound

-1 endlos

0 genau 1 mal

> 0 Anzahl der Wiederholungen

Standard ist 1

1 entspricht 0

Beispiel:

```

<HEAD>
<SCRIPT>
function KanalVerteilung(Wert)
{ ID_bgsound.balance = Wert;}

function GenauEinmal()
{
    ID_bgsound.loop = 1;
    ID_bgsound.src = ID_bgsound.src; // restart
}

function Endlos()
{
    ID_bgsound.loop = -1;
    ID_bgsound.src = ID_bgsound.src; // restart
}
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>

```



```

<BUTTON onclick="Endlos()"></BUTTON>
<BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
<BUTTON onclick="KanalVerteilung(10)"></BUTTON>
<BUTTON onclick="KanalVerteilung (0)"></BUTTON>
<B>Volume control:</B>&nbsp;&nbsp;&nbsp;
<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-10;"
>Mute

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-7;"
>25% Volume

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe" CHECKED
onpropertychange="ID_bgsound.volume=-5;"
>50% Volume

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-2;"
>75% Volume

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

```

.readyState aktueller Status des Objektes beim Füllen des Objektes mit Daten

.src Url der Daten

.volume zulässige Sound-Dateisuffixe sind z.B. wav, mid (laut MIME-Typen des Internet Explorers)

aktuelle Wiedergabe-Lautstärke (Run time volume) per Objekt bgsound, wenn die Media-Datei nicht selbst Lautstärke-Einstellungen während der Wiedergabe vornimmt:

Bsp.: MIDI Volume ohne Wirkung  
WAVE Volume soll Wirkung haben

Hinweis: eventuelle Veränderung der Regler in der Windows-Lautstärke-Regelung beachten.

-10 bis 0  
0 maximal laut

Beispiel:

```

<HEAD>
<SCRIPT>
function KanalVerteilung(Wert)
{ ID_bgsound.balance = Wert;}

function GenauEinmal()
{
    ID_bgsound.loop = 1;
    ID_bgsound.src = ID_bgsound.src; // restart
}

function Endlos()
{
    ID_bgsound.loop = -1;
    ID_bgsound.src = ID_bgsound.src; // restart
}
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
<BUTTON onclick="GenauEinmal()"></BUTTON>
<BUTTON onclick="Endlos()"></BUTTON>
<BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
<BUTTON onclick="KanalVerteilung(10)"></BUTTON>
<BUTTON onclick="KanalVerteilung (0)"></BUTTON>
<B>Volume control:</B>&nbsp;&nbsp;&nbsp;
<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-10;"
>Mute

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-7;"
>25% Volume

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe" CHECKED
onpropertychange="ID_bgsound.volume=-5;"
>50% Volume

```



```

<INPUT TYPE ="radio" NAME ="ID_RadioButtonGruppe"
      onpropertychange="ID_bgsound.volume=-2;"
>75% Volume

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

```

**Methoden:**

keine

**4.3.2.2.4.3.7. document.body Objekt des Internet Explorer**

Rumpf des HTML-Dokumentes

Im Dokument gibt es genau 1 Rumpf also genau 1 Body Objekt

Die Ausführung von document.write() bzw. document.writeln(), das **HTML-Tags** erzeugt, hat 2 Konsequenzen und zwar genau dann, wenn diese Anweisung **nach dem kompletten Laden des Dokumentes, also nach Auslösung des Ereignisses onload** aktiviert wird:

**Fakt 1:**

Die **ERSTE** Erzeugung eines HTML-Tags bewirkt das **automatische Öffnen eines neuen HTML-Dokumentes**, wenn das aktuelle Dokument **bereits komplett geladen**, also das Ereignis onload **bereits** ausgelöst wurde. Letzteres ist immer dann der Fall, wenn der BODY-Teil des Dokumentes komplett geparkt wurde. Grund: Ein komplett geparktes Dokument kann per write() bzw. writeln() nicht um HTML-Elemente verändert werden, da diese Methoden keine des HTML-DOM sind. Mit anderen Worten: Nur die Verwendung von Methoden des HTML-DOM lassen eine **nachträgliche** HTML-Elemente-Veränderung des Dokumentes zu.

Die Methoden write() und writeln() erzeugen einen Datenstrom aus HTML-Elementen und das neue Dokument empfängt diesen so, als würde er aus einer HTML-Datei stammen. Mit Ende des Datenstromes wird quasi ein Dateieinde erkannt und das neue Dokument löst das Ereignis onload aus. Damit wird aber das neue Dokument zum aktuellen Dokument, also im Fenster über dem des alten Dokumentes angezeigt. Da das Fenster des neuen Dokumentes automatisch erzeugt wurde (nicht per Methode open()), sind also die Standards für eine Fenstererzeugung verwendet worden. Damit hat das neue Dokument einen History-Eintrag. Der User kann nun mit diesem zwischen dem alten und neuen Dokument umschalten.

**Fakt 2:**

Im Falle der o.g. nachträglichen Veränderung des Dokumentes um HTML-Elemente per write() bzw. writeln() kennt das neue, automatisch geöffnete Dokument das alte Dokument nur **als Eltern**. Es muss also im neuen Dokument mit dem Zeiger auf die Eltern gearbeitet werden, wenn Daten und Routinen der Eltern benutzt werden sollen (siehe Objekt window bzw. Objekt document). Mit anderen Worten: Das neue Dokument muss dann komplett per Script erzeugt werden, denn dieser Zeiger lässt sich nur über Script ansprechen. Jedes geladene Dokument hat ansonsten seine eigenständige Umgebung.

Wird versucht, per document.write() ein weiteres BODY-Tag zu erzeugen, so wird ein neues Fenster geöffnet und ein leeres Dokument geladen, das damit aktiv wird, oder es erfolgt eine Fehlermeldung.

**Erzeugung in HTML:**

Beispiel:

```

<BODY ID="ID_Body" ....>
....
</BODY>

```

NS und IE **können** den BODY verschieden erzeugen:

Nur der Internet Explorer erzeugt **automatisch** ein BODY-Objekt, wenn dieses im Dokument nicht kodiert wurde, weil z.B. sämtliche HTML-Elemente im HEAD des Dokumentes per Script erzeugt wurden.

Der NS verlangt immer einen BODY-Teil und ignoriert Script-Anweisungen im HEAD des Dokumentes, die vor dem Parsen von </BODY> bereits HTML-Elemente erzeugen wollen (z.B. per document.write()).

Es können also **nur** nachträglich neue HTML-Elemente per Script im HEAD des Dokumentes erzeugt werden.

**Zugriff:**

```

document.all.body.eigenschaft
document.all.body.methode()
document.all.ID_Body.eigenschaft
document.all.ID_Body.methode()

```

Beispiel 1 für Ausdruck des aktuellen Dokumentes:

```

<BODY>
      <INPUT TYPE="button" VALUE="Drucken"onclick="javascript:self.print()">
</BODY>

```

Beispiel 2 für Dokument ohne linken und oberen Standard-Seitenrand:

```

<BODY LEFTMARGIN=0 TOPMARGIN=0>

```

Hinweis: Beim Netscape muss kodiert werden

```

<BODY MARGINWIDTH=0 MARGINHEIGHT=0>

```



Beispiel 3 für Sound mit Sekundenanzeige:

```

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

var PixelBreiteProBalkenErweiterung = 10;

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

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

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

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

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

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

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

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

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

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

```



```

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

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

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

// Sound zu Ende
SoundObjekt.SoundFileBeendet=true;

// Sekundenzähler stoppen
SekundenZaehlen_Stop();

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

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

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

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

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

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

// ##### Dieser Teil wird mit dem Laden des Dokumentes abgearbeitet #####
if (ie)
{

```



```

document.write('<BODY></BODY>');

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

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

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

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

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

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

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

```

**Hinweise:**

bezüglich der Erzeugung von HTML-Elementen per HTML oder Script:

HTML-Elemente können per HTML-Tag **oder** per Methode `.createElement()` erzeugt werden. Die Behandlung von per HTML-erzeugten Elementen kann von der per DOM-Methoden erzeugten **abweichen**, da DOM die Art der Erzeugungen **differenziert**. Für HTML-erzeugte Objekte gibt es **immer noch** eigene Methoden (Spezialfälle), obwohl das nicht nötig wäre (vermutlich wegen der schrittweisen Erweiterung des DOM auf alle Elemente im Dokument). Empfehlung: Man verwende - wenn möglich - **nur** Methoden, die HTML- **und** Script-erzeugte Elemente bearbeiten und erreicht dadurch die Vereinheitlichung im Quellcode.

`.innerHTML` ruft intern `.createElement()` auf

Hinweis zur Pars-Reihenfolge des Internet Explorer innerhalb der HTML-Kodierung bezüglich dem

Tag-Name und den Element-Attributen CLASS, ID, STYLE

1. Element-Bezeichner
2. CLASS-Attribut mit Bezeichner aus Klassendeklaration im HEAD
3. ID-Attribut
4. STYLE-Attribut mit Style-Werten (nicht mit Bezeichner aus Style-Deklaration im HEAD)

Es gilt: Wenn gleiche Bezeichner verwendet, so nur Werte des **zuletzt** geparsten Bezeichners verwendet !

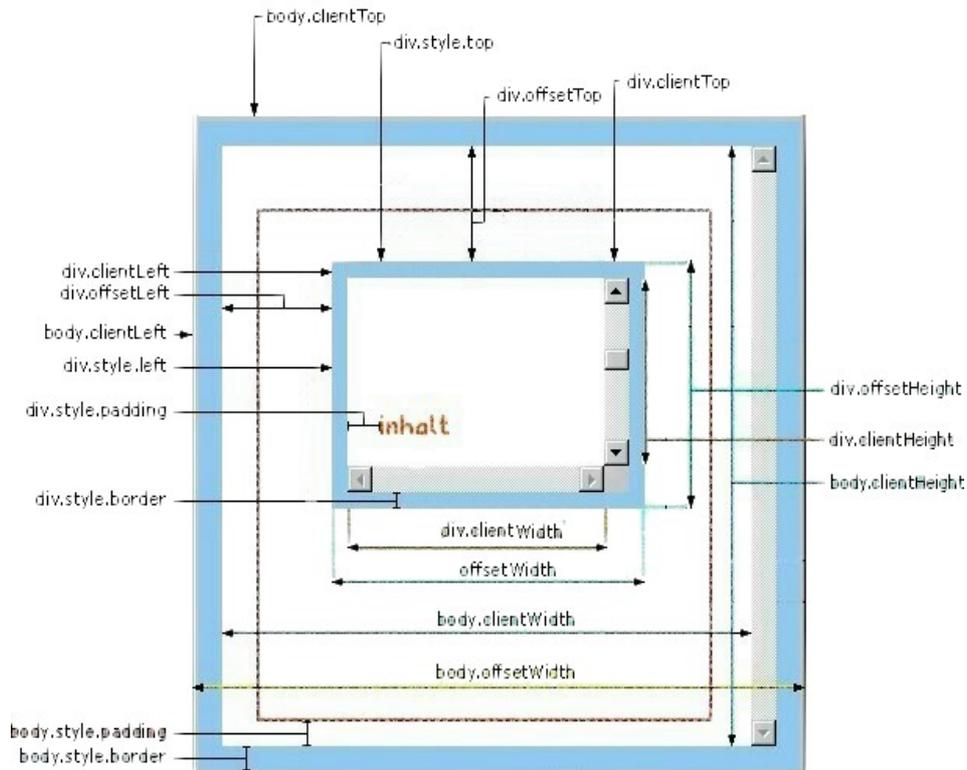
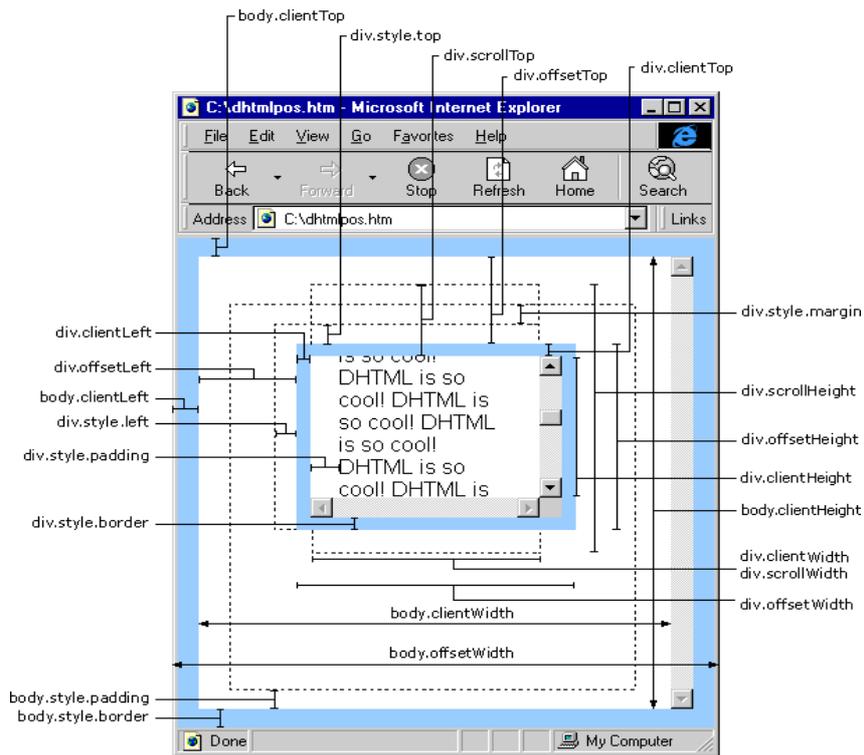
Die CLASS-Deklaration aus dem HEAD des Dokumentes wird wertmäßig durch die Style-Deklaration per Attribut des HTML-Elementes überschrieben, wenn gleiche Style-Eigenschaften betroffen sind (ansonsten hinzufügen).

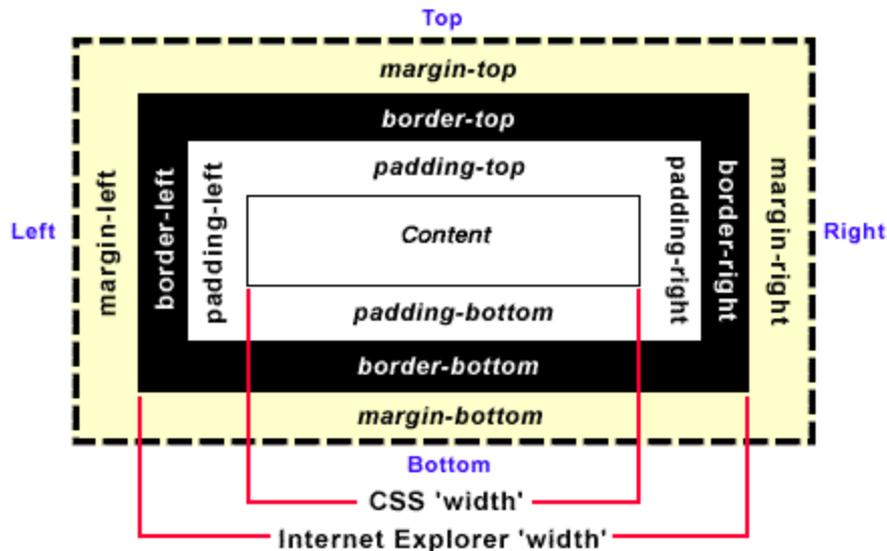
Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:

Die zuletzt während der Laufzeit getätigte Defintion ersetzt wertmäßig den aktuellen Attributwert, wenn gleiche Attributnamen/Eigenschaften betroffen sind (sonst hinzufügen).



Die Eigenschaften des IE als Grafiken:





Hinweis Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
 Padding: Abstand des Objektinhaltes zum Aussenrand

**4.3.2.2.4.3.7.1. Eigenschaften beim Internet Explorer**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.aLink	Farbe eines ALinks
ATOMICSELECTION	Selektierbarkeit einstellen
.background	Hintergrundbild
.bgColor	deprecated und durch STYLE-Attribut zu ersetzen Hintergrundfarbe des Objektes bzw. Dokumentes
.bgProperties	Scroll-Eigenschaften des Hintergrundbildes beim Dokumentscrollen
.blockDirection	Umfluss um ein Objekt
.bottomMargin	Bottom Margin in Pixel
.className	Klassenreferenz
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.



.language	Sprache für Script festlegen
.leftMargin	Left Margin in Pixel des Dokumentes
.link	Farbe eines Links
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.noWrap	Wortumbruch einstellen
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von offsetHeight, offsetLeft, offsetTop und offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.rightMargin	Right Margin in Pixel des Documentes
.scopeName	Namensraum laut XMLNS-Attribut
.scroll	Scrollbar-Anzeige ein/aus vor IE 6.x nur BODY-Objekt ab IE 6.x alle HTML-Elemente wenn DOCTYPE im Kopf des Dokumentes kodiert
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.text	Textfarbe (Vordergrundfarbe)
.timeStartRule	deprecated Startpunkt der Timeline
.title	Tooltip-Text bei Mouse over über Objekt
.topMargin	Top Margin in Pixel des Dokumentes
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektierbarkeit z.B. mit Maus <b>wird nicht an Kinder vererbt</b> bei Wechsel: vorhergehende vorhandene Selektion wird nicht verändert
.vLink	Farbe eine VLINK



**4.3.2.2.4.3.7.2. Methoden beim Internet Explorer**

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createControlRange()	Selektionsbereich erzeugen es kann nur genau 1 Range pro Zeitpunkt existieren: wenn einer vorhanden, so diesen überschreiben
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.doScroll()	Click auf Scrollbar simulieren Achtung: Scrollelemente müssen bereits vorhanden sein !
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle



.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.pause()	deprecated auf Timeline pausieren
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes
.removeAttribute()	Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft.dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu



ersetzenden Text liefern

.replaceChild() DOM wird nicht geändert  
Kind-Objekt ersetzen durch ein Objekt  
ersetzende Objekt muss per Methode createElement() erzeugt worden sein  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde

.replaceNode() DOM wird geändert  
Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern  
sichtbar erst mit parsen des Endetags

.resume() DOM wird geändert  
deprecated

.setActive() Pause auf der Timeline beenden  
Objekt für die Eventdurchreichung aktivieren  
aber ohne es zu fokussieren  
und ohne es scrollbar zu machen

.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

.setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert

.setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.  
ab IE 5.5

.setExpression() Hinweis: ausschalten per Methode .releaseCapture()  
Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
Eigenschaft als Objektreferenz der Form  
objekt.style.eigenschaft.  
dient  
Ausdruck nur als Script kodierbar

.swapNode() DOM wird nicht geändert  
Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.7.3. document.body.timeAll Collection des Internet Explorer**

Feld der Zeiger aller per Behavior time2 verwalteten Elemente (getimte Elemente)  
siehe auch style.time2 Behavior und .timeChildren Collection

**Erzeugung:**

durch den Browser

**Syntax:**

```
[ var ZeigerAufFeld = ] document.all.body.timeAll
[ var ZeigerAufFeldElement = ] document.all.body.timeAll [Index]

[ var ZeigerAufFeld = ] document.all.ID_Body.timeAll
[ var ZeigerAufFeldElement = ] document.all.ID_Body.timeAll [Index]
```

Index Integer ab 0  
oder String Name oder ID des Elementes  
muss in [ ] kodiert sein

ZeigerAufFeldElement.eigenschaft  
ZeigerAufFeldElement.methode()

**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 !!!

**4.3.2.2.4.3.8. button Objekt des Internet Explorer**

HTML-Container und Objekt-Prototyp eines Buttons z.B. auch Button in einem Formular oder Button per Objekt input  
Wird Button innerhalb eines Formulars für das Senden benutzt, so  
wird der Wert laut Eigenschaft .innerText gesendet  
muss das NAME-Attribut für das Button ebenfalls kodiert sein wie für das Formular an sich !).

**Zugriff:**

Beispiel 1:

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



```

        SRC="test.wmv"
        STYLE="position:absolute;top:50px;height:100px"
    >
</t:VIDEO>
<SPAN ID="ID_Span"
    STYLE="position:absolute;top:165px;"
    >
</SPAN>
<BUTTON ID="ID_Button"
    onclick="ID_Span.innerText= ID_Video.abstract"
    >
        Klick
</BUTTON>
</BODY>
</HTML>

```

Beispiel 2:

```

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

```

Beispiel 3:

```

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

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

```

Beispiel 4:

```

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

```

Beispiel 5:

```

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

```

Beispiel 6:

```

<P ID="ID_P">Dieser Text wird verändert</P>
<BUTTON onclick=" ID_P.innerText='Neuer Text'">Neuer Text</BUTTON>
<BUTTON onclick=" ID_P.innerText= Dieser Text wird verändert">Reset</BUTTON>

```

Beispiel 7:

```

<HEAD>
<SCRIPT>
    function EditierbarkeitWechseln()

```



```

    {
        var Editierbarkeit = ID_Span1.isContentEditable;
        var Editierbarkeit_Negation = ! Editierbarkeit;
        ID_Span1.contentEditable = Editierbarkeit_Negation;
        ID_Span2.innerText = Editierbarkeit_Negation;

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

```

Beispiel 8:

```

<HEAD>
<SCRIPT>
    function KanalVerteilung(Wert)
    { ID_bgsound.balance = Wert;}

    function GenauEinmal()
    {
        ID_bgsound.loop = 1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }

    function Endlos()
    {
        ID_bgsound.loop = -1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>
    <BUTTON onclick="Endlos()"></BUTTON>
    <BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung(10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung (0)"></BUTTON>
    <B>Volume control:</B>&nbsp;
    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-10;"
    >Mute

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-7;"
    >25% Volume

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED
        onpropertychange="ID_bgsound.volume=-5;"
    >50% Volume

    <INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-2;"
    >75% Volume

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

```

Beispiel 9 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>

```

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataFormatAs	Datenquelle-Anzeigeart
.dataSrc	Datenquelle als Anker festlegen
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!)



	muss beim Formular für alle zu sendenden Felder kodiert sein !!
	Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrolling	Scrollenbar erzeugen
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup  Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes



	siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Art des Buttons (Standard-Behavior des Buttons)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar



	DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
.hasChildNodes()	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh) prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
.insertAdjacentElement()	DOM nicht geändert Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich
.insertAdjacentHTML()	DOM wird geändert HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
.insertAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes
.insertBefore()	DOM wird geändert Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.mergeAttributes()	DOM wird geändert alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
.normalize()	DOM wird geändert Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
.removeAttributeNode()	DOM wird geändert entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
.removeBehavior()	DOM wird geändert per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
.removeChild()	DOM wird geändert Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
.removeNode()	DOM wird nicht geändert Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern



	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
	DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein
	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
	DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird
	Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern
	DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
	ab IE 5.5
	Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar
	DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.9. comment Objekt des Internet Explorer

Kommentar-Objekt

Kommentar wird nicht angezeigt (gerendert)

##### **Eigenschaften:**

.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.data	Url der Daten des Objektes
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.length	Anzahl der Zeichen im Kommentar (comment-Objekt), ab IE 6.x
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie



.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.text	Text eines Objektes
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

**Methoden:**

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.appendData()	String an das Ende des Objektes anhängen
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.deleteData()	Teilkette aus einem Objekt entfernen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann



	wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.insertData()	Teilkette in ein Objekt einfügen
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceData()	Teilkette in einem Objekt ersetzen
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.substringData()	Teilkette aus einem Objekt lesen
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.10. div Objekt des Internet Explorer

ab IE 4.x

Das DIV- bzw. SPAN-Objekt ähneln sich sehr stark. DIV kann ein automatisches <BR> am DIV-Ende implizieren.

Die übliche Zugriffsweise ist über das ID-Attribut ( Eigenschaft .id) des DIV bzw. SPAN.

DIV bzw. SPAN können im Elternobjekt (Dokument im Fenster oder übergeordneter DIV bzw. SPAN) erzeugt werden, als HTML-Element üblicherweise im BODY-Teil des Dokumentes



per Script durch z.B. document.write("<DIV ..... >...");

Die Anzeige (das Rendern) des DIV bzw.SPAN erfolgt erst mit dem Parsen des Ende-Tags, also </DIV> bzw. </SPAN>. Danach können Komponenten zur Laufzeit nachträglich verändert werden, die aber nur z.T. sofort sichtbar werden (teilweise erst nach dem Reload des betreffenden Dokumentes).

NS und IE können das DIV- und SPAN-Objekt erzeugen, implementieren und rendern diese aber verschiedenartig.

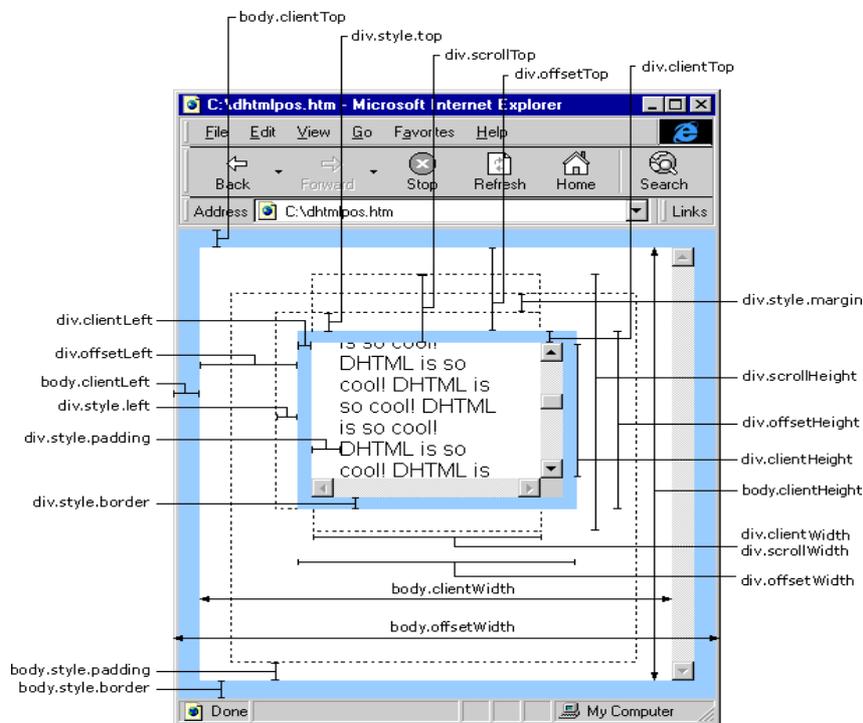
Beispiel: Nur der Internet Explorer erzeugt automatisch ein BODY-Objekt, wenn dieses im Dokument nicht kodiert wurde, weil z.B. sämtliche HTML-Elemente im HEAD des Dokumentes per Script erzeugt wurden.

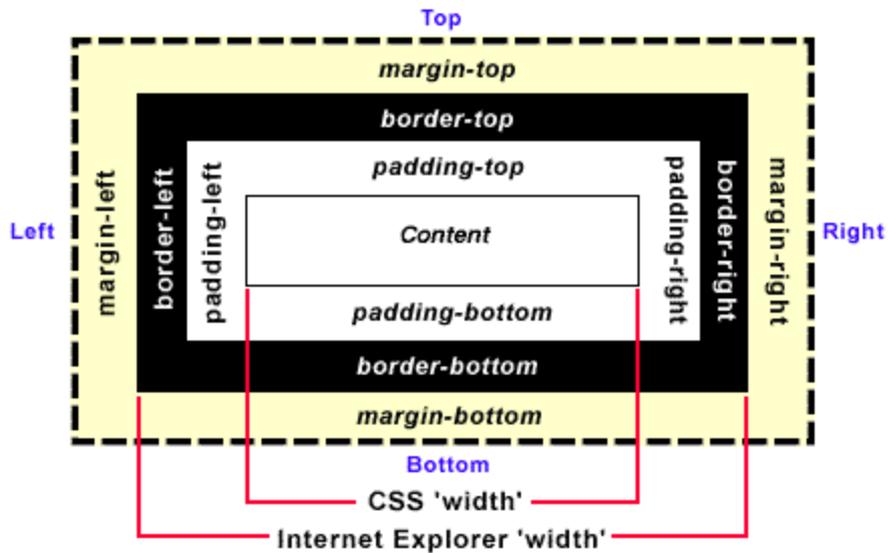
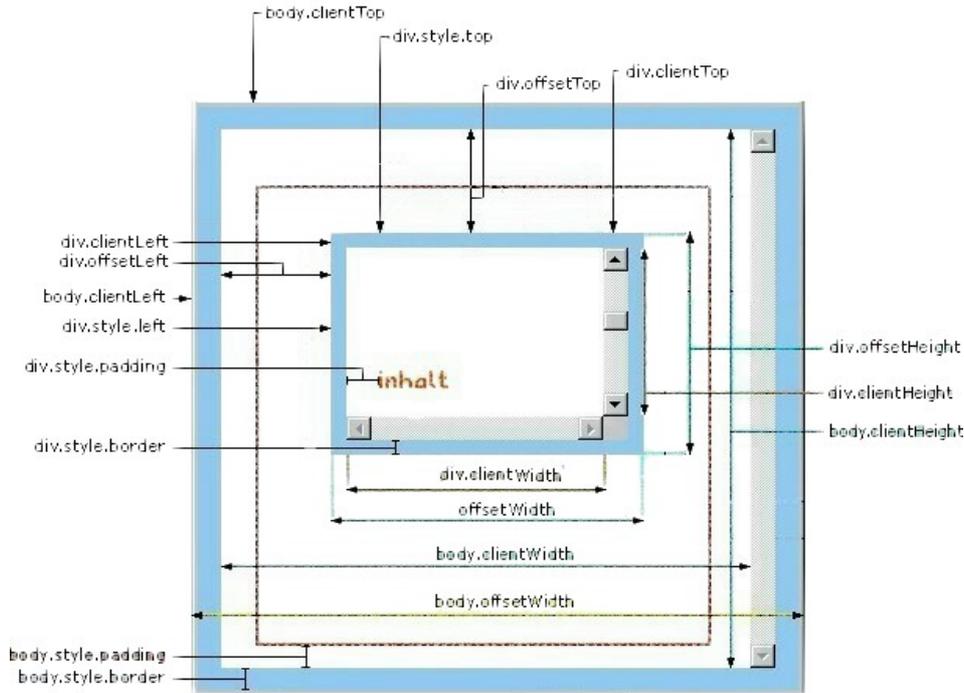
Der NS verlangt immer einen BODY-Teil und ignoriert o.g. Script-Anweisungen im HEAD des Dokumentes, solange der BODY des Dokumentes nicht komplett geparkt wurde (</BODY> noch nicht erkannt wurde). Es können also nur nachträglich neue HTML-Elemente per Script im HEAD des Dokumentes erzeugt werden. Scriptaufrufe aus dem BODY-Teil werden mit dem BODY geparkt, egal ob die Scripte im HEAD oder BODY liegen.

Im NS unter der Version 6.x wird anstelle von DIV- bzw. SPAN-Objekt das LAYER-Objekt favorisiert. Ab NS 6.x wird das Layer-Objekt radikal nicht mehr unterstützt: Es sind für die Ebenendarstellung (überlappende Objekte) keine Layer mehr sondern z.B. DIV oder SPAN verwendbar.

Bezüglich zeitsteuernder Eigenschaften/Methoden werden ab IE 6.x auch Behavior style.time2 und Objekt currTimeState benutzt.

Wichtige Eigenschaften im IE:





Hinweis Margin: Abstand des Aussenrandes eines Objektes zur Umgebung  
 Padding: Abstand des Objektinhaltes zum Aussenrand

Hinweis zur Pars-Reihenfolge des Internet Explorer innerhalb der HTML-Kodierung bezüglich dem Tag-Name und den Element-Attributen CLASS, ID, STYLE

1. Element-Bezeichner
2. CLASS-Attribut mit Bezeichner aus Klassendeklaration im HEAD
3. ID-Attribut
4. STYLE-Attribut mit Style-Werten (nicht mit Bezeichner aus Style-Deklaration im HEAD)

Es gilt: Wenn gleiche Bezeichner verwendet, so nur Werte des **zuletzt** geparsen Bezeichners verwendet !  
 Die CLASS-Deklaration aus dem HEAD des Dokumentes wird wertmäßig durch die Style-Deklaration per Attribut des HTML-Elementes überschrieben, wenn gleiche Style-Eigenschaften betroffen sind (ansonsten hinzufügen).

Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:

Die zuletzt während der Laufzeit getätigte Definition ersetzt wertmäßig den aktuellen Attributwert, wenn gleiche Attributnamen / Eigenschaften betroffen sind (sonst hinzufügen).

**Beispiele:**



Beispiel 1:

```

<HTML>
<HEAD>
<SCRIPT>
    function HandlerFuerOnMoveStart()
    {
        // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
        var ZeigerAufObjektMitEvent = event.srcElement;

        ZeigerAufObjektMitEvent.style.backgroundColor = "green";
        ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
    }

    function HandlerFuerOnMove ()
    {
        ID_Span1.innerHTML = event.srcElement.offsetLeft;
        ID_Span2.innerHTML = event.srcElement.offsetTop;
    }

    function HandlerFuerOnMoveEnd()
    {
        // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
        var ZeigerAufObjektMitEvent = event.srcElement;

        ZeigerAufObjektMitEvent.style.backgroundColor = "red";
        ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
    }

    // 2-D Positionierung einschalten
    document.execCommand("2D-position",false,true);
</SCRIPT>
</HEAD>
<BODY onmovestart="HandlerFuerOnMoveStart();"
onmove="HandlerFuerOnMove();"
onmoveend="HandlerFuerOnMoveEnd();"
>
    offsetLeft = <SPAN ID="ID_Span1"></SPAN>
    <BR>
    offsetTop = <SPAN ID="ID_Span2"></SPAN>
    <BR>
    <DIV CONTENTEDITABLE="true">
        <DIV STYLE=
            "position:absolute;width:300px;height:100px; background-color:red;"
        >
            bewegbarer DIV
        </DIV>
    </DIV>
</BODY>
</HTML>

```

Beispiel 2 für Sound mit Sekundenanzeige:

```

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

    var PixelBreiteProBalkenErweiterung = 10;

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

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

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

```



```

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

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

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

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

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

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

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

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

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

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

        // Sound zu Ende
        SoundObjekt.SoundFileBeendet=true;

        // Sekundenzähler stoppen
        SekundenZaehlen_Stop();

        // und Meldung
        var TimerUngenauigkeit1 = SoundDauerInSekunden - SekundenZahler;

```



```

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

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

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

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

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

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

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

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

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

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

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

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

```



```

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

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

```

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.align	Ausrichtung
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.blockDirection	Umfluss um ein Objekt
.className	Klassenreferenz
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataFormatAs	Datenquelle-Anzeigeart
.dataSrc	Datenquelle als Anker festlegen
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.dur	Dauer Objektaktivitäten laut Eigenschaft .timeAction alternativ: Eigenschaft .end siehe Objekt currTimeState und Behavior .style.time2
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.noWrap	Wortumbruch einstellen
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)



.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scroll	Scrollbar-Anzeige ein/aus vor IE 6.x nur BODY-Objekt ab IE 6.x alle HTML-Elemente wenn DOCTYPE im Kopf des Dokumentes kodiert
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
SYSTEMLANGUAGE	Sprache festlegen für das Objekt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA  Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.timeAction	Aktion des Objektes in der Timeline Aktion nur ausführbar wenn Objekt aktiv ist UND Timeline aktiv ist siehe Objekt currTimeState und Behavior .style.time2
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
UNSELECTABLE	Selektionsfähigkeit eines Objektes
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn



	die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar
.click()	DOM wird geändert simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.doScroll()	Click auf Scrollbar simulieren Achtung: Scrollelemente müssen bereits vorhanden sein !
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.hasFocus()	Objekt im Focus-Zustand
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert



.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-



Eigenschaft als Objektreferenz der Form  
 objekt.style.eigenschaft.  
 dient  
 Ausdruck nur als Script kodierbar  
 DOM wird nicht geändert  
 .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
 nur sichtbar wenn Endetag geparkt  
 DOM wird geändert

**4.3.2.2.4.3.11. document.embeds Collection des Internet Explorer (vermutlich auch im IE 6.x)**

Feld der Zeiger aller als Plugin per EMBED-Tag eingebetteten Objekte im Dokument (inkl. Applets)  
Elementefolge laut HTML-Coding

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 existieren, laufen unter dem IE ab 6.x nicht mehr.

Es ist zu vermuten, dass im IE ab 6.x diese Collection nur eine Dummy-Funktion hat, also existiert, aber nie angefasst wird.

Diese Collection 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. Alle eingebetteten Objekte haben in document.embeds ihren Zeiger (inklusive Plugins, Ausnahme: siehe tiefer).

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 Internet 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.

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:**  
durch Browser

Beispiel für EMBED-Tag beim IE unter 6.x und beim NS:

```
<EMBED
  SRC="url"
  TYPE="mime_typ" // z.B. "image/gif"
  PLUGINSOURCE="plugin_url"
  HEIGHT="hoehe_plugin_objekt"
  WIDTH="breite_plugin_objekt"
  NAME="ID_Embed"
  HIDDEN="true oder false" // true so Objekt unsichtbar
>
  <PARAM Name="parameter_name" VALUE=parameter_wert>
  .....
</EMBED>
```

**Syntax:**

```
[ var ZeigerAufFeld = ] document.embeds
[ var ZeigerAufFeldElement = ] document.embeds[Index, [SubIndex]]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement.eigenschaft  
ZeigerAufFeldElement.methode()

beim IE:  
document.all.ID\_Embed.eigenschaft  
document.all.ID\_Embed.methode()

beim NS:  
document.ID\_Embed.eigenschaft  
document.ID\_Embed.methode()

**Eigenschaften beim Internet Explorer:**

.length Anzahl der Feldelemente also Feldlänge

**Methoden beim Internet Explorer:**

.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
.urns()	Referenz auf Feld aller Elemente mit gemeinsamer URN liefern
<b>4.3.2.2.4.3.12. fieldSet Objekt des Internet Explorer</b>	
Objekt dient zum Rendern einer Box z.B.	um einen Text als optische Gruppierung zusammengehöriger Texte
<b><u>Eigenschaften:</u></b>	
.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.align	Ausrichtung
ATOMICSELECTION	Selektierbarkeit einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.blockDirection	Umfluss um ein Objekt
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente



	nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von offsetHeight, offsetLeft, offsetTop und offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten Einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA  Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektierbarkeit z.B. mit Maus <b>wird nicht an Kinder vererbt</b> bei Wechsel: vorhergehende vorhandene Selektion wird nicht verändert
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen



	Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig).
	ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entferbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut):



	<p>siehe Methode <code>.getElementByName()</code></p> <p>DOM nicht geändert</p> <p>Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern</p> <p>Style-Eigenschaft ist per Methoden <code>expression()</code> oder <code>setExpression()</code> zu definieren</p> <p>DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)</p>
<code>.getExpression()</code>	<p>prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt</p> <p>DOM nicht geändert</p> <p>Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich</p>
<code>.hasChildNodes()</code>	<p>DOM wird geändert</p> <p>HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich</p> <p>HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt</p> <p>eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <code>&lt;SCRIPT DEFER .....&gt;</code> muss kodiert werden</p>
<code>.insertAdjacentElement()</code>	<p>DOM wird geändert</p> <p>Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes</p>
<code>.insertAdjacentHTML()</code>	<p>DOM wird geändert</p> <p>Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode <code>createElement()</code> erzeugt worden sein</p> <p>Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten</p> <p>Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p>
<code>.insertAdjacentText()</code>	<p>DOM wird geändert</p> <p>alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen</p> <p>Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME</p> <p>Achtung: Diese Methode ist mir Vorsicht zu geniessen !!</p>
<code>.insertBefore()</code>	<p>DOM wird geändert</p> <p>Normalisierung des DOM zur Erreichung einer konsistenten Struktur</p> <p>Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen</p>
<code>.mergeAttributes()</code>	<p>Maus-Überwachung ausschalten für ein Objekt</p> <p>Maus-Events sind : <code>onmousedown</code>, <code>onmouseup</code>, <code>onmousemove</code>, <code>onclick</code>, <code>ondblclick</code>, <code>onmouseover</code> und <code>onmouseout</code>.</p>
<code>.normalize()</code>	<p>Hinweis: einschalten per Methode <code>.setCapture()</code></p> <p>entfernen eines per HTML erzeugten Attributes</p> <p>Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode <code>.createAttribute()</code> erzeugte Attribute werden nicht erfasst</p>
<code>.releaseCapture()</code>	<p>DOM wird geändert</p> <p>entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern</p>
<code>.removeAttribute()</code>	<p>DOM wird geändert</p> <p>per Methode <code>.addBehavior()</code> einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)</p>
<code>.removeAttributeNode()</code>	<p>DOM wird geändert</p> <p>Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern</p> <p>Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde</p>
<code>.removeBehavior()</code>	<p>DOM wird geändert</p> <p>Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient.</p> <p>Ausdruck muss mit der Methode <code>.setExpression()</code> gesetzt worden sein</p>
<code>.removeChild()</code>	<p>DOM wird nicht geändert</p> <p>Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern</p> <p>Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p>
<code>.removeExpression()</code>	<p>DOM wird geändert</p> <p>Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern</p>
<code>.removeNode()</code>	<p>DOM wird nicht geändert</p> <p>Kind-Objekt ersetzen durch ein Objekt</p> <p>ersetzende Objekt muss per Methode <code>createElement()</code> erzeugt worden sein</p> <p>Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p>
<code>.replaceAdjacentText()</code>	<p>DOM wird geändert</p> <p>Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags</p>
<code>.replaceChild()</code>	<p>DOM wird geändert</p> <p>Objekt derart scrollen, dass es im Fenster für User sichtbar wird</p>
<code>.replaceNode()</code>	
<code>.scrollIntoView()</code>	



.setActive()	Objekt muss an sich schon renderbar sein Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.13. font Objekt des Internet Explorer

Implementation des Font mit seinen Eigenschaften

##### Eigenschaften:

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz
.color	Farbe des Objektes
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.face	Familie des Font-Typs (Font-Face) z.B. Courier
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes



.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.size	Fontgröße
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde



.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichnung zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout



	(nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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



.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.14. document.form Objekt des Internet Explorer

Dieses Objekt dient der formular-orientierte Datenbeschaffung zum Zweck der Datenübersendung an den Server. Es ist möglich, mit einem Formular versteckt Daten zu transferieren, die dem User nicht angezeigt bzw. bewusst gemacht werden. Der Programmierer sollte bei der Nutzung von Scripten in Formularen die Sorgfaltspflicht bezüglich Userdaten umsetzen. Im Internet Explorer ist dem User das Abschalten der Autovervollständigung auch bezüglich Formulare anzuraten.

Das input Objekt ist der Container für alle Input-Varianten im Formular. Das Objekt img ist der Container für input image. Das Objekt button ist der Container für input button. Das Objekt textarea ist der Container für input textarea. Das Objekt select ist der Container für input option und input select. Container vererben ihre Eigenschaften an die Formularelemente.

#### Erzeugung unter HTML:

Beispiel:

```
<FORM ID="ID_Formular"
NAME="logischer_formular_name" // muss für alle zu sendenden Formular-Elemente ebenfalls
// kodiert sein, wenn die Elemente Daten des
// Formulars darstellen
TARGET="zielfenster" // Ausgabefenster z.B. des CGI-Scripts für
ACTION="auswertungs_url" // Auswertung und Antwort auf abgeschicktes Formular
METHOD="get" oder "post"
ENCTYPE="mime_typ" // z.B. text/* für E-Mail
onreset="eventhandler1" // erzeugt Event onreset
// Reaktion VOR dem Rücksetzen ist zu programmieren per
Eventhandler:
// muß return-Anweisung haben:
// return true; so wird Formular auch
// zurückgesetzt
// return false; so wird Formular nicht
// zurückgesetzt
// Das Rücksetzen an sich macht der Browser per
// Methode .reset()
onsubmit="eventhandler2" // erzeugt Event onsubmit
// Reaktion VOR dem Senden ist zu programmieren per Eventhandler:
// muß return-Anweisung haben:
// return true; so wird Formular auch
// abgesendet
// return false; so wird Formular nicht
// abgesendet
// Das Absenden an sich macht der Browser per
// Methode .submit()
>
formularinhalt
</FORM>
```

Formularinhalte können diverse HTML-Elemente sein., vor allem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann.

#### Beispiele:

Beispiel 1:

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

Beispiel 2:



```

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

```

Beispiel 3:

```

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

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

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

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

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

  // und als UTC-Format erzeugen
  var ZeitpunktUTC = Zeitpunkt.toUTCString();

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

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

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

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

function InputLaden()
{
  // Cache laden
  InputDatenObjekt.load(FreierCacheName);

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

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

```



```

        <INPUT TYPE="button" VALUE="laden der Input-Daten" onclick="InputLaden()">
    </FORM>
</BODY>
</HTML>

```

Beispiel 4:

```

<HTML>
<HEAD>
<SCRIPT>
    function EventHandler_AktionVorReset()
    {
        // ein Hinweis im Textarea anzeigen
        ID_Textarea.value += "Formular zuruecksetzen ????";

        // wirklich rüecksetzen ???
        return( confirm("Wirklich rüecksetzen ?"));
        // true so Reset durch Browser ausführen lassen
        // false so kein Reset durch Browser
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <FORM NAME="Formular" onreset="EventHandler_AktionVorReset();">
            <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
            <BR>
            <BUTTON onclick="form.reset();">OnReset ausloesen</BUTTON>
            <BR><BR>
            <INPUT TYPE="reset" VALUE="oder hiermit zuruecksetzen"
                onclick="form.reset()"
            >
        </FORM>
    </DIV>
    <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>

```

Beispiel 5:

```

<HTML>
<HEAD>
<SCRIPT>
    function EventHandler_AktionVorSubmit()
    {
        // ein Hinweis im Textarea anzeigen
        ID_Textarea.value += "Formular senden ????";

        // wirklich senden ???
        return( confirm("Wirklich senden ?"));
        // true so Submit durch Browser ausführen lassen
        // false so kein Submit durch Browser
    }
</SCRIPT>
</HEAD>
<BODY>
    <DIV>
        <FORM NAME="Formular" ACTION=" ..." METHOD=" "
            onsubmit="EventHandler_AktionVorSubmit();">
            <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
            <BR>
            <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
            <BR><BR>
            <INPUT TYPE="submit" VALUE="oder hiermit senden"
                onclick="form.submit()"
            >
        </FORM>
    </DIV>
    <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>

```

Beispiel 6:

```

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

```



**Zugriff:**

Hinweise: Anstelle des logischen Namens laut NAME-Attribut kann auch der Wert des ID-Attributes verwendet werden. Das NAME-Attribut muss kodiert werden, wenn Daten des Formularelementes zu senden sind per .submit() (zu kodieren im Formular selbst und für jedes Formular-Element, das Daten senden soll).

auf das Formular:

beim NS:

```
logischer_formular_name.eigenschaft  
logischer_formular_name.methode()
```

```
document.forms[index].eigenschaft  
document.forms[index].methode()
```

index: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

beim IE:

```
document.all.logischer_formular_name.eigenschaft  
document.all.logischer_formular_name.methode()
```

```
document.forms[index].eigenschaft  
document.forms[index].methode()
```

index: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

auf ein Formular-Element:

Formularinhalte können diverse HTML-Elemente sein., vorallem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann. **Jedes Element** innerhalb von <FORM> .. </FORM> wird zum Kind des Objektes form. Daher ist Punktnotation nötig, um den Formularinhalt, also die Kinder des Objektes form, referenzieren zu können.

beim NS:

```
logischer_formular_name.logischer_formular_element_name.eigenschaft  
logischer_formular_name.logischer_formular_element_name.methode()
```

```
var ZeigerAufFormular = document.forms[index];  
ZeigerAufFormular.logischer_formular_element_name.eigenschaft  
ZeigerAufFormular.logischer_formular_element_name.methode()
```

index: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

```
var ZeigerAufFormular = document.forms[index1];  
ZeigerAufFormular.elements[inde2].eigenschaft  
ZeigerAufFormular.elements[inde2].methode()
```

index1: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

index2: ab 0  
Nummer des Formular-Elementes im Formular  
muss in [ ] kodiert sein

```
var ZeigerAufFormular = document.forms[index1];  
var ZeigerAufFormularElement = ZeigerAufFormular.elements[index2];  
ZeigerAufFormularElement.eigenschaft  
ZeigerAufFormularElement.methode()
```

index1: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

index2: ab 0  
Nummer des Formular-Elementes im Formular  
muss in [ ] kodiert sein

beim IE:



```
document.all.logischer_formular_name.logischer_formular_element_name.eigenschaft
document.all.logischer_formular_name.logischer_formular_element_name.methode()
```

```
var ZeigerAufFormular = document.forms[index];
ZeigerAufFormular.logischer_formular_element_name.eigenschaft
ZeigerAufFormular.logischer_formular_element_name.methode()
```

index: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

```
var ZeigerAufFormular = document.forms[index1];
ZeigerAufFormular.elements[inde2].eigenschaft
ZeigerAufFormular.elements[inde2].methode()
```

index1: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

index2: ab 0  
Nummer des Formular-Elementes im Dokument  
muss in [ ] kodiert sein

```
var ZeigerAufFormular = document.forms[index1];
var ZeigerAufFormularElement = ZeigerAufFormular.elements[index2];
ZeigerAufFormularElement.eigenschaft
ZeigerAufFormularElement.methode()
```

index1: ab 0  
Nummer des Formulars im Dokument  
muss in [ ] kodiert sein

index2: ab 0  
Nummer des Formular-Elementes im Dokument  
muss in [ ] kodiert sein

**Ereignisse (ausgewählte):**

onreset ausgeführt beim Zurücksetzen des Formulars  
onsubmit ausgeführt vor dem Senden des Formulars, also verwendbar für  
Formularprüfung oder Unterbinden des Sendens wegen  
fehlerhaften Formulardaten

**Eigenschaften zum Internet Explorer:**

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

.action Url des Servers und des dortigen Dokumentes bei Formular

Beispiele für mailto-Formen:

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

Betreff/Mailtext analog für Standard und blind copy

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

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

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

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



```

      + '@'
      + host_name
      + '</A>'
    );

```

ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.autocomplete	Status des Autovervollständigung zum Formular
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.blockDirection	Umfluss um ein Objekt
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.enctype	Multipurpose Internet Mail Extensions (MIME) des Formulars für Encoding nach dem Senden der Daten ab IE 6.x
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.method	Methode des Senden/Empfagen der Daten an/von den Server bei Formular
.name	Name des Objektes (nicht ID !!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)



.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA  Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.target	Name des Ziel-Fenster bzw. Ziel-Frame
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
<b>Methoden zum Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag



	(falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernen DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByName()	Referenz auf ein Feld (Collection) aller im Dokument befindlichen Objekte mit gemeinsamen NAME (analog zum Attribut NAME) liefern Achtung: Objekte, die kein NAME besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per Tag-Name siehe Methode .getElementsByTagName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)



.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.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 !!!
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.namedItem()	Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.reset()	löst für Formular das Event onreset aus, dessen Eventhandler die Reset-Aktion beinhaltet, die gestartet wird VOR dem Reset der Elemente (Browser setzt zurück) Eventhandler muss liefern: return true, für Ausführung des Reset durch Browser



return false; für Nicht-Ausführung des Reset durch Browser

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
function EventHandler_AktionVorReset()
{
    // ein Hinweis im Textarea anzeigen
    ID_Textarea.value += "Formular zuruecksetzen ????";

    // wirklich rü cksetzen ???
    return( confirm("Wirklich rü cksetzen ?"));
    // true so Reset durch Browser ausföh ren lassen
    // false so kein Reset durch Browser
}
</SCRIPT>
</HEAD>
<BODY>
<DIV>
<FORM NAME="Formular" onreset="EventHandler_AktionVorReset();"
<INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
<BR>
<BUTTON onclick="form.reset();">OnReset ausloesen</BUTTON>
<BR><BR>
<INPUT TYPE="reset" VALUE="oder hiermit zuruecksetzen"
onclick="form.reset()"
>
</FORM>
</DIV>
<TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>

```

- .scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
- .setActive() Objekt muss an sich schon renderbar sein  
Objekt für die Eventdurchreichung aktivieren  
aber ohne es zu fokussieren  
und ohne es scrollbar zu machen
- .setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert
- .setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.  
ab IE 5.5
- .setExpression() Hinweis: ausschalten per Methode .releaseCapture()  
Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
Eigenschaft als Objektreferenz der Form  
objekt.style.eigenschaft.  
dient  
Ausdruck nur als Script kodierbar  
DOM wird nicht geändert
- .submit() löst für Formular das Event onsubmit aus, dessen Eventhandler die Submit-Aktion beinhaltet, die gestartet  
wird VOR dem Senden der Elemente (Browser sendet)  
Eventhandler muss liefern: return true, für Ausführung des Submit durch Browser  
return false; für Nicht-Ausführung des Submit durch Browser

Beispiel:

```

<HTML>
<HEAD>
<SCRIPT>
function EventHandler_AktionVorSubmit()
{
    // ein Hinweis im Textarea anzeigen
    ID_Textarea.value += "Formular senden ????";

    // wirklich senden ???
    return( confirm("Wirklich senden ?"));
    // true so Submit durch Browser ausföh ren lassen
    // false so kein Submit durch Browser
}
</SCRIPT>
</HEAD>
<BODY>
<DIV>
<FORM NAME="Formular" ACTION=" ..." METHOD=" "
onsubmit="EventHandler_AktionVorSubmit();"

```



```

        <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
        <BR>
        <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
        <BR><BR>
        <INPUT TYPE="submit" VALUE="oder hiermit senden"
            onclick="form.submit()"
        >
    </FORM>
</DIV>
<TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>
.swapNode()    Positionen von 2 Knoten im DOM tauschen (Zeigertausch)
                nur sichtbar wenn Endetag geparkt
                DOM wird geändert
.urns()        Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

```

#### 4.3.2.2.4.3.14.1. Objekt document.form.input und seine Varianten beim Internet Explorer

Formularinhalte können diverse HTML-Elemente sein., vorallem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann. **Jedes Element** innerhalb

von <FORM> .. </FORM> wird zum Kind des Objektes form. Daher ist Punktnotation nötig, um den Formularinhalt, also die Kinder des Objektes form, referenzieren zu können. Der Objekttyp input.image kann **nicht für eine Formular** referenziert werden, wenn er innerhalb <FORM> ... </FORM> kodiert wurde. Für input.image ist immer die children Collection zu verwenden.

Nachfolgend werden die Varianten des input Objektes kurz beschrieben, wobei nur eine Auswahl von Eigenschaften und Methoden genannt wird. Aus Übersichtsgründen sind weitere Eigenschaften und Methoden direkt bei der Beschreibung vom input Objekt **und nicht hier** zu finden. Ereignisse werden alle beim Objekt event beschrieben. Die Variante **input.image** wird nicht beschrieben.

Die Varianten des input Objekt basieren z.T. auch auf anderen Objekten.

Im Objekt input muss das Attribut TYPE mit dem Objekttyp belegt werden, von dem die Variante des Objektes input abstammt. Der Wert vom Attribut TYPE kann wahlweise mit " " bzw. ' ' oder ohne " " bzw. ' ' kodiert werden, wobei Gross-Kleinschreibung egal ist. Der Objekttyp image kann **nicht für eine Formular** referenziert werden, wenn er innerhalb <FORM> ... </FORM> kodiert wurde.

Die Varianten von input haben nur Sinn, wenn **zugleich** Eventhandler kodiert werden, da ein Formular die User-Interaktion behandeln sollte.

#### 4.3.2.2.4.3.14.1.1. Objekt document.form.input.button des Internet Explorer

Diese Objekt basiert zusätzlich auf dem button Objekt (siehe dort).

##### Erzeugung durch HTML:

Beispiel:

```

<INPUT ID="ID_Input"
  TYPE=button
  VALUE="button_beschriftung"
  NAME="logischer_formular_element_name"
  onblur="eventhandler1"
  onfocus="eventhandler2"
  onclick="eventhandler3"
  onmousedown="eventhandler4"
  onmouseup="eventhandler5"
>

```

##### Beispiel:

```

<HEAD>
<SCRIPT>
    function ValueAendern()
    { ID_Input1.value = "Neuer Wert von VALUE";}

    function FarbeAendern()
    { ID_Input2.style.backgroundColor = "aqua";}

    function Anzeige()
    {alert(event.propertyName + " wurde verändert");}

</SCRIPT>
</HEAD>
<BODY>
    <INPUT TYPE=button ID="ID_Input1"
        VALUE="Click um VALUE zu ändern"
        onclick="ValueAendern()"
        onpropertychange="Anzeige()"
    >
    <INPUT TYPE=button ID="ID_Input2"
        VALUE="Click um Farbe zu ändern"
        onclick="FarbeAendern()"
    >

```



```
onpropertychange="Anzeige()"
>
```

```
</BODY>
```

**Eigenschaften (ausgewählte) :**

```
.form Zeiger auf das Formular (Formular als Container)
ab IE 6.x für Elemente fieldSet, label, legend
.name entspricht NAME
.type liefert immer den Typ button
.value entspricht VALUE
```

**Methoden (ausgewählte):**

```
.blur() entfernt den Cursor vom Button
.click() bewirkt einen Klick auf die Schaltfläche
belegt CHECKED und .checked und .defaultChecked
.focus() setzt den den Cursor auf das Button
```

**Ereignisse (ausgewählte):**

```
onblur ausgelöst, wenn User der Cursor vom Button entfernt
onclick ausgelöst, wenn User auf das Button klickt
onfocus ausgelöst, wenn User den Cursor auf das Button bewegt
onmousedown ausgelöst, wenn User auf dem Button die Maustaste niederdrückt
onmouseup ausgelöst, wenn User auf dem Button die gedrückte Maustaste loslässt
```

**4.3.2.2.4.3.14.1.2. Objekt document.form.input.checkbox (Abhak-Kästchen) des Internet Explorer****Erzeugung durch HTML:**

Beispiel:

```
<INPUT ID="ID_Input"
TYPE=checkbox
VALUE="wert_der_gesendet_wird" // Standardwert ist "on"
NAME="logischer_formular_element_name"
CHECKED // Checkbox ist mit Erstellung bereits markiert
onblur="eventhandler1"
onfocus="eventhandler2"
onclick="eventhandler3"
>
```

**Beispiele:**

Beispiel 1:

```
<HEAD>
<SCRIPT>
function Anzeige()
{alert("Checkbox-Status = " + ID_Checkbox.checked);}
</SCRIPT>
</HEAD>
<BODY>
selektiere !
<INPUT TYPE=checkbox
ID="ID_Checkbox"
NAME="checkboxbox1"
onclick=Anzeige()
>
</BODY>
```

Beispiel 2:

```
<INPUT TYPE=checkbox
ID="ID_InputCheckbox"
CHECKED
DISABLED
>
<SPAN STYLE="font-weight:bold"
onclick="ID_InputCheckbox.status=false"
>
ablehnen
</SPAN>
<SPAN STYLE="font-weight:bold"
onclick="ID_InputCheckbox.status=true"
>
annehmen
</SPAN>
```

Beispiel 3:

```
<HTML>
<HEAD>
<SCRIPT>
function ClickMitFocus()
{
ID_CheckBox.focus();
ClickOhneFocus();
}
```



```

    }

    function ClickOhneFocus ()
    { ID_CheckBox.click();}
</SCRIPT>
<SCRIPT FOR= ID_CheckBox EVENT=onfocus>
    alert("Check Box hat Focus");
</SCRIPT>
</HEAD>
<BODY>
    <INPUT Type="CHECKBOX" ID="ID_CheckBox"></INPUT>
    <BR>
    <BUTTON onclick="ClickMitFocus()">Klick mit Fokus</BUTTON>
    <BR>
    <BUTTON onclick="ClickOhneFocus()">Klick ohne Fokus</BUTTON>
</BODY>
</HTML>

```

**Eigenschaften (ausgewählte):**

.checked nur lesen  
ist true, wenn Checkbox abgehakt

.defaultChecked ist true, wenn CHECKED in <INPUT> kodiert  
kann belegt werden mit true oder false  
false bewirkt CHECKED in <INPUT> wird unwirksam

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

.name entspricht NAME

.type liefert immer Typ checkbox

.value entspricht VALUE

**Methoden (ausgewählte):**

.blur() entfernt Cursor von der Checkbox  
bewirkt markieren oder nicht markieren

.click() belegt CHECKED und .checked und .defaultChecked  
setzt Cursor auf die Checkbox

.focus() setzt Cursor auf die Checkbox

**Ereignisse (ausgewählte):**

onblur ausgelöst, wenn User der Cursor vom Checkbox entfernt

onclick ausgelöst, wenn User auf die Checkbox klickt

onfocus ausgelöst, wenn User den Cursor auf die Checkbox bewegt

Hinweise: onclick für weitere Reaktionen aufgrund markieren bzw. nicht markieren

**4.3.2.2.4.3.14.1.3. Objekt document.form.input.fileupload des Internet Explorer****Erzeugung unter HTML:**

Beispiel:

```

<INPUT ID="ID_Input"
    TYPE=file
    NAME="logischer_formular_element_name"
    onblur="eventhandler1"
    onchange="eventhandler2"
    onfocus="eventhandler3"
>

```

**Eigenschaften (ausgewählte):**

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

.name entspricht NAME

.type entspricht TYPE

.value enthält den Dateinamen, nur lesen

**Methoden (ausgewählte):**

.blur() entfernt Cursor vom Eingabefeld

.focus() setzt den Cursor auf das Eingabefeld

.select() markiert den Inhalt des Eingabefeldes, auf dem per .focus() der Cursor gesetzt sein muss

**Events (ausgewählte):**

onblur ausgelöst, wenn User der Cursor aus der Eingabezeile bewegt

onchange ausgelöst, wenn User auf den Inhalt der Eingabezeile ändert

onfocus ausgelöst, wenn User den Cursor auf die Eingabezeile bewegt

**4.3.2.2.4.3.14.1.4. Objekt document.form.input.hidden des Internet Explorer****Erzeugung unter HTML:**

Beispiel:

```

<INPUT ID="ID_Input"
    TYPE=hidden
    NAME="logischer_formular_element_name"
    VALUE="wert"
>

```

**Beispiel:**

```

<FORM ACTION="mailto:test@test.de" METHOD=GET>

```



```

<INPUT NAME=subject TYPE=hidden
      VALUE="Testt%20Product%20Information%20Anforderung"
>
      volle Mailadresse eingeben
<BR>
<TEXTAREA NAME=body COLS=40></TEXTAREA>
<INPUT TYPE=submit VALUE="absenden "
</FORM>

```

**Eigenschaften (ausgewählte):**

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE

4.3.2.2.4.3.14.1.5. Objekt *document.form.input.password* des Internet Explorer**Erzeugung unter HTML:**

Beispiel:

```

<INPUT ID="ID_Input"
      TYPE=password
      NAME="logischer_formular_element_name"
      VALUE="vorbelegung"
      SIZE="breite_eingabe_feld_in_zeichen"
      onblur="eventhandler1"
      onchange="eventhandler2"
      onfocus="eventhandler3"
      onselect="eventhandler4"
>

```

**Beispiel:**

```

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

```

**Eigenschaften (ausgewählte):**

.defaultValue	ist standardgemäß eine Leerkette kann verändert werden
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE enthält die Benutzereingabe

**Methoden (ausgewählte):**

.blur()	entfernt Cursor vom Eingabefeld
.focus()	setzt Cursor auf Eingabefeld
.select()	markiert den Inhalt des Eingabefeldes, da zuvor mit focus() den Cursor bekommt

**Ereignisse (ausgewählte):**

onblur	ausgelöst, wenn User der Cursor aus dem Eingabefeld bewegt
onfocus	ausgelöst, wenn User den Cursor auf das Eingabefeld bewegt

4.3.2.2.4.3.14.1.6. Objekt *document.form.input.radio* des Internet Explorer**Erzeugung unter HTML:**

Beispiel:

```

<INPUT ID="ID_Input"
      TYPE=radio
      VALUE="radio_wert_der_gesendet_wird_wenn_button_markiert_ist"
      NAME="logischer_radio_name" bzw. "logischer_radio_gruppen_name"
      // Radiobutton mit identischem Namen gehören einer Gruppe an
      // in der Gruppe kann pro Zeitpunkt maximal nur 1
      // Button markiert sein
      CHECKED
      onblur="eventhandler1"
      onfocus="eventhandler2"
      onclick="eventhandler3"
>

```

**Beispiele:**

Beispiel 1:

```

<HEAD>
<SCRIPT>
      function KanalVerteilung(Wert)
      { ID_bgsound.balance = Wert;}

      function GenauEinmal()
      {
          ID_bgsound.loop = 1;
          ID_bgsound.src = ID_bgsound.src; // restart
      }

```



```

function Endlos()
{
    ID_bgsound.loop = -1;
    ID_bgsound.src = ID_bgsound.src; // restart
}
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
<BUTTON onclick="GenauEinmal()"></BUTTON>
<BUTTON onclick="Endlos()"></BUTTON>
<BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
<BUTTON onclick="KanalVerteilung(10)"></BUTTON>
<BUTTON onclick="KanalVerteilung (0)"></BUTTON>
<B>Volume control:</B>&nbsp;
<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-10;"
>Mute

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-7;"
>25% Volume

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe" CHECKED
onpropertychange="ID_bgsound.volume=-5;"
>50% Volume

<INPUT TYPE="radio" NAME="ID_RadioButtonGruppe"
onpropertychange="ID_bgsound.volume=-2;"
>75% Volume

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

```

Beispiel 2:

```

<HTML>
<HEAD>
<SCRIPT>
function Erzeuge()
{
    var Zeiger = document.createElement(
        "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Eins'>"
    );
    document.body.insertBefore(Zeiger);

    Zeiger = document.createElement(
        "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Zwei'>"
    );
    document.body.insertBefore(Zeiger);
}
</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE="BUTTON"
ONCLICK=" Erzeuge()"
VALUE="Zwei Radio Buttons erzeugen">

<BR>
<INPUT TYPE="BUTTON"
ONCLICK="alert(document.body.outerHTML)"
VALUE="Click um HTML zu sehen">

<BODY>
</HTML>

```

#### Eigenschaften (ausgewählte):

.checked	ist true wenn Radiobutton markiert ist (sonst false)
.defaultChecked	ist true wenn CHECKED in INPUT kodiert ist kann neu gesetzt werden --> Wirkung von CHECKED aus INPUT aufgehoben
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.index	Index des ausgeählten Elementes
.length	Anzahl Radiobutton innerhalb der Gruppe
.name	entspricht NAME
.type	entspricht TYPE



.value	entspricht VALUE
<b>Methoden (ausgewählte):</b>	
.blur()	entfernt Cursor vom Radiobutton
.click()	bewirkt markieren bzw. entmarkieren des Buttons (Click auf das Radiobutton) sowie setzen der Eigenschaft checked auf true für das angeklickte Button Eigenschaft checked auf false für alle anderen Button der Gruppe
.focus()	setzt den Cursor auf das Radiobutton
<b>Ereignisse (ausgewählte):</b>	
onblur	ausgelöst, wenn User den Cursor vom Radiobutton entfernt
onclick:	ausgelöst, wenn User auf das Radiobutton klickt Eventhandler für weitere Reaktion nach Anklicken: muss return true; oder return false; besitzen: true, so erfolgt das Senden des radio_wertes auch wirklich false, so erfolgt kein Senden trotz angeklicktem Button
onfocus	ausgelöst, wenn User den Cursor auf das Radiobutton bewegt

#### 4.3.2.2.4.3.14.1.7. Objekt document.form.input.reset des Internet Explorer

##### Erzeugung unter HTML:

Beispiel:

```
<INPUT ID="ID_Input"
TYPE=reset
VALUE="reset_button_beschriftung"
NAME="logischer_formular_element_name"
onblur="eventhandler1"
onfocus="eventhandler2"
onclick="eventhandler3"
onreset="eventhandler4"
>
```

**Beispiel:**

```
<HTML>
<HEAD>
<SCRIPT>
function EventHandler_AktionVorReset()
{
// ein Hinweis im Textarea anzeigen
ID_Textarea.value += "Formular zuruecksetzen ???";

// wirklich rücksetzen ???
return( confirm("Wirklich rücksetzen ?"));
// true so Reset durch Browser ausführen lassen
// false so kein Reset durch Browser
}
</SCRIPT>
</HEAD>
<BODY>
<DIV>
<FORM NAME="Formular" onreset="EventHandler_AktionVorReset();"
<INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
<BR>
<BUTTON onclick="form.reset();">OnReset ausloesen</BUTTON>
<BR><BR>
<INPUT TYPE="reset" VALUE="oder hiermit zuruecksetzen" onclick="form.reset()">
</FORM>
</DIV>
<TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>
```

##### Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE wenn VALUE nicht kodiert, so "Reset" als Standard nur lesbar

##### Methoden (ausgewählte):

.blur()	entfernt den Cursor vom Resetbutton
.click()	bewirkt Anklicken des Resetbuttons und löschen des Formulars
.focus()	setzt den Cursor auf das Resetbutton

##### Ereignisse (ausgewählte):

onblur	ausgelöst, wenn der User den Cursor vom Resetbutton entfernt
onclick	ausgelöst, wenn der User auf das Resetbutton clickt
onfocus	ausgelöst, wenn User den Cursor auf das Resetbutton bewegt
onreset	ausgelöst direkt vor dem Formular-Reset durch den Browser



Eventhandler für weitere Reaktionen aufgrund des Anklicken auslösen:  
 muss return true; oder return false; liefern  
 true, so Reset auch wirklich ausgeführt  
 false, so Reset NICHT ausgeführt trotz Anklicken

#### 4.3.2.2.4.3.14.1.8. Objekt *document.form.input.submit* des Internet Explorer

##### Erzeugung unter HTML:

Beispiel:

```
<INPUT ID="ID_Input"
  TYPE=submit
  NAME="logischer_formular_element_name"
  VALUE="beschriftung_des_button"
  onblur="eventhandler1"
  onfocus="eventhandler2"
  onclick="eventhandler3"
  onsubmit="eventhandler4"
>
```

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
function EventHandler_AktionVorSubmit()
{
    // ein Hinweis im Textarea anzeigen
    ID_Textarea.value += "Formular senden ???";}

    // wirklich senden???
    return( confirm("Wirklich senden ??"));
        // true so Submit durch Browser ausführen lassen
        // false so kein Submit durch Browser
}
</SCRIPT>
</HEAD>
<BODY>
<DIV>
<FORM NAME="Formular" ACTION=" ...." METHOD=" "
  onsubmit="EventHandler_AktionVorSubmit();">
  <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
  <BR>
  <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
  <BR><BR>
  <INPUT TYPE="submit" VALUE="oder hiermit senden" onclick="form.submit()">
</FORM>
</DIV>
<TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
</HTML>
```

##### Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE wenn VALUE in <INPUT> nicht kodiert, so "Submit Query" als Wert nur lesen

##### Methoden (ausgewählte):

.blur()	entfernt den Cursor vom Submitbutton
.click()	bewirkt standardgemäß <b>sofortiges</b> Abschieken des Formulares (aufgrund Anklicken des Buttons), wenn es sich NICHT um E-Mail handelt --> bei E-Mail erfolgt erst Nachfrage
.focus()	setzt den Cursor auf das Submitbutton

##### Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User den Cursor vom Submitbutton entfernt
onclick	ausgelöst, wenn User das Submitbutton clickt
onfocus	ausgelöst, wenn User den Cursor auf das Submitbutton bewegt
onsubmit	ausgelöst direkt vor dem Formular-Senden durch den Browser Eventhandler für weitere Reaktionen aufgrund des Anklicken auslösen: muss return true; oder return false; liefern true, so Senden auch wirklich ausgeführt false, so Senden NICHT ausgeführt trotz Anklicken

#### 4.3.2.2.4.3.14.1.9. Objekt *document.form.input.text* des Internet Explorer

Dieses Objekt basiert zusätzlich auf dem Objekt text (siehe dort).

##### Erzeugung unter HTML:



Beispiel:

```
<INPUT ID="ID_Input"
  TYPE=text
  NAME="logischer_formular_element_name"
  VALUE="zeichenkette_als_vorgabe_wert"
  SIZE="breite_eingabefeld_in_zeichen"
  onblur="eventhandler1"
  onchange="eventhandler2"
  onfocus="eventhandler3"
  onkeydown="eventhandler4"
  onkeypress="eventhandler5"
  onkeyup="eventhandler6"
  onselect="eventhandler7"
>
```

**Beispiele:**

Beispiel 1:

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

Beispiel 2:

```
<HTML>
<HEAD>
<SCRIPT>
  function Antworten(Wert)
  {Wert ? alert("Ja") : alert("Nein");}
</SCRIPT>
</HEAD>
<BODY>
  <P>
    <INPUT type="text" ID="ID_Input" VALUE="Test">
    <BUTTON onclick="Antworten(ID_Input.isMultiLine);">
      Mehrzeiligkeit eines INPUT TYPE=text
    </BUTTON>
  </P>
  <P>
    TEXTAREA:
    <TEXTAREA ID="ID_Textarea">
      Test
    </TEXTAREA>
    <BUTTON onclick="Antworten(ID_Textarea.isMultiLine);">
      Mehrzeiligkeit eines Textbereiches
    </BUTTON>
  </P>
</BODY>
</HTML>
```

Beispiel 3:

```
<INPUT TYPE=text SIZE=33>
```

Beispiel 4 für Auslesen eines Eingabefeldes:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!
  function gruss()
  {alert("Hallo " + document.meinFormular.Eingabe.value + "!");}
// -->
</SCRIPT>
</HEAD>
<BODY>
  Bitte Namen eingeben und danach den Knopf drücken
  <FORM NAME="meinFormular">
    <INPUT TYPE="text"
      NAME="Eingabe"
```



```

        VALUE=""
    >
    <BR>
    <INPUT TYPE="button"
        NAME="Knopf"
        VALUE="Bitte drücken"
        onClick="gruss()"
    >
</FORM>
</BODY>
</HTML>

```

Beispiel 5 für Wert einem Eingabefeld zuweisen:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function umrechnung(celsius)
    {
        var fahrenheit;
        fahrenheit = 9 / 5 * celsius + 32;
        document.Formular.Ausgabe.value = fahrenheit;
    }
// -->
</SCRIPT>
</HEAD>

<BODY>
Bitte geben Sie einen Temperaturwert in Celsius ein:
<FORM NAME="Formular">
    <INPUT TYPE="text"
        NAME="Eingabe"
        VALUE=""
    >
    <BR>
    <INPUT TYPE="button"
        NAME="Knopf"
        VALUE="Berechnung"
        onClick="umrechnung(this.form.Eingabe.value)"
    >
    <BR>
    Entpricht
    <INPUT TYPE="text"
        NAME="Ausgabe"
        VALUE=""
    >
    Fahrenheit
</FORM>
</BODY>
</HTML>

```

Beispiel 6 für Laufschrift in einem Formular

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    text="Laufschrift";

    function anzeigen()
    {
        teilkette=text.substring(0, 1);
        text=text.substring(1, text.length);
        text= text + teilkette;

        window.document.formular.schrift.value=text;
        setTimeout('anzeigen()',200);
    }
//-->
</SCRIPT>
</HEAD>

<BODY>

```



```

<FORM NAME="formular">
  <INPUT TYPE=button
    VALUE="Laufschrift im Formularfeld"
    onClick="anzeigen()"
  >
  <INPUT TYPE="text"
    NAME="schrift"
    SIZE="20"
    READONLY
  >
</FORM>
</BODY>
</HTML>

```

**Eigenschaften (ausgewählte):**

- .defaultValue entspricht VALUE
- .form Zeiger auf das Formular (Formular als Container)  
ab IE 6.x für Elemente fieldSet, label, legend
- .name entspricht NAME
- .type entspricht TYPE
- .value entspricht **nicht** VALUE  
enthält den eingegebenen Text

**Methoden (ausgewählte):**

- .blur() entfernt den Cursor aus dem Eingabefeld (deaktiviert Textfeld für Eingabe)
- .focus() setzt den Cursor auf das Eingabefeld
- .select() markiert den Text im Eingabefeld  
Cursor muss zuvor mit focus() auf das Eingabefeld bewegt worden sein

**Eereignisse (ausgewählte):**

- onblur ausgelöst, wenn User den Cursor vom Eingabefeld entfernt
- onchange ausgelöst, wenn User das Eingabefeld inhaltlich ändert
- onfocus ausgelöst, wenn User den Cursor auf das Eingabefeld bewegt
- onselect ausgelöst, wenn Teil des Inhaltes vom Eingabefeld markiert wird

**4.3.2.2.4.3.14.2. document.form.elements Collection des Internet Explorer**

Felder der Zeiger auf alle Formularkomponenten **außer** Objekt input.image, das damit **im** Formular nicht referenzierbar wird:  
Für input image ist immer die children Collection zu verwenden.

Reihenfolge der Feldelemente laut der Kodierung der Elemente im Formular

**Syntax:**

```

[ var ZeigerAufFeld = ] zeiger_auf_form_objekt.elements
[ var ZeigerAufFeldElement = ] zeiger_auf_form_objekt.elements[Index, [SubIndex]]

```

- Index Integer ab 0  
oder String Name oder ID des Elementes  
muss in [ ] kodiert sein
- SubIndex optional  
nur kodieren wenn Index ein String ist  
Integer als Unterindex also Unterelement eines Elementes

zeiger\_auf\_form\_objekt z.B. laut ID-Attribut

**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
- .urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.15. document.forms Collection des Internet Explorer**

Feld der Zeiger aller Formular-Objekte im Dokument  
Elementefolge laut HTML-Coding

**Syntax:**

```

[ var ZeigerAufFeld = ] document.forms
[ var ZeigerAufFeldElement = ] document.forms[Index, [SubIndex]]

```

- Index Integer ab 0  
oder String Name oder ID des Elementes  
muss in [ ] kodiert sein
- SubIndex optional  
nur kodieren wenn Index ein String ist



Integer als Unterindex also Unterelement eines Elementes

**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

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.16. frame Objekt**

Hinweis: Wenn ein Fenster mit dem Unterstrichsymbol rechts oben in der Fensterecke minimiert wurde, dann ist es durch Windows in die Hintergrund-Prioritätenfolge eingeordnet worden und arbeitet im Hintergrund bzw. garnicht. Mit anschließendem Maximieren (Symbol in der Fensterleiste rechts oben) wird das Fenster wieder angezeigt und das geladene Dokument **erneut** geöffnet. Konsequenz daraus ist, dass dieses Maximieren einem Neustart des Dokumentes entspricht, auch wenn der Programmierer diesen Zustand nicht erwünscht. Sound, der z.B. mit Start des Dokumentes erzeugt wird, erklingt erneut mit Maximierung nach einer Fensterminimierung. Analogon ist die Fenstereinengung im Browser durch z.B. Ein- und Ausblenden der Favoritenleiste. Dieses Verhalten des Fensters mit seinem Dokument ist aber **nicht identisch** mit dem Verhalten nach einer Fenstergrößenänderung z.B. durch Rahmenverschiebung (resize). Die Aktionen des Fensters und seines Dokumentes bezüglich Resize müssen programmiert werden, sind also nicht standardmäßig vorhanden - im Gegensatz zum Verhalten mit/nach Fensterminimierung/-maximierung per Symbole in der rechten oberen Ecke der Fensterleiste. Sollte ein Frameset minimiert werden, dann wird das für den gesamten Frameset getan (Frameset hat 1 Fenster für alle Frames gesamt), allerdings mit Maximierung wird auch das Frameset-Dokument neu geladen. Wichtig dabei sind für Zeiger-Bezüge der Frames auf den Frameset per parent-Zeiger, dass die Daten im Frameset-Dokument mit Maximierung initialisiert werden und somit ebenfalls Daten der Frames, die im Frameset-Dokument abgelegt wurden, also z.B. Daten, die Verhaltensweisen der Frames vor einer Änderung der Frames-Inhalte gespeichert haben.

**Erzeugung unter HTML:**

```
<FRAMESET
    ROWS="zeilen_liste" oder COLS="spalten_liste" // Zeilenliste=Liste der Framehöhen
                                                // mit Kommatrennung
                                                // Spaltenliste=Liste der Framebreiten
                                                // mit Kommatrennung
    onLoad="evenhandler1" // Anweisungen aktiviert NACH laden ALLER Frames
    onUnload="eventhandler2"
    onerror="eventhandler3"
    onBlur="eventhandler4"
    onFocus="eventhandler5"
>
[<FRAME
    SRC="frame_url" // url der HTML-Seite als Frameinhalt
    NAME="logischer_frame_name"
>]
.....
</FRAME>
.....
</FRAMESET>
```

Rahmen zwischen den einzelnen Fenstern des Framesets entfernen:

```
<FRAMESET
    BORDER="0"
    FRAMEBORDER="0"
    FRAMESPACING="0"
    .....
>
```

Hinweise:

- BORDER bei Netscape  
Breite des Rahmens  
>= 0 Pixel
- FRAMEBORDER bei IE  
Rahmenanzeige  
0 oder "no" aus  
1 oder "yes" ein
- FRAMESPACING bei IE  
Rahmenbreite  
>=0 Pixel

**Zugriff:**

document.ID\_Frame.eigenschaft



```
document.ID_Frame.methode()
```

```
document.frames[index].eigenschaft
document.frames[index].methode()
```

```
index:    ab 0
          Nummer des Frames im Dokument
          muss in [ ] kodiert sein
```

### Möglichkeiten des Laden eines Dokumentes:

Laden eines fremden Dokumentes innerhalb eines Frame ist rechtlich nicht zulässig (Abmahnungsgefahr), wenn der Eigentümer der Fremdseite nicht explizit zugestimmt hat.

### Laden eines fremden Dokumentes ohne Framedarstellung:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
      <!--
        function laden()
        { location.href = "http://www.test.de";}
      // -->
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM>
      <INPUT TYPE="button"
        VALUE="www.test.de anwählen "
        onclick="laden()">
    </FORM>
  </BODY>
</HTML>
```

### Eigenes Dokument wird durch fremde Webseite geladen:

#### *CopyRight-Meldung auf fremden Host erzeugen:*

Beispiel 1:

```
// In diesem Beispiel schreibt das Script Text auf die "entführte" Seite!
var HostUrl="www.test.de";

if (    (parent !=null)
    && (parent != self)
    )
{
    var host=parent.location.hostname;

    if(host != HostUrl)
    {
        document.write(
            "Diese Seite wurde ausgeliehen bei "
            + "<A HREF=" + location.href
            + " TARGET=_parent"
            + ">"
            + HostUrl
            + "</A>"
        );
    }
}
```

Beispiel 2:

```
<BODY>
.....
if (    (parent != null)
    && (parent != self)
    )
{
    var  meine_url = "www.test.de"
    var  mein_host_name = "http://" + meine_url;

    var  fremder_host_name = parent.location.hostname;
```



```

        if ( fremder_host_name != mein_host_name)
        {
            document.write(      " Diese Seite liegt auf "
                + "<A HREF=\"" + location.href + "\"\"
                + " TARGET=\"_parent\"\"
                + ">"
                + "</A>"
                + " und stammt von "
                + mein_host_name
            );
        }
    }
    .....
</BODY>

```

***Framedarstellung der eigenen Webseite sofort und ohne Bildschirmmeldung aktivieren:***

Dieses Coding muss in jedes HTML-Dokument, das in einem Frame geladen wird. Bei Einzelaufwurf des Dokumentes wird automatisch die Seite mit dem FRAMSET aktiviert, also die vollständige Framedarstellung.

```

<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.2">
<!--
    var EigenerHost="http://www.test.de";
        // window.location.hostname ist Leerkette, wenn Browser offline
    var StartSeite="_start.html";

    function InaktiveFrameDarstellungAktivieren()
    {
        if (top.frames.length == 0)
        {
            // keine Frame-Darstellung aktiv, also diese aktivieren
            top.location.href = StartSeite; // muss das FRAMESET enthalten !
        }
    }

    if (
        (parent != null) // Elternobjekt existiert
        && (parent != self) // Eltern sind vorhanden Eltern: dieses Dokument (self) ist ein Kind
        )
    {
        // aktuellen ElternHost ermitteln
        var ElternHost=parent.location.hostname;

        // ElternHost prüfen ob eigener und nicht leer, also Browser online ist
        if (
            (ElternHost != "") // Browser ist online
            && (ElternHost != EigenerHost)
            )
        {
            // fremder Host, also als oberstes Fenster nun die Startseite anzeigen
            // und damit den eigenen Host aktivieren
            top.location.href=EigenerHost + '/' + StartSeite;
        }
        else
        {
            // eigener Host und/oder Browser ist offline
            InaktiveFrameDarstellungAktivieren();
        }
    }
    else
    {
        // Elternobjekt existiert nicht und/oder dieses Dokument (self) ist kein Kind
        InaktiveFrameDarstellungAktivieren();
    }
}
-->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>

```

**Reload eines Dokumentes mit allen seinen FRAMES:**



```
function frame_neu_laden()
{
    for (var i=0; i<windows.FRAMES.length; i++)
        { windows.frames[i].location.reload(false); }
}

<FRAMESET onResize="frame_neu_laden()">
```

**Datei ohne FRAMESET in eine Datei mit FRAMESET laden**

```
if (self.location == top.location)
{ location.href="/FRAMESET.html?" + escape(location.pathname); }
```

**Mehrere Frameinhalte gleichzeitig ändern:****Inhalte zweier Frames tauschen:**

Kodierung im Dokument, dass die beiden Frames definiert !

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    function rahmentausch(logischer_name_rahmen1,html_datei1,
        logischer_name_rahmen2,html_datei2
    )
    {
        frames[logischer_name_rahmen1].location.href = html_datei1;
        frames[logischer_name_rahmen2].location.href = html_datei2;
    }
// -->
</SCRIPT>

<A HREF="javascript:parent.rahmentausch(0, 'a.html', 1,b.html)">tauschen</A>
```

**Inhalte beliebig vieler Frames als Ring tauschen:**

Die logischen Framennamen werden als variable Argumenteliste übergeben und dienen der Indizierung der Frame im Dokument. Argumenttrenner sind Doppelpunkte.

Tausch:

```
erster Frame retten und dann mit zweiten Frame überschreiben
zweiten Frame mit drittem Frame überschreiben
.....
vorletzten Frame mit letztem Frame überschreiben
letzten Frame mit geretteten ersten Frame überschreiben
```

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    function rahmentausch()
    {
        if (arguments.length>1) // Länge ab 1, mindestens 2 Argumente
        {
            var pos_doppelpunkt = arguments[0].indexOf(".");

            if (pos_doppelpunkt != -1) // nächstes Argument ist vorhanden
            {
                var rette_logischer_framename= arguments[0].substring(0, pos_doppelpunkt);

                for(var i = 0; i < arguments.length; i++) // Index ab 0
                {
                    pos_doppelpunkt = arguments[i].indexOf(".");

                    if(pos_doppelpunkt != -1) // nächstes Argument ist vorhanden
                    {
                        // ersten zu ersetzenden Framennamen retten

                        if (i=0)
                        {var rette_logischer_framename =
                            arguments[i].substring(0, pos_doppelpunkt)
                        }

                        // tauschen der logischen Framennamen
                        logischer_framename_zu_ersetzender_frame =
                            arguments[i].substring(0, pos_doppelpunkt);
```



```

        logischer_framename_ersetzender_frame =
            arguments[i].substring(0, pos_doppelpunkt + 1);

        frames[logischer_framename_zu_ersetzender_frame].location.href =
            logischer_framename_ersetzender_frame;
    }
}

frames[logischer_framename_ersetzender_frame].location.href =
    rette_logischer_framename;
}
}
}
// -->
</SCRIPT>
<A HREF="javascript:parent.rahmentausch('r1:a.html', 'r2:b.html', 'r3:c.html')">tauschen</A>

```

**Frame und Datenaustausch:**

Zwischen Frames können Daten ausgetauscht werden.

Beispiel Adressierung eines Frame über das FRAMESET

```

<FRAMESET .... >
  <FRAMESET ..... >
    <FRAME SRC="test.html" NAME="test">
    <FRAME SRC="test1.html" NAME="test1">
  </FRAMESET>
  <FRAME SRC="test2.html" NAME="test2">
</FRAMESET>

parent.test2.location.href="neu.html"; // Laden von neu.html in den Frame test2

```

Beispiel: Auf Objekte im FRAMESET-Dokument durch ein FRAME-Dokument zugreifen

```

im FRAMESET-Dokument wird kodiert      var Kette="Hallo !";
im FRAME-Dokument wird auf Kette zugegriffen: alert(parent.Kette);

```

Beispiel: Auf Objekte im FRAME-Dokument durch das FRAMESET-Dokument zugreifen

```

im FRAME-Dokument wird kodiert      var Kette="Hallo !";
<FRAME ...NAME ="test2" ...>
im FRAMESET-Dokument wird auf Kette zugegriffen: alert(parent.test2.Kette);

```

Beispiel: Textdaten-Übergabe durch an Url angehängte Textdaten

**quelle.htm:           lädt ziel.htm  
                          übergibt Textdaten an ziel.htm**

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function ziel_seite_laden_mit_datenuebergabe(uebergabe_daten)
    {location.href = "ziel.htm?" + escape(uebergabe_daten);}
    // Url von ziel.htm als Suchbegriff mit angehangenen Daten merken, die per escape() in das
    //      Url-Format konvertiert wurden
// -->
</SCRIPT>
</HEAD>

<BODY>
An ziel.htm zu &uuml;bergabenden Text eingeben:
<FORM>
  <TEXTAREA       NAME=eingabe
                  ROWS=5
                  COLS=40
  >
  </TEXTAREA>
  <BR>
  <INPUT   TYPE=button

```



```

        VALUE="Zielseite laden"
        onClick=" ziel_seite_laden_mit_datenuebergabe(this.form.eingabe.value)">
</FORM>
</BODY>
</HTML>

```

**ziel.htm:**            **wird durch quelle.htm geladen**  
**erhält Textdaten von quelle.htm**

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    datenuebernahme()
    {
        // Url mit angehangenen Daten holen
        // search liefert ?daten
        uebernahme_daten= location.search;

        // ? abschneiden
        uebernahme_daten= uebernahme_daten.substring(1, uebernahme_daten.length);

        // unescape: von Url-Format nach Zeichenkette
        document.ausgabe.ausgabefeld.value=unescape(uebernahme_daten);
    }

// -->
</SCRIPT>
</HEAD>

<BODY onLoad=" datenuebernahme ()">
Von quelle.htm &uuml;bernommener Text:
    <FORM NAME="ausgabe">
        <TEXTAREA        NAME="ausgabefeld"
        ROWS=10 COLS=40>
    </TEXTAREA>
    </FORM>
</BODY>
</HTML>

```

Beispiel: Textdaten-Übergabe durch Fenster-Handle

Die Textdaten und die der Javascript-Code, der die Textdaten verarbeitet, sind **getrennte** HTML-Dokumente. Nachteil dieser Variante ist, dass in beiden Dokumenten die logischen Namen vom Formular und dem Textarea identisch

kodiert werden müssen, also eine doppelte Verwaltung nötig ist.

**HTML-Dokument, das die Textdaten enthält:**

```

<HTML>
<BODY>
    <FORM NAME="formular">
        <TEXTAREA NAME="text_area">
            // Hier die Textdaten als Wert von TEXTAREA
        </TEXTAREA>
    </FORM>
</BODY>
</HTML>

```

**Dokument, das die Textdaten verarbeitet:**

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function TextAreaWertLesen(url)
    {
        // url enthält den Pfad und den Namen des Dokumentes, dass die
        //     Textarea-Werte enthält
        var handle = window.open(url_der_datendatei, "", "width=100,height=100");

        // Handle erzeugen und Fenster mit dem Textarea anzeigen
        //     (Fenstergröße ist egal)
        var data = handle.document.forms[formular].elements[text_area].value;
    }

```



```

        handle.close();
        return data;
    }
    // -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>

```

**frame Objekt des Internet Explorer:**

Frame muss im FRAMESET liegen.

Die Collection frames wird unter document.frames beschrieben !

**Zugriff auf Frame-Eigenschaft:**

Beispiel

```
var Wert = document.all.ZeigerAufFrame.eigenschaft;
```

mit ZeigerAufFrame laut ID-Attribut

```
<FRAME ID="ZeigerAufFrame" ..... >
```

**Transparenter Content des Frame:**

ab IE 5.5

für den Frame muss das Attribut .ALLOWTRANSPARENCY auf true gesetzt sein

Im Dokument, das in den Frame geladen wird, muss im BODY die Hintergrundfarbe (background-color oder BGCOLOR-Attribut) auf transparent gesetzt sein.

**Eigenschaften:**

.allowTransparency	Transparenz ein/aus Transparenz: z.B. Hintergrundfarbe des Elternobjektes wird zur Hintergrundfarbe des Frames Achtung: Wenn Frame-eigenes STYLE-Attribut mit Wert für background-color also STYLE="background-color: farb_bezeichner" kodierte wurde, so wird die Transparenz ignoriert und die Hintergrundfarbe laut STYLE verwendet
APPLICATION	Umgebung des Frames mit Vertrauen-Status im Rahmen des Browser-Sicherheitsmodells Attribut wird nicht vererbt an Kinder-Frame
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.borderColor	Borderfarbe (Rahmenfarbe) wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no" Eigenschaft .border mit Wert 0
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.contentWindow	Referenz auf das zugehörige Fenster-Objekt laut Collection document.all
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.frameBorder	Borderanzeige ein/aus beim Frame
.height	Höhe des Objektes in Pixel
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.longDesc	Uniform Resource Identifier (URI) zur einer "long description" umwandeln
.marginHeight	Abstand in Pixel vom unteren zum oberen Rand des Objektes
.marginWidth	Abstand in Pixel vom linken zum rechten Rand des Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element)



	also	TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType		Knotentyp laut attributes Collection
.nodeValue		Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.noResize		Frame-Größenänderung ein/aus Größenänderung z.B. durch Maus etc
.offsetHeight		Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft		X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent		Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop		Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth		X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.ownerDocument		Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement		Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode		Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit		Textbereich des Elternobjektes referenzieren
.previousSibling		Referenz auf das Vorgängerkind
.readyState		aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber		Datenquelle-Satznummer eines Datenfeldes
.scopeName		Namensraum laut XMLNS-Attribut
.scrolling		Scrollenbar erzeugen
SECURITY		temporäre Sicherheit des IFRAME wird an Kinder weitervererbt ab IE 6.0
.self		Referenz auf das aktuelle Fenster oder den aktuellen Frame
.sourceIndex		Index des Objektes in der Collection document.all
.src		Url der Daten z.B. vom Image in normaler Auflösung
.tabIndex		Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLETT, DIV, FRAMESET, SPAN, TABLE, TD
.tagName		Tag-Bezeichner des Objektes
.tagUrn		Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title		Tooltip-Text bei Mouse over über Objekt
.uniqueID		durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE		Selektionsfähigkeit eines Objektes
.width		Breite des Objektes in Pixel
<b>Methoden:</b>		
.addBehavior()		DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()		Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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



.clearAttributes()	<p>ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert</p>
.cloneNode()	<p>Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)</p>
.componentFromPoint()	<p>Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.</p>
.contains()	<p>prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert</p>
.detachEvent()	<p>Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)</p>
.dragDrop()	<p>prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element</p>
.fireEvent()	<p>ein Event auslösen</p>
.focus()	<p>Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen</p>
.getAdjacentText()	<p>Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert</p>
.getAttribute()	<p>Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert</p>
.getAttributeNode()	<p>Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert</p>
.getElementsByTagName()	<p>Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementByName()</p>
.hasChildNodes()	<p>DOM nicht geändert prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert</p>
.insertAdjacentElement()	<p>Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert</p>
.mergeAttributes()	<p>alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert</p>
.normalize()	<p>Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen</p>
.removeAttribute()	<p>entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert</p>
.removeAttributeNode()	<p>entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert</p>
.removeBehavior()	<p>per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert</p>



.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endtag geparkt DOM wird geändert

#### 4.3.2.2.4.3.17. document.frames Collection des Internet Explorer

Feld der Zeiger aller Frames (beim IE inklusive der inline-Frames, also IFRAMES)

Elementefolge laut HTML-Coding

Element ist eine Referenz auf das Fenster mit Frame-Eigenschaften

##### Syntax:

Achtung: Für eine Referenz auf das Frame-Objekt ohne diese Collection ist immer document.all zu kodieren.

Bei Fenstererzeugung bitte jedem Fenster seinen eigenen Namen zuweisen (Name nicht doppelt verwenden)

```
[ var ZeigerAufFeld = ] document.frames
[ var ZeigerAufFeldElement = ] document.frames[Index [ ,SubIndex ] ]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

Beispiel für Inhalte zweier Frames tauschen:

Kodierung im Dokument, dass die beiden Frames definiert !

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    function rahmentausch(logischer_name_rahmen1,html_datei1,
        logischer_name_rahmen2,html_datei2
    )
    {
        frames[logischer_name_rahmen1].location.href = html_datei1;
        frames[logischer_name_rahmen2].location.href = html_datei2;
    }
// -->
</SCRIPT>

<A HREF="javascript:parent.rahmentausch('r1','a.html','r2','b.html')">tauschen</A>
```

Beispiel für Inhalte beliebig vieler Frames als Ring tauschen:

Die logischen Framennamen werden als variable Argumenteliste übergeben und dienen der Indizierung der Frame im Dokument. Argumenttrenner sind Doppelpunkte.

Tausch:

erster Frame retten und dann mit zweiten Frame überschreiben  
zweiten Frame mit drittem Frame überschreiben  
.....  
vorletzten Frame mit letztem Frame überschreiben  
letzten Frame mit geretteten ersten Frame überschreiben

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    function rahmentausch()
    {
        if (arguments.length>1) // Länge ab 1, mindestens 2 Argumente
        {
            var pos_doppelpunkt = arguments[0].indexOf(":");

            if (pos_doppelpunkt != -1) // nächstes Argument ist vorhanden
            {
                var rette_logischer_framename= arguments[0].substring(0, pos_doppelpunkt);
```



```

for(var i = 0; i < arguments.length; i++) // Index ab 0
{
    pos_doppelpunkt = arguments[i].indexOf(":");

    if(pos_doppelpunkt != -1) // nächstes Argument ist vorhanden
    {
        // ersten zu ersetzenden Framenamen retten

        if (i=0)
        {var rette_logischer_frame_name =
            arguments[i].substring(0, pos_doppelpunkt)
        }

        // tauschen der logischen Framenamen
        logischer_frame_name_zu_ersetzender_frame =
            arguments[i].substring(0, pos_doppelpunkt);

        logischer_frame_name_ersetzender_frame =
            arguments[i].substring(0, pos_doppelpunkt + 1);

        frames[logischer_frame_name_zu_ersetzender_frame].location.href =
            logischer_frame_name_ersetzender_frame;
    }
}

frames[logischer_frame_name_ersetzender_frame].location.href =
    rette_logischer_frame_name;
}
}
}
// -->
</SCRIPT>
<A HREF="javascript:parent.rahmentausch('r1:a.html', 'r2:b.html', 'r3:c.html')">tauschen</A>

```

**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

**4.3.2.2.4.3.18. frameset Objekt**

Hinweis: Wenn ein Fenster mit dem Unterstrichsymbol rechts oben in der Fensterecke minimiert wurde, dann ist es durch Windows in die Hintergrund-Prioritätenfolge eingeordnet worden und arbeitet im Hintergrund bzw. garnicht. Mit anschließendem Maximieren (Symbol in der Fensterleiste rechts oben) wird das Fenster wieder angezeigt und das geladene Dokument **erneut** geöffnet. Konsequenz daraus ist, dass dieses Maximieren einem Neustart des Dokumentes entspricht, auch wenn der Programmierer diesen Zustand nicht erwünscht. Sound, der z.B. mit Start des Dokumentes erzeugt wird, erklingt erneut mit Maximierung nach einer Fensterminimierung. Analogon ist die Fenstereinstellung im Browser durch z.B. Ein- und Ausblenden der Favoritenleiste. Dieses Verhalten des Fensters mit seinem Dokument ist aber **nicht identisch** mit dem Verhalten nach einer Fenstergrößenänderung z.B. durch Rahmenverschiebung (resize). Die Aktionen des Fensters und seines Dokumentes bezüglich Resize müssen programmiert werden, sind also nicht standardmäßig vorhanden - im Gegensatz zum Verhalten mit/nach Fensterminimierung/-maximierung per Symbole in der rechten oberen Ecke der Fensterleiste. Sollte ein Frameset minimiert werden, dann wird das für den gesamten Frameset getan (Frameset hat 1 Fenster für alle Frames gesamt), allerdings mit Maximierung wird auch das Frameset-Dokument neu geladen. Wichtig dabei sind für Zeiger-Bezüge der Frames auf den Frameset per parent-Zeiger, dass die Daten im Frameset-Dokument mit Maximierung initialisiert werden und somit ebenfalls Daten der Frames, die im Frameset-Dokument abgelegt wurden, also z.B. Daten, die Verhaltensweisen der Frames vor einer Änderung der Frames-Inhalte gespeichert haben.

**Erzeugung unter HTML:**

```

<FRAMESET
    ROWS="zeilen_liste" oder COLS="spalten_liste" // Zeilenliste=Liste der Framehöhen
                                                // mit Kommatrennung
                                                // Spaltenliste=Liste der Framebreiten
                                                // mit Kommatrennung
    onLoad="evenhandler1" // Anweisungen aktiviert NACH laden ALLER Frames
    onUnload="eventhandler2"
    onerror="eventhandler3"
    onBlur="eventhandler4"
    onFocus="eventhandler5"
>

[<FRAME
    SRC="frame_url" // url der HTML-Seite als Frameinhalt
    NAME="logischer_frame_name"

```



```

    >]
    .....
    </FRAME>
    .....
</FRAMESET>

```

Rahmen zwischen den einzelnen Fenstern des Framesets entfernen:

```

<FRAMESET BORDER="0"
           FRAMEBORDER="0"
           FRAMESPACING="0"
           .....
>

```

Hinweise:

BORDER	bei Netscape Breite des Rahmens >= 0 Pixel	
FRAMEBORDER	bei IE Rahmenanzeige 0 oder "no" 1 oder "yes"	aus ein
FRAMESPACING	bei IE Rahmenbreite >=0 Pixel	

#### Zugriff:

```

document.ID_Frameset.eigenschaft
document.ID_Frameset.methode()

```

#### Möglichkeiten des Laden eines Dokumentes:

Laden eines fremden Dokumentes innerhalb eines Frame ist rechtlich nicht zulässig (Abmahnungsgefahr), wenn der Eigentümer der Fremdseite nicht explizit zugestimmt hat.

Das Laden einer neuen Webseite bei vorhandenem Frameset führt nur dann zur Darstellung außerhalb des Framesets, wenn ein Link z.B. <A> benutzt und dort ein **neues** Fenster geöffnet wird (TARGET-Attribut). Mit anderen Worten: Ein instanzierter Frameset ist z.B. per window.location.href oder window.location.replace() **nicht** aufhebbar, auch wenn das neue Dokument selbst einen Frameset instanziiert. Im letzteren Fall passiert optisch gesehen die Überlagerung des alten mit dem neuen Frameset. Allerdings hat der Programmierer mit Javascript ein riesen Problem: Der Zeiger parent als Zeiger auf dasjenige Dokument, das den Frameset deklariert, zeigt **nicht** auf das neu geladene Dokument mit neuem Frameset ! Daher werden alle Versuche, per Zeiger parent sich auf den neuen Eltern-Frameset zu beziehen, stets beim **vorherigen** Frameset landen und **nicht** beim neuen Dokument, das damit seine eigenen Frames nicht kennt. Somit werden nur Instanzen des alten Framesets vererbt, der auch die Kinder des neuen Frameset kennt, jedoch diese **ihn nicht**. Denn woher sollten die Kinder des neuen Framesets wissen, dass vorher bereits ein Frameset existiert und selbst wenn, sie können mit den Zwangs-Eltern in Form des alten Frameset nichts anfangen. Das Setzen des Zeigers parent auf den null-Wert wird vom Browser bemängelt. Die Verwendung des ID-Attributes im FRAMESET und das Null-Setzen per ID wird vom Browser akzeptiert, aber nicht ausgeführt. Das Schliessen des Eltern-Fensters per self.close() vor dem Umleiten per windows.location.href etc. wird ignoriert. Der Zeiger parent ist also nicht umzubiegen.

Einen angenehmen Effekt für den Programmierer hat der Frameset: Wer versucht, sich den Quelltext per Browsermenü Ansicht sich näher zu bringen, wird nur den Quellcode des FRAMESET-Dokumentes zu sehen bekommen.

#### **Laden eines fremden Dokumentes ohne Framedarstellung:**

```

<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
      <!--
        function laden()
        { location.href = "http://www.test.de";}
      // -->
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM>
      <INPUT TYPE="button"
            VALUE="www.test.de anwählen "
            onclick="laden()">
    </FORM>
  </BODY>
</HTML>

```

**Eigenes Dokument wird durch fremde Webseite geladen:**



**CopyRight-Meldung auf fremden Host erzeugen:**

Beispiel 1:

```
// In diesem Beispiel schreibt das Script Text auf die "entführte" Seite!
var HostUrl="www.test.de";

if (    (parent !=null)
    && (parent != self)
    )
{
    var host=parent.location.hostname;

    if(host != HostUrl)
    {
        document.write(
            "Diese Seite wurde ausgeliehen bei "
            + "<A HREF=\"" + location.href
            + "\" TARGET=\"_parent\""
            + ">"
            + HostUrl
            + "</A>"
        );
    }
}
```

Beispiel 2:

```
<BODY>
.....
if (    (parent != null)
    && (parent != self)
    )
{
    var meine_url = "www.test.de"
    var mein_host_name = "http://" + meine_url;

    var fremder_host_name = parent.location.hostname;

    if ( fremder_host_name != mein_host_name)
    {
        document.write(
            " Diese Seite liegt auf "
            + "<A HREF=\"" + location.href + "\"\"
            + " TARGET=\"_parent\""
            + ">"
            + "</A>"
            + " und stammt von "
            + mein_host_name
        );
    }
}
.....
</BODY>
```

**Framedarstellung der eigenen Webseite sofort und ohne Bildschirmmeldung aktivieren:**

Dieses Coding muss in jedes HTML-Dokument, das in einem Frame geladen wird. Bei Einzelaufruf des Dokumentes wird automatisch die Seite mit dem FRAMSET aktiviert, also die vollständige Framedarstellung.

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.2">
<!--
    var EigenerHost="http://www.test.de";
        // window.location.hostname ist Leerkette, wenn Browser offline
    var StartSeite="_start.html";

    function InaktiveFrameDarstellungAktivieren()
    {
        if (top.frames.length == 0)
        {
            // keine Frame-Darstellung aktiv, also diese aktivieren
```



```

        top.location.href = Startseite; // muss das FRAMESET enthalten !
    }
}
if ( (parent != null) // Elternobjekt existiert
    && (parent != self) // Eltern sind vorhanden Eltern: dieses Dokument (self) ist ein Kind
    )
{
    // aktuellen ElternHost ermitteln
    var ElternHost=parent.location.hostname;

    // ElternHost prüfen ob eigener und nicht leer, also Browser online ist
    if ( (ElternHost != "") // Browser ist online
        && (ElternHost != EigenerHost)
        )
    {
        // fremder Host, also als oberstes Fenster nun die Startseite anzeigen
        // und damit den eigenen Host aktivieren
        top.location.href=EigenerHost + '/' + Startseite;
    }
    else
    {
        // eigener Host und/oder Browser ist offline
        InaktiveFrameDarstellungAktivieren();
    }
}
else
{
    // Elternobjekt existiert nicht und/oder dieses Dokument (self) ist kein Kind
    InaktiveFrameDarstellungAktivieren();
}
}
//-->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>

```

**Reload eines Dokumentes mit allen seinen FRAMES:**

```

function frame_neu_laden()
{
    for (var i=0; i<windows.FRAMES.length; i++)
    { windows.frames[i].location.reload(false); }
}

<FRAMESET onResize="frame_neu_laden()">

```

**Datei ohne FRAMESET in eine Datei mit FRAMESET laden**

```

if (self.location == top.location)
{ location.href="/FRAMESET.html?" + escape(location.pathname); }

```

**Mehrere Frameinhalte gleichzeitig ändern:****Inhalte zweier Frames tauschen:**

Kodierung im Dokument, dass die beiden Frames definiert !

```

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    function rahmentausch(logischer_name_rahmen1,html_datei1,
        logischer_name_rahmen2,html_datei2
        )
    {
        frames[logischer_name_rahmen1].location.href = html_datei1;
        frames[logischer_name_rahmen2].location.href = html_datei2;
    }
// -->
</SCRIPT>

<A HREF="javascript:parent.rahmentausch(0, 'a.html', 1,b.html)">tauschen</A>

```

**Inhalte beliebig vieler Frames als Ring tauschen:**

Die logischen Framenamen werden als variable Argumenteliste übergeben und dienen der Indizierung der Frame im Dokument. Argumenttrenner sind Doppelpunkte.

Tausch:

erster Frame retten und dann mit zweiten Frame überschreiben  
 zweiten Frame mit drittem Frame überschreiben  
 .....  
 vorletzten Frame mit letztem Frame überschreiben  
 letzten Frame mit geretteten ersten Frame überschreiben

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
function rahmentausch()
{
    if (arguments.length>1)        // Länge ab 1, mindestens 2 Argumente
    {
        var pos_doppelpunkt = arguments[0].indexOf(":");

        if (pos_doppelpunkt != -1)  // nächstes Argument ist vorhanden
        {
            var rette_logischer_framename= arguments[0].substring(0, pos_doppelpunkt);

            for(var i = 0; i < arguments.length; i++)    // Index ab 0
            {
                pos_doppelpunkt = arguments[i].indexOf(":");

                if(pos_doppelpunkt != -1)  // nächstes Argument ist vorhanden
                {
                    // ersten zu ersetzenden Framenamen retten

                    if (i=0)
                    {var rette_logischer_framename =
                        arguments[i].substring(0, pos_doppelpunkt)
                    }

                    // tauschen der logischen Framenamen
                    logischer_framename_zu_ersetzender_frame =
                        arguments[i].substring(0, pos_doppelpunkt);

                    logischer_framename_ersetzender_frame =
                        arguments[i].substring(0, pos_doppelpunkt + 1);

                    frames[logischer_framename_zu_ersetzender_frame].location.href =
                        logischer_framename_ersetzender_frame;

                }
            }

            frames[logischer_framename_ersetzender_frame].location.href=
                rette_logischer_framename;

        }
    }
}
// -->
</SCRIPT>
<A HREF="javascript:parent.rahmentausch('r1:a.html', 'r2:b.html', 'r3:c.html')">tauschen</A>
```

### **Frame und Datenaustausch:**

Zwischen Frames können Daten ausgetauscht werden.

Beispiel Adressierung eines Frame über das FRAMESET

```
<FRAMESET .... >
    <FRAMESET ..... >
        <FRAME SRC="test.html" NAME="test">
        <FRAME SRC="test1.html" NAME="test1">
    </FRAMESET>
    <FRAME SRC="test2.html" NAME="test2">
</FRAMESET>

parent.test2.location.href="neu.html";    // Laden von neu.html in den Frame test2
```

Beispiel: Auf Objekte im FRAMESET-Dokument durch ein FRAME-Dokument zugreifen



```

im FRAMESET-Dokument wird kodiert      var Kette="Hallo !";
im FRAME-Dokument wird auf Kette zugegriffen:  alert(parent.Kette);

```

Beispiel: Auf Objekte im FRAME-Dokument durch das FRAMESET-Dokument zugreifen

```

im FRAME-Dokument wird kodiert      var Kette="Hallo !";
<FRAME ...NAME ="test2" ...>
im FRAMESET-Dokument wird auf Kette zugegriffen:  alert(parent.test2.Kette);

```

Beispiel: Textdaten-Übergabe durch an Url angehängte Textdaten

**quelle.htm:           lädt ziel.htm  
                          übergibt Textdaten an ziel.htm**

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function ziel_seite_laden_mit_datenuebergabe(uebergabe_daten)
{location.href = "ziel.htm?" + escape(uebergabe_daten);}
// Url von ziel.htm als Suchbegriff mit angehangenen Daten merken, die per escape() in das
// Url-Format konvertiert wurden
// -->
</SCRIPT>
</HEAD>

<BODY>
An ziel.htm zu &uuml;bergabenden Text eingeben:
<FORM>
  <TEXTAREA       NAME=eingabe
                  ROWS=5
                  COLS=40
  >
  </TEXTAREA>
  <BR>
  <INPUT   TYPE=button
          VALUE="Zielseite laden"
          onClick="ziel_seite_laden_mit_datenuebergabe(this.form.eingabe.value)">
</FORM>
</BODY>
</HTML>

```

**ziel.htm:           wird durch quelle.htm geladen  
                          erhält Textdaten von quelle.htm**

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
datenuebernahme()
{
// Url mit angehangenen Daten holen
// search liefert ?daten
uebernahme_daten= location.search;

// ? abschneiden
uebernahme_daten= uebernahme_daten.substring(1, uebernahme_daten.length);

// unescape: von Url-Format nach Zeichenkette
document.ausgabe.ausgabefeld.value=unescape(uebernahme_daten);
}

// -->
</SCRIPT>
</HEAD>

<BODY onLoad="datenuebernahme ()">
Von quelle.htm &uuml;bernommener Text:
  <FORM NAME=ausgabe>
    <TEXTAREA       NAME=ausgabefeld
                  ROWS=10 COLS=40>

```



```

        </TEXTAREA>
    </FORM>
</BODY>
</HTML>

```

Beispiel: Textdaten-Übergabe durch Fenster-Handle

Die Textdaten und die der Javascript-Code, der die Textdaten verarbeitet, sind **getrennte** HTML-Dokumente. Nachteil dieser Variante ist, dass in beiden Dokumenten die logischen Namen vom Formular und dem Textarea identisch

kodiert werden müssen, also eine doppelte Verwaltung nötig ist.

#### HTML-Dokument, das die Textdaten enthält:

```

<HTML>
<BODY>
    <FORM NAME=formular>
        <TEXTAREA NAME=text_area>
            // Hier die Textdaten als Wert von TEXTAREA
        </TEXTAREA>
    </FORM>
</BODY>
</HTML>

```

#### Dokument, das die Textdaten verarbeitet:

```

<HTML>
<HEAD>
    <SCRIPT LANGUAGE="JavaScript">
<!--
    function TextAreaWertLesen(url)
    {
        // url enthält den Pfad und den Namen des Dokumentes, dass die
        // Textarea-Werte enthält
        var handle = window.open(url_der_datendatei,"", "width=100,height=100");

        // Handle erzeugen und Fenster mit dem Textarea anzeigen
        // (Fenstergröße ist egal)
        var data = handle.document.forms[formular].elements[text_area].value;

        handle.close();

        return data;
    }
    // -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>

```

#### frameset Objekt des Internet Explorer:

Container aller Frames

Die Collection frames wird unter document.frames beschrieben !

Das Laden einer neuen Webseite bei vorhandenem Frameset führt nur dann zur Darstellung außerhalb des Framesets, wenn ein Link z.B. <A> benutzt und dort ein **neues** Fenster geöffnet wird (TARGET-Attribut). Mit anderen Worten: Ein instanzierter Frameset ist z.B. per window.location.href oder window.location.replace() **nicht** aufhebbar, auch wenn das neue Dokument selbst einen Frameset instanziiert. Im letzteren Fall passiert optisch gesehen die Überlagerung des alten mit dem neuen Frameset. Allerdings hat der Programmierer mit Javascript ein riesen Problem: Der Zeiger parent als Zeiger auf dasjenige Dokument, das den Frameset deklariert, zeigt **nicht** auf das neu geladene Dokument mit neuem Frameset ! Daher werden alle Versuche, per Zeiger parent sich auf den neuen Eltern-Frameset zu beziehen, stets beim **vorherigen** Frameset landen und **nicht** beim neuen Dokument, das damit seine eigenen Frames nicht kennt. Somit werden nur Instanzen des alten Framesets vererbt, der auch die Kinder des neuen Frameset kennt, jedoch diese **ihn nicht**. Denn woher sollten die Kinder des neuen Framesets wissen, dass vorher bereits ein Frameset existiert und selbst wenn, sie können mit den Zwangs-Eltern in Form des alten Frameset nichts anfangen. Das Setzen des Zeigers parent auf den null-Wert wird vom Browser bemängelt. Die Verwendung des ID-Attributes im FRAMESET und das Null-Setzen per ID wird vom Browser akzeptiert, aber nicht ausgeführt. Das Schliessen des Eltern-Fensters per self.close() vor dem Umleiten per window.location.href etc. wird ignoriert. Der Zeiger parent ist also nicht umzubiegen. Die Anwendung eines Zeigerbezuges vom alten Frameset über den neuen Frameset auf die Kinder des neuen Frameset, also die Verkettung von parent-Zeigern, ist ziemlich unsinnig, da parent ansich der Zeiger auf das aktuelle Frameset sein sollte, auch wenn in der Objekthierarchie der neue Frameset ein Kind vom alten Frameset sein müsste. Alternativ zu dieser unangenehmen Framesetüberlagerung sind Konstruktionen von DIV-Objekten innerhalb eines **gemeinsamen** Dokumentes möglich, wobei in die DIV's dann je ein Dokument geladen wird (document.open()). Es können dann alle Dokumente direkt über das ID des jeweiligen DIV's adressiert werden, allerdings haben diese DIV's eben **keine** Fenstereigenschaften wie Frames im gemeinsamen Frameset-Fenster. (DIV- und FRAME/IFRAME-Objekte unterscheiden sich in Eigenschaften und Methoden). So gesehen stellt document.open() eine eingeschränkte Lösung dar. HTML-orientiert ist die Nutzung eines



Links, z.B. <A> mit Targetwert\_top, also dem Zeiger auf das oberste Fenster, in dem der alte Frameset sitzt. Nur ist ein Link in der Regel visuell vorhanden und muss durch den User aktiviert werden. Letzte Alternative ist die Umlenkung per META-Tag anhand eines Timers.

Hinweis: Wenn ein Fenster mit dem Unterstrichsymbol rechts oben in der Fensterecke minimiert wurde, dann ist es durch Windows in die Hintergrund-Prioritätenfolge eingeordnet worden und arbeitet im Hintergrund bzw. garnicht. Mit anschließendem Maximieren (Symbol in der Fensterleiste rechts oben) wird das Fenster wieder angezeigt und das geladene Dokument **erneut** geöffnet. Konsequenz daraus ist, dass dieses Maximieren einem Neustart des Dokumentes entspricht, auch wenn der Programmierer diesen Zustand nicht erwünscht. Sound, der z.B. mit Start des Dokumentes erzeugt wird, erklingt erneut mit Maximierung nach einer Fensterminimierung. Analogon ist die Fenstereinstellung im Browser durch z.B. Ein- und Ausblenden der Favoritenleiste. Dieses Verhalten des Fensters mit seinem Dokument ist aber **nicht identisch** mit dem Verhalten nach einer Fenstergrößenänderung z.B. durch Rahmenverschiebung (resize). Die Aktionen des Fensters und seines Dokumentes bezüglich Resize müssen programmiert werden, sind also nicht standardmäßig vorhanden - im Gegensatz zum Verhalten mit/nach Fensterminimierung/-maximierung per Symbole in der rechten oberen Ecke der Fensterleiste. Sollte ein Frameset minimiert werden, dann wird das für den gesamten Frameset getan (Frameset hat 1 Fenster für alle Frames gesamt), allerdings mit Maximierung wird auch das Frameset-Dokument neu geladen. Wichtig dabei sind für Zeiger-Bezüge der Frames auf den Frameset per parent-Zeiger, dass die Daten im Frameset-Dokument mit Maximierung initialisiert werden und somit ebenfalls Daten der Frames, die im Frameset-Dokument abgelegt wurden, also z.B. Daten, die Verhaltensweisen der Frames vor einer Änderung der Frames-Inhalte gespeichert haben.

### Zugriff auf Frameset-Eigenschaft:

Beispiel

```
var Wert = document.all.ZeigerAufFrameset.eigenschaft;
```

mit ZeigerAufFrameset laut ID-Attribut

```
<FRAMESET ID="ZeigerAufFrameset" ..... >
```

### Eigenschaften:

ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.border	Rahmendicke in Pixel
.borderColor	Borderfarbe (Rahmenfarbe)
	wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"
	Eigenschaft .border mit Wert 0
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.cols	Breite aller Frames eines Frameset (Breite des Frameset als Container)
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.frameBorder	Borderanzeige ein/aus beim Frame
.frameSpacing	Zwischenraum in Pixel zwischen 2 benachbarten Frames
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
	Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
	Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
	ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
	dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
	also nach dem Laden des Objektes
	aber wirksam erst mit parsen des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit
	nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!)
	muss beim Formular für alle zu sendenden Felder kodiert sein !!
	Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element)
	also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes)
	nur für Text- und Attribut-Elemente
	nicht für Element-Knoten (Knotentyp 1)
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
	wirksam mit parsen des Ende-Tag
	nur nach kompletten Einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren



.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.rows	Höhe aller Frames eines Frameset (Höhe des Frameset als Container)
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes
	Anspringen verbunden mit Focus erhalten
	--> Ereignisse werden ausgelöst !!
	unter IE 5.x
	onblur, onfocus
	ab IE 5.x
	onblur, onfocus, onkeydown, onkeypress, onkeyup
	Anspringen default per TAB-Taste
	für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA
	Anspringen default nicht per TAB-Taste
	für APPLETT, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes
	Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde
	kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.width	Breite des Objektes in Pixel
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt



	(also Standardbehandlung aktiv)
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByTagName()
.hasChildNodes()	DOM nicht geändert prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern



.replaceChild()	DOM wird nicht geändert Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceNode()	DOM wird geändert Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
.setActive()	DOM wird geändert Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
.setAttributeNode()	DOM wird nur bei Erzeugung geändert Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.19. document.html Objekt des Internet Explorer

Das Objekt repräsentiert den HTML-Tag.

Scriptsteuerung erst ab IE 4.x

sonst siehe HTML-Dokument mit Scriptcode

Beispiel

```
<HTML>
<BODY>
  <P>This is an HTML document.</P>
</BODY>
</HTML>
```

#### Eigenschaften:

.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
.isDisabled	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes)



	nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scroll	Scrollbar-Anzeige ein/aus vor IE 6.x nur BODY-Objekt ab IE 6.x alle HTML-Elemente wenn DOCTYPE im Kopf des Dokumentes kodiert
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
.version	Document Type Definition-Version (DTD-Version) ab IE 5.x
XMLNS	Namensraum für Benutzer-Tags zu einem HTML-Tag ab IE 5.x nur für das Tag HTML möglich Namensraum: Präfix des Benutzer-Tags oder Uniform Resource Name (URN)
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos



	erreicht haben
	Overbereich der Maus ist mehr als 1 Pixel gross
	beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
	DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht
	DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern
	DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar
	DOM nicht geändert
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
	DOM nicht geändert
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
	DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
	DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt



	ersetzende Objekt muss per Methode .createElement() erzeugt worden sein
	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern
	sichtbar erst mit parsen des Endetags
	DOM wird geändert
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern
	DOM wird geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch)
	nur sichtbar wenn Endetag geparkt
	DOM wird geändert

#### 4.3.2.2.4.3.20. html comment Objekt des Internet Explorer

repräsentiert den HTML-Kommentar, der weder geparkt noch gerendert wird

Beispiel

```
<!-- This text will not appear in the browser window. -->
```

**Event:** nur onlayoutcomplete

#### Eigenschaften:

.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
	ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
	dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
	also nach dem Laden des Objektes
	aber wirksam erst mit parsen des HTML-Endetags
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
	wirksam mit parsen des Ende-Tag
	nur nach kompletten Einlesen des Dokumentes nutzbar
.tagName	Tag-Bezeichner des Objektes
.text	Text des HTML-Kommentares

#### 4.3.2.2.4.3.21. iframe Objekt des Internet Explorer

ist dem DIV-Objekt sehr ähnlich

ist eine effektivere Variante des FRAME-Objektes

ACHTUNG: Die Ausführung von Scripten etc. im IFRAME können ab IE 6.0 per Browser-Menü "Extra-Internet Optionen-Sicherheit" unter "Programme und Dateien in einem IFRAME starten" durch den User auf deaktivieren gesetzt werden.

Damit ist der IFRAME nicht mehr nutzbar bezüglich Scripts, Protokolle und URL-Aufrufen

z.B. dann im IFRAME nicht ausführbar Code

```
<IFRAME src="http://www.test.de"></IFRAME>
```

Alternativ wird ein neues Fenster geöffnet, wenn

ab IE 4.x

ist ein Dokument in einem Dokument

Eltern des obersten IFRAME ist das BODY-Objekt

benutzt folgende Collections frames

document.all

Die Collection frames wird unter document.frames beschrieben !

Beispiele

```
<IFRAME ID="ID_Frame" IFrame1 FRAMEBORDER=0 SCROLLING=NO SRC="sample.htm">
  <A HREF="http://www.test.de" TARGET="ID_Frame"> www.test.de </A>
</IFRAME>
```

```
var ZeigerAufDocumentAllImFrame = document.frames("IFrame1").document.all
```

```
var HintergrundFarbeImFrame = document.frames("sFrameName").document.body.style.backgroundColor;
```

```
var BoderWert = document.all.zeiger_auf_frame.style.border;
```

```
var MarginWidthWert = document.all.id_des_frame.marginWidth
```

Erweiterungen ab IE 5.5

Transparenz: möglich, wenn

im IFRAME das Attribut ALLOWTRANSPARENCY auf "true" gesetzt ist

Bsp.: <IFRAME NAME="Frame1" SRC="frame.htm" ALLOWTRANSPARENCY="true">

```
</IFRAME>
```

UND Eltern des IFRAME, z.B. BODY, mit background-color auf "transparent" gesetzt ist

```
<BODY STYLE="background-color:transparent">
```

Hinweis: Attribut BGCOLOR ist deprecated !!!

IFRAME ohne Fensterrahmen-Elemente möglich, auch mit Überlappung

sind schneller in der Darstellung als FRAME mit Fensterrahmen-Elementen



nutzen weniger RAM  
scrollen schneller

Attribut z-index nutzbar für Überlappung von IFRAME

z.B. <IFRAME SRC="frame.htm" STYLE="z-index:1" ></IFRAME>

je höher der z-index, um so höher der IFRAME in der Hierarchie  
oberster IFRAME hat höchsten z-index  
z-index kann auch < 0 sein

### Eigenschaften:

.align	Ausrichtung
.allowTransparency	Transparenz ein/aus Transparenz: z.B. Hintergrundfarbe des Elternobjektes wird zur Hintergrundfarbe des Frames Achtung: Wenn Frame-eigenes STYLE-Attribut mit Wert für background-color also STYLE="background-color: farb_bezeichner" kodierte wurde, soe wird die Transparenz ignoriert und die Hintergrundfarbe laut STYLE verwendet
APPLICATION	Umgebung des Frames mit Vertrauen-Status im Rahmen des Browser-Sicherheitsmodells Attribut wird nicht vererbt an Kinder-Frame
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.border	Rahmendicke in Pixel
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.contentWindow	Referenz auf das zugehörige Fenster-Objekt laut Collection document.all
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.frameBorder	Borderanzeige ein/aus beim Frame
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.height	Höhe des Objektes in Pixel
.hideFocus	Focussierbarkeit
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.longDesc	Uniform Resource Identifier (URI) zur einer "long description" umwandeln
.marginHeight	Abstand in Pixel vom unteren zum oberen Rand des Objektes
.marginWidth	Abstand in Pixel vom linken zum rechten Rand des Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)



.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrolling	Scrollenbar erzeugen
SECURITY	temporäre Sicherheit des IFRAME wird an Kinder weitervererbt ab IE 6.0
.sourceIndex	Index des Objektes in der Collection document.all
.src	Url der Daten z.B. vom Image in normaler Auflösung
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLETT, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.vspace	vertikaler Abstand in Pixel zum Elternobjekt
.width	Breite des Objektes in Pixel
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern



	DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar
.cloneNode()	DOM wird geändert Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
.detachEvent()	DOM nicht geändert Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten



	(Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endtags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient



Ausdruck nur als Script kodierbar  
 DOM wird nicht geändert  
 .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
 nur sichtbar wenn Endtag geparkt  
 DOM wird geändert

#### 4.3.2.2.4.3.22. img Objekt

##### Erzeugung unter HTML:

Beispiele:

Bild nicht im Formular:

```
<IMG SRC="url_des_bildes_in_voller_auflösung"
NAME="logischer_bild_name"
LOWSRC="url_des_bildes_in_geringer_auflösung"
ALT="alternativer_text"
ALIGN=ausrichtung
HEIGHT=hoehe
WIDTH=breite
BORDER=rahmenbreite
HSPACE=abstand_links_und_rechts_zur_umgebung
VSPACE=abstand_oben_und_unten_zur_umgebung
ISMAP
USEMAP="map_url#image_map"
oder "map_name"
onAbort="eventhandler1"
onerror="eventhandler2"
onLoad="eventhandler3"
>
```

Bild im Formular:

```
<INPUT TYPE=image
SCR=url_des_bildes_in_voller_auflösung...
>
```

Beispiel für Linie mit variabler Dimension:

Es wird ein Bild aus 1x1 neu dimensioniert, das eine kleine Dateigröße hat und nur 1 mal geladen werden muss.  
 Durch die Angaben von Breite und Höhe kann eine vertikale oder horizontale Linie erzeugt werden. Die  
 Linienpositionierung erfolgt per STYLE-Attribut.

```
horizontale Linie mit 10 Pixel Dicke:
<IMG SRC="1x1bild.gif" WIDTH="300" HEIGHT="10" BORDER="0" ALT="" STYLE=" ....">
vertikale Linie mit 10 Pixel Dicke:
<IMG SRC="1x1bild.gif" WIDTH="10" HEIGHT="300" BORDER="0" ALT="" STYLE=" ....">
```

##### Erzeugung in Script:

```
var Bild = new Image([breite, hoehe]);
```

erzeugt Instanz und lädt zugleich das Bild  
 zeigt das Bild NICHT an  
 Verwendung z.B. bei animiertem Bild, wobei die Animation geladene Bilder voraussetzt

##### Zugriff:

Bild nicht im Formular:

```
document.ID_Img.eigenschaft
document.images[index].eigenschaft

index: ab 0
muss in [ ] kodiert sein
```

Bild im Formular:

```
document.ID_Img.eigenschaft
document.images[index].eigenschaft
document.ID_Formular.elements[index].eigenschaft

index: ab 0
muss in [ ] kodiert sein
```

Bild in Script: per Variable aus new

Beispiel für Belegen des Attributes SRC am Beispiel des Vorladens eines Bildes:

```
var Bild_Hoehe=45; var Bild_Breite=60; var Bild_Url="test.gif";
var BildObjekt=new Image(Bild_Breite,Bild_Hoehe);
BildObjekt.src=Bild_Url; // ohne "" kodieren da Zeiger
document.write('<IMG NAME="IMG_Bild"
```



```

+ ' SRC="' + BildObjekt.src + ""
+ ' HEIGHT=' + Bild_Hoehe
+ ' WIDTH=' + Bild_Breite
+ '>'
);

```

**Eigenschaften (ausgewählte):**

.border	entspricht BORDER nur lesen
.complete	wenn mit <b>true</b> belegt, so Bild komplett geladen wenn mit <b>false</b> belegt, so Bild noch nicht komplett geladen nur lesen
.height	entspricht HEIGHT nur lesen
.hspace	entspricht HSPACE nur lesen
.lowsrc	entspricht LOWSRC
.name	entspricht NAME nur lesen
.src	entspricht SRC
.vspace	entspricht VSPACE; Standard ist 0 nur lesen
.width	entspricht WIDTH nur lesen

Beispiel für Belegen des Attributes SRC am Beispiel des Vorladens eines Bildes:

```

var Bild_Hoehe=45; var Bild_Breite=60; var Bild_Url="test.gif";
var BildObjekt=new Image(Bild_Breite,Bild_Hoehe);
BildObjekt.src=Bild_Url; // ohne "" kodieren da Zeiger
document.write(
    '<IMG NAME="IMG_Bild"'
    + ' SRC="' + BildObjekt.src + ""
    + ' HEIGHT=' + Bild_Hoehe
    + ' WIDTH=' + Bild_Breite
    + '>'
);

```

**Events bei sich überlagernden Objekten:**

Beispiel: Sollte ein Image mit seiner Erzeugung ein anderes Image überlagern, so ist die Eventübergabe von und an das untere Image unterbrochen: Ereignisse onXXX kommen nicht mehr durch, solange das untere Bild nicht den **Fokus** erhält. Alternativ ist die Style-Eigenschaft z-index zu kodieren und dann der z-index auf den obersten zu setzen.

**img Objekt des Internet Explorer:**

HTML-Container für ein Bild, Videoclip, VRML-Datei  
Scriptsteuerung erst ab IE 4.x  
Anwendung auch im Formular per input image Objekt  
Die Collection images wird unter document.images beschrieben !

ab IE 5.x werden folgende Image-Dateiarten unterstützt:

*.avi	Audio-Visual Interleaved (AVI)
*.bmp	Windows Bitmap (BMP)
*.emf	Windows Enhanced Metafile (EMF)
*.gif	Graphics Interchange Format (GIF)
*.jpg, *.jpeg	Joint Photographic Experts Group (JPEG)
*.mov	Apple QuickTime Movie (MOV)
*.mpg, *.mpeg	Motion Picture Experts Group (MPEG)
*.png	Portable Network Graphics (PNG)
*.wmf	Windows Metafile (WMF)
*.xbm	X Bitmap (XBM)

Hinweis: Ereignis onfocus nicht ausgelöst bei einem MAP-Image

Kodierung der Datei-Url:

für statisches Bild das Attribut SRC verwenden

für Videoclip oder VRML-Datei (virtual reality modeling language-Datei) das Attribut DYN SRC verwenden

Beispiel

```
<IMG SRC=mygraphic.bmp>
```

Beispiel für Aus- und Einblende eines Bildes:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
var Sichtbar=true; // DIV ist natürlich nach dem Dokument-Laden sichtbar
var StandardFarbe="green";

```



```

function Blenden()
{
    // Fade zurücksetzen
    DIV_ID.filters[0].Apply();

    if (Sichtbar)
    {
        Sichtbar=false;
        DIV_ID.style.visibility="hidden";
    }
    else
    {
        Sichtbar=true;
        DIV_ID.style.visibility="visible";
        DIV_ID.style.backgroundColor=StandardFarbe;
    }

    // Fade starten
    DIV_ID.filters[0].Play();
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<BUTTON onclick="Blenden()">
    Aus- und Einblenden
</BUTTON>
<BR>
<BR>
<DIV ID="DIV_ID"
    STYLE="height:250px;width:250px;background-color:red;
        filter:progid:DXImageTransform.Microsoft.Fade(duration=1);"
    // alles EINE Zeile !!
>
    <IMG SRC="Bild.gif" ...HEIGHT=250 WIDTH=250>
    // Bild-Dimension muss in die DIV-Dimension reinpassen !
</DIV>
</BODY>
</HTML>

```

Beispiel für Aus- und Einblende von Bildern mit Bildwechsel:

#### **Variante 1**

Es werden zwei DIV an absoluter Position überlagert. Jedes DIV hat einen eigenen Inhalt, hier im Beispiel sein eigenes Bild. Durch das Ein- und Ausblenden wird ein Slide-Show-Effekt der beiden Bilder erreicht. Pro Bild wird ein eigenes DIV erzeugt.

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var DIV_ID1_Sichtbar=true; // DIV_ID1 ist nach dem Dokument-Laden sichtbar
    // DIV_ID2 aber nicht

    function BildWechsel()
    {
        // Fade zurücksetzen
        DIV_ID1.filters[0].Apply;
        DIV_ID2.filters[0].Apply;

        if (DIV_ID1_Sichtbar)
        {
            DIV_ID1_Sichtbar = false;
            DIV_ID1.style.visibility="hidden";
            DIV_ID2.style.visibility="visible";
        }
        else
        {
            DIV_ID1_Sichtbar = true;
            DIV_ID1.style.visibility="visible";
            DIV_ID2.style.visibility="hidden";
        }
        // Fade starten
    }

```



```

        DIV_ID1.filters[0].Play();
        DIV_ID2.filters[0].Play();
    }
// -->
</SCRIPT>
</HEAD>
<BODY>
    <BUTTON onclick="BildWechsel()">
        Bild wechseln mit Aus- und Einblenden
    </BUTTON>
    <BR>
    <BR>
    <DIV ID="DIV_ID1"
        STYLE="height:250px;position:absolute;top:50;width:250px;
            background-color:red; visibility:visible
            filter:progid:DXImageTransform.Microsoft.Fade(duration=1);"
        // alles EINE Zeile
    >
        <IMG SRC="DIV_ID1_Bild.gif" ....HEIGHT=250 WIDTH=250>
        // Bild-Dimension muss in die DIV-Dimension reinpassen !
    </DIV>

    <DIV ID="DIV_ID2"
        STYLE="height:250px;position:absolute;top:50;width:250px;
            background-color:red; visibility:hidden
            filter:progid:DXImageTransform.Microsoft.Fade(duration=1);"
        // alles EINE Zeile
    >
        <IMG SRC="DIV_ID2_Bild.gif" ....HEIGHT=250 WIDTH=250>
        // Bild-Dimension muss in die DIV-Dimension reinpassen !
    </DIV>
</BODY>
</HTML>

```

**Variante 2**

Es wird der innere DIV-Bereich zwischen <DIV> und </DIV> ersetzt und damit je ein Bild dargestellt. Durch das Ein- und Ausblenden wird ein Slide-Show-Effekt der Bilder erreicht.

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var BildUrlFeld=Array
    (
        // hier beliebig viele Bilder eintragen
        "test1.jpg",
        "test2.jpg",
        "test3.jpg"
    );

    var BildUrlFeld_Laenge= BildUrlFeld.length;

    // fuer alle Bilder die identischen Dimensionen !!!
    var BildHoehe=444;
    var BildBreite=640;

    // Zeigerfeld zum Vorladen der Grafiken
    var BildZeigerFeld = Array();

    var Index=1; // Starten mit test2.jpg, da test1.jpg bereits angezeigt mit Laden des DIV

    function BildWechsel()
    {
        // Fade zurücksetzen
        DIV_ID.filters[0].Apply();

        // Bild im DIV anzeigen
        document.AktuellesBild.src= BildZeigerFeld[Index].src;

        // Fade starten
        DIV_ID.filters[0].Play();

        // nächstes Bild einstellen
        Index++;
    }

```



```

        // Index korrigieren
        if (Index >= BildUrlFeld_Laenge)
        {Index=0;}
    }

    // nachfolgender Code wird mit dem Laden des HEAD-Teiles abgearbeitet

    // Bildobjekte vorladen: allokalieren und Zeiger merken per Prototyping
    for (i=0; i<= BildUrlFeld_Laenge; i++)
    {
        // Image-Zeiger bilden als Feldeintrag
        BildZeigerFeld[i] = new Image();

        // Url dem Zeiger zuweisen und Bild somit vorladen
        BildZeigerFeld[i].src= BildUrlFeld[i];
    }

    // DIV erzeugen mit test1.gif als Startbild
    //                               in Dimension aller Bilder
    document.write(
        '<DIV '
        + 'ID="DIV_ID" '
        + 'STYLE="height:' + BildHoehe + 'px;width:' + BildBreite + 'px;'
        + 'filter:progid:DXImageTransform.Microsoft.Fade(duration=1);'
        + ' "'
        + '>\n'
    );

    document.write(
        '<IMG NAME="AktuellesBild" '
        + 'SRC="' + BildZeigerFeld[0].src + "' '
        + 'WIDTH=' + BildBreite + ' '
        + 'HEIGHT=' + BildHoehe
        + '>\n'
    );

    document.write( '</DIV>\n');

    // Button zum Bildwechsel anzeigen
    document.write(
        '<BUTTON onclick="BildWechsel()">'
        + 'Naechstes Bild'
        + '</BUTTON>\n'
    );

    // -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>

```

Beispiel für mehrere Bilder in den RAM laden und mit Wartezeit preloaden:

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
    var bild_feld = [ "b1.gif",
                    // hier alle Bilder eintragen.
                    ];

    var wartezeit = 500; // Zeit in ms zwischen zwei Ladevorgaengen
    var feld_index = 0;

    function preloaden()
    {
        var bild = new Image();
        bild.src = bild_feld[feld_index];

        feld_index ++;

        if(feld_index < bild_feld.length) // Länge ab 1, Index ab 0
        {setTimeout('preloaden()', load_next)}
    }

    function start()

```



```

        {setTimeout('preloaden()', wartezeit)}
    // -->
</SCRIPT>
</HEAD>
<BODY ... onLoad="start()">
</BODY>
</HTML>

```

Beispiele für Bildfolge:

#### Beispiel 1:

10 Bilder mit Namen b1.gif bis b10.gif. Wichtig ist der numerische Namenteil 1 bis 10, da als Zähler verwendet

```

<HTML>
<HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

        verzoegerung = 120; // in Millisekunden
        bildNummer = 2;

        bilder= new Array(); //nimmt die Bilder der Animation auf

        // hier werden die Bilder im Hintergrund geladen
        for (i = 1; i <= 10; i++)
        {
            bilder[i] = new Image();
            bilder[i].src = "b" + i + ".gif";
        }

        function naechstesBild()
        {
            document.animation.src = bilder[bildNummer].src;
            bildNummer++;
            if (bildNummer > 10) bildNummer = 1;
        }

    // -->
</SCRIPT>
</HEAD>
<BODY>
    <IMG SRC="dp1.gif"
        NAME="animation"
        WIDTH="165"
        HEIGHT="185"
        onLoad="setTimeout('naechstesBild()', verzoegerung)">

</BODY>
</HTML>

```

#### Beispiel 2:

Es können **geladene Bilder** bild1.gif bis bild5.gif an fester Position im Wechsel ausgegeben werden. geladenen Bilder erzeugen per Image-Objekt  
 erstes Bild wird angezeigt per <IMG>  
 alle Bilder müssen gemeinsame Dimension haben, sonst wird gestreckt oder gestaucht  
 Durch Aktualisierung vom Attribut SRC aus <IMG> mit der jeweiligen Url des Bildes wird der sichtbare Bildwechsel vollzogen.

```

<HEAD>
<SCRIPT ...>
<!-- // alle Bilder vorladen

    // globale Größen festlegen

    var bild_anzahl=5;           // ab 1

    var feld=new Array(bild_anzahl); // Inhalt des Feldes ist hier noch unbekannt

    feld[1]=new Image();        // Feldelement 1 ist Image-Objekt
                                // Bild 1 wird vorgeladen und nicht nicht agenzeigt,
                                // da der Javascript-Code VOR dem IMG-Tag
                                // aus <BODY> .. </BODY> abgearbeitet wird !

    feld[1].url="bild1.gif";    // Prototyping: Eigenschaft url hinzufügen für Index 1
    .....

```



```

feld[5]=new Image();           // Feldelement 5 ist ein Image-Objekt
feld[5].url="bild5.gif";       // Prototyping: Eigenschaft url hinzufügen für Index 5

var index=bild_anzahl;         // auch möglich           var index= feld.length;

function bild_wechsel()
{
    // Nummer des aktuellen Bild ermitteln
    if (index == bild_anzahl)
    {index=0;}

    index+=1;

    // Kopieren der Url nach <IMG>-Attribut SRC
    //      also gleichzeitig anzeigen
    self.document.logischer_img_name.url=feld[index].url;
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<IMG NAME="logischer_img_name"
SRC="bild1.gif"                // vorgeladenes Bild anzeigen
>
<SCRIPT>
<!--
    setInterval(bild_wechsel,2000); // periodische Abarbeitung alle 2 Sekunden
// -->
</SCRIPT>
</BODY>

```

Beispiele für festes, nicht mitscrollendes Bild:

#### Beispiel 1:

festes Grafik in einem Fenster für Internet Explorer oder Netscape

```

<HTML>
<HEAD>
<TITLE> .... </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
    posX=20;                // horizontal
    posY=20;                // vertikal
    idName="grafik_name";

    function merkeYPosition()
    {
        if (document.all) // IE
        {merkeY=document.body.scrollTop;}
        else // Netscape
        {
            if (document.layers)
            {merkeY=pageYOffset;}
        }
    }

    funktion setzeYPosition()
    {
        merkeYPosition();
        if (document.all)
        {document.all[idName].style.top=merkeY+posY;}
        else
        {
            if (document.layers)
            {document.layers[idName].top=merkeY+posY;}
        }
    }

    function merkeXPosition()
    {
        if (document.all) // IE
        {merkeX=document.body.scrollLeft;}
        else // Netscape
        {
            if (document.layers)
            {merkeX=pageXOffset;}
        }
    }

```





```

//*****
//
//      nachfolgenden Code nicht verändern
//
//*****
//      Variablen hier ohne extra Funktion intitialisieren
//      werden automatisch beim Dokumentladen belegt
//
//      BrowserTyp-Ermittlung
//      Annahme Netscape läuft
var BrowserTyp=true;
if (document.all)
{
    // IE läuft
    BrowserTyp=false;
}

// Browser-Fenster-Variablen
var BrowserFenster_AktuelleBreite    =0;
var BrowserFenster_AktuelleHoehe    =0;
var BrowserFenster_AktuelleScrollPositionX =0;
var BrowserFenster_AktuelleScrollPositionY =0;

// Grafik-Positions-Korrektur-Variablen
var GrafikPositionsKorrekturX=0;
var GrafikPositionsKorrekturY=0;

function GrafikPositionKorrigieren()
{
    // aktuelle Browserfenster-Daten ermitteln
    if (BrowserTyp)
    {
        // Netscape läuft
        BrowserFenster_AktuelleBreite    =window.innerWidth;
        BrowserFenster_AktuelleHoehe    =window.innerHeight;
        BrowserFenster_AktuelleScrollPositionX =window.pageXOffset;
        BrowserFenster_AktuelleScrollPositionY =window.pageYOffset;
    }
    else
    {
        //IE läuft
        BrowserFenster_AktuelleBreite    =document.body.clientWidth;
        BrowserFenster_AktuelleHoehe    =document.body.clientHeight;
        BrowserFenster_AktuelleScrollPositionX =document.body.scrollLeft;
        BrowserFenster_AktuelleScrollPositionY =document.body.scrollTop;
    }

    // Korrekturpositionen für Grafik ermitteln
    //      Achtung: Die Klammerung ist hier WICHTIG !!!
    GrafikPositionsKorrekturX=      BrowserFenster_AktuelleBreite
        - (GrafikBreiteLautIMGinBODY+GrafikAbstandZurFensterEckeRechtsUntenX)
        + BrowserFenster_AktuelleScrollPositionX;

    GrafikPositionsKorrekturY=      BrowserFenster_AktuelleHoehe
        - (GrafikHoeheLautIMGinBODY+GrafikAbstandZurFensterEckeRechtsUntenY)
        + BrowserFenster_AktuelleScrollPositionY;

    // Grafikposition korrigieren
    // eval () ermittelt String und liefert dessen Inhalt anstelle eval()
    // da der String ein StyleSheed-Kommando ist, wird das somit ausgeführt
    if (BrowserTyp)
    {
        // Netscape läuft
        eval("document.StyleSheedID.left=" + GrafikPositionsKorrekturX);
        eval("document.StyleSheedID.top=" + GrafikPositionsKorrekturY);
    }
    else
    {
        // IE läuft
        eval("StyleSheedID.style.pixelLeft=" + GrafikPositionsKorrekturX);
        eval("StyleSheedID.style.pixelTop=" + GrafikPositionsKorrekturY);
    }

    setTimeout("GrafikPositionKorrigieren()",GrafikPositionsKorrekturGeschwindigkeit);
}

```



```

}
//-->
</SCRIPT>
</HEAD>
<BODY onload="GrafikPositionKorrigieren();">
<DIV ID="StyleSheedID" STYLE="position:absolute; visibility:show; left:0px; top:0px; z-index:2">
  <CENTER>
    <IMG SRC="picture.gif" WIDTH="30" HEIGHT="49" BORDER="0">
  </CENTER>
</DIV>
Zeile<BR>
Zeile<BR>
Zeile<BR>
Zeile<BR>
Zeile<BR>
....
</BODY>
</HTML>

```

Beispiele für Bild verschwinden lassen:

Variante 1: Bild ausserhalb des sichtbaren Bereiches positionieren

```

if (document.layers)
{
    document.layers[DIV_ID].left = (-1 * 40);
    document.layers[DIV_ID].top = (-1 * 20);
}
else
{
    if (document.all)
    {
        document.all.DIV_ID.style.left = (-1 * 40);
        document.all.DIV_ID.style.top = (-1 * 20);
    }
}

<DIV ID="DIV_ID"
STYLE="position:absolute;left=0px;top=0px;"
>
    <IMG NAME="IMG_ID"
        SCR="alt.jpg"
        HEIGHT=20
        WIDTH=40
        STYLE="...."
    >
</DIV>

```

Variante 2: Bild ersetzen durch transparentes 1x1 Pixel

```

document.IMG_ID.src="neu.gif"; // neu.gif ist transparent und 1x1 Pixel

<DIV ID="DIV_ID"
STYLE="position:absolute;left=0px;top=0px;"
>
    <IMG NAME="IMG_ID"
        SCR="alt.jpg"
        HEIGHT=20
        WIDTH=40
        STYLE="...."
    >
</DIV>

```

Vairante 3: Sichtbarkeit verändern per style.visibility

Achtung: style.visibility=hidden lässt den Platz des Bildes im Layout **nicht** bestehen, was dann wichtig ist, wenn Umgebung des Bildes **nicht** per STYLE="position:absolute ..." erzeugt wurde.

Beispiel 1:

```

<HEAD>
<STYLE>
    .vis1 { visibility:visible }
    .vis2 { visibility:hidden }
</STYLE>

```



```

</HEAD>
<BODY>
  <IMG ID="ID_IMG" SRC="test.jpg">
  <P   onmouseover="ID_IMG.className='vis1'"
     onmouseout="ID_IMG.className='vis2'"
  >
    Testtext
  </P>
</BODY>

```

Beispiel 2:

```

<SCRIPT>
  function Verstecken()
  {ID_IMG.style.visibility="hidden"; }

  function Anzeigen()
  {ID_IMG.style.visibility="visible"; }
</SCRIPT>
<IMG  ID="ID_IMG"
  SRC="test.jpeg">
<SPAN onmouseover="Verstecken()"
  onmouseout="Anzeigen()"
  >
  Testtext
</SPAN>

```

Vairante 4: Sichtbarkeit verändern per style.display

Achtung: style.visibility=none lässt den Platz des Bildes im Layout bestehen, was dann wichtig ist, wenn Umgebung des Bildes **nicht** per STYLE="position:absolute ..." erzeugt wurde.

Beispiel 1:

```

<HEAD>
<STYLE>
  .vis1 { display:inline }
  .vis2 { display:none }
</STYLE>
<HEAD>
<BODY>
  <IMG ID="ID_IMG" SRC="test.jpg">
  <P   onmouseover="ID_IMG.className='vis1'"
     onmouseout="ID_IMG.className='vis2'"
  >
    Testtext
  </P>
</BODY>

```

Beispiel 2:

```

<SCRIPT>
  function Verstecken()
  {ID_IMG.style.display="none"; }

  function Anzeigen()
  {ID_IMG.style.display="inline"; }
</SCRIPT>
<IMG  ID="ID_IMG"
  SRC="test.jpeg">
<SPAN onmouseover="Verstecken()"
  onmouseout="Anzeigen()"
  >
  Testtext
</SPAN>

```

#### **Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.align	Ausrichtung
.alt	Alternativer Text als Tooltip
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen



.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.border	Rahmendicke in Pixel
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.complete	Zustand des Ladens des Objektes
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.dynsrc	Adresse von Videoclip oder VRML
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.fileCreateDate	Datum der Dokumenterstellung
.fileModifiedDate	Datum der letzten Dokumentveränderung
.fileSize	Größe des Dokumentes
.fileUpdatedDate	Datum des letzten Datei-Updates
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.galleryImg	Toobar "My Pictures Photo Support image toolbar" ein/ausschalten für aktuelles Image
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.height	Höhe des Objektes in Pixel
.hideFocus	Focussierbarkeit
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMap	server-seitige Image-Map
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.longDesc	Uniform Resource Identifier (URI) zur einer "long description" umwandeln
.loop	Anzahl der Wiederholungen nur Objekt BGSOUND bzw. IMG mit Sound bzw. INPUT-Element mit Sound
.lowsrc	Url des Images in geringerer Auflösung
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nameProp	Dateiname-Teil einer Url ohne Path und ohne Protokoll liefern
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem



	des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag
.outerText	nur nach kompletten Einlesen des Dokumentes nutzbar Referenz auf den gesamten Plain-Text im Objekt
.ownerDocument	nur nach kompletten einlesen des Dokumentes nutzbar Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.protocol	Protokoll-Teil einer Url inklusive http und ftp liefern
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
.src	Url der Daten z.B. vom Image in normaler Auflösung
.start	Start eines Videoclips oder VRML-Datei
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.useMap	Url oder Anker für client-seitige Image-Map
.vspace	vertikaler Abstand in Pixel zum Elternobjekt
.width	Breite des Objektes in Pixel
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern



	DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernenbar
.click()	DOM wird geändert simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
.detachEvent()	DOM nicht geändert Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht
.getAttribute()	DOM nicht geändert Wert eines per HTML erzeugten Attributes liefern
.getAttributeNode()	DOM nicht geändert Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
.hasChildNodes()	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh) prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten



	(Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu genießen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.23. document.images Collection des Internet Explorer

Feld der Zeiger aller img Objekte

Elementefolge laut HTML-Kodung

Achtung: Per new Image() erzeugte Bilder sind **nicht** Bestandteil der Collection !!



Diese Collection umfasst **nicht** das Objekt input.image, da für dieses Objekt die childrenCollection verwendet werden muss !

**Syntax:**

```
[ var ZeigerAufFeld = ] document.images
[ var ZeigerAufFeldElement = ] document.images [Index [, SubIndex] ]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

**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

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.24. input Objekt und seine Varianten**

Varianten des Input Controls (Eingabe-Steuerelementes) z.B. im Formular

festlegbar per TYPE-Attribut:

- button
- checkbox
- file
- hidden
- image
- password
- radio
- reset
- submit
- text

wobei der Wert des TYPE-Attributes **wahlweise**  
in " " bzw. ' '  
oder ohne " " bzw. ' '  
kodiert werden kann.

Das input Objekt ist Prototyp für das Objekt document.form.input nutzt weitere Objekte wie button und img.

Sämtliche Input-Varianten besitzen das ID-Attribut, über das der Script-Zugriff möglich ist.

Nachfolgende Syntax-Beschreibungen zeigen die HTML-Verwendung der Input-Varianten, wobei aus Gründen der Übersichtlichkeit auf die Kodierung des ID-Attributes verzichtet wurde. Eigenschaften und Methoden **nur** beim Internet Explorers.

**Erzeugung in HTML:**

Beispiel Alle Elemente im Formular müssen **NAME**-Attribut erhalten, wenn die Daten der Elemente gesendet werden sollen !

```
<FORM ACTION="http://intranet/test" METHOD=POST>
  <INPUT TYPE="TEXT" NAME="CONTROL1" VALUE="Ihr Name">
  <BR>
  <P>Password</P>
  <INPUT TYPE="PASSWORD" NAME="CONTROL2">
  <P>Farbe-Auswahl</P>
  <BR>
  <INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="0" CHECKED>Rot
  <INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="1">Gelb
  <INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="2">Blau
  <P>Ihr Kommentar</P>
  <BR>
  <INPUT TYPE="TEXT" NAME="CONTROL4" SIZE="20,5" MAXLENGTH="250">
  <BR>
  <INPUT TYPE="CHECKBOX" NAME="CONTROL5" CHECKED>Senden
```



```

<BR>
<INPUT TYPE="SUBMIT" VALUE="OK">
<INPUT TYPE="RESET" VALUE="Reset">
</FORM>

```

**Eigenschaften beim Internet Explorer:**

Diese Eigenschaften sind nicht für alle INPUT-Varianten sinnvoll anwendbar  
Bsp.: Loop für ein Image-Map

.accept	Liste von Multipurpose Internet Mail Extensions (MIME)-Typen
.align	Ausrichtung
.alt	Alternativer Text als Tooltip
.complete	Zustand des Ladens des Objektes
.dynsrc	Adresse von Videoclip oder VRML
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.lowsrc	Url des Images in geringerer Auflösung
.start	Start eines Videoclips oder VRML-Datei
.useMap	Url oder Anker für client-seitige Image-Map
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.vspace	vertikaler Abstand in Pixel zum Elternobjekt

**Methoden beim Internet Explorer:**

keine

**4.3.2.2.4.3.24.1. input button Objekt**

Taste-Control-Element (Button-Control-Element)  
Hinweis: Es gibt noch das button Objekt (siehe dort)

**Erzeugung in HTML:**

```
<INPUT TYPE=button ....> auch als String "button" kodierbar
```

Beispiel

```

<INPUT TYPE=button
VALUE="betaetigen!"
NAME="button1"
onClick="alert('Das input button Objekt in Aktion!)"
>

```

**Eigenschaften beim Internet Explorer:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataFormatAs	Datenquelle-Anzeigeart
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2



.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master bei der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes



	<p>Anspringen verbunden mit Focus erhalten  --&gt; Ereignisse werden ausgelöst !!  unter IE 5.x  onblur, onfocus  ab IE 5.x  onblur, onfocus, onkeydown, onkeypress, onkeyup</p>
	<p>Anspringen default per TAB-Taste  für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT,  SELECT, TEXTAREA</p>
	<p>Anspringen default nicht per TAB-Taste  für APPLET, DIV, FRAMESET, SPAN, TABLE, TD</p>
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner



	zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern



- Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
- DOM wird geändert
- .removeExpression() Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form `objekt.style.eigenschaft` dient.  
Ausdruck muss mit der Methode `.setExpression()` gesetzt worden sein
- DOM wird nicht geändert
- .removeNode() Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde
- DOM wird geändert
- .replaceAdjacentText() Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
- DOM wird nicht geändert
- .replaceChild() Kind-Objekt ersetzen durch ein Objekt  
ersetzende Objekt muss per Methode `.createElement()` erzeugt worden sein  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde
- DOM wird geändert
- .replaceNode() Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern  
sichtbar erst mit parsen des Endetags
- DOM wird geändert
- .scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird  
Objekt muss an sich schon renderbar sein
- .select() Bereich des Input-Objektes im Formular markieren  
setzt **nicht** den Focus (dafür Methode `.focus()` verwenden)
- .setActive() Objekt für die Eventdurchreichung aktivieren  
aber ohne es zu fokussieren  
und ohne es scrollbar zu machen
- .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
- .setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert
- .setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : `onmousedown`, `onmouseup`, `onmousemove`, `onclick`, `ondblclick`,  
`onmouseover` und `onmouseout`.
- ab IE 5.5
- Hinweis: ausschalten per Methode `.releaseCapture()`
- .setExpression() Wert definieren, der als Ausdruck für die Methode `.getExpression()` zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form `objekt.style.eigenschaft` dient  
Ausdruck nur als Script kodierbar
- DOM wird nicht geändert
- .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.24.2. input checkbox Objekt**

Checkbox-Control

Standardwert ist "on"

Style-Wert zu height und width ab IE 5.x (innere Höhe bzw. Breite): wenn height bzw. width

>= 20 Pixel

Dicke Padding um die Checkbox	4 Pixel
innere Höhe bzw innere Breite	8 Pixel

< 20 Pixel **und** > 13 Pixel

Dicke Padding um die Checkbox	(height -13) / 2 Pixel bzw. (width -13) / 2 Pixel
innere Höhe bzw innere Breite	unverändert übernommen

< 13 Pixel

Dicke Padding um die Checkbox	0 Pixel
innere Höhe bzw innere Breite	unverändert übernommen

**Erzeugung in HTML:**

<INPUT TYPE=checkbox ....> auch als String "checkbox" kodierbar

Beispiel

```

<SCRIPT>
function Anzeige1()
{ID_Span.insertAdjacentHTML("Checkbox 1 aktiv");}

function Anzeige2()
{ID_Span.insertAdjacentHTML("Checkbox 2 aktiv");}
</SCRIPT>
<INPUT TYPE=checkbox
NAME="checkbox1"
CHECKED
onclick="Anzeige1()"

```



```

>Aktion 1
<INPUT TYPE=checkbox
      NAME="checkbox2"
      onclick="Anzeige2()"
>Aktion 2
<SPAN ID="ID_Span">
  Test-Text
</SPAN>

```

**Eigenschaften beim Internet Explorer:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.checked	Selektionsstatus des Checkbox-Control bzw. Radio Button-Control bezüglich Selektion durch User
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultChecked	Selektionsstatus des Checkbox-Control bzw. Radio Button-Control bzw. Radio Button-Control bezüglich Standard-Selektion
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.indeterminate	Grauzustand (Dimmed) <b>und</b> Selektiertheit des Checkbox-Control
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nextSibling	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)



.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
.status	Selektionsstatus eines Control-Elementes Control-Element: kann durch User interaktiv verändert werden
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes



	Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression()



	zu definieren
	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
	DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich
	DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich
	HTML- und Script-Code müssen syntaktisch korrekt sein
	wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt
	eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist
	bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
	DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes
	DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein
	Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten
	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen
	Attribute sind: HTML
	Events
	Styles
	ab IE 5.01 auch ID, NAME
	Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
	DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur
	Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt
	Maus-Events sind : onmousedown, onmouseup, onclick, ondblclick, onmouseover und onmouseout.
	Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes
	Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!
	per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
	DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
	DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
	DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
	Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
	DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient.
	Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
	DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern
	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
	DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt
	ersetzende Objekt muss per Methode .createElement() erzeugt worden sein
	Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern
	sichtbar erst mit parsen des Endtags
	DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird
	Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren
	setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchführung aktivieren
	aber ohne es zu fokussieren



und ohne es scrollbar zu machen

.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

.setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert

.setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.  
ab IE 5.5  
Hinweis: ausschalten per Methode .releaseCapture()

.setExpression() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
Eigenschaft als Objektreferenz der Form  
objekt.style.eigenschaft.  
dient  
Ausdruck nur als Script kodierbar  
DOM wird nicht geändert

.swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.24.3. input file Objekt**

File Upload-Control mit Dialogbox  
ab IE 4.x

folgende Voraussetzungen **müssen** erfüllt werden:

- Control muss in einem Formular liegen
- NAME-Attribut muss kodiert sein
- Formular mit Attribut METHOD auf "post" gesetzt  
ENCTYPE auf "multipart/form-data"

Server muss den Enctype "multipart/form-data" verarbeiten können (eventuell CGI- oder ASP-Script dort nötig)

**Erzeugung in HTML:**

Beispiel mit Serverscript per ASP:

**HTML-Dokument:**

```
<FORM NAME="Formular"
ACTION="script.asp"
ENCTYPE="multipart/form-data"
METHOD="post"
>
<INPUT TYPE="file" NAME="testdatei">
<INPUT TYPE="submit" VALUE="Upload der Datei">
</FORM>
```

**script.asp:**

```
<%@ LANGUAGE = JScript %>
<%
Response.buffer=true;
%>
<HTML>
<BODY>
<H1>Upload Status</H1>
<P>
Zielort: <% Response.Write(Request.Form("TargetURL")) %>
</P>
<%
Response.write("<P>Dateiname: " + Request.Form("FileName") + "</P>");
Response.write("<P>Dateigroesse: " + Request.Form("FileSize") + "</P>");
Response.write("<P>Dateipfad: " + Request.Form("FilePath") +
"</P>");
%>
</BODY>
</HTML>
```

**Eigenschaften beim Internet Explorer:**

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

ATOMICSELECTION Selektierbarkeit des Objektes einstellen

.begin Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction  
siehe Objekt currTimeState und Behavior .style.time2

.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann

.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf



.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
.isDisabled	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes



	nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberem sichtbarem Rand des Umgebungsobjektes zum oberen Rand des Objektes
	nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	Wird durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht!
.attachEvent()	Einschalten des Registrierens eines Events durch Eventhandler Hinweis: Abschalten mit Methode .detachEvent() Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider NICHT verkettet sondern in <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute



	Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME



	Achtung: Diese Methode ist mir Vorsicht zu genießen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
.removeBehavior()	DOM wird geändert per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
.removeChild()	DOM wird geändert Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
.removeNode()	DOM wird nicht geändert Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.replaceChild()	DOM wird nicht geändert Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceNode()	DOM wird geändert Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
.scrollIntoView()	DOM wird geändert Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
.setAttributeNode()	DOM wird nur bei Erzeugung geändert Attribut einem Knoten zuweisen und Referenz liefern
.setCapture()	DOM wird geändert Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.setExpression()	ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar
.swapNode()	DOM wird nicht geändert Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.24.4. **input hidden Objekt**

ist **kein** Control-Element

Ersatz für Cookies: Übertragung von für den User nicht sichtbarer Daten zum Server  
gesendet wird der Wert

zu sendender Wert durch Programmierer festlegbar und nicht durch User beeinflussbar

**Achtung: Dieses Objekt kann missbräuchlich benutzt werden !**

#### Erzeugung in HTML:

<INPUT TYPE=hidden ...> auch als "hidden" kodierbar



**Eigenschaften beim Internet Explorer:**

ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeSibling	
.nodeName	
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)



.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar

**Methoden beim Internet Explorer:**

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert



.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.24.5. input image Objekt

Image-Control für Senden eines Formulars durch Klick auf das Bild (interaktives Bild **nur** zum Zweck des Submit) gesendet werden **nur** die Koordinaten der linken oberen Ecke (in Pixel) gesendet

wobei es sich darin um interne Eigenschaften handelt, die nicht manipulierbar sind

.x für Spalte  
.y für Zeile

ab IE 5.0 werden folgende Image-Medien unterstützt:

.avi	Audio-Visual Interleaved (AVI)
.bmp	Windows Bitmap (BMP)
.emf	Windows Enhanced Metafile (EMF)
.gif	Graphics Interchange Format (GIF)
.jpg, .jpeg	Joint Photographic Experts Group (JPEG)
.mov	Apple QuickTime Movie (MOV)
.mpg, .mpeg	Motion Picture Experts Group (MPEG)



.png	Portable Network Graphics (PNG)
.wmf	Windows Metafile (WMF)
.xbm	X Bitmap (XBM)

basiert auf dem img Objekt

### Erzeugung in HTML:

<INPUT TYPE=image .....> auch als "image" kodierbar

### Eigenschaften beim Internet Explorer:

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.align	Ausrichtung
.alt	Alternativer Text als Tooltip
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.complete	Zustand des Ladens des Objektes
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.dynsrc	Adresse von Videoclip oder VRML
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.loop	Anzahl der Wiederholungen nur Objekt BGSOUND bzw. IMG mit Sound bzw. INPUT-Element mit Sound
.lowsrc	Url des Images in geringerer Auflösung
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker



.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
.src	Url der Daten z.B. vom Image in normaler Auflösung
.start	Start eines Videoclips oder VRML-Datei
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde



UNSELECTABLE	kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
.value	Selektionsfähigkeit eines Objektes Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichnung zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle



.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu genießen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren



.setActive()	setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden) Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.24.6. document.images Collection des Internet Explorer

Feld der Zeiger aller img Objekte

Elementefolge laut HTML-Kodung

Achtung: Per new Image() erzeugte Bilder sind **nicht** Bestandteil der Collection !!

Diese Collection umfasst **nicht** das Objekt input.image, da für dieses Objekt die childrenCollection verwendet werden muss !

##### Syntax:

```
[ var ZeigerAufFeld = ] document.images
[ var ZeigerAufFeldElement = ] document.images [Index [, SubIndex] ]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

##### 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
.urns()	Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

#### 4.3.2.2.4.3.24.7. input password Objekt

Text-Control aus einer Zeile mit Dummy-Anzeige während der Eingabe der Zeichen durch User

##### Erzeugung in HTML:

<INPUT TYPE=password ...> auch als "password" kodierbar

Beispiel <SCRIPT>

```
function Pruefen()
{
    if (ID_InputButton.value == "Otto")
    {
        if (ID_InputPassword.value == "Waalkes")
        {alert("Password korrekt!");}
    }
}
</SCRIPT>
```

```
Username <INPUT TYPE=button ID="ID_InputButton" onclick="Pruefen()">
Password <INPUT TYPE=password ID="ID_InputPassword" onclick=Pruefen()">
```

##### Eigenschaften beim Internet Explorer:

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste



	bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst
	vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.autocomplete	Status des Autovervollständigung zum Formular
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.maxLength	Maximale Anzahl der durch User eingebaren Zeichen in einem Text-Control
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x



.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readOnly	Editierbarkeit eines Text-Controls Editierung bedeutet Focuserhalt !
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup  Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.vcard_name	Werte für vCard für Autocomplete (Autovervollständigung) bei Formular per Text-Control
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert



	Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM



	nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,



onmouseover und onmouseout.  
 ab IE 5.5  
 Hinweis: ausschalten per Methode .releaseCapture()  
 Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
 Eigenschaft als Objektreferenz der Form  
 objekt.style.eigenschaft.  
 dient  
 Ausdruck nur als Script kodierbar  
 DOM wird nicht geändert  
 .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
 nur sichtbar wenn Endetag geparkt  
 DOM wird geändert

**4.3.2.2.4.3.24.8. input radio Objekt**

Radio-Button-Control z.B. für Formular  
ermöglicht Auswahl (Selektion) genau eines Eintrages aus Menge von Einträgen:

jeder Eintrag ist Radio-Button-Control  
kann nur selektiert werden wenn NAME-Attribut kodiert wurde  
zu jedem Zeitpunkt kann nur ein Eintrag ausgewählt sein

Menge von Einträgen ist Gruppe:  
alle Elemente der Gruppe haben identischen Wert im NAME-Attribut  
mehrere Gruppen zulässig mit je eigenem Namen  
Standard-Element markierbar per Eigenschaft .checked (siehe dort)

senden beim Formular: nur das ausgewählte (selektierte) Element der Gruppe:  
gesendet wird der Wert und NAME-Attribut-Wert

**Erzeugung in HTML:**

Beispiel

```

<SCRIPT>
function Anzeigen()
{
    if (radio[0].checked)
    {alert("Eintrag 1");}
    else
    {
        if (radio[1].checked)
        {alert("Eintrag 2");}
        else
        {alert("Eintrag 3");}
    }
}
</SCRIPT>
Gruppe <B>GemeinsamerName<B>
<BR>
<INPUT TYPE=radio NAME="GemeinsamerName" CHECKED>Eintrag 1
<INPUT TYPE=radio NAME="GemeinsamerName">Eintrag 2
<INPUT TYPE=radio NAME="GemeinsamerName">Eintrag 3
<INPUT TYPE=button
VALUE="Anzeige des aktuell selektierten Eintrages"
onClick="Anzeigen()"
>

```

**Eigenschaften beim Internet Explorer:**

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste  
 bei Ausführung der Tastenkombination wird  
 das Sprungziel wird focussiert  
 die Sprungquelle defocussiert  
 das Focus-Ereignis ausgelöst  
 vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt  
 ATOMICSELECTION Selektierbarkeit des Objektes einstellen  
 .begin Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction  
 siehe Objekt currTimeState und Behavior .style.time2  
 .canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann  
 .canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf  
 .checked Selektionsstatus des Checkbox-Control bzw. Radio Button-Control  
 bezüglich Selektion durch User  
 .className Klassenreferenz, Klassenname  
 .clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt  
 ohne Rahmen  
 ohne Scrollbalken  
 .clientLeft Abstand in Pixel zum linken Rand des Fensters  
 .clientTop Abstand in Pixel zum oberen Rand des Fensters



<code>.clientWidth</code>	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
<code>.contentEditable</code>	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
<code>.dataFld</code>	Datenquelle-Name vergeben (ID)
<code>.dataSrc</code>	Datenquelle als Anker festlegen
<code>.defaultChecked</code>	Selektionsstatus des Checkbox-Control bzw. Radio Button-Control bzw. Radio Button-Control bezüglich Standard-Selektion
<code>.defaultValue</code>	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
<code>.dir</code>	Umflussrichtung
<code>.disabled</code>	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
<code>.end</code>	Objektaktivitäten laut Eigenschaft <code>.timeAction</code> beenden ab IE 6.x alternativ: Eigenschaft <code>.dur</code> siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
<code>.firstChild</code>	Zeiger auf das ERSTE Kind laut <code>childNodes</code> -Collection eines Objektes
<code>.form</code>	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente <code>fieldSet</code> , <code>label</code> , <code>legend</code>
<code>.hasMedia</code>	Objekt ist HTML-Media-Objekt siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
<code>.hideFocus</code>	Focussierbarkeit
<code>.id</code>	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft <code>.uniqueID</code> ermittelt und anstelle der Eigenschaft <code>.id</code> verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft <code>.uniqueID</code> kennen). Zeiger aus ID bilden <code>var Zeiger = eval(object.id);</code>
<code>.isContentEditable</code>	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
<code>.isDisabled</code>	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
<code>.isMultiLine</code>	Mehrzeiligkeit des Objektinhaltes
<code>.isTextEdit</code>	Erzeugbarkeit eines Textbereiches
<code>.lang</code>	Sprache für Anzeige von Sonderzeichen etc.
<code>.language</code>	Sprache für Script festlegen
<code>.lastChild</code>	Zeiger auf das LETZTE Kind laut <code>childNodes</code> collection eines Objektes
<code>.name</code>	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode <code>.createElement()</code> erzeugt worden sein
<code>.nextSibling</code>	Zeiger auf das NACHFOLGENDE Kind laut <code>childNodes</code> collection eines Objektes
<code>.nodeName</code>	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
<code>.nodeType</code>	Knotentyp laut <code>attributes</code> Collection
<code>.nodeValue</code>	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
<code>.offsetHeight</code>	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetLeft</code>	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetParent</code>	Referenz der Eltern für Nutzung von <code>.offsetHeight</code> , <code>.offsetLeft</code> , <code>.offsetTop</code> und <code>.offsetWidth</code>
<code>.offsetTop</code>	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetWidth</code>	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.onOffBehavior</code>	deprecated ab IE 5.x Unterstützung von <code>DirectAnimation</code> z.B. für 2D, 3D, Sound
<code>.outerHTML</code>	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit <code>parse</code> des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
<code>.outerText</code>	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
<code>.ownerDocument</code>	Referenz auf das <code>document</code> Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
<code>.parentElement</code>	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
<code>.parentNode</code>	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
<code>.parentTextEdit</code>	Textbereich des Elternobjektes referenzieren
<code>.previousSibling</code>	Referenz auf das Vorgängerkind
<code>.readyState</code>	aktueller Status des Objektes beim Füllen des Objektes mit Daten
<code>.recordNumber</code>	Datenquelle-Satznummer eines Datenfeldes
<code>.scopeName</code>	Namensraum laut XMLNS-Attribut
<code>.scrollHeight</code>	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes



.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
.status	Selektionsstatus eines Control-Elementes Control-Element: kann durch User interaktiv verändert werden
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup  Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichnung zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen



	Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5
.setExpression()	Hinweis: ausschalten per Methode .releaseCapture() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

**4.3.2.2.4.3.24.9. input reset Objekt**

Reset-Button für Formular-Reset: Standardgemäß alle Formular-Elemente auf Initialwert setzen.  
VALUE-Attribut für Titel (Label) des Reset-Button verwenden  
Wert wird nicht gesendet



**Erzeugung in HTML:**

<INPUT TYPE=reset .....> auch als "reset" kodierbar

**Eigenschaften beim Internet Explorer:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nextSibling	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem



	des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag
.outerText	nur nach kompletten Einlesen des Dokumentes nutzbar Referenz auf den gesamten Plain-Text im Objekt
.ownerDocument	nur nach kompletten einlesen des Dokumentes nutzbar Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
	Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA
	Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag



	(falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert



.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmouseover, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmouseover, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5



Hinweis: ausschalten per Methode `.releaseCapture()`  
`.setExpression()` Wert definieren, der als Ausdruck für die Methode `.getExpression()` zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form `objekt.style.eigenschaft`.  
 dient  
 Ausdruck nur als Script kodierbar  
 DOM wird nicht geändert  
`.swapNode()` Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
 nur sichtbar wenn Endetag geparkt  
 DOM wird geändert

#### 4.3.2.2.4.3.24.10. *input submit Objekt*

Submit-Button für Formular  
 Label im VALUE-Attribut kodieren (Standardtext ist "Submit Query")  
 Senden nur möglich, wenn NAME-Attribut kodiert wurde:  
 gesendet wird Wert von NAME- und VALUE-Attribut

#### Erzeugung in HTML:

`<INPUT TYPE=submit .....>` auch als "submit" kodierbar

#### Eigenschaften beim Internet Explorer:

`.accessKey` Tastaturzugriff auf ein Objekt per Alt + Taste  
 bei Ausführung der Tastenkombination wird  
 das Sprungziel wird focussiert  
 die Sprungquelle defocussiert  
 das Focus-Ereignis ausgelöst  
 vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt  
 ATOMICSELECTION  
`.begin` Selektierbarkeit des Objektes einstellen  
 Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft `.timeAction`  
 siehe Objekt `currTimeState` und Behavior `.style.time2`  
`.canHaveChildren` prüfen ob Kind möglich ist, also ob Objekt Parent sein kann  
`.canHaveHTML` prüfen ob Objekt HTML-Tags enthalten darf  
`.className` Klassenreferenz, Klassenname  
`.clientHeight` Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt  
 ohne Rahmen  
 ohne Scrollbalken  
`.clientLeft` Abstand in Pixel zum linken Rand des Fensters  
`.clientTop` Abstand in Pixel zum oberen Rand des Fensters  
`.clientWidth` Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt  
 ohne Rahmen  
 ohne Scrollbalken  
`.contentEditable` Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat)  
 Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.  
`.defaultValue` Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars  
`.dir` Umflussrichtung  
`.disabled` Interaktionsfähigkeit  
 nur wenn sichtbar so User-Interaktion möglich  
`.end` Objektaktivitäten laut Eigenschaft `.timeAction` beenden  
 ab IE 6.x  
 alternativ: Eigenschaft `.dur`  
 siehe Objekt `currTimeState` und Behavior `.style.time2`  
`.firstChild` Zeiger auf das ERSTE Kind laut `childNodes`-Collection eines Objektes  
`.form` Zeiger auf das Formular (Formular als Container)  
 ab IE 6.x für Elemente `fieldSet`, `label`, `legend`  
`.hasMedia` Objekt ist HTML-Media-Objekt  
 siehe Objekt `currTimeState` und Behavior `.style.time2`  
`.hideFocus` Focussierbarkeit  
`.id` Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)  
 Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft `.uniqueID` ermittelt  
 und anstelle der Eigenschaft `.id` verwendet werden kann (falls Browser und  
 betroffenes Objekt die Eigenschaft `.uniqueID` kennen).  
 Zeiger aus ID bilden `var Zeiger = eval(object.id);`  
`.isContentEditable` Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)  
 Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.  
`.isDisabled` Interaktionsfähigkeit  
 nur wenn sichtbar so User-Interaktion möglich  
`.isMultiLine` Mehrzeiligkeit des Objektinhaltes  
`.isTextEdit` Erzeugbarkeit eines Textbereiches  
`.lang` Sprache für Anzeige von Sonderzeichen etc.  
`.language` Sprache für Script festlegen  
`.lastChild` Zeiger auf das LETZTE Kind laut `childNodes` collection eines Objektes  
`.name` Name des Objektes (nicht ID !!!)  
 muss beim Formular für alle zu sendenden Felder kodiert sein !!  
 Element darf nicht per Methode `.createElement()` erzeugt worden sein  
`.nextSibling` Zeiger auf das NACHFOLGENDE Kind laut `childNodes` collection eines Objektes



.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parse des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-geniertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde



	kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0



	pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu genießen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein



.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode .focus() verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.24.11. input text Objekt

Einzeilige Textbox, in der User eingeben kann  
Breite der Textbox in Zeichen im SIZE-Attribut kodieren.  
Maximale Anzahl der einzeiligen Zeichen in Attribut MAXLENGTHs kodieren.  
Wenn SIZE < MAXLENGTH so wird automatisch horizontaler Scrollbalken erzeugt.

#### Erzeugung in HTML:

Beispiel

```
<SCRIPT>
function Anzeige()
{alert("Es wurde eingegeben" + textbox.value); }
</SCRIPT>

<INPUT TYPE=text VALUE="" NAME="textbox" SIZE=15 onclick="Anzeige()">
```

#### Eigenschaften beim Internet Explorer:

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.autocomplete	Status des Autovervollständigung zum Formular
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Standardwert laut Objektinitialisierung für Element innerhalb eines Formulars
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes



.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.maxLength	Maximale Anzahl der durch User eingebbaren Zeichen in einem Text-Control
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readOnly	Editierbarkeit eines Text-Controls Editierung bedeutet Focuserhalt ! aktueller Status des Objektes beim Füllen des Objektes mit Daten
.readyState	
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.size	Dimension eines Input-Objektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf !



	siehe Objekt currTimeState und Behavior .style.time2
	siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
	Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.type	Typ (Variante) des Input-Objektes (Art der Control-Elemente vom gemeinsamen Typ INPUT)
.uniqueID	durch den Browser automatisch-geniertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
.vcard_name	Werte für vCard für Autocomplete (Autovervollständigung) bei Formular per Text-Control
.width	Breite des Objektes in Pixel
<b>Methoden beim Internet Explorer:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben



	Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.createTextRange()	Textbereich erzeugen
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdrucks berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst



.removeAttributeNode()	DOM wird geändert entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
.removeBehavior()	DOM wird geändert per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
.removeChild()	DOM wird geändert Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient. Ausdruck muss mit der Methode <code>.setExpression()</code> gesetzt worden sein
.removeNode()	DOM wird nicht geändert Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.replaceChild()	DOM wird nicht geändert Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode <code>.createElement()</code> erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceNode()	DOM wird geändert Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
.scrollIntoView()	DOM wird geändert Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Bereich des Input-Objektes im Formular markieren setzt <b>nicht</b> den Focus (dafür Methode <code>.focus()</code> verwenden)
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
.setAttributeNode()	DOM wird nur bei Erzeugung geändert Attribut einem Knoten zuweisen und Referenz liefern
.setCapture()	DOM wird geändert Maus-Überwachung einschalten für ein Objekt Maus-Events sind : <code>onmousedown</code> , <code>onmouseup</code> , <code>onmousemove</code> , <code>onclick</code> , <code>ondblclick</code> , <code>onmouseover</code> und <code>onmouseout</code> .
.setExpression()	ab IE 5.5 Hinweis: ausschalten per Methode <code>.releaseCapture()</code> Wert definieren, der als Ausdruck für die Methode <code>.getExpression()</code> zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient Ausdruck nur als Script kodierbar
.swapNode()	DOM wird nicht geändert Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.25. label Objekt

Aufschrift/Beschriftung (Etikett) eines Elementes der Webseite. Nicht verwechseln mit Titel, da Titel in HTML anders verwendet wird.

Label sind nicht verschachtelbar.

Input-Elemente (Objekt `input xxxx`) haben z.T. eigenen Labelkodierung im Attribut `VALUE` und benötigen das Objekt `label` nicht.

#### Erzeugung in HTML:

##### **Kodierung der Beschriftung:**

Dieses Objekt verlangt eine Kodierung des Attributes `ID` im Label **und** im zu beschriftenden Element. Das `ID` ist das Bindeglied zur Realisierung einer Beschriftung. Es ist nicht das `ID`-Attribut !!!

```
Bsp.: <LABEL FOR="ID_Label">Beschriftung</LABEL>
      <INPUT TYPE="text"
            ID="ID_Label"
            VALUE="Textbox mit Beschriftung"
            SIZE="20"
      >
```

##### **Elementzugriff per Tastaturkürzel:**

Label wird vorrangig zur Realisierung des Zugriffes auf das Element per Tastaturkürzel verwendet:  
im Label das Attribut `ACCESSKEY` kodieren

```
Bsp.: <LABEL FOR="ID_Label" ACCESSKEY="1">
```



Zugriffserklärung z.B. per <SPAN> mit Unterstreichung eines Buchstabens als Tastaturkürzel  
per STYLE="text-decoration: underline"

Bsp.: <SPAN STYLE="text-decoration:underline;">1</SPAN>: Press Alt+1 für Anwahl des Elementes

Beispiel

```
<LABEL FOR="ID_Label" ACCESSKEY="1">
  <SPAN STYLE="text-decoration:underline;">1</SPAN>: Press Alt+1 für Anwahl der Textbox
</LABEL>
<INPUT TYPE="text" ID="ID_Label" NAME="TXT1" VALUE="Textbox" SIZE="20" TABINDEX="1">
```

NAME-Attribut nur nötig bei Formular  
<SPAN ....> dient nur der Erklärung des Tastaturkürzel und der Unterstreichung des Kürzelzeichens

Hinweis: Im Formular müssen alle Elemente - wie das Formular selbst - das Attribut NAME erhalten.  
Mausklick auf Label erzeugt das Event onclick, bewirkt aber keinen Aufruf per Tastaturkürzel (es sei denn, es wird manuell als Ersatz von ACCESSKEY programmiert).

**LABEL Objekt des Internet Explorer:**

ab IE 4.x

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataFormatAs	Datenquelle-Anzeigeart
.dataSrc	Datenquelle als Anker festlegen
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hideFocus	Focussierbarkeit
.htmlFor	ID des Label <b>nur für die Realisierung der Beschriftung</b> ist nicht das ID laut ID-Attribut per label Objekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit



	nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objekthinhalte
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x
.outerHTML	Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes

**Methoden:**

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig).
----------------	--



.appendChild()	<p>ab IE 5.x bis unter IE 5.5 Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde</p>
.applyElement()	<p>Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !</p>
.attachEvent()	<p>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 <b>Zufallsfolge</b>, es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)</p>
.blur()	<p>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</p>
.clearAttributes()	<p>alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert</p>
.click()	<p>simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus</p>
.cloneNode()	<p>Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)</p>
.componentFromPoint()	<p>Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.</p>
.contains()	<p>prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert</p>
.detachEvent()	<p>Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)</p>
.dragDrop()	<p>prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element</p>
.fireEvent()	<p>ein Event auslösen</p>
.focus()	<p>Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen</p>
.getAdjacentText()	<p>Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert</p>
.getAttribute()	<p>Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert</p>
.getAttributeNode()	<p>Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert</p>
.getBoundingClientRect()	<p>Referenz auf TextRectangle-Objekt im Element holen</p>
.getClientRects()	<p>Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle</p>
.getElementsByTagName()	<p>Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementByName()</p>



.getExpression()	<p>DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren</p> <p>DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)</p>
.hasChildNodes()	<p>prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt</p> <p>DOM nicht geändert</p>
.insertAdjacentElement()	<p>Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich</p> <p>DOM wird geändert</p>
.insertAdjacentHTML()	<p>HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: &lt;SCRIPT DEFER .....&gt; muss kodiert werden</p> <p>DOM wird geändert</p>
.insertAdjacentText()	<p>Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes</p> <p>DOM wird geändert</p>
.insertBefore()	<p>Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p> <p>DOM wird geändert</p>
.mergeAttributes()	<p>alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME</p> <p>Achtung: Diese Methode ist mir Vorsicht zu geniessen !!</p> <p>DOM wird geändert</p>
.normalize()	<p>Normalisierung des DOM zur Erreichung einer konsistenten Struktur</p> <p>Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen</p>
.releaseCapture()	<p>Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.</p> <p>Hinweis: einschalten per Methode .setCapture()</p>
.removeAttribute()	<p>entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst</p> <p>DOM wird geändert</p>
.removeAttributeNode()	<p>entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern</p> <p>DOM wird geändert</p>
.removeBehavior()	<p>per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)</p> <p>DOM wird geändert</p>
.removeChild()	<p>Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde</p> <p>DOM wird geändert</p>
.removeExpression()	<p>Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein</p> <p>DOM wird nicht geändert</p>
.removeNode()	<p>Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p> <p>DOM wird geändert</p>
.replaceAdjacentText()	<p>Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern</p> <p>DOM wird nicht geändert</p>
.replaceChild()	<p>Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde</p> <p>DOM wird geändert</p>
.replaceNode()	<p>Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags</p> <p>DOM wird geändert</p>
.scrollIntoView()	<p>Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein</p>



.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.  ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft.  dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.26. link Objekt

siehe auch Objektes a

##### Erzeugung unter HTML:

```
<LINK
    HREF="url" // protocol://[host_name[:port]]/dateiname
                // protocol://[host_name[:port]]/dateiname#hashtext
                // protocol://[host_name[:port]]/dateiname?serachtext
                // Bsp: http://www.test.com:8888/test.html
    NAME="anker_name_ohne_#" // ist nicht der logische link_name !!
                                // Netscape + Internet Explorer
oder ID="anker_name_ohne_#" // ist der logische link_name !!
                                // nur Internet Explorer ab 4.x
    TARGET="logischer_window_name"
    onClick="eventhandler1"
    onDbClick="eventhandler2"
    onMouseOut="eventhandler3"
    onMouseOver="eventhandler4"
    onMouseDown="eventhandler5"
    onMouseUp="eventhandler6"
>
link_text_immer_schreiben
</LINK>
```

##### Zugriff:

```
document.links[index]

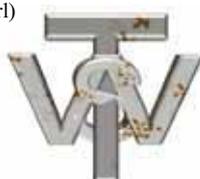
index: ab 0
        Nummer des Links im Dokument
Anzahl aller Links lautet document.links.length
```

Beispiel für globalen Style per HEAD:

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

##### Eigenschaften (ausgewählte):

.className	Klassenreferenz
.hash	entspricht #hashtext zum Anspringen eines Ankers hier den Ankernamen mit vorgesetztem # ablegen
.host	entspricht hostname:port
.hostname	entspricht nur hostname
.href	gesamter Url (kompletter Url)



zum Anspringen eines Ankers  
hier den Ankernamen ohne vorgesetztem # ablegen

.id entspricht ID  
nur IE ab 4.x

.innerText ist der link\_text  
nur IE ab 4.x  
lesen und schreiben

.name entspricht NAME

.offsetHeight Pixelhöhe  
nur IE ab4.x

.offsetLeft Pixelpos bezüglich linken Rand des HTML-Dokumentes  
nur IE ab 4.x

.offsetTop Pixelpos bezüglich oberen Rand des HTML-Dokumentes  
nur IE ab 4.x

.offsetWidth Pixelbreite  
nur IE ab4.x

.offsetLeft Pixelpos bezüglich linken Rand des HTML-Dokumentes

.pathname entspricht aus url /dateiname  
bzw. /dateiname#hashtext  
bzw. /dateiname?searchtext  
lesen und schreiben

.port entspricht port; lesen und schreiben

.protocol entspricht Protokoll mit Doppelpunkt z.B. "http:"  
lesen und schreiben

.search entspricht ?searchtext  
lesen und schreiben

.style Zeiger auf das style-Objekt  
nur IE ab 4.x

.tagName enthält "A"  
nur IE ab 4.x

.target entspricht TARGET  
kann auch sein     \_blank  
                          \_parent  
                          \_search  
                          \_self  
                          \_top

lesen und schreiben

.text enthält den link\_text  
nur lesen  
nur NS ab 4.x

.x horizontale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes  
nur lesen  
nur NS ab 4.x

.y vertikale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes  
nur lesen  
nur NS ab 4.x

**Methoden (ausgewählte):**

.attachEvent() siehe Eventbehandlung des IE ab 5.x  
nur IE ab 5.x

.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

.detachEvent() siehe Eventbehandlung des IE ab 5.x  
nur IE ab 5.x

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

.getBoundingClientRect() liefert Kooerinatens des Rechtecktes um den Link  
Funktionswert ist Zeiger auf die Rechteck-Instanz vom Objekt-Typ TextRectangle  
mit den Eigenschaften     .left  
                                  .top  
                                  .right  
                                  .bottom  
                                  alles Pixelpositionen

nur IE ab 5.x

.handleEvent() siehe Eventbehandlung zum NS ab 4.x  
nur NS ab 4.x

.releaseCapture() siehe Eventbehandlung zum IE ab 5.x  
nur IE ab 5.x

.reload([true]) wenn true entfällt: Dokument vom Cache auf Festplatte laden  
wenn true kodiert: Dokument vom Server und nicht Cache laden  
Achtung: Server kann selbst Cache haben und  
                                  daraus das Dokument liefern



.replace(url)	aktuelle Seite mit neuer geladener Seite überschreiben sowie den History-Eintrag zur aktuellen Seite überschreiben (keinen neuen bilden) --> BACK-Button wechselt danach nicht zur überschriebenen Seite, da diese in der History nicht mehr bekannt ist
.scrollIntoView(true oder false)	Oberkante des Links in den sichtbaren Bereich scrollen true, so bis zum oberen Fensterrand false, so bis zum unteren Fensterrand nur IE ab 5.x
.setCapture()	siehe Eventbehandlung zum IE ab 5.x nur IE ab 5.x

**link Objekt des Internet Explorer:**

Das Objekt kann nur innerhalb <HEAD> ... <HEAD> benutzt werden und wird nicht angezeigt. Es dient zum Einbinden externer Dateien, die den Quellcode des Dokumentes manipulieren sollen.

Beispiel:

```
<LINK REL=stylesheet HREF="styles.css" type="text/css">
```

**Eigenschaften:**

.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.charset	Zeichensatz zum Encoden eines Objektes im Dokument
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.href	Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag) siehe auch Eigenschaften .rel und .rev
.hreflang	Sprachcode des Objektes laut RFC1766
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.rel	Beziehung zwischen Objekt in Quellseite und Nachfolger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen beim durchforsten der Seiten und Nachfolgerseiten vorrangig kodiert in <A> oder <LINK> es <b>muss</b> zugleich die Eigenschaft .href kodiert und mit <b>gültigen</b> Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)
.rev	Beziehung zwischen Objekt in Quellseite und Vorgänger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steuerungshinweis für Suchmaschinen beim durchforsten der Seiten und Vorgängerseiten vorrangig kodiert in <A> oder <LINK> es <b>muss</b> zugleich die Eigenschaft .href kodiert und mit <b>gültigen</b> Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.target	Name des Ziel-Fenster bzw. Ziel-Frame
.type	MIME-Typ des Objektes (Multipurpose Internet Mail Extension) wenn die Eigenschaft .classid nicht kodiert wird, dann immer .type kodieren bzw.



	die im Browserstandard enthaltenen MIME verwenden
	Hinweis: MIME wird von Simple Mail Transfer Protocol (SMTP) benutzt z.B. bei Audio, Images, Video, Texten
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementByName()
.hasChildNodes()	DOM nicht geändert prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert



- .insertAdjacentElement() Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich  
DOM wird geändert
- .mergeAttributes() alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen  
Attribute sind: HTML  
Events  
Styles  
ab IE 5.01 auch ID, NAME  
Achtung: Diese Methode ist mir Vorsicht zu geniessen !!  
DOM wird geändert
- .normalize() Normalisierung des DOM zur Erreichung einer konsistenten Struktur  
Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
- .removeAttribute() entfernen eines per HTML erzeugten Attributes  
Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!  
per Methode .createAttribute() erzeugte Attribute werden nicht erfasst  
DOM wird geändert
- .removeAttributeNode() entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern  
DOM wird geändert
- .removeBehavior() per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)  
DOM wird geändert
- .replaceAdjacentText() Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern  
DOM wird nicht geändert
- .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
- .setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert
- .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.27. document.links Collection des Internet Explorer (HTML-Element mit HREF-Attribut)**

Feld der Zeiger aller Objekte mit HREF-Eigenschaft (Attribut) sowie aller AREA-Objekte im Dokument, wobei **alle** diese Objekte das Attribut NAME und oder ID besitzen **müssen**, um in dieser Collection referenziert zu sein. Elementefolge laut HTML-Coding

**Syntax:**

```
[ var ZeigerAufFeld = ] document.links
[ var ZeigerAufFeldElement = ] document.links[Index]
```

Index	Integer ab 0 muss in [ ] kodiert sein
-------	--

**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
- .urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.28. map Objekt des Internet Explorer**

Das map-Objekt wird anhand eines img Objektes auf dem Client gebildet und stellt eine Sammlung (Mappe) von logischer Unterteilungen des Bildes in Teilbereiche dar. Jeder Teilbereich kann vom User agiert werden. Z.B. Selektive Auswahl des Users anhand der Teilbereiche des Bildes, um Bildteile in einem Extrafenster darstellen zu können. Teilbereiche werden nicht gerendert. Man sieht also nur das Bild als Gesamtheit.

Zur Bildung der Mappe wird der Anker als Wert des Attributes USEMAP benutzt.

```
Beispiel: <IMG SRC="test.gif" USEMAP="#freier_mappen_name">
```

DerAnker stellt den freiwählbaren Mappennamen dar. Zum schnelleren Rendern des Bildes sollten die Attribute WIDTH und HEIGHT kodiert werden.

Die Mappe selbst wird durch Bezug auf den Anker kodiert.

```
Beispiel: <MAP NAME="freier_mappen_name">
```



Zur Bildung eines Teilbereiches muss der Koordinatenbereich innerhalb der Bilddimensionen festgelegt werden. Dazu dient das area Objekt. Es sind beliebig viele Teilbereiche festlegbar, die alle die Dimension des Bildes einhalten müssen.

Beispiel: <AREA SHAPE="rect" COORDS="0,0,82,126" ALT="Teilbereich 1" HREF="/graphics/test.gif">

Die Aktionsmöglichkeiten des Users können z.B. per HREF-Attribut und optional per Eventhandler kodiert werden. Da Teilbereiche nicht gerendert werden, sollt als Hilfe für den User das ALT-Attribut kodiert werden, um einen Text zu sehen, der erscheint, wenn die Maus über den Teilbereich fährt.

Beispiel:

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

#### Eigenschaften:

.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
.isDisabled	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !!
.nextSibling	Element darf nicht per Methode .createElement() erzeugt worden sein
.nodeName	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem



	des Elternobjektes (.offsetParent)
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

**Methoden:**

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern



	DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar
	DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByTagName()
	DOM nicht geändert
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
	DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich
	DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
	DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes
	DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
	DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
	Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
	DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
	DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
	DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
	DOM wird geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
	DOM wird nicht geändert



- .replaceChild() Kind-Objekt ersetzen durch ein Objekt  
ersetzende Objekt muss per Methode .createElement() erzeugt worden sein  
Sichtbarkeit erst wenn Ende-Tag geparst wurde  
DOM wird geändert
- .replaceNode() Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern  
sichtbar erst mit parsen des Endetags  
DOM wird geändert
- .scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird  
Objekt muss an sich schon renderbar sein
- .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
- .setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern  
DOM wird geändert
- .setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.  
ab IE 5.5
- .swapNode() Hinweis: ausschalten per Methode .releaseCapture()  
Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparst  
DOM wird geändert

**4.3.2.2.4.3.29. map.areas Collection des Internet Explorer**

Zeigerfeld aller area Objekte eines map Objektes (siehe auch dort)  
Erzeugung eines Elementes kann nur erfolgen durch die Methoden  
.createElement() mit anschliessendem .add()  
oder .insertAdjacentHTML()

**Syntax:**

```
[ var ZeigerAufFeld = ] zeiger_auf_map_objekt.areas
[ var ZeigerAufFeldElement = ] zeiger_auf_map_objekt.areas[Index , SubIndex]
```

zeiger_auf_map_objekt	laut ID-Attribut
Index	Integer ab 0 oder String z.B. laut ID-Attribut muss in [ ] kodiert sein
SubIndex	optional Unterindex also Unterelement eines Elementes nur kodieren wenn Index ein String ist Integer
ZeigerAufFeld	ist null, wenn keine area Objekte vorhanden
ZeigerAufFeldElement	ist null, wenn Feldelement nicht vorhanden

**Eigenschaften:**

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

**Methoden:**

- .add() ein bereits erzeugtes Element einer Collection hinzufügen  
hinzufügen erst nach dem kompletten Laden des Dokumentes  
Element erzeugen per Methode .createElement()
- .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
- .remove() ein Element entfernen aus einer Collection
- .tags() Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern  
siehe tags Collection des DOM
- .urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.30. marquee Objekt des Internet Explorer**

Scrolltext, der horizontal oder vertikal in den Dimensionen des Elternobjektes laufen kann.  
Standardbreite ist die des Elternobjektes. Breite sollte kodiert werden, wenn marquee Objekt innerhalb TD liegt und der TD-Tag keine  
Breitenangabe enthält.

Für vertikales Scrollen ist die Eigenschaft .scrollLeft auf 0 zu setzen.  
Für horizontales Scrollen ist die Eigenschaft .scrollTop auf 0 zu setzen.

- .behavior 'scroll' Default.  
Scrollrichtung laut .direction
- 'alternate'
- 'slide' Scrollrichtung laut .direction, kein reinscrollen



Die Eigenschaften .scrollLeft und .scrollTop sind nur beschreibbar, wenn das Objekt **nicht** scrollt (Methode .start() noch nicht aktiviert bzw. Methode .stop() wurde aktiviert).

Die Methode .start() löst das Event onstart leider **nicht** aus.  
Die Methode .stop() löst das Event onstop leider **nicht** aus.

Beispiel 1:

```

<MARQUEE DIRECTION=RIGHT
          BEHAVIOR=SCROLL
          SCROLLAMOUNT=10
          SCROLLDELAY=200
>
    Dieser Text scrollt
</MARQUEE>

```

Beispiel 2:

```

<MARQUEE ID="ID_marquee"
          DIRECTION=right
          WIDTH=200
          HEIGHT=200
          STYLE="border-width:2px;border-style:solid;"
>
    Dieser Text scrollt rechts
</MARQUEE>
<BR>
<BUTTON onclick="alert(
    'scrollLeft: ' + ID_marquee.scrollLeft
    + ' scrollRight: ' + ID_marquee.scrollRight
    + ' scrollTop: ' + ID_marquee.scrollTop
    )"
>
    Anzeigen
</BUTTON>
<BUTTON onclick="ID_marquee.stop();ID_marquee.scrollLeft = 190;">
    Stop und scrollLeft=190
</BUTTON>
<BUTTON onclick=" ID_marquee.start();">
    Start
</BUTTON>

```

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst
ATOMICSELECTION	vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.begin	Selektierbarkeit des Objektes einstellen Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.behavior	Verhalten des Scrollens des marquee Objektes
.bgColor	deprecated und durch STYLE-Attribut zu ersetzen Hintergrundfarbe des Objektes bzw. Dokumentes
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataFormatAs	Datenquelle-Anzeigeart
.dataSrc	Datenquelle als Anker festlegen
.dir	Umflussrichtung
.direction	Start-Scrollrichtung des marquee Objektes
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden



	ab IE 6.x
	alternativ: Eigenschaft <code>.dur</code>
	siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
<code>.firstChild</code>	Zeiger auf das ERSTE Kind laut <code>childNodes</code> -Collection eines Objektes
<code>.hasMedia</code>	Objekt ist HTML-Media-Objekt
	siehe Objekt <code>currTimeState</code> und Behavior <code>.style.time2</code>
<code>.height</code>	Höhe des Objektes in Pixel
<code>.hideFocus</code>	Focussierbarkeit
<code>.hspace</code>	horizontaler Abstand in Pixel zum Elternobjekt
<code>.id</code>	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
	Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft <code>.uniqueID</code> ermittelt und anstelle der Eigenschaft <code>.id</code> verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft <code>.uniqueID</code> kennen).
	Zeiger aus ID bilden <code>var Zeiger = eval(object.id);</code>
<code>.innerHTML</code>	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
	ein Tag ohne Ende-Tag kann kein <code>innerHTML</code> haben, z.B. <code>&lt;B&gt;</code>
	dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
	also nach dem Laden des Objektes
	aber wirksam erst mit parsen des HTML-Endetags
<code>.innerText</code>	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
	ein Tag ohne Ende-Tag kann kein <code>innerHTML</code> haben, z.B. <code>&lt;B&gt;</code>
	dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
	also nach dem Laden des Objektes
	aber wirksam erst mit parsen des HTML-Endetags
<code>.isContentEditable</code>	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
	Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
<code>.isDisabled</code>	Interaktionsfähigkeit
	nur wenn sichtbar so User-Interaktion möglich
<code>.isMultiLine</code>	Mehrzeiligkeit des Objektinhaltes
<code>.isTextEdit</code>	Erzeugbarkeit eines Textbereiches
<code>.lang</code>	Sprache für Anzeige von Sonderzeichen etc.
<code>.language</code>	Sprache für Script festlegen
<code>.lastChild</code>	Zeiger auf das LETZTE Kind laut <code>childNodes</code> collection eines Objektes
<code>.loop</code>	Anzahl der Scrollaktionen des <code>marquee</code> Objektes
<code>.nextSibling</code>	Zeiger auf das NACHFOLGENDE Kind laut <code>childNodes</code> collection eines Objektes
<code>.nodeName</code>	String als Name des Kindes (Knoten, Node, Element)
	also TAG-Bezeichner, Attribut-Name; <code>#text</code> für Anker
<code>.nodeType</code>	Knotentyp laut <code>attributes</code> Collection
<code>.nodeValue</code>	Knotenwert (Wert des Kindes, Node, Elementes)
	nur für Text- und Attribut-Elemente
	nicht für Element-Knoten (Knotentyp 1)
<code>.offsetHeight</code>	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetLeft</code>	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetParent</code>	Referenz der Eltern
	für Nutzung von <code>.offsetHeight</code> , <code>.offsetLeft</code> , <code>.offsetTop</code> und <code>.offsetWidth</code>
<code>.offsetTop</code>	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.offsetWidth</code>	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes ( <code>.offsetParent</code> )
<code>.onOffBehavior</code>	deprecated ab IE 5.x
	Unterstützung von <code>DirectAnimation</code> z.B. für 2D, 3D, Sound
<code>.outerHTML</code>	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
	wirksam mit parsen des Ende-Tag
	nur nach kompletten Einlesen des Dokumentes nutzbar
<code>.outerText</code>	Referenz auf den gesamten Plain-Text im Objekt
	nur nach kompletten einlesen des Dokumentes nutzbar
<code>.ownerDocument</code>	Referenz auf das <code>document</code> Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
<code>.parentElement</code>	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
<code>.parentNode</code>	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
<code>.parentTextEdit</code>	Textbereich des Elternobjektes referenzieren
<code>.previousSibling</code>	Referenz auf das Vorgängerkind
<code>.readyState</code>	aktueller Status des Objektes beim Füllen des Objektes mit Daten
<code>.recordNumber</code>	Datenquelle-Satznummer eines Datenfeldes
<code>.scopeName</code>	Namensraum laut XMLNS-Attribut
<code>.scrollAmount</code>	Scrollschrittweite in Pixel des <code>marquee</code> Objektes
<code>.scrollDelay</code>	Wartezeit in Millisekunden <b>und</b> laut Eigenschaft <code>.trueSpeed</code> zwischen zwei Scrollschritten eines <code>marquee</code> Objektes
	Scrollgeschwindigkeit, Fließgeschwindigkeit
<code>.scrollHeight</code>	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
<code>.scrollLeft</code>	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
	nur nutzbar nach dem kompletten Laden des Dokumentes



	Hinweis: Eigenschaften .scrollLeft und .scrollTop sind <b>nicht</b> beschreibbar, solange eine Scrollaktion aktiv ist.
.scrollTop	Abstand von oberem sichtbarem Rand des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes Hinweis: Eigenschaften .scrollLeft und .scrollTop sind <b>nicht</b> beschreibbar, solange eine Scrollaktion aktiv ist.
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex STYLE	Index des Objektes in der Collection document.all direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.trueSpeed	zeitliche Takteinheit zur Erzeugung der Pause laut Eigenschaft .scrollDelay eines marquee Objektes
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.vspace	vertikaler Abstand in Pixel zum Elternobjekt
.width	Breite des Objektes in Pixel
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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



.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName() DOM nicht geändert
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann



	nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu genießen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endtags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style- Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.start()	Start der Scrollaktion eines marquee Objektes Anzahl der Scrollaktionen laut Eigenschaft .loop erzeugt <b>nicht</b> das Ereignis onstart



Hinweis: Eigenschaften .scrollLeft und .scrollTop sind **nicht** beschreibbar, solange eine Scrollaktion aktiv ist.

.stop() Stopp der aktuellen und laut Eigenschaft .loop noch offener Scrollaktionen eines marquee Objektes erzeugt **nicht** das Ereignis onstop

Hinweis: Eigenschaften .scrollLeft und .scrollTop sind **nicht** beschreibbar, solange eine Scrollaktion aktiv ist.

.swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.31. meta Objekt des Internet Explorer**

dient der Beschaffung von Informationen über Dokument, Client und Server durch Suchmaschinen, Browser und Server  
nur im HEAD des Dokumentes kodierbar  
nicht verschachtelbar

Kodierung des META-Objektes in HTML

```
<META HTTP-EQUIV="....." CONTENT="....." SCHEME=".....">
oder <META NAME="....." CONTENT="....." SCHEME=".....">
```

HTTP-EQUIV und NAME beschreiben die Art der Information  
CONTENT enthält den Wert der Information  
SCHEME-Attribut ist optional

Beispiel: Dokument für Suchmaschine vorbereiten

```
<META NAME="description"
oder NAME="author"
oder NAME="keywords"
oder NAME="date"
oder NAME="robots"

CONTENT="text_bzw_angaben_zu_name"
LANG="xx"
>
```

Beispiele:

```
<META NAME="author " CONTENT="Ich ">
<META NAME="date" CONTENT="Mittwoch, 26. April 2000">
<META NAME="description" CONTENT="freier text der von suchmaschinen indiziert wird">
<META NAME="keywords" LANG="xx" CONTENT="stichwortliste freier art">
```

Hinweis zum Attribut LANG:

Verwendung für mehrsprachige Stichwortliste  
lang="xx" z.B. "de" für Deutschland  
"en" für Englisch

Bsp.:

```
<META NAME="keywords" CONTENT="....." LANG="de">
<META NAME="keywords" CONTENT="....." LANG="en">

<META NAME="revisit-after" CONTENT="20 days">
```

Suchmaschine beim Scannen des Dokumentes beeinflussen:

Suchmaschine darf alles tun

```
<META HTTP-EQUIV="Robots" CONTENT="all" >
```

Suchmaschine darf Dokument nichts scannen

```
<META HTTP-EQUIV="Robots" CONTENT="noindex" >
```

Suchmaschine darf Dokument scannen

```
<META HTTP-EQUIV="Robots" CONTENT="index" >
```

Suchmaschine darf Verweisen folgen

```
<META HTTP-EQUIV="Robots" CONTENT="follow" >
```

Suchmaschine darf Verweisen nicht folgen

```
<META HTTP-EQUIV="Robots" CONTENT="nofollow" >
```

Beispiel: Um sicher zu gehen, daß immer die Seite mit garantiert aktuellem Stand geladen wird

--> eventuell ist Proxyserver nicht aktuell

```
<META HTTP-EQUIV="expires" CONTENT=sekunden_wert>
```

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

Tag und Zeitpunkt an dem unter Umgehung des Proxyserver-Cache geladen wird



"day, tt mon jjjj hh:mm:ss GMT"

Bsp: "Sat, 14 Dec 1999 12:00:00 GMT"

day: Mon Tue Wed Thu Fri Sat Sun

mon: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

TITLE-Tag kodieren  
falls dieser nicht kodiert, so "No Title" von der Suchmaschine verwendet

jede Seite MUSS einen Rücksprung zur Startseite enthalten, da eine Suchmaschine Links auf der Seite abklappert

Beispiel für die Nutzung von browsereigenen Erweiterungen ab IE 4.x:

HTML-Seite ein- bzw.ausblenden per Filter

```
<META HTTP-EQUIV="Page-Enter" content="revealTrans(Duration=2.0, Transition=x)">
<META HTTP-EQUIV="Page-Exit" content="revealTrans(Duration=2.0, Transition=x)">
```

mit x	für	0	sich schliessender Kasten
		1	sich öffnender Kasten
		2	sich schliessender Kreis
		3	sich öffnender Kreis
		4	Vorhang nach oben
		5	Vorhang nach unten
		6	Vorhang nach rechts
		7	Vorhang nach links
		8	vertikaler Streifen
		9	horizontaler Streifen
		10	Schachbrettmuster nach rechts
		11	Schachbrettmuster nach unten
		12	zerbröseln
		13	Vorhang senkrecht zu Mitte hin
		14	Vorhang senkrecht von Mitte weg
		15	Vorhang horizontal zu Mitte hin
		23	Zufallsform

Beispiel für Vermeidung von Smart-Tags ab IE 6.x:

```
<META NAME="MSSmartTagsPreventParsing" CONTENT="TRUE">
```

Beispiel Dokument alle 2 Sekunden neu laden

```
<META HTTP-EQUIV="REFRESH" CONTENT=2>
```

Beispiel Zeichensatz des Dokumentes

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

Beispiel Themenunterstützung abschalten

```
<META HTTP-EQUIV="MSTHEMECOMPATIBLE" CONTENT="no">
```

Beispiel Grafik-Smarttag abschalten ab IE 6.x (Deaktivierung der automatischen Symbole bei Maus-Over über ein Bild)

```
<META HTTP-EQUIV="IMAGETOOLBAR" CONTENT="no">
```

Alternative: pro IMG das Attribut GALLERYIMG mit Wert "yes" bzw. "no" kodieren

**Eigenschaften:**

.charset	Zeichensatz zum Encoden eines Objektes im Dokument
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.content	Wert der Informationen laut Eigenschaften .httpEquiv und .name des meta Objektes
.disabled	Interaktionsfähigkeit



.httpEquiv	nur wenn sichtbar so User-Interaktion möglich Informationen des HTTP Response Header per meta Objekt Beschaffung der Informationen per Serverfunktionen HttpQueryInfo und QueryInfo Dieser Kopf wird von Suchmaschine, Server und Browser verwendet der Wert der Information wird in der Eigenschaft .content abgelegt
.isTextEdit	Erzeugbarkeit eines Textbereiches
.name	Informationen diverser Arten per meta Objekt der Wert der Information wird in der Eigenschaft .content abgelegt
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.scheme	Schema der Interpretation des Wertes laut Eigenschaft .content per meta Objekt z.B. von Datum und Zeit oder Tag-Content-Beschreibung
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrolling	Scrollenbar erzeugen
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
.tagName	Tag-Bezeichner des Objektes
<b>Methoden:</b>	
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.remove()	ein Element aus einer Collection entfernen
.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

**4.3.2.2.4.3.32. noFrames Objekt des Internet Explorer**

Container für denjenigen Code der Homepage, der nur durch Browser abgearbeitet wird, die FRAMESET-Tag nicht kennen.

Beispiel:

```

<FRAMESET>
  <NOFRAMES>
    Ihr Browser kennt das FRAMESET-Tag nicht
  </NOFRAMES>
</FRAMESET>

```

**Zugriff auf noFrames-Eigenschaft:**

Beispiel

```
var Wert = document.all.ZeigerAufnoFrames.eigenschaft;
```

mit ZeigerAufFrameset laut ID-Attribut

```
<NOFRAMES ID="ZeigerAufnoFrames" ..... >
```

**Eigenschaften:**

.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout



	onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.fireEvent()	ein Event auslösen
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert

#### 4.3.2.2.4.3.33. noScript Objekt des Internet Explorer

Container für denjenigen Code der Homepage, der nur durch Browser abgearbeitet wird, die das SCRIPT-Tag nicht kennen.

##### Zugriff auf noScript-Eigenschaft:

Beispiel

```
var Wert = document.all.ZeigerAufnoScript.eigenschaft;
```

mit ZeigerAufFrameset laut ID-Attribut

```
<NOSCRIPT ID="ZeigerAufnoScript" ..... >
```

##### Eigenschaften:

.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut

##### Methoden:

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.fireEvent()	ein Event auslösen
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)





	dient als Ersatz für die browserspezifischen MIME-Typen
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.code	Url der *.class-Datei (kompilierter Javacode)
.codeBase	Url der Komponente für Download der Komponente vom Server auf den Client optionale Versionsprüfung möglich (nicht bei Applet-Komponente) um unnötigen Download zu ersparen: vorheriger Ableich der Version der Komponente auf Server und Client auch wenn der Versionsabgleich keinen Download ergab, wird immer HTTP Header-Transaktion ausgelöst
.codeType	Internet Media Type (Mimety) des Codes zum Objekt, das per OBJECT-Tag eingebunden
wurde	
.data	Url der Daten des Objektes
.declare	Zeichenkette für Declare-Funktionalität des Objektes, das per OBJECT-Tag eingebunden wurde
.dir	Umlflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.height	Höhe des Objektes in Pixel
.hideFocus	Focussierbarkeit
.hspace	horizontaler Abstand in Pixel zum Elternobjekt
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.object	Zeiger auf das Objekt laut Applet bzw. laut Objekt object
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordset	Zeiger des Standard-Record-Set in einem Datasource-Objekt (DSO) Methode . namedRecordset() liefert den Zeiger für Datenquellen-Objekt mit Namen



	also eines beliebige Mitgliedes im Datasource-Objekt (DSO)
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes Hinweis: Eigenschaften .scrollLeft und .scrollTop sind <b>nicht</b> beschreibbar, solange eine Scrollaktion aktiv ist.
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes Hinweis: Eigenschaften .scrollLeft und .scrollTop sind <b>nicht</b> beschreibbar, solange eine Scrollaktion aktiv ist.
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
.standby	Zeichenkette für Standby-Funktionalität des per OBJECT-Tag eingebundenen Objektes
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
	Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA
	Anspringen default nicht per TAB-Taste für APPLETT, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.type	MIME-Typ des Objektes (Multipurpose Internet Mail Extension) wenn die Eigenschaft .classid nicht kodiert wird, dann immer .type kodieren bzw. die im Browserstandard enthaltenen MIME verwenden Hinweis: MIME wird von Simple Mail Transfer Protocol (SMTP) benutzt z.B. bei Audio, Images, Video, Texten
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.useMap	Url oder Anker für client-seitige Image-Map
.vspace	vertikaler Abstand in Pixel zum Elternobjekt
.width	Breite des Objektes in Pixel
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernen DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus



.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Überbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichnet zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop() .fireEvent() .focus()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element ein Event auslösen Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect() .getClientRects()	Referenz auf TextRectangle-Objekt im Element holen Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.namedRecordset()	Zeiger desjenigen Datenquellen-Record-Objektes mit Namen, das den Standard-Record-Set enthält Hinweis: Namen zeigt die Mitgliedschaft in einem Datasource-Objekt (DSO) an Eigenschaft .recordset liefert den Zeiger für den Standard-Record-Set in einem Datasource-Objekt (DSO)
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)



.removeExpression()	DOM wird geändert Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient. Ausdruck muss mit der Methode <code>.setExpression()</code> gesetzt worden sein
.removeNode()	DOM wird nicht geändert Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
.replaceAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.replaceNode()	DOM wird nicht geändert Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
.scrollIntoView()	DOM wird geändert Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
.setAttributeNode()	DOM wird nur bei Erzeugung geändert Attribut einem Knoten zuweisen und Referenz liefern
.setCapture()	DOM wird geändert Maus-Überwachung einschalten für ein Objekt Maus-Events sind : <code>onmousedown</code> , <code>onmouseup</code> , <code>onmousemove</code> , <code>onclick</code> , <code>ondblclick</code> , <code>onmouseover</code> und <code>onmouseout</code> .
.setExpression()	ab IE 5.5 Hinweis: ausschalten per Methode <code>.releaseCapture()</code> Wert definieren, der als Ausdruck für die Methode <code>.getExpression()</code> zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> dient Ausdruck nur als Script kodierbar
.swapNode()	DOM wird nicht geändert Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.35. document.embeds Collection des Internet Explorer (vermutlich auch im IE 6.x)

Feld der Zeiger aller als Plugin per EMBED-Tag eingebetteten Objekte im Dokument (inkl. Applets)  
Elementefolge laut HTML-Coding

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 existieren, laufen unter dem IE ab 6.x nicht mehr.

Es ist zu vermuten, dass im IE ab 6.x diese Collection nur eine Dummy-Funktion hat, also existiert, aber nie angefasst wird.

Diese Collection 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. Alle eingebetteten Objekte haben in `document.embeds` ihren Zeiger (inklusive Plugins, Ausnahme: siehe tiefer).

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 Internet 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.

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:**  
durch Browser

Beispiel für EMBED-Tag beim IE unter 6.x und beim NS:

```
<EMBED
  SRC="url"
  TYPE="mime_typ"           // z.B. "image/gif"
  PLUGINSOURCE="plugin_url"
  HEIGHT="hoehe_plugin_objekt"
  WIDTH="breite_plugin_objekt"
  NAME="ID_Embed"
  HIDDEN="true oder false" // true so Objekt unsichtbar
>
<PARAM Name="parameter_name" VALUE=parameter_wert>
```



```
.....
</EMBED>
```

**Syntax:**

```
[ var ZeigerAufFeld = ] document.embds
[ var ZeigerAufFeldElement = ] document.embds[Index, [SubIndex]]
```

Index	oder	Integer ab 0 String Name oder ID des Elementes muss in [ ] kodiert sein
SubIndex		optional nur kodieren wenn Index ein String ist Integer als Unterindex also Unterelement eines Elementes

```
ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()
```

beim IE:

```
document.all.ID_Embed.eigenschaft
document.all.ID_Embed.methode()
```

beim NS:

```
document.ID_Embed.eigenschaft
document.ID_Embed.methode()
```

**Eigenschaften beim Internet Explorer:**

.length Anzahl der Feldelemente also Feldlänge

**Methoden beim Internet Explorer:**

.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

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.36. document.select Objekt des Internet Explorer**

Drop-Down-Liste oder Liste in einer Box (List-Box) als Objekt:

Die Zeiger aller Optionen werden in der Collection document.select.options gesammelt.

Ab IE 5.5 werden option Objekte zusätzlich auch über die Collection document.all referenzierbar.

Achtung: .z-index Attribut wird nicht unterstützt.

Beispiel 1:

```
<SELECT NAME="Katzen" SIZE="1">
  <OPTION VALUE="1">Mischling
  <OPTION VALUE="2">Siamese
  <OPTION VALUE="3" SELECTED>Kratzkekel
</SELECT>
```

Beispiel 2:

```
<SELECT NAME="Autos" SIZE="3" MULTIPLE>
  <OPTION VALUE="1" SELECTED>BMW
  <OPTION VALUE="2">Porsche
  <OPTION VALUE="3" SELECTED>Mercedes
</SELECT>
```

Beispiel 3:

```
<SCRIPT LANGUAGE="JScript">
  var OptionObjekt = document.createElement("OPTION");
  OptionObjekt.text="Ferrari";
  OptionObjekt.value="4";

  // Annahme: selection Objekt sei vorhanden
  SelektionObjekt.add(OptionObjekt);
</SCRIPT>
```

Beispiel 4: Leere Textarea per Textwert einer selektierten Option aus Drop-Down-Liste füllen:

```
<SCRIPT LANGUAGE="JScript">
  function TextAreaErweitern()
  {
    // Index der selektierte Option holen
    var Index = ID_Select.selectedIndex;

    // Text-Wert der selektieren Option holen
```



```

var TextWert = ID_Select.options[Index].text;

// Textwert an Textarea anhängen
ID_Textarea.value+= TextWert + "\n";
}
</SCRIPT>
<SELECT ID="ID_Select" SIZE="1" onchange="TextAreaErweitern()">
  <OPTION VALUE="1">BMW
  <OPTION VALUE="2">PORSCHE
  <OPTION VALUE="3" SELECTED>MERCEDES
</SELECT>
<TEXTAREA ID="ID_Textarea"></TEXTAREA>

```

Beispiel 5:

```

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

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

```

Beispiel 6:

```

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

```

Beispiel 7:

```

<SELECT ID="ID_select" MULTIPLE>
  <OPTION>Auswahl 1</OPTION>
  <OPTION> Auswahl 2</OPTION>
  <OPTION> Auswahl 3</OPTION>
</SELECT>
<BUTTON onclick="ID_select.multiple=false">keine multiple Auswahl</BUTTON>
<BUTTON onclick="ID_select.multiple=true">multiple Auswahl</BUTTON>

```

Beispiel 8:

```

<SCRIPT>
var ZeigerAufNeueOption = document.createElement("OPTION");
ID_Select.options.add(ZeigerAufNeueOption);
ZeigerAufNeueOption.innerText = "Option 2";
ZeigerAufNeueOption.Value = "2";
</SCRIPT>
<SELECT ID="ID_Select" >
  <OPTION VALUE="1">Option 1</OPTION>
</SELECT>

```

Beispiel 9:

```

<SCRIPT>
function Erzeugen()
{
  ID_Span.innerHTML="";
  var FeldDerZeigerAufOption = ID_Select.options[ID_Select.selectedIndex];

  if(FeldDerZeigerAufOption.text.length>0)
  {
    var ZeigerAufElement= document.createElement(FeldDerZeigerAufOption.text);
    eval(
      " ZeigerAufElement."
      + FeldDerZeigerAufOption.value

```



```

+ ""
+ ID_Input.value
+ ""
);

if(FeldDerZeigerAufOption.text=="A")
{ ZeigerAufElement.href="javascript:alert('A link.');" }
}
ID_Span.appendChild(ZeigerAufElement);
}
</SCRIPT>
<SELECT ID="ID_Select" onchange="Erzeugen()">
  <OPTION VALUE="innerText">A
  <OPTION VALUE="value">&lt;INPUT TYPE="button"&gt;
</SELECT>
<INPUT TYPE="text" ID="ID_Input" VALUE="Beispiel Text">
<SPAN ID="ID_Span" ></SPAN>

```

**Zugriff:**

```

document.all.zeiger_auf_select_objekt.eigenschaft
document.all.zeiger_auf_select_objekt.methode()

```

zeiger\_auf\_select\_objekt z.B. laut ID-Attribut

**Eigenschaften:**

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
.align	Ausrichtung
.ATOMICSELECTION	Selektierbarkeit einstellen
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.dir	Umlflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.length	Anzahl der Feldelemente also Feldlänge der Collection document.select.options (siehe Eigenschaft .options)
.multiple	Multiple Auswahl bei document.select



.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von offsetHeight, offsetLeft, offsetTop und offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.options	Zeiger auf die Collection document.select.options des document.select Objektes
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.selectedIndex	Index der Option in der Collection document.select.options des Objektes select jedes option Objekt eines select Objektes wird im Feld, das Teil des select Objektes ist, referenziert es kann immer nur eine Option innerhalb einer Optionsgruppe selektiert sein
.size	Anzahl der Zeilen in einer List-Box als select Objekt
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen Achtung: Bezüglich der Style-Eigenschaften .background-color und .color kann nur genau 1 Style deklariert werden für <b>alle</b> Objekte document.select.option <b>und</b> das document.select Objektes insgesamt. Es wird der Style des ersten in der Kodierungsfolge gefundenen Objektes document.select.option für alle anderen <b>und</b> das Objekt document.select selbst verwendet. Es wird angeraten, <b>nur</b> im document.select das STYLE-Attribut zu verwenden. Das Objekt document.select.option kann <b>nur</b> genau folgende Style-Eigenschaften erhalten: .background-color und .color
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup  Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA



	Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.type	prüfen auf multiples select Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektierbarkeit z.B. mit Maus <b>wird nicht an Kinder vererbt</b> bei Wechsel: vorübergehende vorhandene Selektion wird nicht verändert
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar
<b>Methoden:</b>	
.add()	ein bereits erzeugtes Element einer Collection hinzufügen hinzufügen erst nach dem kompletten Laden des Dokumentes Element erzeugen per Methode .createElement()
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entferbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht



	Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. Hinweis: einschalten per Methode .setCapture()
.remove()	ein Element aus einer Collection entfernen
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient.



	Ausdruck muss mit der Methode <code>.setExpression()</code> gesetzt worden sein
	DOM wird nicht geändert
<code>.removeNode()</code>	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
<code>.replaceAdjacentText()</code>	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
	DOM wird nicht geändert
<code>.replaceChild()</code>	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode <code>.createElement()</code> erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
<code>.replaceNode()</code>	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags
	DOM wird geändert
<code>.scrollIntoView()</code>	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
<code>.setActive()</code>	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
<code>.setAttribute()</code>	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
<code>.setAttributeNode()</code>	Attribut einem Knoten zuweisen und Referenz liefern
	DOM wird geändert
<code>.setCapture()</code>	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : <code>onmousedown</code> , <code>onmouseup</code> , <code>onmousemove</code> , <code>onclick</code> , <code>ondblclick</code> , <code>onmouseover</code> und <code>onmouseout</code> .
	ab IE 5.5
	Hinweis: ausschalten per Methode <code>.releaseCapture()</code>
<code>.setExpression()</code>	Wert definieren, der als Ausdruck für die Methode <code>.getExpression()</code> zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form <code>objekt.style.eigenschaft</code> . dient
	Ausdruck nur als Script kodierbar
	DOM wird nicht geändert
<code>.swapNode()</code>	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt
	DOM wird geändert
<code>.urns()</code>	Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

#### 4.3.2.2.4.3.36.1. **document.select.option Objekt des Internet Explorer**

Objekt einer Option aus einer Selektion per `document.select` Objekt

Option ist anwählbar

Die Zeiger aller Optionen werden in der Collection `document.select.options` gesammelt.

**Ab IE 5.5** werden option Objekte zusätzlich auch über die Collection **document.all** referenzierbar.

Zum Hinzufügen eines option Objektes müssen das `select` und `option` Objekt in einem **gemeinsamen** Fenster liegen.

Beispiel 1:

```
<SELECT NAME="Katzen" SIZE="1">
  <OPTION VALUE="1">Mischling
  <OPTION VALUE="2">Siamese
  <OPTION VALUE="3" SELECTED>Kratzkegel
</SELECT>
```

Beispiel 2:

```
<SELECT NAME="Autos" SIZE="3" MULTIPLE>
  <OPTION VALUE="1" SELECTED>BMW
  <OPTION VALUE="2">Porsche
  <OPTION VALUE="3" SELECTED>Mercedes
</SELECT>
```

Beispiel 3:

```
<SCRIPT LANGUAGE="JScript">
  var OptionObjekt = document.createElement("OPTION");
  OptionObjekt.text="Ferrari";
  OptionObjekt.value="4";

  // Annahme: selection Objekt sei vorhanden
  SelektionObjekt.add(OptionObjekt);
</SCRIPT>
```

Beispiel 4: Leere Textarea per Textwert einer selektierten Option aus Drop-Down-Liste füllen:

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



```

function TextareaErweitern()
{
    // Index der selektierte Option holen
    var Index = ID_Select.selectedIndex;

    // Text-Wert der selektieren Option holen
    var TextWert = ID_Select.options[Index].text;

    // Textwert an Textarea anhängen
    ID_Textarea.value+= TextWert + "\n";
}
</SCRIPT>
<SELECT ID="ID_Select" SIZE="1" onchange="TextareaErweitern()">
  <OPTION VALUE="1">BMW
  <OPTION VALUE="2">PORSCHE
  <OPTION VALUE="3" SELECTED>MERCEDES
</SELECT>
<TEXTAREA ID="ID_Textarea"></TEXTAREA>

```

Beispiel 5:

```

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

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

```

Beispiel 6:

```

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

```

Beispiel 7:

```

<SELECT ID="ID_select" MULTIPLE>
  <OPTION>Auswahl 1</OPTION>
  <OPTION> Auswahl 2</OPTION>
  <OPTION> Auswahl 3</OPTION>
</SELECT>
<BUTTON onclick="ID_select.multiple=false">keine multiple Auswahl</BUTTON>
<BUTTON onclick="ID_select.multiple=true">multiple Auswahl</BUTTON>

```

Beispiel 8:

```

<SCRIPT>
var ZeigerAufNeueOption = document.createElement("OPTION");
ID_Select.options.add(ZeigerAufNeueOption);
ZeigerAufNeueOption.innerText = "Option 2";
ZeigerAufNeueOption.Value = "2";
</SCRIPT>
<SELECT ID="ID_Select" >
  <OPTION VALUE="1">Option 1</OPTION>
</SELECT>

```

Beispiel 9:

```

<SCRIPT>
function Erzeugen()
{
    ID_Span.innerHTML="";
    var FeldDerZeigerAufOption = ID_Select.options[ID_Select.selectedIndex];

```





	document.select Objekt
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parse des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEditable	Erzeugbarkeit eines Textbereiches
.label	Label einer Option, also Objektes document.select.option des Objektes document.select
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von offsetHeight, offsetLeft, offsetTop und offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberem sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.selected	Selektionsstatus der Option, also Objektes document.select.option des Objektes document.select standardgemäß ist immer die oberste Option selektiert es kann aus einer Optionengruppe immer nur genau 1 Option selektiert sein direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen Achtung: Bezüglich der Style-Eigenschaften .background-color und .color kann nur genau 1 Style deklariert werden für <b>alle</b> Objekte document.select.option <b>und</b> das document.select Objektes insgesamt. Es wird der Style des ersten in der Kodierungsfolge gefundenen Objektes document.select.option für alle anderen <b>und</b> das Objekt document.select selbst verwendet. Es wird angeraten, <b>nur</b> im document.select das STYLE-Attribut zu verwenden. Das Objekt document.select.option kann <b>nur</b> genau folgende Style-Eigenschaften erhalten: .background-color und .color
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit).



	nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2 siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemLanguage	Standard-Sprache des Betriebssystems (Sprache der Installation) per navigator Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.text	interner Kettenwert einer Option, , also Objektes document.select.option des Objektes document.select ändert nicht den angezeigten Text hinter <OPTION ...> ist nur ein interner Wert und beim Formular der gesendete Wert der Option
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
.value	Wert eines Objekt-Attributes Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar im Formular: wenn Eigenschaft .text kodiert, so wird deren Wert gesendet und der Wert von .value bleibt nur angezeigt wenn Eigenschaft .text nicht kodiert, so wird der Wert von .value gesendet
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert
.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_ bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht



	DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern
	DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar
	DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getExpression()	Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
	DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
	DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich
	DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
	DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes
	DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu genießen !!
	DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
	Hinweis: einschalten per Methode .setCapture()
.removeAttribute()	entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
	DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
	DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
	DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde
	DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft.dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
	DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde
	DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu



- ersetzenden Text liefern
- DOM wird nicht geändert
- .replaceChild() Kind-Objekt ersetzen durch ein Objekt  
ersetzende Objekt muss per Methode .createElement() erzeugt worden sein  
Sichtbarkeit erst wenn Ende-Tag geparkt wurde
- DOM wird geändert
- .replaceNode() Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern  
sichtbar erst mit parsen des Endetags
- DOM wird geändert
- .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
- .setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern
- DOM wird geändert
- .setCapture() Maus-Überwachung einschalten für ein Objekt  
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,  
onmouseover und onmouseout.
- ab IE 5.5
- Hinweis: ausschalten per Methode .releaseCapture()
- .setExpression() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-  
Eigenschaft als Objektreferenz der Form  
objekt.style.eigenschaft.  
dient  
Ausdruck nur als Script kodierbar
- DOM wird nicht geändert
- .swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)  
nur sichtbar wenn Endetag geparkt  
DOM wird geändert

**4.3.2.2.4.3.36.2. document.select.options Collection des Internet Explorer**

Eine Option ist über dieses Feld der Zeiger aller Optionen anwählbar:

Die Zeiger aller Optionen werden in der Collection document.select.options gesammelt.

**Ab IE 5.5** werden option Objekte zusätzlich auch über die Collection **document.all** referenzierbar.

Eine Option aus dem Feld löschen, erfolgt durch Zuweisung des Null-Zeiger (null). Zugleich wird automatisch das Feld kondensiert, also alle Lücken werden entfernt.

Zum Hinzufügen eines option Objektes müssen das select und option Objekt in einem **gemeinsamen** Fenster liegen.

**Syntax:**

```
[ var ZeigerAufFeld = ] document.zeiger_auf_select_objekt.options
[ var ZeigerAufFeldElement = ] document.zeiger_auf_select_objekt.options [Index [, SubIndex]]
```

- Index Integer ab 0  
oder String Name oder ID des Elementes  
(analog zu NAME und ID-Attribut)  
muss in [ ] kodiert sein
- SubIndex optional  
nur kodieren wenn Index ein String ist  
Integer als Unterindex also Unterelement eines Elementes
- ZeigerAufFeldElement  
ist null, wenn Feldelement nicht vorhanden  
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection  
**aus diesen** Script-Objekten geliefert

zeiger\_auf\_select\_objekt z.B. laut ID-Attribut

Beispiel:

```
var Feld = document.all.tags("SELECT");

if (Feld.length>0)
{
  for (var i=0; i < Feld[0].options.length; i++)
  {
    alert("Element " + i
      + " : mit internem Text = " + Feld[0].options(i).text
      + " und angezeigtem Wert = " + Feld[0].options(i).value);
  }
}
```

**Eigenschaften:**

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

**Methoden:**

- .add() ein bereits erzeugtes Element einer Collection hinzufügen  
hinzufügen erst nach dem kompletten Laden des Dokumentes  
Element erzeugen per Methode .createElement()
- .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

.remove() ein Element aus einer Collection entfernen

.tags() Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern  
 siehe tags Collection des DOM

.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

**4.3.2.2.4.3.37. document.selection Objekt des Internet Explorer**

**Aktive Selektion** eines Dokumentes markierter Textblock (document.selection.textrange Collection, Objekt textrange)  
 Control-Element(e) (document.selection.controlRange Collection) nur für body Objekt

Selektion erzeugbar durch User  
 Methode .select()

im Dokument kann zu jedem Zeitpunkt nur **genau 1** Selektion aktiv sein: ID-Attribut wird daher nicht unterstützt, weil nicht nötig

Leere Selektion entspricht: User setzt Cursor auf einen leeren Bereich im Dokument.

Elternobjekte mit Textrange sind body Objekt  
 button Objekt  
 textarea Objekt  
 input text Objekt  
 selection Objekt (nur wenn ein Text selektiert wurde (mit oder ohne HTML))

wobei diese weitere HTML-Elemente enthalten können, die ebenfalls Textbereiche besitzen können

**Zugriff:**

document.selection.eigenschaft  
 document.selection.methode()

Hinweis: Im Dokument kann zu jedem Zeitpunkt nur genau 1 Selektion aktiv sein. Deshalb reicht **document.selection** weil es eindeutig ist.

**Eigenschaften:**

.type Type der Selektion per document.selection Objekt des Dokumentes  
 siehe auch Methoden .createRange() .createControlRange() und .createTextRange()  
 Objekt textrange

.typeDetail Type der Selektion per document.selection Objekt des Dokumentes, wobei die Hostanwendung  
 diese Eigenschaft unterstützen und mit Wert belegen muss

**Methoden:**

.clear() Selektion als Markierung aufheben per document.selection Objekt des Dokumentes per document.selection  
 nicht Selektion löschen  
 siehe Methode .empty()

.createRange() Zeiger auf einen Universal-Bereich erzeugen per document.selection Objekt des Dokumentes  
 Universal-Bereich kann enthalten:  
 Text (document.selection.textrange Collection  
 textrange Objekt)  
 oder Control-Element(e) (document.selection.controlRange Collection)  
 siehe auch Methoden .createControlRange() .createTextRange() und .createRangeCollection()  
 Eigenschaft .type

.createRangeCollection() document.selection.textrange Collection erzeugen per document.selection Objekt des Dokumentes  
 nur wenn der Browser multiple Selektion unterstützt, so mehr als 1 Feld-Element textrange Objekt  
 vorhanden bei Mehrfach-Selektion durch User  
 siehe auch textrange Objekt  
 Methoden .createRange() .createControlRange() und .createTextRange()

.empty() Selektion löschen per document.selection Objekt des Dokumentes  
 setzt zugleich Eigenschaft  
 .item auf null von document.selection.textrange Collection  
 bzw. von document.selection.controlRange Collection  
 .type auf "none" von document.selection

**4.3.2.2.4.3.37.1. document.selection.controlrange Collection des Internet Explorer**

Feld der Zeiger aller Control-Elemente, die in der **aktiven** Selektion per document.selection Objekt markiert wurden  
**nur für body Objekt**  
 ein Objekt controlrange existiert nicht  
 ab IE 5.x

**Syntax:**

[ var ZeigerAufFeld = ] document.selection.controlrange  
 [ var ZeigerAufFeldElement = ] document.selection.controlrange [Index]

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

ZeigerAufFeldElement



ist null, wenn Feldelement nicht vorhanden  
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection  
**aus diesen** Script-Objekten geliefert

**Eigenschaften:**

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

**Methoden:**

.add() ein bereits erzeugtes Element einer Collection hinzufügen  
hinzufügen erst nach dem kompletten Laden des Dokumentes  
Element erzeugen per Methode .createElement()  
.execCommand() Kommando ausführen z.B. im aktuellen Dokument  
in aktueller Selektion  
im aktuellen Bereich  
erst nach dem kompletten Laden des Dokumentes zulässig  
Hinweis: Selektion = Markierung z.B. von Textbereich (Block)  
Control = Element zur Steuerung analog zum HTML-Element (Tag)  
Input-Control = Element mit Eingabeeigenschaft  
.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 !!!  
.queryCommandEnabled() prüfen ob Kommando ausführbar ist  
.queryCommandIndeterm() prüfen ob Kommando-Status bestimmbar ist oder nicht  
.queryCommandState() Status des aktuellen Kommando ermitteln: ob ausgeführt wurde oder nicht  
.queryCommandSupported() prüfen ob Kommando im aktuellen Bereich unterstützt wird  
.queryCommandValue() Wert eines Kommandos liefern  
.remove() ein Element aus einer Collection entfernen  
.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird  
Objekt muss an sich schon renderbar sein  
.select() Selektion eines Textranges (Textbereich, Objekt textrange) oder ControlRange (Control-Elemente)  
nur unter Windows 32-Bit  
siehe Objekt document.selection  
Methoden .createTextRange() .createControlRange() und .createRange()

**4.3.2.2.4.3.37.2. document.selection.textrange Collection des Internet Explorer**

Feld der Zeiger aller textrange Objekte der **aktiven** Selektion  
ab IE 5.5

Elternobjekte mit Textrange sind  
body Objekt  
button Objekt  
textarea Objekt  
input text Objekt  
selection Objekt (nur wenn ein Text selektiert wurde (mit oder ohne HTML))  
wobei diese weitere HTML-Elemente enthalten können, die ebenfalls Textbereiche besitzen können

**Syntax:**

```
[ var ZeigerAufFeld = ] document.selection.textrange
[ var ZeigerAufFeldElement = ] document.selection.textrange[Index [, SubIndex] ]
```

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

SubIndex optional  
nur kodieren wenn Index ein String ist  
Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement  
ist null, wenn Feldelement nicht vorhanden  
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection  
**aus diesen** Script-Objekten geliefert

**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

**4.3.2.2.4.3.38. span Objekt**

Das DIV- bzw. SPAN-Objekt ähneln sich sehr stark.. SPAN wird als üblicherweise als Textcontainer benutzt.

Beispiel: <SPAN STYLE="color: blue">blaues</SPAN>Wort



DIV bzw. SPAN können im Elternobjekt (Dokument im Fenster oder übergeordneter DIV bzw. SPAN) erzeugt werden,

als HTML-Element üblicherweise im BODY-Teil des Dokumentes  
per Script durch z.B. `document.write("<DIV ..... >...");`

Die Anzeige (das Rendern) des DIV bzw. SPAN erfolgt erst mit dem Parsen des Ende-Tags, also `</DIV>` bzw. `</SPAN>`. Danach können Komponenten zur Laufzeit nachträglich verändert werden, die dann aber nur z.T. sofort sichtbar werden (teilweise erst nach dem Reload des betreffenden Dokumentes).

NS und IE können das DIV- und SPAN-Objekt erzeugen, implementieren und rendern diese aber verschiedenartig.  
Beispiel:

Nur der Internet Explorer erzeugt automatisch ein BODY-Objekt, wenn dieses im Dokument nicht kodiert wurde, weil z.B. sämtliche HTML-Elemente im HEAD des Dokumentes per Script erzeugt wurden.

Der NS verlangt immer einen BODY-Teil und ignoriert o.g. Script-Anweisungen im HEAD des Dokumentes, solange der BODY des Dokumentes nicht komplett geparkt wurde (`</BODY>` noch nicht erkannt wurde). Es können also nur nachträglich neue HTML-Elemente per Script im HEAD des Dokumentes erzeugt werden. Scriptaufrufe aus dem BODY-Teil werden mit dem BODY geparkt, egal ob die Scripte im HEAD oder BODY liegen.

Unter NS 6.x wird anstelle von DIV- bzw. SPAN-Objekt das LAYER-Objekt favorisiert. Ab NS 6.x wird das Layer-Objekt radikal nicht mehr unterstützt: Es sind für die Ebenendarstellung (überlappende Objekte) keine Layer mehr sondern z.B. DIV oder SPAN verwendbar.

**Zugriff:**

Die übliche Zugriffsweise ist über das ID-Attribut (Eigenschaft `.id`) des DIV bzw. SPAN.

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
function HandlerFuerOnMoveStart()
{
    // anstelle des ID vom SPAN falls mehrere bewegbare Objekte vorhanden sind
    var ZeigerAufObjektMitEvent = event.srcElement;

    ZeigerAufObjektMitEvent.style.backgroundColor = "green";
    ZeigerAufObjektMitEvent.innerText = "SPAN wird bewegt ";
}

function HandlerFuerOnMove ()
{
    ID_Span1.innerHTML = event.srcElement.offsetLeft;
    ID_Span2.innerHTML = event.srcElement.offsetTop;
}

function HandlerFuerOnMoveEnd()
{
    // anstelle des ID vom DIV falls mehrere bewegbare Objekte vorhanden sind
    var ZeigerAufObjektMitEvent = event.srcElement;

    ZeigerAufObjektMitEvent.style.backgroundColor = "red";
    ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
}

// 2-D Positionierung einschalten
document.execCommand("2D-position",false,true);
</SCRIPT>
</HEAD>
<BODY onmovestart="HandlerFuerOnMoveStart();"
onmove="HandlerFuerOnMove();"
onmoveend="HandlerFuerOnMoveEnd();"
>
offsetLeft = <SPAN ID="ID_Span1"></SPAN>
<BR>
offserTop = <SPAN ID="ID_Span2"></SPAN>
<BR>
<DIV CONTENTEDITABLE="true">
<DIV STYLE=
"position:absolute;width:300px;height:100px; background-color:red;"
>
```



bewegbarer DIV

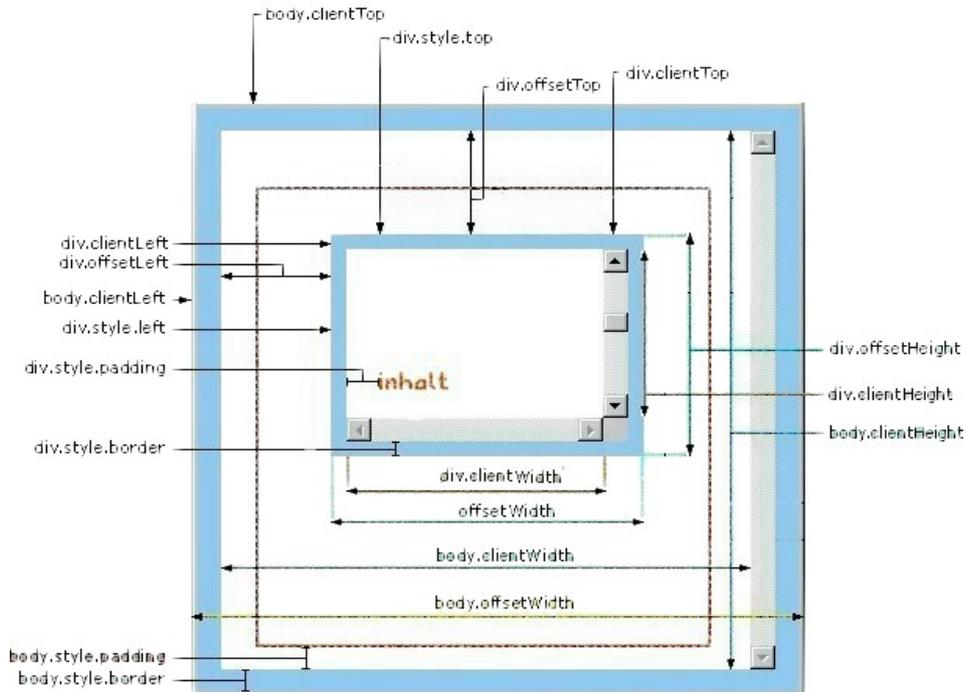
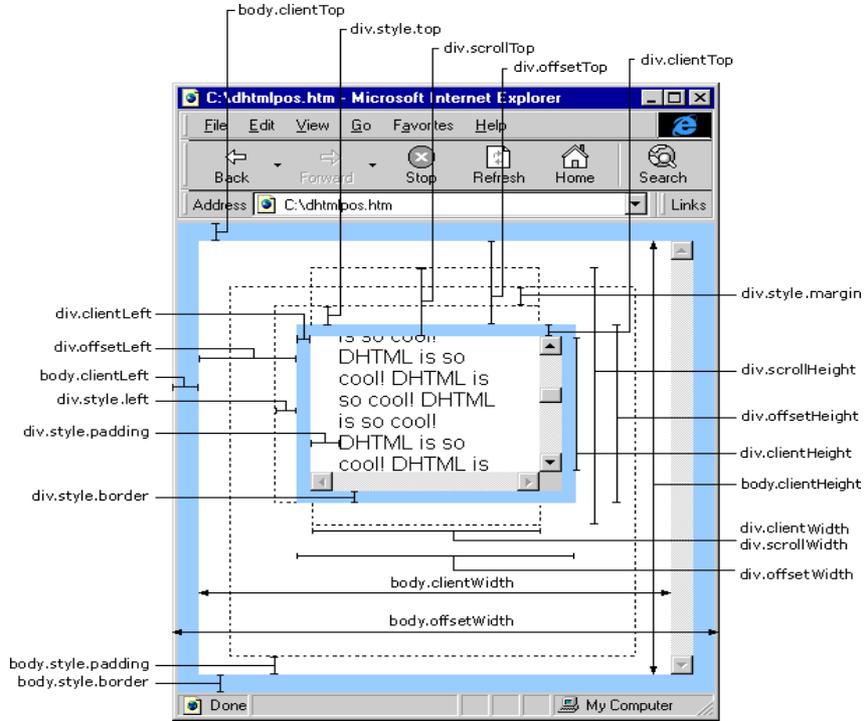
```

</DIV>
</BODY>
</HTML>

```

**span Objekt des Internet Explorer:**

Bezüglich zeitsteuernder Eigenschaften/Methoden werden auch Behavior style.time2 und Objekt currTimeState ab IE 6.x benutzt. Wichtige Eigenschaften im IE:





	siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia	Objekt ist HTML-Media-Objekt
	siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID)
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEditable	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
STYLE	direkt im HTML-Element kodierter Style (Inline-Style) Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster	Synchronisierung der Animation des im Container liegenden Elementes auf Timeline (Container ist Master in der Synchronisierung), wobei ein Master <b>nur</b> genau ein zu synchronisierendes Element haben darf (Eindeutigkeit). nur sinnvoll bei Zwangssynchronisierung (siehe .syncBehavior) ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf! siehe Objekt currTimeState und Behavior .style.time2



	siehe .syncTolerance und .syncBehavior
.systemBitrate	wird hier nicht erklärt
.systemCaptions	wird hier nicht erklärt
.systemLanguage	Sprache festlegen für das Objekt
.systemLanguage	Standard-Sprache des Betriebssystems (Sprache der Installation) per navigator Objekt
.systemOverdubOrSubtitle	wird hier nicht erklärt
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA  Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer	Typ der Timeline des Objektes siehe Objekt currTimeState und Behavior .style.time2
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
<b>Methoden:</b>	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endtag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.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 <b>Zufallsfolge</b> , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.clearAttributes()	alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert
.click()	simuliert einen Klick auf das Element und löst onclick-Event aus manipuliert nicht den Focus
.cloneNode()	Objekt klonen und Referenz des erzeugten Klone liefern DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM- Objektes im Hauptspeicher außerhalb des DOM)
.componentFromPoint()	Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt auch für CSS-Layout onmouseover-Event hat <b>nicht die Pixelgenauigkeit</b> wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.contains()	prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist DOM nicht geändert



.detachEvent()	Abschalten des Registrieren eines Events durch Eventhandler wobei Registrierung mit Methode .attachEvent() aktiviert wurde Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das <b>nicht</b> behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)
.doScroll()	Click auf Scrollbar simulieren Achtung: Scrollelemente müssen bereits vorhanden sein !
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert
.getAttribute()	Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert
.getAttributeNode()	Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft. Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt Wert des Attributes wird somit über die Referenz laut DOM erreichbar DOM nicht geändert
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementByName()
.getExpression()	DOM nicht geändert Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)
.hasChildNodes()	prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen <b>nicht</b> ausgeführt eingefügter Code wird <b>nur</b> dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden DOM wird geändert
.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.mergeAttributes()	alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !! DOM wird geändert
.normalize()	Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen
.releaseCapture()	Maus-Überwachung ausschalten für ein Objekt



	Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.
.removeAttribute()	Hinweis: einschalten per Methode .setCapture() entfernen eines per HTML erzeugten Attributes Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst DOM wird geändert
.removeAttributeNode()	entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern DOM wird geändert
.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.removeChild()	Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern Sichtbarkeit erst, wenn Ende-Tag geparkt wurde, also das Dokument neu geladen wurde DOM wird geändert
.removeExpression()	Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient. Ausdruck muss mit der Methode .setExpression() gesetzt worden sein DOM wird nicht geändert
.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5
.setExpression()	Hinweis: ausschalten per Methode .releaseCapture() Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

#### 4.3.2.2.4.3.39. style Objekt des Internet Explorer

ab IE 4.x

z.T. erst ab IE 5.5 bzw. IE 6.x

##### Ansatz:

Aufgrund der HTML-Integration von CSS sollte der Browser CSS unterstützen. Ob diese Unterstützung gegenüber den aktuellen Standards zu HTML und CSS vollständig ist, hängt vom Willen des Browserherstellers ab. Letztendlich wird CSS sinnigerweise als grundlegende Komponente und Eigenschaft aller Objekte integriert. CSS bietet elementare Vielfalt bei der dynamischen Scriptprogrammierung, ist gewöhnungsbedürftig, aber durch Normung universell auf diverse Objekte anwendbar, die das STYLE-Attribut bzw. die Eigenschaft .style unterstützen.

Es sei darauf hingewiesen, dass

die Verwendung von CSS-Klassen, Kodierung von <STYLE> bzw. dem STYLE-Attribut bzw. der Eigenschaft .style nur **scheinbar**

alle synonym sind:

CSS-Klassen im HEAD können dokumentweit verwendet werden.

<STYLE>, STYLE-Attribut und .style sind stets HTML-komponentenbezogen.

der Browser diese Formen intern z.T. verschieden (auch bezüglich der Collectionen) verwaltet.

