

onabort	Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
	Tag
onload	Ereignis ausgelöst, wenn Bild vollständig geladen wurde zu animiertes Bild (Gif-Datei): Ereignis load bei jedem Bildwechsel der Animation ausgelöst es kann also für jedes Bild das Ereignis behandelt werden
	Tag <BODY>, <DIV>, <FRAMESET>,

4.3.2.2.5.12.1.2. Lade-Ereignisse für HTML-Dokument und dessen Elemente (außer Bild)

onload	Ereignis ausgelöst, wenn vollständig geladen wurde HTML-Dokument mit all seinen Elementen jeder Art Framset (innerhalb <FRAMESET> kodieren) DIV Layer
	Tag <BODY>, <DIV>, <FRAMESET>,
onunload	Ereignis ausgelöst, wenn das HTML-Dokument verlassen wird durch Schliessen Browserfenster Wechsel zu anderer Seite auch per Browser-Button window.document.open() window.document.write()
	Tag <BODY>, <FRAMESET>

4.3.2.2.6. window.external Objekt des Internet Explorer

Dieses Objekt betrifft direkt die Privatsphäre des Users und ist mit Sorgfalt zu verwenden ! Jeder Interessenkonflikt zwischen dem Webseitenanbieter, der dieses Objekt nutzt, und dem User ist zu vermeiden. Die Nutzung des Objektes ist dem User transparent zu machen, so dass er die Wahl hat, ob das Objekt wirksam werden soll oder nicht.

Das Objekt ermöglicht den Zugriff auf

- das Kontextmenü
- die Favoritenliste
- den Microsoft Active Desktop (Desktop muss vom User per Desktop-Eigenschaften als Active Desktop eingestellt worden sein)
- die Microsoft Active Channel (Channel muss vom User per Desktop-Eigenschaften als Active Channel eingestellt worden sein)
- das Autocomplete bei Formularen (Autokomplete kann vom User in den Browser-Optionen eingerichtet werden)
- das permanente Speichern von User-Daten
- das Autoscan von Urls
- die Suche in Webseiten
- die Einstellungen zu Sprache, Favoriten, Sicherheit

kompletter Zugriff erst ab IE 5.x

Eigenschaften:

.menuArguments Zeiger auf dasjenige offene Fenster, indem ein Eintrag aus dem Kontextmenü ausgeführt wurde

Beispiel:

```
<SCRIPT LANGUAGE = "JavaScript">
var ZeigeraufWindow = window.external.menuArguments;

// beispielsweise einen selektierten Text im Fenster verarbeiten
var ZeigerAufDokuemntImWindow = ZeigeraufWindow.document;
var SelektionImDokument = ZeigerAufDokuemntImWindow.selection;
var TextBereichImDokument = SelektionImDokument.createRange();
var SelektierterTextImTextBereich = TextBereichImDokument.text;

if (SelektierterTextImTextBereich.length == 0)
{ TextBereichImDokument.text = "Einfuege Text";}
else
{TextBereichImDokument.text = SelektierterTextImTextBereich.toUpperCase();}
</SCRIPT>
```

Methoden:

.AddChannel() Dialogbox zur Channel-Einstellung öffnen um einen Channel zu installieren (Microsoft Active Channel)
erzeugt im Fehlerfall eine Dialogbox und das Event onerror

Beispiel:

```
window.external.AddChannel("http://test /file.cdf");
```

.AddDesktop() eine Webseite oder Bild zum installierten Microsoft Active Desktop hinzufügen
wenn Active Desktop nicht installiert, so passiert nichts

Beispiel:

```
window.external.AddDesktopComponent("http://www.test.de","website",100,100,200,200);
```

.AddFavorite() Dialogbox zum Hinzufügen einer Url in die Favoritenliste für den User öffnen
Syntax:

```
logischer_window_name.external.AddFavorite(Kette1 [, Kette2])
```

Kette1

String

URL des Favoriteneintrages, also diejenige Url
die gebokkmarkt werden soll



Kette2	String freier Vorgabe-Text des Favoriteneintrages wird von Usereingabe in die Dialogbox überschrieben
--------	--

logischer_window_name	Zeiger laut open()
-----------------------	--------------------

liefert nichts

Beispiel:

```
window.external.AddFavorite(location.href, document.title);
```

.AutoCompleteSaveForm() permanentes Speichern von Daten eines Formulars in den AutoComplete-Data Store
Diese Methode ist mit Vorsicht zu geniessen und richtet sich nach dem aktuellen Einstellungen von AUTOCOMplete, die der User in den Browser-Optionen treffen kann.
 Gespeichert werden die Werte von INPUT's im Formular, die das Attribut NAME besitzen, wobei auch INPUT TYPE=password speicherbar ist
 Hinweis: Im Formular-Tag ist ebenfalls NAME zu kodieren
 NAME-Attribut ist auch Voraussetzung für das Senden von Formulardaten an den Server der Webseite
 AutoComplete-Data Store kann vom User gelöscht werden (auch komponentenweise).

Beispiel:

```
<SCRIPT>
function Speichern()
{
    // erst speichern
    window.external.AutoCompleteSaveForm(ID_Formular);
    // dann löschen
    ID_Formular.ID_Input1.value="";
    ID_Formular.ID_Input2.value="";
}
</SCRIPT>
<FORM NAME="ID_Formular">
  Dieser Text wird gespeichert:
  <INPUT TYPE="text" NAME="ID_Input1">
  Dieser Text wird nicht gespeichert:
  <INPUT TYPE="text" NAME="ID_Input2" AUTOCOMplete="off">
</FORM>
<INPUT TYPE=button VALUE="Speichern" onclick="Speichern()">
```

.AutoScan() Url einer beliebigen Webseite festlegen, die als Fehlermeldung vom Autoscan angezeigt wird
 Webseite wird aufgerufen, wenn Autoscan nicht erfolgreich war
 erklärt den Misserfolg
muss immer erreichbar, also anzeigbar sein
 Standardwebseite auf Festplatte des User-PC vorhanden (falls der User diese nicht manuell gelöscht hat)
 Autoscan: Suchen nach einer anzuzeigenden Web-Seite im Internet durch den Browser nach Eingabe der Url (auch mit Autovervollständigung)
 Voraussetzungen für Autoscan:
 es muss aktiv sein
 Autoscan leider nur möglich für Domains mit "www" am Anfang der Url
 Suffixe der Url laut Registry-Eintrag
 HKEY_LOCAL_MACHINE\software\microsoft\internet explorer\main\urltemplate
 (standardgemäß steht dort .com, .org, .net, und .edu)
 Suche in der Reihenfolge der Einträge laut dem Registry-Eintrag
 Hinweis: Der Registry-Eintrag ist manuell änderbar:
 Bsp.: .de als Eintrag hinzufügen an gewünschte Position
 Syntax:
 logischer_window_name.external.AutoScan(Kette1, Kette2 [, Kette3])

Kette1	String Url-Teil nach www und vor dem Suffix wie z.B. .com, .org, .net, oder .edu (siehe oben)
--------	---

Kette2	String Url der Webseite, die bei Fehler angezeigt werden soll Webseite kann im Internet liegen muss immer erreichbar sein Standard: IE-Fehlermeldung-Webseite lokal auf der Festplatte des User-PC
--------	---



Kette3 String
 Referenz auf dasjenige Fenster, in dem die Webseite der Fehlermeldung angezeigt wird
 "_parent"
 "_self" (Standard)
 "_top"
 "_main"

logischer_window_name Zeiger laut open()
 liefert leider nichts

Beispiel:
 window.external.AutoScan("test","error.htm","_main");
 für **www.test.xxx** mit xxx als Domain-Suffix laut Registry-Eintrag (siehe oben)

.ImportExportFavorites() Import und Export der Favoritenliste aus dem/ in das Netscape-Bookmark-Format
 User muss Import/Export bestätigen per Dialogbox
 es wird HTTP benutzt
 Ziel-bzw. Quellort: Es wird durch Import immer hinzugefügt
 Es wird nach Export nicht gelöscht.
 Syntax:
 logischer_window_name.external.ImportExportFavorites(Wert, Kette)

Wert true., so importieren
 false, so exportieren

Kette String
 Pfad des Ziel- (bei Export) bzw. Quellortes (bei Import)
 auf der User-PC-Festplatte
 bzw. im User-Netzwerk
 Server-Url auch möglich
 Url muss mit http:// beginnen
 Ort muss bereits vorhanden sein (kein automatisches Anlegen)
 wenn Leerkette, so wird automatisch eine Dialogbox für User-Auswahl geöffnet

logischer_window_name Zeiger laut open()
 liefert nichts

Beispiel:
 window.external.ImportExportFavorites(true,"http://www.test.com");

.IsSubscribed() auf Verfügbarkeit der Unterschrift zu einer Microsoft Active Channel-Datei prüfen
 (Channel Definition Format-Datei mit Dateisuffix ist *.cdf)
 nur für Dateien in gemeinsamer Domain, also nicht domain-übergreifend
 liefert immer Scriptfehler, wenn Domain-Wechsel durch Url der CDF-Datei erzeugt wird

.NavigateAndFind() Webseite laden, dann Text dort suchen und wenn gefunden, so diesen in der Webseite markieren
 (analog zu CTRL-F in der Webseite für Suchen und Finden)
 auch Suche in Frames bzw. Unterseiten der Webseite
 Syntax:
 logischer_window_name.external.NavigateAndFind(Kette1, Kette2, Kette3)

Kette1 String
 Url der Webseite oder lokaler Pfad
 also z.B. auch http:// oder c:\

Kette2 String
 zu findender Text

Kette3 String
 Referenz auf Frame/Fenster in der Webseite laut URL
 Leerkette möglich

logischer_window_name Zeiger laut open()
 liefert nichts

Beispiel
 <HEAD>
 <SCRIPT>
 function Suchen()
 {



```

window.external.NavigateAndFind( "http://www.test.de/file.htm",
                                ID_Select.options[ID_Select.selectedIndex].text,
                                ""
                                );
}
</SCRIPT>
</HEAD>
<BODY>
    bitte selektieren
    <SELECT ID="ID_Select" onchange="Suchen()">
        <OPTION>Eintrag1
        <OPTION>Eintrag2
    </SELECT>
</BODY>

```

.ShowBrowserUI() öffnen einer IE-Dialogbox (UI = User-Interface) bezüglich
 Spracheinstellungen
 oder Favoritenverwaltung
 oder Sicherheitseinstellungen
 Inwieweit eine geöffnete Dialog-Box dann per Windows Script Host z.B. per VBScript programmierbar ist, wird hier nicht betrachtet.
 Ein User, dem diese Boxen durch eine Webseite geöffnet werden, sollte stets misstrauisch ein.

Beispiel:

```
<BUTTON onclick="window.external.ShowBrowserUI('LanguageDialog', null)">Spracheinstellungen</BUTTON>
```

4.3.2.2.7. window.history Objekt (history Collection)

Dieses Objekt ist eigentlich eine Collection.

Erzeugung: keine, da vom Browser erzeugt

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

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

index: ab 0
Nummer des History-Eintrages im Dokument

Eigenschaften (Auswahl):

- .current Zeichenkette mit Url der aktuellen Seite
nur NS ab 3.x mit signiertem Script
- .length Anzahl der History-Einträge
nur lesen
- .next Zeichenkette mit Url der nächsten Seite (falls vorhanden)
nur NS ab 3.x mit signiertem Script
- .previous Zeichenkette mit Url der vorhergehenden Seite (falls vorhanden)
nur NS ab 3.x mit signiertem Script

Methoden (Auswahl):

- .back() vorhergehende Seite laut previous laden
- .forward() nächste Seite laut next laden
- .go(position) Position der zu ladenden Seite innerhalb der History
0 entspricht erste geladene Seite
history.length-1 entspricht aktuelle Seite
Bsp.: onClick="window.history.go(0)

history Object des Internet Explorer:

ist eigentlich eine Collection:

beinhaltet die Urls, die der User auf dem Client besucht hat (entspricht einem Verlauf des Surfens des Users)
allerdings sortiert nach Urls und dort nach zugehörigen Unter-Urls (nicht sequentiell)

Urls können sequentiell-relativ zur aktuell besuchten Seite nur durch die Methoden des history-Objektes verwendet werden
wobei Sprünge möglich sind

anhand eines Index 0 = aktuelle Seite
< 0 zuvorliegende Seiten
> 0 nachliegende Seiten

nur nutzbar wenn aktuelle Seite (Index 0) durch Rückwärtsgehen
in der History eingestellt wurde

Alle Seiten werden immer aus dem Browser-Cache gelesen, wenn
im HTML-Dokument nichts anderes vereinbar wurde
in den Internet-Optionen des Browsers nichts anderes vereinbart wurde

Eigenschaften:

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



Methoden:

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

Beispiel: Reload des Dokumentes nach Resize des Browserfensters

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function neuLaden()
    {
        if (document.all)
        {history.go(0);}
    }
// -->
</SCRIPT>
</HEAD>
<BODY onResize=neuLaden();">
</BODY>
```

4.3.2.2.8. window.location Objekt

Erzeugung: keine, da vom Browser erzeugt

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

sonst auch möglich
location.eigenschaft
location.methode()

Eigenschaften (Auswahl):

.hash	entspricht #hashtext zum Anspringen eines Ankers hier den Ankernamen mit vorgesetztem # ablegen
.host	entspricht hostname:port lesen und schreiben
.hostname	entspricht nur hostname lesen und schreiben
.href	gesamter Url (kompletter Url) zum Anspringen eines Ankers
.pathname	hier den Ankernamen ohne vorgesetztem # ablegen entspricht aus url /dateiname bzw. /dateiname#hashtext bzw. /dateiname?searchtext
.port	lesen und schreiben entspricht port
.protocol	lesen und schreiben entspricht Protokoll mit Doppelpunkt z.B. "http:" lesen und schreiben
.search	entspricht ?searchtext lesen und schreiben

Methoden (Auswahl):

.assing(url)	HTML-Doument laden, nur IE ab 4.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 HistoryEintrag zur aktuellen Seite überschreiben (keinen neuen bilden) --> BACK-Button wechselt danach nicht zur überschriebenen Seite, da diese in der History nicht mehr bekannt ist

location Objekt des Internet Explorer:

Container der Informationen der aktuellen Url

Eigenschaft .href enthält die komplette Url-Information und ist die Standard-Eigenschaft (entspricht HREF-Attribut):

location='http://www.test.de' ist gleichwertig mit location.href='http://www.test.de'

Bsp.: Quelltext anzeigen: Öffnet lokalen Standard-Text-Editor z.B. Notepad

```
<INPUT TYPE="button" VALUE="Quelltext ansehen" onclick="window.location = 'view-source:' + window.location.href">
```

Eigenschaften:

.hash	Teil des Wertes der Eigenschaft .href also Teil der Url als Anker
-------	---



Teil folgt direkt hinter dem Nummernkreuz # und ohne Nummernkreuz

.host **Hostname** und **Port** einer Location oder Url in der Form "**hostname:port**"

.hostname **Hostname** einer Location oder Url in der Form "**hostname:port**"
kann Domain oder IP sein

.href Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag)
siehe auch Eigenschaften .rel und .rev

.pathname Datei und Pfad eines Objektes

.port **Port** einer Location oder Url in der Form "**hostname:port**"
basierend auf dem aktuellen **Protokoll** laut Eigenschaft .protocol

z.B.	Standard-Ports	Protokoll
	21	FTP
	80	HTTP

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

.search Teil des Wertes des Eigenschaft .href

Teil folgt direkt dem Fragezeichen
wird als Querystring oder Formdata bezeichnet
hat nichts mit der Suche auf Webseiten zu tun

Hinweis: Fragezeichen-Anhang an der HREF dient zur Übergabe von String-Werten einer Webseite an eine andere:
Quellseite besitzt HREF mit " ...?....."
ruft Zielseite mit diesem HREF auf
Zielseite wurde von Quellseite aufgerufen
liest den Teil von HREF hinter dem ? als Zeichenkettendaten

Methoden:

.assign() neues Dokument zuweisen und laden
im Verlauf (History) wird ein neuer Eintrag hinzugefügt (history Objekt)
Altes Dokument ist per Vorwärts- und Zurück-Button einstellbar.

.reload() aktuelles Dokument neu laden

.replace() neues Dokument zuweisen und laden
im Verlauf (History) wird der Eintrag des alten, vorherigen Dokumentes durch den Eintrag des neuen zu ladenden Dokumentes ersetzt.
Altes Dokument ist damit **nicht** mehr per Vorwärts- und Zurück-Button einstellbar.

4.3.2.2.9. window.navigator Objekt

Container der Informationen über den Browser, Betriebssystem, regionale Einstellung des Users, CPU und Java-Maschine.

Die Informationen sind z.T. browserherstellerspezifisch und können sich von Version zu Version ändern. Es ist daher zu empfehlen, bei einer Browserprüfung auch Javascript zu verwenden und browsersepezifische Funktionen aufzurufen. Deren Rückkehrcode liefert den Beweis, ob der gewünschte Browser auch benutzt wird vom User (siehe DOM). Ein typisches Verhalten zeigt der Opera-Browser: Er gibt sich als Internet Explorer aus.

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

navigator.eigenschaft
navigator.methode()

4.3.2.2.9.1. navigator Objekt im Netscape

4.3.2.2.9.1.1. Eigenschaften

alle Eigenschaften sind nur lesbar

.appName Browser-Codename z.B. "Mozilla"

.appName Browsername z.B. "Netscape"

.appVersion Plattform und Browserversion
Aufbau: versionsnummer (system; land)
Bsp: "3.0B2 (Win16;I)"
"4.1 (WinNT;I)"
Haupt-Versionsnummer ermittelbar per
parseFloat(navigator.appVersion)
z.B. 4 des IE 4.1
Minor-Version --> siehe .appMinorVersion

.cookieEnabled wenn true so Cookiesverwaltung aktiv
wenn false so Cookies abgeschaltet

.language Browser-Sprachversion z.B. "en" "de"

.mimeTypes Zeiger auf Feld aller im Browser verfügbaren Mimetypen
Index kann Typ als Zeichenkette sein z.B. "image/png"

.oscpu nur NS 6.x
in der Dokumentation nicht weiter behandelt

.platform Browser-Betriebssystem
z.B. "Win32" oder "Win16"

.plugins Zeiger auf Feld aller im Browser verfügbaren Plugins

.product nur NS 6.x
in der Dokumentation nicht weiter behandelt

.productSub nur NS 6.x
in der Dokumentation nicht weiter behandelt



.securityPolicy	Sicherheitseinstellungen vom Netscape nur NS 6.x in der Dokumentation nicht weiter behandelt
.userAgent	Browser-Codename UND -Version Bsp: "Mozilla / 3.0B2 (Win16;1)"
.vendor	nur NS 6.x in der Dokumentation nicht weiter behandelt
.vendorSub	nur NS 6.x in der Dokumentation nicht weiter behandelt

4.3.2.2.9.1.2. Methoden

.javaEnabled()	liefert true, wenn Browser Java kann und Java nicht abgeschaltet ist
.preference()	Benutzereinstellungen des Browsers verändern benötigt Privileg-Manger
.savePreferences()	aktuelle Benutzereinstellungen des Browsers sichern benötigt Priveleg-Manger

4.3.2.2.9.1.3. navigator.plugins Collection des Netscape

Feld der Zeiger aller Plugins

Feldeintrag ist true, wenn Plugin im Feld vorhanden ist, also der Browser das Plugin besitzt.

Erzeugung:

keine, da vom Browser bereitgestellt

Zugriff:

navigator.plugins[index].eigenschaft	
navigator.plugins[index].methode()	
index:	Zeichenkette z.B. "Shockwave" Integer Index ab 0 muss in [] kodiert sein

Beispiel: Auf Adobe Acrobat-Reader-Plugin prüfen

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

Beispiel: Alle Plugins auflisten

```
<BODY>
  <SCRIPT LANGUAGE="JavaScript1.2">
  <!--
    for (i=0; i < navigator.plugins.length; i++)
    {
      document.writeln(navigator.plugins[i].name);
      document.writeln(navigator.plugins[i].description);
      document.writeln(navigator.plugins[i].filename);
    }
  //-->
</BODY>
```

Eigenschaften:

.description	Zeichenkette der Plugin-Kurzbeschreibung als String
.filename	Plugin-Dateiname als String
.length	Anzahl der Plugins, ab 1
.mimeTypes	Zeiger auf navigator.mimeTypes Collection Beispiel für Zeigerbezug: navigator.plugins.mimeTypes[index].eigenschaft
	mit index String Mimetyp Integer als Index ab 0 muss in [] kodiert sein
.name	Name des Plugin als String

Methoden:

.refresh()	Funktionswert true HTML-Dokument mit seinem per EMBED eingebetteten Objekten neu laden falls Plugin noch nicht installiert ist, so es dabei installieren false kein Refresh
------------	--



.taintEnabled() Data-Tainting (Daten verwerfen) per navigator Objekt

4.3.2.2.9.1.4. navigator.mimeTypes Collection

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

Ein Plugin wird einem Mimetyp zugeordnet.

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

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

Feldelement:

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

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

navigator.mimeTypes[mime_typ].eigenschaft

mime_typ: String, der als Feldindex dient
z.B. "text/html"
muss in [] kodiert sein

Beispiel 1: Feld auslesen

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

Beispiel 2: Beschreibung zum Plugin anzeigen

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

Beispiel 3: Daten an Plugin für VRML-Dateiverarbeitung übergeben

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

Beispiel 4: Suffix prüfen

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

Eigenschaften:

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

Methoden:

keine

4.3.2.2.9.2. navigator Objekt im Internet Explorer

4.3.2.2.9.2.1. Eigenschaften

- .appName Codename des Browsers per navigator Objekt
- .appMinorVersion Update der Browserversion per navigator Objekt
- .appName Name des Browsers per navigator Objekt
- .appVersion Betriebssystem-Plattform und Browserversion per navigator Objekt
- .browserLanguage Sprache des Betriebssystems und **nicht** des Browsers (Browsersprache kann andere sein als die des Betriebssystems z.B. französischer Browser auf deutschem Windows)



.cookieEnabled	Cookiesstatus lesen aber nicht verändern per navigator Objekt
.cpuClass	CPU-Klasse per navigator Objekt
.mimeTypes	Zeiger auf Collection mimeTypees
.onLine	Onlinestatus des Browsers ermitteln per navigator Objekt keine Überprüfung auf Netzwerkverbindung
.platform	Name des Betriebssystems per navigator Objekt
.plugins	Zeiger auf Collection plugin
.systemLanguage	Standard-Sprache des Betriebssystems (Sprache der Installation) per navigator Objekt
.userAgent	Browser-Codename und -Version per navigator Objekt
.userLanguage	vom User eingestellte Sprache des Betriebssystems (nicht die Sprache der Installation, sondern die regionale Sprache des Betriebssystems)
.userProfile	Zeiger auf Collection userProfile

4.3.2.2.9.2.2. Methoden

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

4.3.2.2.9.2.3. navigator.mimeTypees Collection (auch bei IE 6.x)

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

Ein Plugin wird einem Mimetype zugeordnet.

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

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

Feldelement:

Reihenfolge der Feldelemente ist browserspezifisch.

Es besteht die Möglichkeit, dass Mimetypeen, die keine Plugins repräsentieren, ebenfalls mit in der Collection liegen

Bsp.: "image/jpeg".

Beispiel für einen Mimetype für echten Plugin: "application/pdf" für Adobe Acrobat-Plugin

Beim Internet Explorer muss diese Collection nicht komplett gefüllt sein, kann aber (probieren!).

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

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

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

navigator.mimeTypees[mime_typ].eigenschaft

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

Beispiele 1: Feld auslesen

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

Beispiel 2: Beschreibung zum Plugin anzeigen

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

Beispiel 3: Daten an Plugin für VRML-Dateiverarbeitung übergeben

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

Beispiel 4: Suffix prüfen

```
if (navigator.mimeTypees["image/jpeg"])
```



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

Eigenschaften:

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

Methoden:

keine

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

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

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

Das Ansprechen von Plugins kann über die Collection navigator.mimeTypes per Eigenschaft .enablePlugin 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. Beim Internet Explorer muss diese Collection nicht komplett gefüllt sein, kann aber (probieren!).

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

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

Syntax:

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

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

ZeigerAufFeldElement
                    ist null, wenn Feldelement nicht vorhanden
```

Beispiel: Auf Adobe Acrobat-Reader-Plugin prüfen

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

Eigenschaften:

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

Methoden:

.item()	Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern außer bei Formular mit <INPUT TYPE=image ...> da dafür die children-Collection verwendet werden muss !!!
.namedItem()	Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern
.tags()	Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern siehe tags Collection des DOM

4.3.2.2.9.2.5. navigator.userProfile Objekt des Internet Explorer

Objekt zur nur lesenden Verwaltung von User-Profile-Informationen (vCard-Daten) ab IE 4.x

Die vCard-Daten werden beim Internet Explorer im Objekt document.form verwertet, wenn im Formular das AutoComplete aktiviert ist.



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

Hinweis: AutoComplete betrifft auch Werte laut VALUE-Attribute von Textfeldern im Formular, aber NUR, wenn diese Felder auch das NAME-Attribut besitzen. VALUE-Werte werden automatisch für die automatische Wiederverwendung im Browser gespeichert (auch bei Password-Feldern !!!). Daher sollte die Nutzung von AutoComplete reiflich überlegt sein.

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

Beispiel 1:

```
// Request Queue füllen

//          es soll der Wert zu "vcard.displayname" ermittelt werden
navigator.userProfile.addReadRequest("vcard.displayname");
//          es soll der Wert zu "vcard.gender" ermittelt werden
navigator.userProfile.addReadRequest("vcard.gender");

// Datenermitteln per Queue
navigator.userProfile.doReadRequest(12, "Daten von unbekannt verwendet");

// Queue leeren
navigator.userProfile.clearRequest();
```

Beispiel 2:

```
// lesen vom Wert zu "vcard.displayname"
var Name = navigator.userProfile.getAttribute("vcard.displayname");

// lesen vom Wert zu "vcard.gender"
var Gender = navigator.userProfile.getAttribute("vcard.gender");
```

Eigenschaften:

keine

Methoden:

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

4.3.2.2.10. window.popup Objekt des Internet Explorer

Objekt für Popup-Fenster z.B. für Dialog, Meldungen, Tooltip
ab IE 5.5

Diese Fensterform ist eine stark reduzierte Instanz des window Objektes, wobei der Inhalt per Script kodiert werden muss (z.B. HTML-Tags).



Ein PopUp-Fenster

wird aktuell, wenn es angezeigt wird
 wird automatisch geschlossen, sobald ein anderes Objekt den Fokus erhält, also aktiv wird
 z.B. durch User-Klick außerhalb des PopUp-Fensters

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

Diese fortlaufenden Änderungen muss der Programmierer in Erfahrung bringen.

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

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

Prinzipielle Lizenzprobleme für den Programmierer

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

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

Ein Windows mit Vorversion zum Servicepack bedarf einer anderen Lizenz.

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

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

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

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

Abänderungen wegen Sicherheitspatches der jeweiligen Windows-Versionen

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

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

Im Rahmen der Sicherheitspatches ist es Microsoft sogar gelungen, Webseiten, die den MS-Encoder zur Komprimierung von HTML- und JScript-Code nutzen, schlagartig unnutzbar zu machen: Ein Bug in einem Patch zu Windows XP - Q918899 Das Patch verursacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit Kompression genutzt wird z.B. bei
 onclick-Handler auf IMG
 klick ins Fenster per aktivem Popup

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

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

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

Patch Q918899 für

Windows XP SP1Download für jedermann bereitgestellt

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

Microsoft explizit die User registriert haben will, bei denen das

Patchproblem auftritt (User muss sich Telefonnummer besorgen)

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

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

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

IE-Seiten mit komprimierten Code dauerhaft nicht nutzen können.

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

Abänderungen wegen Browser-Inkompatibilität*Popublocker-Fehler*

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

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

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

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

Bedingung:

Scriptfehleranzeige ist erlaubt im IE 7

Popublocker ist im IE abgeschaltet

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

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

beide (Register) liegen in einer gemeinsamen IE-Instanz

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

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

Der Popublocker für die leere Seite verursacht den Programmfehler im Dokument mit .show(). Es wird folgende



Meldung angezeigt (in der Informationsleiste):

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

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

Der Popublocker hat ein Popupfenster geblockt. Sie können den Popublocker deaktivieren oder Popups temporär zulassen, indem Sie auf die Informationsleiste klicken.

Die Realität zur obigen Meldung ist völlig anders:

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

Popublocker einschalten
weitere Informationen

jedoch keine Möglichkeit wie laut Bedeutung

Damit gilt: Der abgeschaltete Popublocker ist in Wirklichkeit aktiv.

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

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

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

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

Der Popublocker ist nicht als Filter aufgesetzt sondern reingestrickt worden.

Der Popublockfehler verändert die Eventverwaltung:

Es werden u.a. ignoriert

onfocus
onblur
onfocusin
onfocusout

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

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

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

focus-Methode beim IE 7

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

Hinweis:

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

animierte Gif (mit Timer)

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

Abänderungen wegen Rechtstreitigkeiten von Microsoft mit Fremdanbietern

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

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

Es wurde das Eventmodell gleichzeitig geändert:

Es werden alle Events solange unterdrückt, bis der User die Sprechblase geklickt hat.
Das Klicken muss auf das Objekt im Sprechblasenrahmen erfolgen, der so groß ist, wie die Dimension, in der gerendert wurde.
Es muss also ERST per Mausclick das Control aktiviert werden, ehe das Control klickbar und damit die Eventsteuerung aktiviert ist.



Ein Control, das programmtechnisch zwar was rendert, aber ansonsten ohne sichtbare programmtechnisch startet, muss ebenfalls geklickt werden, obwohl es bereits läuft und es nichts zu klicken gäbe (wenn keine Eventsteuerung eingebaut wurde).

Wegen blockierter Eventsteuerung ist also die Sprechblase z.B. nicht automatisch klickbar.

Die Eventauslösung per nicht-objekteigenen Eventhandler, der für das Objekt per fireEvent() ein Event auslöst, ist solange blockiert, bis der User die Sprechblase geklickt hat.

style.visibility='hidden' wird ignoriert

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

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

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

Abänderung wegen Abschaltungen

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

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

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

ActiveX-Controls und Unterstützung/Verbot 20041215

erlaubt sind noch

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

http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp(http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp)

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

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

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

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

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

• http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp
(http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp)

verboten sind

Datenbindung RDS {BD96C556-65A3-11D0-983A-00C04FC29E36} {BD96C556-65A3-11D0-983A-00C04FC29E33} Die RDS (Remote Data Service) Steuerelemente ermöglichen dem Browser, client-basierte



SQL Abfragen an einen Webserver zu stellen. Inzwischen wurde RDS jedoch durch neuere Standards wie SOAP abgelöst, von einer weiteren Verwendung von RDS wird daher abgeraten. Weitere Informationen:• 184375 (<http://support.microsoft.com/kb/184375/DE/>) - Sicherheitsaspekte bei RDS 1.5, IIS 3.0 oder 4.0 und ODBC

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

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

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

Internet Explorer

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

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

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

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

Media Player / Windows Media Player {22D6F312-B0F6-11D0-94AB-0080C74C7E95} Dies ist das Steuerelement für Windows Media Player version 6.4 und war Installationsbestandteil bis einschließlich Windows Media Player Version 8. Ab Windows Media Player 9 wurde diese ClassID durch die neue ClassID {6BF52A52-394A-11D3-B153-00C04F79FAA6} abgelöst, deren Verwendung stattdessen empfohlen wird. Ab Windows Media Player Version 9 wird ferner die alte ClassID anhand eines Wrappers automatisch auf die neue ClassID umgeleitet. Die ClassID für Windows Media Player Version 9 ist jedoch nicht in der Liste der vom Administrator genehmigten Steuerelemente enthalten, und muss bei Bedarf manuell hinzugefügt werden.



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

IE Label-Steuerelement

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

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

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

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

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

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

Microsoft Agent Control - Version 1.5 {F5BE8BD2-7DE6-11D0-91FE-00C04FD701A5} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Diese Version des Steuerelements ist jedoch nicht mehr aktuell und wird nicht mehr unterstützt. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • <http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm> (<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

Aktive Inhalte im Internet Explorer

Ab IE 6.0 ist das Blockieren aktiver Inhalte möglich, z.B. als Standardeinstellung. Es wird also dem IE verboten, JScript zu nutzen.



Daher muss mit Start der Webseite auf das Blockieren von Inhalten der Webseite, die auf JScript basieren, aufmerksam gemacht werden. Bleibt die Blockierung aktiv, so muss die Webseite ALLE Elemente, die per Script angesteuert werden, inaktiv machen: Am besten garnicht erst anzeigen. Oder es wird eine scriptfreie Version der Webseite per <NOSCRIPT> aktiviert, wobei dann Browser vorzuziehbar sind, die z.B. CSS

exakter rendern als der IE (will man keine IE-spezifischen HTML-Elemente verwenden).

Wenn der IE 6.x aktive Inhalte blockiert, wird NOSCRIPT-Tag aktiviert, Ausnahme: Frameset

FRAMESET ist ein aktiver Inhalt:

Da der Frameset anstelle <BODY> kodiert sein muss, gilt:

Alle Tags, die für BODY zulässig sind, werden ignoriert, auch NOSCRIPT.

Wird neben Frameset noch BODY kodiert, so wird Frameset ignoriert.

Die Freigabe der Scriptblockierung erzeugt Ausführung aller Script-Teile inklusive der Eventauslösungen

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

```
function Y_unload(X00){X85[X00].close();}
var X85=new Array();var X86=new Array();
X85[0]=window.open(...);
var X87='parent.Y_unload(0);'; X86[0]=new Function(",X87);
X85[0].document.body.onunload=X86[0];
```

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

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

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

```
<META HTTP-EQUIV="imagetoolbar" CONTENT="no">
unterdrückt NICHT IE-Kontextmenü rechte Maus auf Bild
<META HTTP-EQUIV="site-enter" CONTENT="revealtrans(duration=0.3, transition=12)">
<META HTTP-EQUIV="site-exit" CONTENT="revealtrans(duration=0.3, transition=12)">
```

Achtung: Für das Hinzufügen von Elementen in den BODY (document.body) per DOM-Funktion createElement() MUSS der Body komplett geparkt sein (document.body.readyState == 'complete').

Grund: Es wird standargemäß immer am Ende des BODY angefügt. Für das Hinzufügen nicht an das Ende des BODY muss im HTML-Code ein Platzhalter z.B. DIV kodiert sein,

innerhalb dessen dann die neuen HTML-Elemente erzeugt werden.

Erzeugung:

.createPopup() Popupfenster ohne irgendwelche Fensterelemente öffnen z.B. für Anzeige von Tooltips
Fenster wird mit der Anzeige per Methode .show() zum aktuellen Fenster
erst geschlossen, wenn **anderes Fenster** aktuell wird
z.B. durch Klicken außerhalb des Popupfensters
ab IE 5.5
Syntax: [var Zeiger =] logischer_window_name.createPopup()
Zeiger Referenz auf das Popupfenster (Zeiger entspricht ID),
wird für die Verwendung der Eigenschaften und
Methoden benutzt per
Zeiger.eigenschaft
Zeiger.methode()
logischer_window_name Zeiger laut open()

Zugriff:

zeiger_auf_popup_fenster.eigenschaft
zeiger_auf_popup_fenster.methode
zeiger_auf_popup_fenster laut Erzeugung

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
// erzeugen
var PopupFenster = window.createPopup(); // windows referenziert das aktuelle und instanzierte Fenster
// Körper erzeugen also body Objekt
var PopupFensterKoerper = PopupFenster.document.body;
```



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

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

Beispiel 2 für diverse Meldungsfenster per **jeweiligem** PopUp-Fenster
(es kann immer nur genau 1 Popup-Fenster angezeigt werden):

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

    var PopUpFensterFeld = new Array();

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

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

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

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

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

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

    var PopUpFenster_Breite_Koordinate_Feld = new Array
    (
        80,
        140,
        200
    );

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

    var AnzahlPopUpFenster = PopUpFenster_Hoehe_Koordinate_Feld.length;

    function PopupFensterInstanzieren(NummerDesPopUpFensters, ZeigerAufElternFenster)
        // NummerDesPopUpFensters ab 0
```



```

    {
        PopupFensterFeld[NummerDesPopupFensters] =
            ZeigerAufElternFenster.createPopup();
    }

function PopupFensterFuellen(NummerDesPopupFensters)
{
    // Body des Popup-Fensters gestalten
    var PopupFenster_Body =
        PopupFensterFeld[NummerDesPopupFensters].document.body;

    PopupFenster_Body.style.backgroundColor =
        PopupFenster_HintergrundFarbe_Feld[NummerDesPopupFensters];

    PopupFenster_Body.style.border =
        PopupFenster_RahmenStyle_Feld[NummerDesPopupFensters];

    PopupFenster_Body.innerHTML =
        PopupFenster_Inhalt_Feld[NummerDesPopupFensters];
}

function PopupFensterAnzeigen(NummerDesPopupFensters,
    ObjektZuDemPopupFensterRelativPositioniertIst
)
{
    // Popup-Fenster anzeigen und damit öffnen
    PopupFensterFeld[NummerDesPopupFensters].show(
        PopupFenster_X_Koordinate_Feld[NummerDesPopupFensters],
        PopupFenster_Y_Koordinate_Feld[NummerDesPopupFensters],
        PopupFenster_Breite_Koordinate_Feld[NummerDesPopupFensters],
        PopupFenster_Hoeche_Koordinate_Feld[NummerDesPopupFensters],
        ObjektZuDemPopupFensterRelativPositioniertIst
    );
}

function PopupFensterSchliessen(NummerDesPopupFensters)
{
    var Zeiger = PopupFensterFeld[NummerDesPopupFensters];

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

function Init()
{
    // PopupFenster instanzieren im aktuellen Fenster (window)
    for (var i = 0 ; i < AnzahlPopupFenster; i++)
    {
        PopupFensterInstanzieren(i, window);
        PopupFensterFuellen(i);
    }
}
</SCRIPT>
</HEAD>
<BODY onload="Init();">
    Durch Klick ausserhalb des Popup-Fensters wird dieses auch automatisch geschlossen.
    <BR>
    <INPUT TYPE=button
        VALUE="PopupFenster 0 anzeigen"
        onclick=" PopupFensterAnzeigen(0, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopupFenster 1 anzeigen"
        onclick=" PopupFensterAnzeigen(1, document.body);"
    >
    <INPUT TYPE=button
        VALUE="PopupFenster 2 anzeigen"
        onclick=" PopupFensterAnzeigen(2, document.body);"
    >
    <BR>
    <INPUT TYPE=button
        VALUE="PopupFenster 0 schliessen"
        onclick=" PopupFensterSchliessen(0);"

```



```

>
<INPUT TYPE=button
VALUE="PopUpFenster 1 schliessen"
onclick=" PopUpFensterSchliessen(1);"
>
<INPUT TYPE=button
VALUE="PopUpFenster 2 schliessen"
onclick=" PopUpFensterSchliessen(2);"
>
</BODY>
</HTML>

```

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

```

function OnClickHandler()
{
    alert(FensterZeiger.ID_TextArea.value;
    FensterZeiger.close());
}

....

// Fenster öffnen
var FensterZeiger=window.open("", null, "height=250,width=300,status=no,toolbar=no,menubar=no,location=no");
var FensterDokumentZeiger = FensterZeiger.document;

var Kette= '<HTML>'
+ '<HEAD></HEAD>'
+ '<BODY >'
+ '<TEXTAREA ID="ID_TextArea" ROWS="5" COLS="20"></TEXTAREA>'
+ '<BR>'
+ '<INPUT TYPE="button" VALUE="Text an Aufrufer übergeben"'
+ ' onclick="opener.OnClickHandler();"'
+ '>'
+ '</BODY>'
+ '</HTML>';

FensterDokumentZeiger.open("text/html");
FensterDokumentZeiger.write(Kette);
FensterDokumentZeiger.close();

```

Eigenschaften:

.document	Zeiger auf das HTML-Dokument im Popup-Fenster, das per .show() instanziiert wird Es können damit alle Eigenschaften und Methoden des Objektes document referenziert werden siehe Objekt window.popup Syntax: [var Zeiger =] zeiger_auf_popup_fenster.document
	zeiger_auf_popup_fenster laut Erzeugung per .createPopup()
.isOpen	nur lesen prüfen ob ein per .createPopup() instanziiertes Popup-Fenster angezeigt wird siehe Objekt window.popup Syntax: [var Wert =] zeiger_auf_popup_fenster.isOpen
	Wert true, so angezeigt false, so nicht angezeigt
	zeiger_auf_popup_fenster laut Erzeugung per .createPopup()

Methoden:

.hide()	nur lesen ein angezeigtes Popup-Fenster schliessen Hinweis: Das Popup-Fenster wird automatisch geschlossen, wenn ein anderes Objekt den Focus erhält bzw. aktiv wird, z.B. durch Klick des Users außerhalb des Popupfensters. ändert Wert der Eigenschaft .isOpen siehe Objekt window.popup Syntax: zeiger_auf_popup_fenster.hide()
	zeiger_auf_popup_fenster laut Erzeugung per .createPopup()
	liefert nichts



<code>.show()</code>	ein per <code>.createPopup()</code> instanziiertes Popup-Fenster anzeigen Hinweis: Es kann immer nur genau 1 Popup-Fenster angezeigt werden. Das Popup-Fenster wird automatisch geschlossen, wenn ein anderes Objekt den Focus erhält bzw. aktiv wird, z.B. durch Klick des Users außerhalb des Populfensters ändert Wert der Eigenschaft <code>.isOpen</code> siehe Objekt <code>window.popup</code> Syntax: <code>zeiger_auf_popup_fenster. .show(Wert, Wert2, Wert3, Wert4 [, Zeiger])</code>
Wert1	X-Koordinate der linken oberen Fensterecke bezüglich eines Objektes oder des Bildschirms Koordinaten-Ursprung (0,0) des Bildschirms liegt in der linken oberen Ecke Integer, in Pixel, >= 0
Wert2	Y-Koordinate der linken oberen Fensterecke bezüglich eines Objektes oder des Bildschirms Koordinaten-Ursprung (0,0) des Bildschirms liegt in der linken oberen Ecke Integer, in Pixel, >= 0
Wert3	Breite des Fensters in Pixel Integer, > 0
Wert4	Höhe des Fensters in Pixel Integer, > 0
Zeiger	auf Objekt zu dem Wert1 und Wert2 relativ sind Standard: Bildschirm
<code>zeiger_auf_popup_fenster</code>	laut Erzeugung per <code>.createPopup()</code>
liefert nichts	

4.3.2.2.11. **window.screen** Objekt

Dieses Objekt beschreibt den Bildschirm

Die linke obere Browserfensterecke ist der Ursprung des Grafiksystems, also (0,0) mit (x,y)

x für horizontal
y für vertikal

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

`screen.eigenschaft`

4.3.2.2.11.1. **screen** Objekt im Netscape

Eigenschaften:

<code>.availHeight</code>	maximal verfügbare Fensterhöhe in Pixel laut max. Bildschirmauflösung
<code>.availLeft</code>	minimale Pixel-X-Position der Browserfensterecke links oben (horizontal) hängt von Position der Windows-Taskleiste ab
<code>.availTop</code>	minimale Pixel-Y-Position der Browserfensterecke links oben (vertikal) hängt von Position der Windows-Taskleiste ab
<code>.availWidth</code>	maximale verfügbare Fensterbreite in Pixel laut max. Bildschirmauflösung
<code>.colorDepth</code>	Farbtiefe der aktuellen Farbpalette, also Anzahl der verfügbaren Farben
<code>.height</code>	aktuelle Höhe in Pixel der Bildschirmauflösung
<code>.left</code>	Abstand links in Pixel
<code>.pixelDepth</code>	aktuelle Anzahl Farbbits pro Bildpunkt laut aktueller Bildschirmauflösung 1 oder 4 oder 8 oder 15 oder 16 oder 24 oder 32
<code>.top</code>	Abstand oben in Pixel
<code>.updateInterval</code>	Verzögerung der Anzeigegeschwindigkeit in Millisekunden z.B. bei Animationen nur IE
<code>.width</code>	aktuelle Breite in Pixel der Bildschirmauflösung

Methoden:

keine

4.3.2.2.11.2. **screen** Objekt im Internet Explorer

Eigenschaften:

<code>.availHeight</code>	verfügbare Höhe des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
<code>.availWidth</code>	verfügbare Breite des Arbeitsbereiches auf dem Bildschirm ohne Windows-Taskbar per screen Objekt
<code>.bufferDepth</code>	Anzahl der Bits pro Pixel für eine Farbe UND Verwendung des off-screen bitmap buffer per screen Objekt
<code>.colorDepth</code>	Anzahl der Bits pro Pixel für eine Farbe UND Verwendung destination device or buffer



	wird durch den Wert der Eigenschaft .bufferDepth überschrieben, wenn .bufferDepth mit Wert > 0
	per screen Objekt
.deviceXDPI	Aktuelle Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm
	per screen Objekt
.deviceYDPI	Aktuelle Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm
	per screen Objekt
.fontSmoothingEnabled	Fontglättung auf Bildschirm
	per screen Objekt
.height	Auflösung des Bildschirms in Höhe, also Anzahl der vertikalen Pixel
	per screen Objekt
.logicalXDPI	Standard-Anzahl der horizontalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm
	per screen Objekt
.logicalYDPI	Standard-Anzahl der vertikalen Bildpunkte per Inch (DPI = Dot per Inch) auf Bildschirm
	per screen Objekt
.updateInterval	Refresh-Intervall des Bildschirms: aus dem Puffer neu schreiben
	per screen Objekt
.width	Auflösung des Bildschirms in Breite, also Anzahl der horizontalen Pixel
	per screen Objekt

Methoden:

keine

4.3.2.2.12. Sonstige Objekte und Collectionen (Auswahl)

4.3.2.2.12.1. regexp Objekt als Instanz des Script-Objektes RegExp

RegExp dient der Definition eines regulären Ausdruckes für die Zeichenkettensuche, wobei dabei automatisch wird z.T. auch von Methoden anderer Objekte z.B. String für Zeichenkettensuche benutzt.

regexp ist eine Instanz des Objektes RegExp
dient der detaillierten Festsetzung des Suchmusters.

Erzeugung:

```

var regexp_name= suchmuster;
oder var regexp_name=new RegExp(suchmuster);

```

suchmuster: regulärer ausdrück vom Objekttyp RegExp
verwendet für Suche
syntaktischer Aufbau:
/suchmuster_elemente_mit_kommatrennung/optionen

/ / sind zu kodierende Zeichen anstelle " "
markieren ein Suchmuster und keine Zeichenkette

suchmuster_elemente: verwendet für detaillierte Musterdefinition
stehen innerhalb von / / (NICHT " ")
Kommatrennung:

ein Element kann zum Beispiel sein:

- \zeichen hebt die ursprüngliche Bedeutung eines Zeichens auf
Bsp: + ist Addition oder Verketten
 \+ ist das Zeichen '+'
- ^zeichenfolge markiert eine feste Zeichenfolge am **Anfang** des Suchbegriffes
Bsp:^Otto entspricht "Otto Waalkes"
 entspricht NICHT "Heinrich Waalkes"
- \$zeichenfolge markiert eine feste Zeichenfolge am **Ende** des Suchbegriffes
Bsp:^Waalkes entspricht "Otto Waalkes"
 entspricht NICHT "Otto der Komiker"
- zeichen* zeichen kann mehrfach auftreten (beliebig viel) auch Null-Mal
Bsp: t* entspricht "tto Waalkes"
 entspricht "to Waalkes"
 entspricht "o Waalkes"
- zeichen+ zeichen kann mehrfach auftreten (beliebig viel) aber mindestens einmal
Bsp: t* entspricht "tto Waalkes"
 entspricht "to Waalkes"
 entspricht NICHT "o Waalkes"
- zeichen? zeichen kann höchstens einmal oder keimnal auftreten
Bsp: t* entspricht NICHT "tto Waalkes"
 entspricht "to Waalkes"
 entspricht NICHT "o Waalkes"
- . Punkt im Suchmuster ersetzt genau ein beliebiges Zeichen an seiner Position
- zeichenfolge_1|zeichefolge_2 entweder eine von beiden Zeichenfolgen oder beide zusammen



[zeichen_menge]	alternative Zeichen als Menge einschliessend Bsp: [Oo] "[Oo]tto Waalkes" für "Otto Waalkes" oder "otto Waalkes"
[^zeichen_menge]	alternative Zeichen als Menge ausschliessen Bsp: [^O] "[^O]tto Waalkes" für "tto Waalkes" oder "otto Waalkes"
\bzeichenfolge	Worttrenner z.B. Blank oder Tab werden beachtet Bsp: "Otto\bWaalkes" für "Otto Waalkes" nicht für "OttoWaalkes"
\Bzeichenfolge	Worttrenner z.B. Blank oder Tab dürfen nicht enthalten sein Bsp: "Otto\BWaalkes" für "OttoWaalkes" nicht für "Otto Waalkes"
\dzeichenfolge	oder [0] [9] oder [0-9] Ziffer 0 bis 9 dürfen enthalten sein Bsp: "Otto Waalkes der \d" für "Otto Waalkes der 1." nicht für "Otto Waalkes der V." Bsp: "Otto Waalkes der [1]" für "Otto Waalkes der 1." nicht für "Otto Waalkes der 2." Bsp: "Otto Waalkes der [0-9]" für "Otto Waalkes der 1." für "Otto Waalkes der 2." nicht für "Otto Waalkes der V."
\Dzeichenfolge	oder [^0] [^9] oder [^0-9] Ziffer 0 bis 9 dürfen nicht enthalten sein Bsp: "Otto Waalkes der \D" für "Otto Waalkes der Erste" nicht für "Otto Waalkes der 1." Bsp: "Otto Waalkes der [^1]" für "Otto Waalkes der 2." nicht für "Otto Waalkes der 1." Bsp: "Otto Waalkes der [^0-9]" für "Otto Waalkes der Erste" nicht für "Otto Waalkes der 1." nicht für "Otto Waalkes der 2."
\fzeichenfolge	Seitenvorschub muss vorhanden sein vor zeichenfolge, wenn sie gefunden werden soll
\mzeichenfolge	Mehrzeiligkeit zulässig, nur NS ab 6.x
\nzeichenfolge	Zeilenvorschub muss vorhanden sein vor zeichenfolge, wenn sie gefunden werden soll
\rzeichenfolge	Wagenrücklauf muss vorhanden sein vor zeichenfolge, wenn sie gefunden werden soll
\tzeichenfolge	Tabulator muss vorhanden sein vor zeichenfolge, wenn sie gefunden werden soll
\vzeichenfolge	vertikaler Tabulator muss vorhanden sein vor zeichenfolge, wenn sie gefunden werden soll
\wzeichenfolge	zeichenfolge muss Ziffern und oder Buchstaben enthalten, wenn sie gefunden werden soll
\Wzeichenfolge	zeichenfolge darf keine Ziffer und oder Buchstaben enthalten, wenn sie gefunden werden soll
[f\n\r\t\v]zeichenfolge	zeichenfolge muss mindestens einen der in [] gesetzten Steuerzeichen enthalten, wenn sie gefunden werden soll
[^f\n\r\t\v]zeichenfolge	zeichenfolge darf keinen der in [] gesetzten Steuerzeichen enthalten, wenn sie gefunden werden soll
[A-Za-z0-9]zeichenfolge	zeichenfolge muss mindestens einen der in [] gesetzten Zeichen enthalten, wenn sie gefunden werden soll
[^A-Za-z0-9]zeichenfolge	zeichenfolge darf keinen der in [] gesetzten Zeichen enthalten, wenn sie gefunden werden soll
(suchmuster_element)	Bsp: ([0-9])
\szeichenfolge	zeichenfolge muss mindestens ein beliebigen Trenner z.B. Blank, Tab, Seitenvorschub etc. enthalten, wenn sie gefunden werden soll
\Szeichenfolge	zeichenfolge darf keinen Trenner z.B. Blank, Tab, Seitenvorschub etc.



enthalten, wenn sie gefunden werden soll

- optionen: sind wahlweise kodierbar
 - i ignoriere Groß und Klein bei der Suche
 - oder g mehrfaches Auftreten möglich
 - oder gi entspricht i UND g
 - Standard ist Unterscheidung Groß und Klein
 - sowie Suche NUR bis zum ersten Auffinden

Zugriff:

regexp_name.eigenschaft
regexp_name.methode()

Eigenschaften (Auswahl):

- .global ist true wenn Option g kodiert
- .ignoreCase ist true wenn Option i kodiert
- .lastIndex Angabe derjenigen Position, die direkt hinter dem ZULETZT gefunden Teilstring liegt, der dem Suchmuster entspricht
Verwendung z.B. für Weitersuchen
nur ab NS 6.x
- .multiline st true wenn Option m kodiert
- .source enthält das gesamte Suchmuster aber ohne "/"
nur lesbar

Methoden (Auswahl):

- .compile(suchmuster,options) suchmuster --> siehe oben
optionen --> siehe oben
automatisch ausgeführt
- .exec(suchmuster) auch kodierbar als regexp(suchmuster) also ohne .exec
füllt Eigenschaften von RegExp und regexp, wobei RegExp.input die zu durchsuchende Kette enthält
liefert Array mit index ab 0:
 - als Position im Suchbegriff an der Übereinstimmung besteht
 - 0 letzte gefundene Übereinstimmung
 - ab 1 bis unbegrenzt: jeweils gefundene Übereinstimmung mit den Teil des Suchbegriffes innerhalb von ()
 - Feldelement enthält gefundene Übereinstimmung
 - ist null-Zeiger wenn Auswertung fehlerhaft war

Beispiel:

```
function getInfo()
{
    re = /(w+)s(d+)/;
    var m = re.exec();
    alert(m[1] + ", your age is " + m[2]);
}

<FORM>
  <INPUT TYPE="text" NAME="Alter" onChange="getInfo(this);">
</FORM>
```

- .test(suchmuster) Suche laut suchmuster
liefert true, wenn Suche erfolgreich (sonst false)

Beispiel:

```
function testinput(re, str)
{
    if (re.test(str))
    {midstring = " contains ";}
    else
    {midstring = " does not contain ";}

    document.write (str + midstring + re.source);
}
```

Beispiel: <SCRIPT>
<!--

```
// Vorbereitung der Suche
var zu_durchsuchende_kette="Otto Waalkes";
// oder RegExp.input="Otto Waalkes"

var such_muster=/(\wOtto)/g;
// such_muster ist ein regulärer Ausdruck
// (Objekt RegExp)
// Suchmuster ist Otto, wobei Otto gefunden
// werden muss für eine erfolgreiche Suche
// anstelle von " ist / zu kodieren
// alternativ nicht möglich
// RegExp.input="Otto"
// dann kein detailliertes Suchmuster
```



```

// Option g alternativ kodierbar per
//   RegExp.multiline=true;

// Start der Suche nur durch eine Zeichenketten-Funktion, die RegExp beachtet,
//   z.B. durch replace() als Methode des Objektes String

var ergebnis_kette=zu_durchsuchende_kette.
    replace(such_muster,"Heinrich");

// Start der Suche nach Otto
//wenn gefunden, so durch Heinrich ersetzen
// Ergebnis der Suche und des Ersetzens
//   nach ergebnis_kette bringen
// oder alternativ z.B.
//   var ergebnis_kette=RegExp.lastMatch
// --> Suchmuster hat ( ), also auch $1 bis $9
//   verwendbar

//oder bei exec() bzw. regexp(such_muster)
var ergebnis_array=such_muster.exec(zu_durchsuchende_kette);
//-->
</SCRIPT>

```

.toSource() liefert String mit Quellcode

.toString() liefert String des Objekthinhaltes

Beispiel

```

myExp = new RegExp("a+b+c");
alert(myExp.toString()) displays "/a+b+c/"

```

4.3.2.2.12.2. **RegExp Objekt (nicht regexp Objekt)**

Dieses Objekt ist eine Komponente der Scriptsprache.

RegExp dient der Definition eines regulären Ausdruckes für die Zeichenkettensuche, wobei dabei automatisch wird z.T. auch von Methoden anderer Objekte z.B. String für Zeichenkettensuche benutzt.

regexp ist eine Instanz des Objektes RegExp
dient der detaillierten Festsetzung des Suchmusters.

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

RegExp.eigenschaft

Beispiel: <SCRIPT>
 <!--

```

// Vorbereitung der Suche
var zu_durchsuchende_kette="Otto Waalkes";
// oder RegExp.input="Otto Waalkes"

var such_muster=/Otto/g; // ist Objekt vom Typ regexp und NICHT RegExp
// Suchmuster ist Otto
//   anstelle von " ist / zu kodieren
//   alternativ auch möglich
//   RegExp.input="Otto"
//   dann aber ohne detailliertes Suchmuster

RegExp.multiline=true; // mehrfaches Vorkommen beachten

// Start der Suche nur durch eine Zeichenketten-Funktion, die RegExp beachtet,
//   durch replace() als Methode des Objektes String
//   Hinweis: Die Methode search() des Objektes String wertet nicht RegExp-
//   Vorgaben aus

var ergebnis_kette=zu_durchsuchende_kette. replace(such_muster,"Heinrich");

// Start der Suche nach Otto
//wenn gefunden, so durch Heinrich ersetzen
// Ergebnis der Suche und des Ersetzens
//   nach ergebnis_kette bringen
// oder alternativ z.B.
//   var ergebnis_kette=RegExp.lastMatch
// --> Suchmuster hatte keine ( ), also
//   z.B. $1 nicht verwendbar

//-->
</SCRIPT>

```



Eigenschaften (Auswahl):

sämtliche Steuerzeichen .\$ sind im NS ab 6.x deprecated
IE ab Version 5.5 verwendbar

- .input oder .\$_ Zeichenkette, die durchsucht werden soll
Wertzuweisung per RegExp.input="zeichenkette"
- .multiline oder .\$* auf true setzen für Suche über mehrere Zeilen
auf false setzen für Suche nur bis zum nächsten Zeilenumbruch
- .lastMatch oder .\$& enthält Zeichenfolge, die zuletzt gefunden wurde **und** dem Suchbegriff (Suchmuster) entspricht
ab IE 5.5
- .lastParent oder .\$+ enthält die Zeichenfolge, die zuletzt gefunden wurde **und** dem Suchbegriff innerhalb ()
aus regexp (**NICHT** RegExp !!!) entspricht
- .leftContext oder .\$` nicht .\$`
enthält Teilkette, der LINKS neben dem gefundenen Muster steht
- .rightContext oder .\$' nicht .\$'
enthält Teilkette, der Rechts neben dem gefundenen Muster steht
- .\$1 bis .\$9 die letzten 9 gefundenen Zeichenketten, die dem Muster innerhalb ()
aus regexp (**NICHT** RegExp) entsprechen
- .\$01 bis .\$09 die letzten 100 gefundenen Zeichenketten, die dem Muster innerhalb ()
aus regexp (**NICHT** RegExp) entsprechen

Methoden:

keine

4.3.2.2.12.3. script Objekt des Internet Explorer

Container für Scriptcode z.B. Javascript oder XML

Achtung: Jedes Script hinter dem Ende-Tag vom FRAMESET-Element wird ignoriert und nicht geparkt !!

Der erste Script-Block mit .language Eigenschaft legt die Sprache aller nachfolgenden fest, wenn dort kein .language codiert wurde.

Fehlt generell die .language Eigenschaft, dann so wird die microsoft-spezifisches Scriptmaschine zum Parsen verwendet.

Bps.: JScript ist Microsoft-Javascript-Standard
Javascript ist allgemeiner der Javascript-Standard

Zugriff:

document.all["id_des_scripts"].eigenschaft
document.all["id_des_scripts"].eigenschaft

id_des_scripts laut ID-Attribut

Beispiel 1: für XML-Script

```
<HEAD>
<SCRIPT LANGUAGE="Javascript">
var XMLScriptObjekt = document.all["ID_XML"].XMLDocument;
</SCRIPT>

<SCRIPT LANGUAGE="XML" ID="ID_XML">
.....
</SCRIPT>
</HEAD>
```

Beispiel 2: für Javascript

```
<HEAD>
<SCRIPT LANGUAGE="Javascript">
var JavaScriptObjekt = document.all["ID_JavaScript"]
</SCRIPT>

<SCRIPT LANGUAGE="Javascript" ID="ID_JavaScript">
.....
</SCRIPT>
```

Eigenschaften:

- .canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf
- .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
- .defer Pars-Status des Scriptes per script Objekt



	download eines Scriptes kann beschleunigt werden, wenn es vom Parsen zurückgestellt wird
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.event	On-Eventbezeichner mit angefügten Klammerspaar Script soll das Event verwalten per script Objekt immer mit Eigenschaft .htmlFor kodieren
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.htmlFor	Referenz auf Objekt, das das Event laut Eigenschaft .event verwalten soll und dafür einen Scriptaufruf im Ereignishandler onxxx besitzt per script Objekt immer mit Eigenschaft .event kodieren
.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. 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. 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)
.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 Hinweis: Eigenschaften .scrollLeft und .scrollTop sind nicht 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 nicht 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
.src	Url der Daten z.B. vom Image in normaler Auflösung
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.text	Text eines Objektes
.type	Mimetyt des Scriptes per script Objekt Mimetyt wird von der Scriptmaschine zum Parsen verwendet Achtung: Wert muss passend zum Wert der Eigenschaft .language sein !
.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
.XMLDocument	Referenz auf XML-Dokument (XML-DOM)



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
.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 Zufallsfolge , 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 nicht die Pixelgenauigkeit wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.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 nicht 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()
.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 !!



.normalize()	DOM wird geändert Normalisierung des DOM zur Erreichung einer konsistenten Struktur
.removeAttribute()	Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen 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)
.replaceAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.setAttribute()	DOM wird nicht geändert 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
.swapNode()	DOM wird geändert Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt DOM wird geändert

4.3.2.2.12.4. document.scripts Collection des Internet Explorer

Feld der Zeiger aller script Objekte im Dokument
Elementefolge laut HTML-Koding, also auch in der Reihenfolge HEAD --> BODY

Syntax:

```
[ var ZeigerAufFeld = ] document.scripts
[ var ZeigerAufFeldElement = ] document.scripts[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
.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

4.3.2.2.12.5. var Objekt des Internet Explorer

Basisobjekt zu einer Variablen

Erzeugung:

in Script mit der Anweisung var deklariert werden kann (aber nicht muss)

in HTML mit dem VAR-Tag deklariert wird

Beispiel: Das ist eine HTML-<VAR>Variable</VAR>die mit italic font gerendert wird

Zugriff:

in Script per Bezeichner der Variablen als Referenz
ID-Attribut

in HTML per 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
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.
.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
.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. dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag Plain-Text HTML-Elemente je nach Objekt möglich Script: muss mit DEFER-Attribut kodiert sein schreiben: wenn Bereich nicht leer, so komplett überschreiben wenn Bereich leer so einfügen nur lesen bei COL, COLGROUP, FRAMESET, HTML, STYLE, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.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)
.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 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 nur 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 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
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



	<p>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 !</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 Zufallsfolge, 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 entfernenbar 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 nicht die Pixelgenauigkeit wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.</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 nicht behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)</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 .getElementsByName()</p>
.getExpression()	<p>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)</p>
.hasChildNodes()	<p>prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert</p>



.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 nicht ausgeführt eingefügter Code wird nur dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER> muss kodiert werden DOM wird geändert
.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

4.3.2.2.12.6. xml Objekt im Internet Explorer

Dieses Objekt definiert eine Daten-Insel als eigenständige Datenmenge im HTML-Dokument, die per XML kodiert wird.

Daten sind dabei nur vom String-Typ werden nicht gerendert (angezeigt) liegen direkt im HTML-Dokument

ab IE 5.x

XML-Daten werden nach dem XML-DOM verarbeitet, siehe Objekt XMLHttpRequest (ab IE 7) und Objekt userProfile.

Erzeugung:

durch Browser

Zugriff:

solange das HTML-Dokument als Container instanziiert ist

Beispiel:

```
<XML ID="ID_DatenInsel">
  <METADATA>
    <AUTHOR>Ich </AUTHOR>
    <GENERATOR> Notepad</GENERATOR>
    <PAGETYPE>Reference</PAGETYPE>
    <ABSTRACT>Dateninsel </ABSTRACT>
  </METADATA>
</XML>
```

```
alert("Daten komplett geparkt = " + (ID_DatenInsel.readyState == "complete"));
```

Eigenschaften:

.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf
.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) ID-Attribut in HTML: Wert ist String alphanumerisch muss mit Buchstaben beginnen Unterstrich _ verwendbar kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.
.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. false Content nicht editierbar true Content editierbar
.isDisabled Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich false User kann mit Objekt interagieren true User kann mit Objekt nicht interagieren
.isMultiLine Mehrzeiligkeit des Objektinhaltes false Objekt hat genau 1 Zeile true Objekt hat mindestens 1 Zeile
.parentElement Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.readyState aktueller Status des Objektes beim Füllen des Objektes mit Daten "uninitialized" Objekt ist nicht initialisiert "loading" Objekt ist nicht intialisiert aber lädt gerade Daten zur Initialisierung "loaded" Objekt hat alle Daten komplett geladen und ist komplett intitialisiert "interactive" Objekt kann vom User interaktiv verwendet werden zum Füllen mit Daten "complete" Object hat alle Daten geladen und ist mit diesen komplett initialisiert
.recordset Zeiger des Standard-Record-Set in einem Datasource-Objekt (DSO) Methode . namedRecordset() liefert den Zeiger für Datenquellen-Objekt mit Namen also eines beliebige Mitgliedees im Datasource-Objekt (DSO)
.scopeName Namensraum laut XMLNS-Attribut
.src Url der Daten z.B. vom Image in normaler Auflösung
.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.XMLDocument Referenz auf XML-Dokument (XML-DOM)



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 nicht die Pixelgenauigkeit wie die Angaben der Methode .componentFromPoint() also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben Overbereich der Maus ist mehr als 1 Pixel gross beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.
.fireEvent()	ein Event auslösen true Event erfolgreich ausgelöst false Event nicht ausgelöst
.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
.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
.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
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert

4.3.2.2.12.7. Collectionen des Internet Explorers - Übersicht (englisch)

Hinweis:

Wenn ein HTML-Tag als Zeigernamen in der Punktnotation verwendet wird, dann ist der konkrete Zeiger auf ein Objekt mit Typ laut HTML-Tag gemeint (Objekt erstellbar z.B. per .createElement(tag_bezeichner_als_string))
Das gleiche gilt für Dialog Helper (Objekt dlgHelper) etc..

4.3.2.2.12.7.1. Collection .all**Objekte mit der Collection:**

A, ACRONYM, ADDRESS, APPLET, AREA, B, BASE, BASEFONT, BDO, BGSOUND, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, COL, COLGROUP, CUSTOM, DD, DEL, DFN, DIR, DIV, DL, document, DT, EM, EMBED, FIELDSET, FONT, FORM, FRAME, FRAMESET, HEAD, hn, HR, HTML, I, IFRAME, IMG, INS, KBD, LABEL, LEGEND, LI, LINK, LISTING, MAP, MARQUEE, MENU, OBJECT, OL, P, PLAINTEXT, PRE, Q, S, SAMP, SCRIPT, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TITLE, TR, TT, U, UL, VAR, XMP

Syntax:

```
[ collAll = ] object.all
[ oObject = ] object.all(vIndex [, iSubIndex])
```

collAll Array of elements contained by the object.

oObject Reference to an individual item in the array of elements contained by the object.

vIndex Required. Integer or string that specifies the element or collection to retrieve. If this parameter is an integer, the method returns the element in the collection at the given position, where the first element has value 0, the second has 1, and so on. If this parameter is a string and

there is more than one element with the name or id property equal to the string, the method returns a collection of matching elements.

iSubIndex Optional. Position of an element to retrieve. This parameter is used when vIndex is a string. The method uses the string to construct a collection of all elements that have a name or id property equal to the string, and then retrieves from this collection the element at the position specified by iSubIndex.

Eigenschaften:

.length Sets or retrieves the number of objects in a collection.

Methoden:

.item Retrieves an object from the all collection or various other collections.
.namedItem Retrieves an object or a collection from the specified collection.
.tags Retrieves a collection of objects that have the specified HTML tag name.
.urns Retrieves a collection of all objects to which a specified behavior is attached.

