wenn Anzahl > object.length, so kein Fehler

liefert nichts

Beispiel:

```
var TextKnoten = document.createTextNode("Test 1");
TextKnoten.deleteData(4, 2);          // ergibt "Test"
```

Textdaten einfügen:

.insertData() Teilkette in ein Objekt einfügen

Syntax:

object.insertData(Offset, Kette)

Offset	Integer	Startposition ab der eingefügt werden soll ab 0
Kette	Zeichenkette	

liefert nichts

Beispiel:

```
var TextKnoten = document.createTextNode("Test 1");
TextKnoten.insertData(4, "reihe");          // ergibt "Testreihe 1"
```

Textdaten anhängen:

.appendData() String an das Ende des Objektes anhängen

Syntax:

object.appendData(Kette)

Kette	Zeichenkette	

liefert nichts

Beispiel:

```
var TextKnoten = document.createTextNode("Test 1");
TextKnoten.appendData("0");          // ergibt "Test 10"
```

4.3.2. window Objekt

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

Das window Objekt hat diverse Zeiger auf andere Objekte:

z.B. beim IE und NS:

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

z.B. beim IE:

.screen	Zeiger auf das Objekt screen

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

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

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

Erzeugung:

per Methode .open()

neues Fenster als unterstes Fenster in der Hierarchie instanzieren und öffnen, also anzeigen
mit öffnen wird Referezn auf das Fenster geliefert: Erstes Fenster wird beim Öffnen des Internet Explorer automatisch geöffnet.

Hinweis zu Scrollbalken: **Neu** erscheinende Scrollbalken verkleinern die Dimension des Browserfensters. Das ist zu beachten, wenn die Fensterdimension ohne bereits exisitierende Scrollbalken ermittelt wurde.

Beispiel:
```
var logischer_window_name;

function oeffne_fenster()
{logischer_window_name=window.open();}

        <A HREF="javascript:oeffne_fenster();">Fenster oeffnen</A>
        <A HREF= "javascript:logischer_window_name.close();">Fenster schliessen</A>
```

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

....

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

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

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

Zugriff:

auf Fenster mit Zeiger laut Erzeugung:

```
logischer_window_name.eigenschaft
logischer_window_name.methode()

        logischer_window_name        laut open()
```

auf aktuelles Fenster:
```
window.eigenschaft
window.methode()

self.eigenschaft
self.methode()

eigenschaft
methode()
```

Hinweise: window ist synonym zu self

Innerhalb von Eventhandlern muss der korrekte Fensterbezug kodiert werden, egal ob es das aktuelle Fenster ist oder nicht. Folgende Kodierungen sind nicht zulässig:

```
eigenschaft
methode()
```

auf oberstes Fenster in der Fenster-Hierarchie:

```
top.eigenschaft
```


top.methode()

auf das Elternfenster:

parent.eigenschaft
parent.methode()

auf den Erzeuger des Fensters, der das open() enthält:

opener.eigenschaft
opener.methode()

Beispiel:

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

....

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

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

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

4.3.2.1. window Objekt des Netscape (Übersicht)

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

Das window Objekt hat diverse Zeiger auf andere Objekte:

z.B. beim IE und NS:

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

z.B. beim IE:

.screen	Zeiger auf das Objekt screen

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

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

Hinweis zu signiertem Script beim NS

Es muss das Privileg "UniversalBrowserWrite" vom Privilegmanger angefordert sein.

Hinweis zum Netscape-Privilegmanager: Diese Manager wird aktiv z.B. für bei Verwendung der Methode .enableExternalCapture(), wenn also versucht wird, dass eine fremde Seite in einem Frame diejenige Seite überwachen will, die den Frame besitzt. Dazu muss das Privileg "UniversalBrowserWrite" beim Manager angefordert werden.

Die fremde Seite kann z.B. folgenden Code besitzen:

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite");
```


```
window.enableExternalCapture();
window.captureEvents(Event.CLICK | Event.MOUSEDOWN);
```

4.3.2.1.1. Eigenschaften

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

._content
Zeiger auf das Window-Objekt
nur lesen
nur NS 6.x

.appCore
nur NS 6.x

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

Wert false so ist Fenster offen
true so ist Fenster geschlossen
nur lesen

.Components
nur NS 6.x

.controllers
nur NS 6.x

.crypto
Zeiger auf das gleichnamige Objekt für Verschlüsselung anhand von Zertifikaten
nur NS 6.x

.defaultStatus
Text der Statuszeile
Verwendung in Eventhandler-Funktion: Es muss **return true** kodiert werden
z.B. onmouseover=" ...; return true;"
Syntax:
window.defaultStatus [= Kette]
[var Kette =] window.defaultStatus

Kette String
Plaintext
lesen und schreiben

.directories
nur NS 6.x

.document
Zeiger auf das Objekt document im Fenster
nur lesen

.event
Zeiger auf das Objekt event im Fenster
nur lesen

.frames
Zeiger auf die Collection frames des Fensters
nur lesen

.history
Zeiger auf das Objekt History
nur lesen

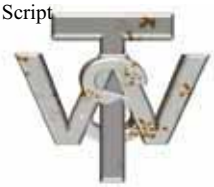
.innerHeight
Höhe des Anzeigebereiches im Fenster in Pixel, ohne Leisten, Menü, Statuszeile etc.
lesen und schreiben

.innerWidth
Breite des Anzeigebereiches im Fenster in Pixel, ohne Leisten, Menü, Statuszeile etc.
lesen und schreiben

.length
Anzahl aller Frames bzw. IFrames im HTML-Dokument
Integer, ab 0
nur lesen

.location
Zeiger auf das gleichnamige Objekt
nur lesen

.locationbar
Zeiger auf die Adresszeile mit folgenden Eigenschaften:
.locationbar.visible liefert true wenn Adresszeile sichtbar
liefert false wenn Adresszeile unsichtbar
veränderbar per signiertem Script



.menubar Zeiger auf die Menüzeile mit folgenden Eigenschaften:
.menubar.visible liefert true wenn Menüzeile im Fenster sichtbar
liefert false wenn Menüzeile im Fenster unsichtbar
veränderbar per signiertem Script

.name physischer Fenster-Name eines Fensters laut open()
entspricht Wert des Attributes TARGET eines Links
Syntax:
window.name [= Kette]
[var Kette =] window.name

Kette String
lesen und schreiben

Beispiel:

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

.navigator Zeiger auf das Objekt navigator
nur lesen

.opener Zeiger auf dasjenige Fensters, das open() enthält also das das Kind-Fenster öffnet, welches in dieser .opener-Eigenschaft den Zeiger auf das Elternfenster enthält
Bezug im geöffneten Fenster auf Instanzen des Aufrufers ist möglich
Bezug des Aufrufers auf geöffnetes Fenster ist möglich
Hinweis: Bei FRAME und IFRAME anstelle opener den Zeiger parent verwenden
Öffnet ein Frame / Iframe ein Fenster, das damit nicht im Framset läuft, so ist der Bezug vom Fenster aus auf das Frameset oder auf einen Frame im Frameset per window.opener.parent möglich, solange opener existiert.
Syntax:
window.opener [= Zeiger]
[Zeiger =] window.opener

lesen und schreiben

.outerHeight Fensterhöhe in Pixel inkl. Anzeigebereich, Leisten, Menü, Statuszeile etc
ohne signiertes Script minimal nur 100 Pixel

.outerWidth Fensterbreite in Pixel inkl. Anzeigebereich, Leisten, Menü, Statuszeile etc.
ohne signiertes Script minimal nur 100 Pixel

.pageXOffset horizontale Position im gescrollten Dokument, die genau auf der linken oberen Ecke des Anzeigebereiches liegt, also Offset-Angabe gegenüber der linken obere Ecke des ungescrollten Dokumentes, also gegenüber 0
Hinweis: Dokument scrollbar im Anzeigebereich, wenn Größe des Dokumentes die des Anzeigebereiches überschreitet
Offset ist 0, wenn Dokument noch nie gescrollt wurde

.pageYOffset vertikale Position im gescrollten Dokument, die genau auf der linken oberen Ecke des Anzeigebreiches liegt, also Offset-Angabe gegenüber der linken obere Ecke des ungescrollten Dokumentes, also gegenüber 0
Hinweis: Dokument scrollbar im Anzeigebereich, wenn Größe des Dokumentes die des Anzeigebereiches überschreitet
Offset ist 0, wenn Dokument noch nie gescrollt wurde

.parent Referenz auf Elternfenster
z.B. Zeiger auf das Fenster, das die FRAMESET-Deklaration enhtält, oder das diesem Fenster übergeordnet ist
muß nicht opener sein
Test auf Existenz von parent per if (parent != self) ....
Syntax:
[var Zeiger =] window.parent
nur lesen

.personalbar ist selbst Objekt und enthält die Toolbar mit folgenden Eigenschaften:
.visible ist true wenn Personal-Toolbar im Fenster sichtbar
veränderbar per signiertem Script

.pkcs11 nur NS 6.x

.prompter nur NS 6.x

.screen — Zeiger auf das Objekt screen

.screenX — X-Koordinate links oben des Browser-Fensters
0 = linke obere Ecke des Bildschirmes

.screenY — Y-Koordinate links oben des Browser-Fensters
0 = linke obere Ecke des Bildschirmes

.scrollbars — Zeiger auf Objekt scrollbars, das die Scrolleitsten mit folgenden Eigenschaften enthält:
.visible ist true wenn Scrollbar im Fenster sichtbar
veränderbar per signiertem Script

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

.sidebar — Sidebar
nur NS6.x
Das Dokument, welches in die Sidebar geladen werden soll, muss folgende META-Tags besitzen:

```
<HEAD>
<META HTTP-EQUIV="expires" CONTENT="0">
<META HTTP-EQUIV="Pragma" CONTENT ="no-cache">
<HEAD>
```

Das Dokument, welches in die Sidebar geladen werden soll, kann einen Link z.B. per A-Tag besitzen, der im TARGET-Attribut den Wert "_content" besitzen muss:
Bsp.: <A TARGET="_content" HREF="test.htm">Dokument in der NS-Sidebar oeffnen</A>

Das Dokument, welches die Sidebar aufrufen will, kann dieses mit folgender Funktion tun:

```
</HEAD>
<SCRIPT LANGUAGE ="JavaScript">
<!--
function DokumentInSideBar Laden(TitelKette, UrlKette)
{
                    // TitelKette       String in " " bzw. ' ' mit freiem Wert
                    // UrlKette         String in " " bzw. ' ' mit absoluter Url-Angabe

        // prüfen ob Netscape-Objekt window.sidebar instanziert ist
        if (typeof window.sidebar == "object")
        {
                    //  prüfen ob Methode addPanel verfügbar ist
            if (typeof window.sidebar.addPanel == "function") // ohne () kodieren !
            {
                    // in der Sidebar neues Fenster öffnen und Dokument laden
                    window.sidebar.addPanel(TtitelKette, UrlKette, "");
            }
        }
}
//-->
</SCRIPT>
</HEAD>
```

.status — enthält aktuellen Plain-Text der Statuszeile des Fensters (nicht Standardtext)
auch schreibbar: wenn per Eventhandler, so muss dieser **return true;** enthalten

Beispiel für Text buchstabenweise in Statuszeile anzeigen:

Der aktuelle Statuszeilentext wird als Teilkette von Position 0 bis zeichen_nr angezeigt, wobei zeichen_nr pro Anzeige um 1 erhöht wird.

```
<HTML>
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var statuszeile     = new Array();
        statuszeile[0]      = "Text0";
        statuszeile[1]      = "Text1";
        statuszeile[2]      = "Text2";
```


```
                var statuszeile_nr      = 0;
                var zeichen_nr          = 0;

                function textanzeigen ()
                {
                        if (position < statuszeile[statuszeile_nr].length)  // Länge ab 1
                        {
                                // nächste Teilkette des aktuellen Statuszeilentextes anzeigen
                                window.status = statuszeile[statuszeile_nr].substring(0, zeichen_nr);

                                // nächste Position
                                position++;
                                setTimeout("textanzeigen()", 100);
                        }
                        else
                        {
                                // aktuelle Statuszeilentext komplett anzeigen
                                window.status = statuszeile[statuszeile_nr];

                                // nächste Statuszeile adressieren
                                statuszeile_nr ++;
                                if (statuszeile_nr >= statuszeile.length)
                                { statuszeile_nr = 0;}

                                // Position 0 einstellen
                                zeichen_nr = 0;

                                // Start der buchstabenweise Anzeige
                                setTimeout("textanzeigen()", 1000);
                        }
                }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>
```

Beispiel für automatisch wechselnder Text in der Statuszeile:

```
<HTML>
<HEAD>
 <SCRIPT  LANGUAGE="JavaScript">
 <!--
        var statuszeile = new Array();
        statuszeile[0] = "Text0";
        statuszeile[1] = "Text1";
        statuszeile[2] = "Text2";

        var statuszeile_nr = 0;

        function textanzeigen()
        {
                window.status = statuszeile[statuszeile_nr];
                statuszeile_nr ++;
                if (statuszeile_nr >= statuszeile.length) {statuszeile_nr = 0;}
                setTimeout("textanzeigen()", 1000);
        }
// -->
 </SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>
```

Beispiel für dauerhaftes Scrollen eines Textes in der Statuszeile:

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


```
<!--
        var scrolltext_gesamt = "Dieser Text scrollt!";
        var scrolltext        = "";
        var zahler            = scrolltext_gesamt.length;


        function scrollen()
        {
                if (zahler == scrolltext_gesamt.length)
                {
                        zahler = 0;
                        scrolltext = scrolltext_gesamt;
                }
                else
                {
                        zahler++;  // 1  bis  scrolltext_gesamt.length
                        // Blank vorsetzen, also Kette um 1 Zeichen verlängern
                        scrolltext = " "+scrolltext;

                        // rausgerutschtes Zeichen abschneiden
                        scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
                                                        // Angaben ab 0
                }

                window.status = scrolltext;

                setTimeout("scrollen()", 100);
        }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="scrollen()">
</BODY>
</HTML>
```

Beispiel für einen Scrolltext in der Statusleiste, der angehalten wird, wenn Mauszeiger sich über einen Linkt bewegt:

```
<HTML>
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var scrolltext_gesamt = "Dieser Text scrollt!";
        var scrolltext = "";
        var zahler = scrolltext_gesamt.length;
        var id;

        function scrollen()
        {
                 if (zahler >= scrolltext_gesamt.length)
                 {
                        zahler = 0;
                        scrolltext = scrolltext_gesamt;
                 }
                 else
                 {
                        zahler++;  // 1  bis  scrolltext_gesamt.length
                        // Blank vorsetzen, also Kette um 1 Zeichen verlängern
                        scrolltext = " "+scrolltext;

                        // rausgerutschtes Zeichen abschneiden
                        scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
                                                        // Angaben ab 0
                }

                 window.status = scrolltext;
                id=setTimeout("scrollen()", 100);
        }

        function scrollen_stoppen()
        {clearTimeout(id);}
// -->
```


```
</SCRIPT>
</HEAD>

<BODY onLoad="scrollen()">
        Bewegen Sie den Mauspfeil &uuml;ber diesen
        <A      HREF="http://www.test.de"
                onMouseOver="scrollen_stoppen()"
                onMouseOut="scrollen()"
        >
        Link
        </A>
</BODY>
</HTML>
```

Beispiel für Anzeige eines Textes zu einem Hyperlink (HREF) für eine feste Zeitspanne in der Statuszeile:

```
<HTML>
</HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var id;
        var anzeige_aktiv = false;
        var anzeige_zeit = 3000;        // 3000 Millisekunden = 3 Sekunden

        function anzeige_starten(href_text)
        {
                if(anzeige_aktiv)
                {clearTimeout(id);}

                window.status = href_text;

                id = setTimeout("anzeige_stoppen()",anzeige_zeit);

                anzeige_aktiv = true;

                return true;            // Wichtig !!!
        }

        function anzeige_stoppen ()
        {
                anzeige_aktiv = false;

                window.status = "";
        }
//-->
</SCRIPT>
</HEAD>
<BODY>
        <A HREF= ... onMouseOver="javascript:return anzeige_starten('Hinweistext...')">
        ...
        </A>
</BODY>
</HTML>
```

Beispiel zur Anzeige eines Formularfeld-Hinweises in der Statuszeile:

```
<HTML>
</HEAD>
        <SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
        <!--
                function hinweis_anzeigen(hinweis_text)
                {window.status = hinweis_text;}

                function hinweis_loeschen()
                {window.status = "";}
        // -->
        </SCRIPT>
</HEAD>
<BODY>
        <FORM>
                <INPUT    TYPE=TEXT
```


```
                                        onFocus="hinweis_anzeigen('Hinweis');"
                                        onBlur="hinweis_loeschen();"
                                >
                        </FORM>
                </BODY>
                </HTML>
```

Beispiel für blinkende Anzeige mit Zeitspanne von Text in der Statuszeile:

```
<HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
        var zeitspanne  = 6000;         // 6 Sekunden blinken lassen, danach
                                        //      Statuszeile löschen
        var anzeigezeit = 500;          // 0,5 Sekunden warten nach Setzen
                                        //      bzw. Löschen der Statuszeile
        var id_blinken  = null;         // muss auf null initialsiert sein !!

        function blinken_timer_loeschen
        {
                if  (id_blinken != null)
                {
                        clearTimeout(id_blinken)};  // blinken stoppen falls aktiv
                        id_blinken = null;
                 }
        }

        function blinken_stop()
        {
                 blinken_timer_loeschen;
                 window.status="";
        }

        function blinken_start(status_text)
        {
                blinken_timer_loeschen;
                blinken(true, status_text);                 // Blinken anstossen für
                                                            // paralleles Arbeiten per Timer
                setTimeout("blinken_stop()",zeitspanne);    // warten und danach blinken stoppen
        }

        function blinken(ein_aus, text)
        {
                 if (ein_aus)
                 {       window.status  = text;
                         ein_aus=false;          // im nächsten Aufruf die Statuszeile löschen
                 )
                 else
                 (       window.status = "";
                         ein_aus=true;           // im nächsten Aufruf die Stautuszeile setzen
                 )

                 id_blinken = setTimeout("blinken(" + ein_aus + ")", anzeigezeit)
                                                 // nächsten Aufruf nach Ablauf der anzeigezeit starten
        }
//-->
</SCRIPT>
</HEAD>
<BODY ... onLoad="blinken_start('Ich blinke !')">
</BODY>
</HTML>
```

.statusbar — Zeiger auf Objekt statusbar, das die Statuszeile mit folgenden Eigenschaften enthält:
- .visible — ist true wenn Statuszeile im Fenster sichtbar
  veränderbar per signiertem Script

.title — String mit dem physischen Fensternamen, also den Text im Titel, also Titel des HTML-Dokumentes

.toolbar — Zeiger auf das Objekt toolbar, das die Toolbar mit folgenden Eigenschaften enthält:
- .visible — ist true wenn Toolbar im Fenster sichtbar

veränderbar per signiertem Script

.top Zeiger auf das oberste Fenster in der Fenster-Hierarchie
Syntax:
[var Zeiger =] window.top
nur lesen

4.3.2.1.2. Methoden

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

.alert() Dialogbox erzeugen
1. Anzeige von:
Ausrufungszeichen-Symbol
variablen String
OK-Button und
2. warten auf Drücken des OK-Button

Syntax:
alert(String)

String String oder Stringausdruck
in " " bzw. " kodieren
Steuerzeichen wie "\n" zulässig
sonst nur Plain-Text

liefert nichts

.atob() liefert String, der das nach BASE64-Algorithmus kodierte String-Argument enthält
Syntax:
[var Kette2 =] atob(Kette1)

Kette1 String
zu kodierender Text

Kette2 kodierter String

.back() lädt vorhergehendes HTML- Dokument laut History, wobei dabei Frames eines HTML-Dokumentes **nicht** beachtet werden
Syntax:
back()
liefert nichts

.blur() Element den Focus wegnehmen und Event onblur auslösen
Der Focus wird **nicht** automatisch auf irgend ein anderes Element gesetzt !
Syntax:
blur()
liefert nichts

.btoa() liefert String, der das nach BASE64-Algorithmus kodierte String-Argument enthält
Syntax:
[var Kette2 =] atob(Kette1)

Kette1 String
BASE64- kodierter Text

Kette2 dekodierter String

.captureEvents() ordnet dem Window-Objekt Ereignisse zu, die anstelle der Standardbehandlung durch eine per Überschreiben des onXXX-Ereignis-Handlers gleichnamige Funktion (Überschreibung der Objektmethode) oder eine nicht-gleichnamige Funktion behandelt werden sollen
Syntax:
window. captureEvents(event_liste)

event_liste: Folge von Eventbezeichnern
Trennung durch | also logisches Oder
Bsp. für Eventbezeichner

onXXX	Event.XXX
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS

onload	Event.LOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER

liefert nichts

Hinweise zum Eventhandler:
wenn das Ereignis nur einmal bearbeitet und danach wieder der Standardbehandlung zugeordnet werden soll, so muss die Funktion am Ende die Methode .releaseEvents() zum Ereignis aufrufen
wenn das Ereignis nicht durch nachgelagerte Eventhandler in der Handlerhierarchie bearbeitet werden soll, so muss die Funktion mit return false; enden
wenn das Ereignis durch nachgelagerte Eventhandler in der Handlerhierarchie bearbeitet werden soll, so muss die Funktion mit return true; enden

Beispiel:

```
function MouseOverHandler(){..}
    window.onmouseover=MouseOverHandler;
                    // Achtung: nicht () kodieren, da die Funktion sonst
                    //                  sofort aktiviert wird !
    window.captureEvents(Event.MOUSEDOWN);
```

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

Syntax:

```
clearInterval(timer_id)
```

timer_id Integer
ist der Rückgabewert von .setInterval()

liefert nichts

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

Syntax:

```
clearTimeout(timeout_id)
```

timeout_id Integer
ist der Rückgabewert von .setTimeout()

liefert nichts

.close() Fenster schliessen, das offen ist:
Beim Schliessen des ersten Fensters, das mit dem Starten des Browsers angezeigt wird, also beim Schliessen des Browsers, wird automatisch eine Dialogbox angezeigt.
Fenster muss nicht explizit mit open-Methode erzeugt worden sein
wenn Fenster mit .open() erzeugt wurde:
vor .close() ist der logische Windowname zu kodieren, der von
. .open() geliefert wurde, also logischer_window_name.close()
Fenster des Dokumentes schliessen: document.close()
innerhalb von Eventhandler immer **window**.close() anstelle von close() kodieren

Syntax:

```
.close()
logischer_window_name.close()
```

liefert nichts

Beispiel:

```
<SCRIPT LANGUAGE="JScript">
    function Test()
    {window.close();}
</SCRIPT>
<BODY onclick="Test();">
    Klick zum Ausloesen von window.close()
</BODY>
```

Beispiel für Fenster schliesst sich selbst nach Wartezeit:

```
</HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function fenster_offnen_und_schliessen()
    {

        var fenster;
        fenster=window.open("","Fenster","width=180,height=100");
```


```
                        fenster.document.write("<H1>Ich schließe mich nach 4 Sekunden</H1>");
                        fenster.setTimeout('window.close()',4000);  // 4 Sekunden
                }
        //-->
        </SCRIPT>
        </HEAD>
        <BODY onload="fenster_offnen_und_schliessen()">
        </BODY>
        </HTML>
```

Beispiel:

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

        ....

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

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

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

.confirm()

Dialogbox erzeugen mit

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

Syntax:

```
[ var Wert = ] confirm(Kette)

Kette    String

Wert     true für OK
         false für Abbrechen
```

.disableExternalCapture()

schaltet Wirkung von .enableExternalCapture() ab
Ereignisüberwachung in einem Fenster abschalten
Beispiel für Anwendung: Fremdseite in einem Frame kann die Seite nicht überwachen, die die Fremdseite im Frame anzeigt

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

Syntax:

```
disableExternalCapture()
```

liefert nichts

.dump()

nur NS 6.x

.enableExternalCapture()

aktiviert die Kontrolle einer Fremdseite, die in einem Frame dargestellt wird, über die Seite, die den Frame inne hat
Ereignisüberwachung in einem Fenster einschalten
Beispiel für Anwendung: Fremdseite im Frame kann Seite überwachen
Überwachung per .captureEvents() aktivierbar

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

nur mit signiertem Script
Syntax:

```
enableExternalCapture()
```


liefert nichts

.find() Suche im Dokument nach Zeichenkette
Syntax:
[var Wert =] find([zeichenkette],gross_klein,such_richtung)

zeichenkette	wenn nicht kodiert, so automatisch Suchfenster geöffnet
gross_klein	wenn true, so Unterscheidung von Gross und Klein
such_richtung	wenn true, so rückwärts suchen
Wert	true, sobald zeichenkette gefunden

.focus() Focus setzen und Focus-Event auslösen
nur nach dem kompletten Laden des Dokumentes anwendbar
Syntax:
object.focus()
liefert nichts

.forward() lädt nachfolgendes HTML- Dokument laut History, wobei dabei Frames eines HTML-Dokumentes **nicht** beachtet werden
Syntax:
forward()
liefert nichts

.getAttention() nur NS 6.x

.getSelection() aktuelle Selektion referenzieren
nur NS 6.x

Beispiel: Suchmaschine einbinden

```
var markierter_text=document.getSelection();
    // oder var markierter_text=prompt('Suchbegriff: ');
var suchmaschinen_url='http:// .....';
var suchmaschinen_parameter='.....';

if (markierter_text)
{location.HREF=suchmaschinen_url + suchmaschinen_parameter + escape(markierter_text);}
else
{location.HREF=suchmaschine_url; }
```

Beispiel für altavista: suchmaschinen_url 'http://altavista.de/'
suchmaschinen_parameter 'cgi-bin/query?pg=q&what=web&q='

Hinweis: escape() setzt Umlaute und Leerzeichen in korrekte Darstellung um

.handleEvent() Eventhandler aufrufen, der dem Ereignis laut Parameter event_objekt gerade zugeordnet ist
diese Methode wird also innerhalb des Programmcodes eines Eventhandlers
verwendet, der das Ereignis von einem anderen Eventhandler als Argument bekommen hat
verwendet für eine andere Eventverarbeitungs-Hierarchie als die Standardhierarchie
Syntax:
handleEvent(event_objekt_zeiger)
liefert nichts

.home() lädt Startseite laut Browsereinstellung, also nicht die Startseite des aktuellen HTML-Dokument-Projektes
Syntax:
home()
liefert nichts

.moveBy(x,y) Fenster verschieben um Pixeldifferenz
Fenster ganz aus dem BS schieben ist eventuell möglich
Syntax:
moveBy(iX, iY)

iX	Integer, X-Koordinatendifferenz, auch negativ
iY	Integer, Y_Koordinatendifferenz, auch negativ

liefert nichts

Beispiel für Fenster des Browser rausschieben und danach neu anzeigen:

```
<HTML>
```


```
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
        var BrowserFenster_AktuelleBreite=2000;  // sollte größer als max. übliche Auflösung sein
        var BrowserFenster_AktuelleHoehe=2000;  // sollte größer als max. übliche Auflösung sein

        function moveWin()
        {
                for (var i = 1; i < BrowserFenster_AktuelleBreite; i++)
                {window.moveBy(1, 1);}

                window.moveBy((-1)* BrowserFenster_AktuelleBreite,(-1) * BrowserFenster_AktuelleHoehe);
        }
//-->
</SCRIPT>
</HEAD>
<BODY onload="moveWin();">
</BODY>
</HTML>
```

.moveTo() Fenster verschieben auf Pixelpos bezüglich linke obere Ecke des Screen
(Ursprung (0,0) liegt in der linken oberen Ecke)
Fenster ganz aus dem BS schieben ist eventuell möglich
Syntax:
moveTo(x, y)

x horizontale absolute Position in Pixel der linken oberen Fensterecke, Integer >=0
x vertikale absolute Position in Pixel der linken oberen Fensterecke, Integer >=0

liefert nichts

.open() neues Fenster erzeugen als unterstes der aktuellen Fensterhierarchie
und dann oberstes sichtbares Fenster öffnen (anzeigen, rendern)
Fenster mit kodiertem .open() ist der .opener
Syntax:
[var ZeigerAufWindow =] open([URL] [, Name] [, Features] [, Replace])

innerhalb von Eventhandler immer **window**.open() anstelle von open() kodieren

URL String mit Url des Dokumentes, das nach dem Öffnen in das Fenster geladen wird
kann Leerkette sein
Daten per '?' + escape() **nicht** übergebbar

Name String
physische Window-Name: identisch mit Wert laut Attribut
TARGET z.B. im Link
dient der Referenz
auch 'null' kodierbar

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

alwaysLowered = { yes | no }

yes: Fenster öffnen und permanent als
unterstes in der Fensterhierarchie
belassen
nur per signiertem Script
beachte .setZOptions()

alwaysRaised = { yes | no}
yes: Fenster öffnen und permanent als
oberstes in der Fensterhierarchie belassen
nur per signiertem Script
beachte .setZOptions()

dependent = { yes | no }
yes: Fenster an den .opener bezüglich
Fensterschliessen koppeln: auch
schliessen, wenn .opener geschlossen
wird
sowie geöffnetes Fenster nicht in
der Taskleiste von Windows
anzeigen

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

height = Pixelwert, Fensterhöhe
nur mit signiertem Script < 100
ohne signiertem Script: wenn < 100,
so auf 100 automatisch gesetzt

hotkeys = { yes | no}
yes: alle sicherheitsrelevanten Tasten
verwendbar machen
no: nur noch ALT+F4 funktioniert
(schliessen des Fensters)

innerHeight= anzeigebereich_höhe_in_pixel
ohne signiertes Script: >= 100
Anzeigebereich = Fenster abzüglich
Leisten, Scrollbars etc.

innerWidth= anzeigebereich_breite_in_pixel
ohne signiertes Script: >= 100
Anzeigebereich = Fenster abzüglich
Leisten, Scrollbars etc.

location = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw 1 für URL-Eingabezeile anzeigen
(Adresszeile anzeigen)
menubar = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Menübar anzeigen
(Leiste der Pull-DownMenüs anzeigen)

outerHeight= fenster_höhe_in_pixel
ohne signiertes Script: >= 100
Fenster = Anzeigebereich sowie Leisten,
Scrollbars etc.

outerWidth= fenster_breite_in_pixel
ohne signiertes Script: >= 100
Fenster = Anzeigebereich sowie Leisten,
Scrollbars etc.

personalbar = { yes | no }
yes: persönliche Toolbar anzeigen

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

Fensters erlaubt per Mausziehen
beachte .setResizable()

screenX= horizontale_pixel_pos des Fensters
bezüglich dem Bildschirm,
dessen Ursprung (0,0) in der
linken oberen Ecke liegt

screenY= vertikale_pixel_pos des Fensters
bezüglich dem Bildschirm,
dessen Ursprung (0,0) in der
linken oberen Ecke liegt

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

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

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

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

width = Pixelwert, Fensterbreite,
nur mit signiertem Script < = 100
ohne signiertem Script: wenn < 100,
so auf 100 automatisch gesetzt

z-lock = { yes | no}
yes: Fenster kann kein anderes
überlagern
nur per signiertem Script
beachte .setZOptions()

Replace true, so History-Eintrag des Dokumentes, in dem das neue
Fenster erzeugt wird, ersetzen durch den Eintrag
des neuen Fensters, also Zurück zum Dokument,
dass das Fenster öffnet, nicht möglich
false, so neuen History-Eintrag zum neuen Fenster
erzeugen, also zurück zum alten Fenster möglich

ZeigerAufWindow
Referenz (ID, logischer Windowname) auf das neue
Fenster z.B. für close-Methode

Beispiel window.open("Sample.htm",null,"height=200,width=400,status=yes,toolbar=no,menubar=no,location=no");

Beispiel:

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

....

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

var Kette=    '<HTML>'
```


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

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

.print() ruft Dialog-Box-Druckerfenster auf zum Druck des Dokumentes im aktuellen Fenster (Frame)
entspricht Druckbutton bzw. Datei-Menü-Drucken
löst folgende Ereignisse aus:
onbeforeprint
onafterprint
Syntax:
print()
liefert nichts

Beispiel für Verwendung der Ereignisse per Handler
onbeforeprint Handler blendet Teile der Seite ein/aus, die gedruckt werden sollen
onafterprint Handler hebt Veränderungen von onbeforeprint wieder auf

Beispiel für Ausdruck des aktuellen Dokumentes:

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

.prompt() Eingabefenster erzeugen, Meldungstext anzeigen, Eingabezeile vorbelegen und
anzeigen, OK- bzw. CANCEL-Button anzeigen und auf Eingabe
in die Zeile und anschliessendem Druck auf OK warten
Syntax:
[var Kette =] prompt("meldungstext"[,"vorgabe_text"])

meldungstext	String, frei wählbar
vorgabe_text	String, frei wählbar wenn nicht kodiert, so "undefined" als Standard
Kette	Zeichenketten-Ergebnis der Eingabe

Beispiel: var eingabe = prompt(.....);

.releaseEvents() ordnet dem Window-Objekt Ereignisse zu, die durch die Standardbehandlung bearbeitet werden
hebt zugleich ein aktuelles .captureEvents() zum Ereignis auf
Syntax:
releaseEvents(event_liste)

event_liste: Folge von Eventbezeichnern
Trennung durch | also logisches Oder
Bsp. für Eventbezeichner

onXXX	Event.XXX
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS
onload	EventLOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER

liefert nichts

.resizeBy() Fenstergröße um Pixeldifferenz verändern
funktioniert nur, wenn open-Merkmal resizable=yes kodiert wurde
Syntax:
resizeBy(x,y)

x ist horizontale Spanne in Pixel , Integer, auch negativ
y ist vertikale Spanne in Pixel , Integer, auch negativ
Fenstergröße auf weniger als 100 x 100 nur mit signiertem Script
liefert nichts

.resizeTo() Fenstergröße neu dimensionieren
Syntax:
resizeTo(x, y)

x ist Breite in Pixel, Integer, >0
y ist Höhe in Pixel, Integer, > 0
Fenstergröße auf weniger als 100 x 100 nur mit signiertem Script
liefert nichts

Beispiel für Fensterauflösung ändern:

javascript:window.resizeTo(**640,480**); javascript:window.resizeTo(**800,600**); javascript:window.resizeTo(**1024,768**)

Beispiel für sich aufblasendes Fenster von 100x100 bis auf 640x480

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
	var start_hoehe=100;
	var start_breite=100;
	var max_hoehe=480;
	var max_breite=640;
	var aktuelle_hoehe=start_hoehe;
	var aktuelle_breite=start_breite;
	var y=5;
	var TimerID=null;
	var fenster;

function start()
{
	fenster=window.open("","","scrollbars");

	if ( document.layers  || document.all)
	{
		fenster.resizeTo(start_breite,start_hoehe);
		fenster.moveTo(0,0);
		blasen();
	}
	else
	{alert("Weder Netscape noch Microsoft erkannt !")};
}

function blasen()
{
	if (aktuelle_hoehe>=max_hoehe)
	{x=0;}

	fenster.resizeBy(5,y);
	aktuelle_hoehe+=5;
	aktuelle_breite+=5;

	if (aktuelle_breite>=max_breite)
	{
		alert("Maximal aufgeblasen !");
		fenster.close();
		x=5;
		TimerID=null;
	}
	else
	{TimerID=setTimeout("blasen()",50);}
}
//-->
</SCRIPT>
</HEAD>
```


```
<BODY>
<A      HREF="javascript:start()"
        onMouseOver="javascript:window.status='Oeffne Fenster';return true;"    // Text nach Statuszeile
        onMouseout="javascript:window.status='';"                               // Statuszeile löschen
>Oeffne NEUES Fenster mit 100x100 Pixel und blase es auf  640x480 Pixel !
</A>
</BODY>
</HTML>
```

.routeEvent()

aktiviert das Weiterreichen des Events zum in der Eventhierarchie untergeordneten Eventhandler
standardgemäß landen die Events immer beim Window-Objekt und werden dort behandelt, also nicht nach unten durchgereicht
Syntax:
routeEvent(ereignis_objekt_zeiger)
liefert nichts

Beispiel

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript">
<!--
        function MouseDownFuerInputButton()
        { ....  }

        function OnMouseDownEventWeiterreichen(event)
        {window.routeEvent(event);}

        window.onmousedown= OnMouseDownEventWeiterreichen;
                //  Achtung: ohne () kodieren, sonst wird Funktion sofort ausgeführt
        window.captureEvents(Event.MOUSEDOWN);
// -->
</SCRIPT>
</HEAD>
<BODY>
        <FORM>
                <INPUT   TYPE=button
                         VALUE="Weiterreichen"
                         onmousedown= "MouseDownFuerInputButton();"
                >
        </FORM>
</BODY>
</HTML>
```

Ablauf:

Ablauf im Script-Teil:
Der Script-Teil im Head wird zuerst abgearbeitet, also gilt:
Mousedown-Event durch OnMouseDownEventWeiterreichen bearbeiten lassen und nicht durch Standardbehandlung, also Ereignis weiterrunter leitbar.
Hinweis: Standardbehandlung erfolgt durch das Window-Objekt, das Ereignisse nicht weiter runter leitet.

Ablauf im Body-Teil:
Es wird auf das Input-Button gedrückt.
Wegen captureEvents() wird das Mausereignis ZUERST von OnMouseDownEventWeiterreichen bearbeitet. Damit wird auf das Weiterleiten aktiviert.
Das untergeordnete Element ist der Input-Button. Damit wird MouseDownFuerInputButton aktiviert.

Hinweis: Dieses Beispiel hinkt, da die Standardbehandlung letztendlich das gleiche bewirkt aber auf anderen Wegen.

.scroll()

deprecated und zu ersetzen durch scrollBy bzw. .scrollTo()
linke obere Ecke des Dokumentes verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters, also scrollen
bewirkt Veränderung von .pageXOffset und .pageXOffset
Hinweis: wenn Dokumentdimension > Anzeigebereichdimension, so ist Scrollen nötig
Syntax:
scroll(x, y)

x	horizontal scroll Offset in Pixels, Integer, > 0
y	vertikal scroll Offset in Pixels, Integer, > 0

liefert nichts

.scrollBy()

linke obere Ecke des Dokumentes verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters, also scrollen

ist anstelle von .scroll() zu verwenden
bewirkt Veränderung von .pageXOffset und .pageXOffset
Hinweis: wenn Dokumentdimension > Anzeigebereichdimension, so ist Scrollen nötig
Syntax:
scrollBy(x, y)

x horizontal scroll Offset in Pixels
Integer
> 0 so Verschiebung nach rechts
< 0 so Verschiebung nach links

y vertikal scroll Offset in Pixels
Integer
> 0 so Verchiebung nach unten
< 0 so Verschiebung nach oben

liefert nichts

.scrollByLines() nur NS 6.x

.scrollByPages() nur NS 6.x

.scrollTo() linke obere Ecke des Dokumentes verschieben auf neue Pixelposition bezüglich linker oberer Ecke des Fensters, also scrollen
ist anstelle von .scroll() zu verwenden
bewirkt Veränderung von .pageXOffset und .pageXOffset
Syntax:
scrollTo(x, y)

x horizontale Position in Pixel
Integer, >=0

y vertikale Position in Pixel
Integer, >=0

liefert nichts

.scrollX nur NS 6.x

.scrollY nur NS 6.x

.setActive() Objekt für die Eventdurchreichung aktivieren, aber ohne es zu fokussieren und ohne es scrollbar zu machen
Syntax:
setActive()
liefert nichts

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    var ID_Fenster;

    function FensterErzeugen()
    {
        ID_Fenster = window.open( "test.htm",
                                  " ID_Fenster",
                                  "top=10px,left=480px,height=375px,width=200px,resizable=1"
                                );
        this.focus();
    }

    function ButtonAktivieren()
    {window.parent.ID_Fenster.ID_Button.setActive();}
</SCRIPT>
</HEAD>
<BODY onload="FensterErzeugen();">
    <BUTTON ID="ID_Button" onclick="ButtonAktivieren();">
        Button aktivieren
    </BUTTON>
</BODY>
</HTML>
```

.setCursor() nur NS 6.x

.setHotKeys() Hotkeys des Fensters (außer im Menü) aktivieren bzw. deaktivieren
Standard: immer aktiviert
Syntax:
setHotKeys(Wert)

Wert	true, so Hotkeys aktiviert false, so Hotkeys deaktiviert

liefert nichts

.setInterval() endlos-periodischer Aufruf eines Codes mit jeweiligem vorherigen Warten in Millisekunden (Timer)
(getimte Rekursion)
Der mit setInterval() aufgerufene Code wird zyklisch aktiviert, wobei nach dem ERSTEN Aufruf von setInterval() mit der Folgeanweisung **hinter** .setInterval() sofort weitergemacht wird, während die Rekursion parallel läuft. Damit gilt: setInterval() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da sonst diese Anweisungen während der parallelen rekursiven Ereignisüberwachung bereits abgearbeitet werden.
Syntax:
[var TimerID =] setInterval(Code, MilliSeconds [, Language])

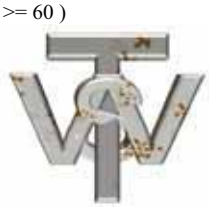
Code	auszuführender Code bzw. Zeiger auf Codebaustein Zeiger empfehlenswert, wenn Code in externer Datei liegt String empfehlendwert, wenn Code im aktuellen Dokument liegt Übergabe von Argumenten möglich
MilliSeconds	Integer, Wartezeit, erst nach deren Ablauf wird Code aktiviert
Language	String als Sprache des Codes (analog zum LANGUAGE-Attribut) z.B. "JavaScript".
TimerID	Integer ID für den Stopp der periodischen Aufruffolge mit der Methode .clearInterval()

Beispiel 1

String
```
window.setInterval("someFunction()", 5000);
```
Zeiger
```
window.setInterval(someFunction, 5000);
```

Beispiel 2
```
function callback1(){alert("callback1");}

function callback2(){alert("callback2");}

function chooseCallback(Nummer)
{
    switch (Nummer)
    {
        case 0:  return callback1;
        case 1:  return callback2;
        default: return "";
    }
}
var Nummer = 1;
window.setInterval(chooseCallback(Nummer), 5000);
```

Beispiel 3
```
var SekundenZahler=0;
var timer_id=null;

function ZyklischeAktion()
{
    SekundenZahler++;
    window.status= SekundenZahler + " Sekunden ";

    if (    ( SekundenZahler >= 60 )
```

```
                    &&  ( timer_id != null)
                   )
                {window.clearInterval(timer_id);}
        }

        timer_id=window.setInterval(ZyklischeAktion,1000);
```

Beispiel 4

```
var SekundenZahler=0;
var timer_id=window.setInterval("window.status= SekundenZahler++",1000);
```

.setResizeable() Veränderbarkeit der Fenstergröße aktivieren/ deaktivieren

Syntax:

setResizeable(Wert)

Wert	true für Veränderbarkeit aktiv false für Veränderbarkeit nicht möglich

liefert nichts

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

Der mit setTimeout() aufgerufene Code wird **nicht zyklisch** aktiviert, wobei nach dem Aufruf von setTimeout() mit der Folgeanweisung hinter setTimeout() sofort weitergemacht wird. Damit gilt: setTimeout() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da diese Anweisungen bereits während der Ereignisüberwachung abgearbeitet werden.

Syntax:

[var TimerID =] setTimeout(Code, MilliSeconds [, Language])

Code	auszuführender Code bzw. Zeiger auf Codebaustein Zeiger empfehlenswert, wenn Code in externer Datei liegt String empfehlendwert, wenn Code im aktuellen Dokument liegt Übergabe von Argumenten möglich
MilliSeconds	Integer, Wartezeit, erst nach deren Ablauf wird Code aktiviert
Language	String als Sprache des Codes (analog zum LANGUAGE-Attribut) z.B. "JavaScript"
TimerID	Integer ID für den Stopp der periodischen Aufruffolge mit der Methode .clearTimeout() wird nur benötigt, wenn der Code auf eine rekursive Funktion weist, wobei der Zyklusabbruch durch .clearTimeout() erfolgen muss Hinweis: ist keine Rekursion nötig, soll aber zyklisch aufgerufen werden, dann .setInterval() verwenden

Beispiel 1

```
window.setTimeout("alert('Hallo')", 1000);
```

Beispiel 2

```
var Text = "Hallo ";
window.setTimeout(    "alert("
                        + Text
                        + ")",
                        1000
                      );
```

Beispiel 3

```
<SCRIPT>
        function Start(ZeigerAufObjekt)
        {window.setTimeout("Kode(" + ZeigerAufObjekt.id + ")", 3000);}

        function Kode(ID)
        {
                var Zeiger = eval(ID);
                Zeiger.style.display="none";
        }
```


```
</SCRIPT>
<INPUT TYPE=button VALUE="Unsichtbar in 3 Sekunden"
        ID="ID_Button" onclick=" Start(this)"
```

Beispiel 4

```
var SekundenZahler=0;
var timeout_id=null;

function ZyklischeAktion()
{
        SekundenZahler++;
        window.status= SekundenZahler + " Sekunden ";

        if (    ( SekundenZahler >= 60 )
            &&  ( timeou_id != null)
           )
        {window.clearTimeout(timeout_id);}
}

timeout_id=window.setTimeout(ZyklischeAktion,1000);
```

Beispiel 5

```
var timeout_id=window.setTimeout("window.status= 'Es ist 1 Sekunde vergangen !'",1000);
```

.setZOptions() Art und Weise der Fensterüberlagerung unabhängig vom Aktivzustand des Fensters definieren
Syntax:
setZOptions(zustand)

zustand: Zeichenkette mit folgenden Werten
"alwaysRaised" Fenster immer oberstes auch wenn nicht aktiv
"alwaysLowered" Fenster immer unterstes auch wenn aktiv
"z-lock" Fenster bleibt in aktueller Überlagerungssituation egal ob aktiv oder nicht

liefert nichts

.sizeToContent () nur NS 6.x

.stop() stoppt das Laden des Dokumentes in das Fenster
Syntax:
stop()
liefert nichts

.updateCommands() nur NS 6.x

4.3.2.1.3. window.document Objekt des Netscape

Erzeugung:
durch den Browser

Das HTML-Dokument
umfasst sämtlichen Code **zwischen** <HTML...> und </HTML>
ist der Container für die Elemente der Webseite z.B. BODY oder DIV
Code hinter </HTML> wird nicht geparst aber eventuell als Plain-Text angezeigt
Ein Dokument muss zwar in ein Fenster geladen werden, aber das Dokument wird nur dann visualisiert (gerendert), wenn es sichtbare Elemente besitzt.

Ein HTML-Element beim IE und NS:
als HTML-Tag: Der IE und NS unterstützen z.T. verschiedene Tags
als Objekt: Es gibt Objekte, die im IE **und** NS implementiert sind, aber z.T. auf verschiedene Weise (inkl. Syntax)

Collectionen zum Objekt document:
Zur Verwaltung des Dokumentes existieren Collectionen als Felder, die zugleich als Schnittstelle zum Programmierer dienen.
Der Netscape besitzt teilweise Collectionen, die auch der Internet Explorer bedient.

Objekt document und Browserfenster:
Der **logische** Fenstername muss nur dann kodiert werden, wenn der Bezug nicht auf das aktuelle Dokument gehen soll.
Der logische Fenstername stammt aus
der Methode .open() zum window Objekt und kann ein frei festgelegter oder vordefinierter Name sein (vordefiniert bei automatischen Öffnen eines Fensters)
aus den ID-Attributen bei FRAMESET mit FRAMES.
Die Kodierung des Zeigers von document kann dann entfallen, wenn das aktuelle Dokument referenziert wird.
Allerdings ist von der Kodierung ohne window bzw. ohne window.document abzuraten, denn

es könnten gleichnamige Eigenschaften und Methoden geben, die in anderen Objekten auch implementiert sind
es wird damit die Browserperformance gesenkt.

Zugriff:

window.document.eigenschaft	oder	document.eigenschaft	oder	eigenschaft
window.document.methode()	oder	document.methode()	oder	methode()

logischer_window_name.document.eigenschaft
logischer_window_name.document.methode()

logischer_window_name laut open()

Der **logische** Fenstername muss nur dann kodiert werden, wenn der Bezug nicht auf das Dokument im aktuellen Fenster gehen soll.
Der logische Fenstername stammt aus
der Methode .open() zum window Objekt und kann ein frei festgelegter oder vordefinierter Name sein (vordefiniert bei automatischen Öffnen eines Fensters)
aus den ID-Attributen bei FRAMESET mit FRAMES.
Die Kodierung des Zeigers von document kann dann entfallen, wenn das aktuelle Dokument referenziert wird.
Allerdings ist von der Kodierung ohne window bzw. ohne window.document abzuraten, denn
es könnten gleichnamige Eigenschaften und Methoden geben, die in anderen Objekten auch implementiert sind
es wird damit die Browserperformance gesenkt.

Eigenschaften:
.alinkColor
.bgColor
.characterSet
.cookie
.defaultView
.dir
.doctype
.documentElement
.domain
.expando
.fgColor
.firstChild
.height
.implementation
.lastChild
.lastModified
.linkColor
.location
.localName
.namespaceURI
.nextSibling
.nodeName
.nodeType
.nodeValue
.ownerDocument
.parentNode
.prefix
.previousSibling
.referrer
.title
.URL
.vlinkColor
.width

Methoden:
.captureEvents()
.close()
.createElement()
.createTextNode()
.getElementById()
.getElementsByName()
.getElementsByTagName()
.mergeAttributes()
.open()
.releaseEvent()
.routeEvent()
.write()
.writeln()

4.3.2.1.3.1. Collectionen zum Objekt window.document des Netscape

Nachfolgende Collectionen dienen der Verwaltung des HTML-Dokumentes bzw. verschiedener HTML-Elemente.
In den Syntax-Beschreibungen wird aus Gründen der Vereinfachung immer vom aktuellen Fenster ausgegangen. Damit erfolgt nicht die Kodierung von window.document sondern nur document, da beide Formen für das aktuelle Fenster synonym sind.

4.3.2.1.3.1.1. window.document.anchors Collection des Netscape

Feld der Zeiger aller Anker im Dokument
Feldelementefolge laut HTML-Coding
Erzeugung:
durch den Browser

Beispiel für Anker:

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

Hinweis zu HREF= :

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

zu Anspringen eines Ankers:
        den Wert laut NAME ablegen in
                window.location.hash    ohne vorgesetztes  #
                window.location.href     mit   vorgesetztem #
```

Syntax:

```
[ var ZeigerAufFeld = ] document.anchors
[ var ZeigerAufFeldElement = ] document.anchors[index]

        index           Integer ab 0
                        muss in [ ] kodiert sein

ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()
```

Eigenschaften:

```
.length         Anzahl der Feldelemente, ab 1
```

4.3.2.1.3.1.2. window.document.applets Collection des Netscape

Feld der Zeiger aller Applet-Objekte im Dokument
Elementefolge laut HTML-Coding
Applet muss alle Eigenschaften und Methoden, die im HTML-Dokument benutzt werden sollen, als **public** deklarieren
Erzeugung:
durch den Browser

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

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

Syntax:

```
[ var ZeigerAufFeld = ] document.applets
[ var ZeigerAufFeldElement = ] document.applets[index]

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


ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()

document.**ID_Applet**.eigenschaft
document.**ID_Applet**.methode()

document.**Name_Applet**.eigenschaft
document.**Name_Applet**.methode()

Eigenschaften:
.length Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.3. window.document.cookie Collection des Netscape (Cookie-Verwaltung im HTML-Dokument)

Feld der Zeiger auf alle Cookies zum Dokument
Ein Cookie kann bis zu 4 KBytes groß sein.
Es sind bis zu 300 verschiedene Cookies speicherbar.

Wenn Cookieverwaltung nicht erwünscht wird, so muss sie im Browser abgeschaltet werden --> eventuell laufen dann Webseiten nicht mehr korrekt ! Alternativ: Nach der Onlinesitzung die regelmäßige Löschung des **Inhaltes** des Cookie-Ordners **und** des Browser-Cache (Ordner selbst auf keinen Fall löschen !).
Nutzung Firewall

Cookies haben den gleichen Aufbau aber verschiedene Inhalte.
Erzeugung:
durch den Browser
Zugriff:
[var ZeigerAufFeld =] document.cookie
[var ZeigerAufFeldElement =] document.cookie**[**index**]**

index Integer ab 0
muss in **[ ]** kodiert sein

Eigenschaften:
.length Anzahl der Feldelemente, ab 1
Methoden:
keine
Auf den Wert des Cookies sind alle Methoden des String-Objektes anwendbar, da der Wert immer eine Zeichenkette ist.
Aufbau eines Cookie:
Cookies haben den gleichen Aufbau aber verschiedene Inhalte.
Optionale Eigenschaften müssen ein Semikolon vorgesetzt bekommen und können beliebig kombiniert werden.

Mindestdeklaration eines Cookie (Wert eines Cookies):

```
document.cookie[0] = wert_des_cookie_1;

oder  var cookie_feld = document.cookie;
      cookie_feld[0] = wert_des_cookie_1;
```

wert_des_cookies: Zeichenkette
kann HTML-gerecht kodierte Sonderzeichen enthalten
darf kein Blank enthalten, da jedes Cookie intern als Endekennzeichen ein Blank besitzt
Aufbau:
freierwert[;eigenschaften_liste]

Semikolon trennt die optionale Eigenschaftenliste ab
Eigenschaften in der Liste durch Semikolon trennen

freierwert: cookie_titel=freier_wert

= ist festkodiert

Bsp: "Filmbeginn=23:15;path=...;expires=....."

Anwendungsbereich des Cookie also nicht die Lage auf der Festplatte der Users:

path=pfad_angabe
path= ist festkodiert
pfad_angabe:
Bsp.: 'Otto=der_erste;c:\privat\html_files\'
also Anwendbarkeit nur möglich, wenn HTML-Dokument innerhalb bzw. unterhalb von c:\privat\html_files liegt
wenn nicht kodiert, wo wird der Pfad des Browser-Cache verwendet

Servergruppezugehörigkeit:

domain=server_name_fragment

domain= ist festkodiert
server_name_fragment:
Bsp.: "Otto=der_erste;domain=online.de"
also alle Server, die im Namen den Rest online.de haben
wenn nicht kodiert, so wird der volle Name von dem Server abgelegt, der das Cookie verwaltet und nutzt

Verfallsdatum:

expires=datum_angabe

expires= ist festkodiert
datum_angabe: wochentag, tt-mm-jj hh-mm-ss nach Greenwich-Time
Bsp.: var loeschdatum = new Date();
"Otto=der_erste;epires= " +
loeschdatum.toGMTString;
wenn nicht kodiert, so Cookie mit Ende der Onlinesitzung von der Festplatte des Users automatisch gelöscht
wenn kodiert, so am Verfallsdatum bzw. danach von der Festplatte des Users automatisch gelöscht (danach, wenn der User den PC nach dem Verfallsdatum erst wieder nutzt)

Cookie nur senden, wenn HTTPS-Protokoll also sichere Verbindung benutzt wird:

secure
kein Wert

Beispiel für Cookie verwalten:

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

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

    function LeseCookieWert(Name)
    // Name des zu lesenden Cookie ist String
    // liefert Cookiewert aus unescape() als String
    {
        var CookieWert="";   // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

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

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


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

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

                                        return CookieWert;
                        }
                </SCRIPT>
```

4.3.2.1.3.1.4. window.document.embeds Collection des Netscape

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

Diese Collection ist ein Alias für die plugins Collection **nur** bezüglich von Plugins (und nicht anderen einbettbaren Objekten), wobei letztere nur der Kompatibilität mit anderen plugin-fähigen Browsern dient. Alle eingebetteten Objekte haben in document.embeds ihren Zeiger (inklusive Plugins, Ausnahme: siehe tiefer).

Das Ansprechen von Plugins kann über die Collection navigator.mimeTypes per Eigenschaft .enabledPlugin erfolgen, die einen Zeiger auf das installierte Plugin enthält (wenn nicht installiert, so null-Zeiger). Wenn der Zeiger nicht null ist, dann sind die plugin-eigenen Methoden und Eigenschaften über Punktnotation ansprechbar.

Erzeugung:
durch Browser
Beispiel:

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

Syntax:

```
[ var ZeigerAufFeld = ] document.embeds
[ var ZeigerAufFeldElement = ] document.embeds[index]

        index           Integer ab 0
                        muss in [ ] kodiert sein

ZeigerAufFeldElement.eingenschaft
ZeigerAufFeldElement.methode()
```

Eigenschaften:

.length	Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.5. window.document.forms Collection des Netscape

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

Syntax:

```
document..forms[index]

        Index           Integer, ab 0
                        muss in [ ] kodiert sein
```

Eigenschaften:

.length Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.6. window.document.frames Collection des Netscape

Feld der Zeiger aller Frames
Elementefolge laut HTML-Coding
Element ist eine Referenz auf das Fenster mit Frame-Eigenschaften

Syntax:

[var ZeigerAufFeld =] document.frames
[var ZeigerAufFeldElement =] document.frames**[**index**]**

index Integer ab 0
muss in **[ ]** kodiert sein

Beispiel für Inhalte zweier Frames tauschen:

Kodierung im Dokument, dass die beiden Frames definiert !

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

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

Eigenschaften:

.length Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.7. window.document.images Collection des Netscape

Feld der Zeiger aller img Objekte (auch INPUT mit Image)
Elementefolge laut HTML-Koding

Syntax:

document..images**[**index**]**

index Integer, ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

.length Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.8. window.document.layers Collection des Netscape (nicht mehr ab Version 6.x)

Feld der Zeiger aller Layer-Objekte im Dokument

Syntax:

document..layers**[**index**]**

index Integer, ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

.length Anzahl der Feldelemente, ab 1

Beispiele:

unverschachtelter Layer:

```
<LAYER ID="ID_Layer" STYLE="position: absolute;">Ich bin eine Ebene</LAYER>
```

Layer-Zugriff per Script:

document. **ID_Layer**
oder document.layers[**"ID_Layer"**]
oder document.layers[0]

Verwendung eines numerischen Indexes:

Die Indexfolge ist nicht die Folge laut HTML-Kodierung (außer Indexwert 0 = unterster Layer)
dafür z-index-Attribut kodieren und dieses verwenden

Anzahl der obersten Ebenen auf gleichem Level also ohne deren innere Layer:

document.layers.length

verschachtelte Layer:

```
<LAYER ID="ebene1" STYLE="position: absolute; left: 100; top: 100;">
```


```
                Nur ein Test...
                <LAYER ID="ebene2" STYLE="position: absolute; left: 50; top: 50;">
                        Jaja!
                </LAYER>
        </LAYER>
```

es wird vererbt: Bsp.: Verschiebung des äußeren Layers (Eltern) verschiebt den inneren (Kind),
wobei innerer in gleicher relativer Position zum äußeren bleibt wie vor der Verschiebung.

document.layers.length liefert 1

Zugriff auf inneren Layer:

```
document.ebene1.document.ebene2
```

Den Inhalt von Ebenen ändern:

```
document.ebene1.document.open();                        // anzeigen
document.ebene1.document.write("Ein neuer Inhalt");     // füllen
document.ebene1.document.close();                       // und schliessen
                                                        // bleibt aber sichtbar
```

oder

```
with(document.ebene1.document.ebene2)
{
        open();
        write("So geht's auch");
        close();
}
```

Layer dynamisch erzeugen NACH dem kompletten Laden des Dokumentes
z.B. per <BODY onload= .....>

Achtung:
Ein Versuch, Ebenen direkt im HEAD per Script zu erzeugen, wird einfach ignoriert, wenn der BODY-Teil des HTML-Dokumentes nicht bereits komplett geparst wurde.

Bsp.:

```
var neueEbene=new Layer(breite_in_Pixel);           // instanzieren

neueEbene.document.open();                          // anzeigen
neueEbene.document.write("test");                   // und füllen
neueEbene.document.close();                         // und schliessen
                                                    // bleibt aber sichtbar !!
```

Inhalt einersLayers:

laden	Aufruf der Methode "load" oder neueEbene.src="seite1.html";
anzeigen	neueEbene.visibility="show";

verschachtelte Layer dynamisch erzeugen

```
var neueEbene=new Layer(breite_in_Pixel);           // instanzieren
var neueEbene1=new Layer(breite_in_pixel, document.neueEbene);
```

Eigenschaften von Layer lesen und/oder ändern
z.B. X-Position document.neueEbene.left=200; // Pixel
Y-Position document.neueEbene.top= 300; // Pixel

Layer verschieben:

```
Bsp.        document.ebene1.moveBy(50, 20);
oder  document.ebene1.left+=50;  document.ebene1.top+=20;
```

4.3.2.1.3.1.9. window.document.links Collection des Netscape

Feld der Zeiger aller Objekte mit HREF-Eigenschaft (Attribut) sowie aller AREA-Objekte im Dokument
Elementefolge laut HTML-Coding

Syntax:

document.links**[**index**]**

index Integer, ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

.length Anzahl der Feldelemente, ab 1

4.3.2.1.3.1.10. window.document.plugins Collection des Netscape

siehe navigator.plugins Collection des Netscape

4.3.2.1.3.2 Objekte des HTML-Dokumentes des Netscape (Auswahl)

In den Beschreibungen wird aus Gründen der Vereinfachung immer vom aktuellen Fenster ausgegangen. Damit erfolgt nicht die Kodierung von window.document sondern nur document, da beide Formen für das aktuelle Fenster synonym sind.

Objekte sind in Script z.B. per document.write() bzw. document.writeln() oder per HTML-DOM-Methoden erzeugbar.

4.3.2.1.3.2.1. window.document.body Objekt des Netscape

Rumpf des HTML-Dokumentes

Erzeugung in HTML:

Beispiel:

```
<BODY....>
        ....
</BODY>
```

Zugriff:

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

Hinweise zu den Methoden document.write() und document.writeln():

Diese Methoden schreiben immer in den komplett geparsten BODY-Teil des Dokumentes und können neben Text und HTML-Code auch Scriptcode schreiben. Im Falle von Scriptcode gilt: Mit der Abarbeitung dieser Methoden bewirkt das Schreiben des Scriptcodes in den Body immer die **sofortige** Ausführung des Scriptcodes (analog zur Methode eval(), die aber keinen HTML-Code ausführen kann).

Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:

Die **zuletzt** während der Laufzeit getätigte Wertsetzung zu einem Attribut definiert dessen **aktuellen** Attributwert.

4.3.2.1.3.2.2. window.document.frame Objekt des Netscape

Erzeugung unter HTML:

Beispiel:

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

Rahmen zwischen den einzelnen Fenstern des Framesets entfernen:

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

Hinweise:

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

Rahmenbreite
>=0 Pixel

Zugriff:

document.**ID_Frame**.eigenschaft
document.**ID_Frame**.methode()

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

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

Eigenschaften (Auswahl):

.name	entspricht NAME
.length	entspricht logischer_window_name.frames.length
.parent	Fenster, das das Frame-Objekt erhält
.self	aktueller Frame

Methoden (Auswahl):

.blur()	deaktiviert Frame
.focus()	aktiviert Frame

Eventhandler (Auswahl):

onLoad
onUnload
onMove
onResize

Collectionen:

window.document.frames Collection
Feld der Zeiger aller Frames
Elementefolge laut HTML-Coding
Element ist eine Referenz auf das Fenster mit Frame-Eigenschaften

Syntax:

[var ZeigerAufFeld =] document.frames
[var ZeigerAufFeldElement =] document.frames**[**index**]**

index Integer ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

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

Möglichkeiten des Laden eines Dokumentes:

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

Laden eines fremden Dokumentes ohne Framedarstellung:

```
<HTML>
	<HEAD>
		<SCRIPT  LANGUAGE="JavaScript">
		<!--
			function laden()
			{ location.href = "http://www.test.de";}
		// -->
		</SCRIPT>
	</HEAD>

	<BODY>
		<FORM>
			<INPUT  TYPE="button"
				VALUE="www.test.de anwaehlen "
				onclick="laden()">
		</FORM>
	</BODY>
</HTML>
```

Eigenes Dokument wird durch fremde Webseite geladen:

CopyRight-Meldung auf fremden Host erzeugen:

Beispiel 1:

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


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

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

Beispiel 2:

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

        var  fremder_host_name = parent.location.hostname;

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

Framedarstellung der eigenen Webseite sofort und ohne Bildschirmmeldung aktivieren:

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

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

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

        if (        (parent != null) // Elternobjekt existiert
             && (parent != self) // Eltern sind vorhanden Eltern: dieses Dokument (self) ist ein Kind
```


```
                                        )
                        {
                                // aktuellen ElternHost ermitteln
                                var ElternHost=parent.location.hostname;

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

Reload eines Dokumentes mit allen seinen FRAMES:

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

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

Datei ohne FRAMESET in eine Datei mit FRAMESET laden

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

<u>Mehrere Frameinhalte gleichzeitig ändern</u>:

Inhalte zweier Frames tauschen:

Kodierung im Dokument, dass die beiden Frames definiert !

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

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

Inhalte beliebig vieler Frames als Ring tauschen:

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

Tausch:

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

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

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

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

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

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

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

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

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

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

Frame und Datenaustausch:

Zwischen Frames können Daten ausgetauscht werden.

Beispiel Adressierung eines Frame über das FRAMESET

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

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

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

im FRAMESET-Dokument wird kodiert var Kette="Hallo !";

im FRAME-Dokument wird auf Kette zugegriffen: alert(parent.Kette);

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

im FRAME-Dokument wird kodiert var Kette="Hallo !";
<FRAME ...NAME ="test2" ...>

im FRAMESET-Dokument wird auf Kette zugegriffen: alert(parent.test2.Kette);

Beispiel: Textdaten-Übergabe durch an Url angehängte Textdaten

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

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

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

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

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

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

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

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

<BODY onLoad=" datenuebernahme ()">
 Von quelle.htm &uuml;bernommener Text:
```


```
                <FORM NAME=ausgabe>
                        <TEXTAREA       NAME=ausgabefeld
                                        ROWS=10 COLS=40>
                        </TEXTAREA>
                </FORM>
        </BODY>
        </HTML>
```

Beispiel: Textdaten-Übergabe durch Fenster-Handle

Die Textdaten und die der Javascript-Code, der die Textdaten verarbeitet, sind **getrennte** HTML-Dokumente. Nachteil dieser Variante ist, dass in beiden Dokumenten die logischen Namen vom Formular und dem Textarea identisch kodiert werden müssen, also eine doppelte Verwaltung nötig ist.

HTML-Dokument, das die Textdaten enhält:

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

Dokument, das die Textdaten verarbeitet:

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

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

                handle.close();

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

4.3.2.1.3.2.3. window.document.frameset Objekt des Netscape

Erzeugung unter HTML:

Beispiel:

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


```
        </FRAME>
        .....
</FRAMESET>
```

Rahmen zwischen den einzelnen Fenstern des Framesets entfernen:

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

Hinweise:

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

Eigenschaften (Auswahl):

.name	entspricht NAME
.length	entspricht logischer_window_name.frames.length
.parent	Fenster, das das Frame-Objekt erhält
.self	aktueller Frame

Methoden (Auswahl):

.blur()	deaktiviert Frame
.focus()	aktiviert Frame

Eventhandler (Auswahl):

onLoad
onUnload
onMove
onResize

Collectionen:

window.document.frames
Feld der Zeiger aller Frames
Elementefolge laut HTML-Coding
Element ist eine Referenz auf das Fenster mit Frame-Eigenschaften

Syntax:

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

Index	Integer ab 0 muss in **[ ]** kodiert sein

Eigenschaften:

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

Möglichkeiten des Laden eines Dokumentes:

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

Laden eines fremden Dokumentes ohne Framedarstellung:

```
<HTML>
        <HEAD>
                <SCRIPT  LANGUAGE="JavaScript">
                <!--
                        function laden()
                        { location.href = "http://www.test.de";}
                // -->
                </SCRIPT>
        </HEAD>

        <BODY>
                <FORM>
                        <INPUT  TYPE="button"
                                VALUE="www.test.de anwaehlen "
```


```
                                onclick="laden()">
            </FORM>
        </BODY>
</HTML>
```

Eigenes Dokument wird durch fremde Webseite geladen::

CopyRight-Meldung auf fremden Host erzeugen:

Beispiel 1:

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

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

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

Beispiel 2:

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

        var  fremder_host_name = parent.location.hostname;

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

Framedarstellung der eigenen Webseite sofort und ohne Bildschirmmeldung aktivieren:

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

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.2">
<!--
        var EigenerHost="http://www.test.de";
```


```
                        // window.location.hostname ist Leerkette, wenn Browser offline
                var StartSeite="_start.html";

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

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

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

Reload eines Dokumentes mit allen seinen FRAMES:

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

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

Datei ohne FRAMESET in eine Datei mit FRAMESET laden

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

Mehrere Frameinhalte gleichzeitig ändern:

Inhalte zweier Frames tauschen:

Kodierung im Dokument, dass die beiden Frames definiert !

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
        function rahmentausch(logischer_name_rahmen1,html_datei1,
                              logischer_name_rahmen2,html_datei2
```


```
                                        )
                {
                        frames[logischer_name_rahmen1].location.href = html_datei1;
                        frames[logischer_name_rahmen2].location.href = html_datei2;
                }
// -->
</SCRIPT>

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

Inhalte beliebig vieler Frames als Ring tauschen:

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

Tausch:

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

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

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

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

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

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

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

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

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

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

Frame und Datenaustausch:

Zwischen Frames können Daten ausgetauscht werden.

Beispiel Adressierung eines Frame über das FRAMESET

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

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

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

im FRAMESET-Dokument wird kodiert var Kette="Hallo !";

im FRAME-Dokument wird auf Kette zugegriffen: alert(parent.Kette);

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

im FRAME-Dokument wird kodiert var Kette="Hallo !";
<FRAME ...NAME ="test2" ...>

im FRAMESET-Dokument wird auf Kette zugegriffen: alert(parent.test2.Kette);

Beispiel: Textdaten-Übergabe durch an Url angehängte Textdaten

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

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

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

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

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


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

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

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

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

Beispiel: Textdaten-Übergabe durch Fenster-Handle

Die Textdaten und die der Javascript-Code, der die Textdaten verarbeitet, sind **getrennte** HTML-Dokumente. Nachteil dieser Variante ist, dass in beiden Dokumenten die logischen Namen vom Formular und dem Textarea identisch kodiert werden müssen, also eine doppelte Verwaltung nötig ist.

HTML-Dokument, das die Textdaten enhält:

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

Dokument, das die Textdaten verarbeitet:

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

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

                handle.close();

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

4.3.2.1.3.2.4. window.document.img Objekt des Netscape

Erzeugung unter HTML:

Beispiele:

Bild nicht im Formular:

```
<IMG
```

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

Bild im Formular:

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

Beispiel für Linie mit variabler Dimension:

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

horizontale Linie mit 10 Pixel Dicke:

```
<IMG SRC="1x1bild.gif" WIDTH="300" HEIGHT="10" BORDER="0" ALT="" STYLE=" ....">
```

vertikale Linie mit 10 Pixel Dicke:

```
<IMG SRC="1x1bild.gif" WIDTH="10" HEIGHT="300" BORDER="0" ALT="" STYLE=" ....">
```

Erzeugung in Script:

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

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

Zugriff:

Bild nicht im Formular:

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

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

Bild im Formular:

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

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

Bild in Script: per Variable aus new

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

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

Eigenschaften (ausgewählte):

.border entspricht BORDER

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

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

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

Events bei sich überlagernden Objekten:

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

Collectionen:

window.document.images Collection des Netscape

Feld der Zeiger aller img Objekte
Elementefolge laut HTML-Koding

Syntax:

document..images**[**index**]**

index Integer, ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

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

4.3.2.1.3.2.5. window.document.layer / window.document.ilayer Objekt des Netscape unter 6.x

Objekt eines LAYER bzw. ILAYER (ILAYER steht für inline Layer)
Netscape hat nur bis unter Version 6.x das Layer-Objekt implementiert.

HTML 4 unterstützt kein LAYER- und ILAYER-Tag mehr.
Die Tags LAYER und ILAYER werden vom Internet Explorer nicht erkannt.

Das layer Objekt untergliedert das HTML-Dokument in Ebenen, wobei diese sich wie Folien überlappen oder ganz verdecken können.
kann ein HTML-Dokument laden
kann einen Ausschnitt als Sichtfenster erzeugen

Das Layer-Objekt ist an sich **völlig veraltet**:

Foliendarstellung: Die Eigenschaften bezüglich Position des Layers, bezüglich Ausschnitt im Layer und bezüglich Überlagerung von Layern sind z..B. auch anhand von DIV's in Verbindungmit Style realisierbar. Vorteil: DIV und Style kennt jeder moderne Browser, dagegen Layer nicht.

Laden eines HTML-Dokumentes: Die Anzeige eines Fensters ohne Fensterelemente (beim Internet Explorer auch durch zusätzliche Fensterarten) kann einen Layer ersetzen.

Erzeugung unter HTML:

Beispiel:

```
<LAYER
        NAME="logischer_layer_name"      fuer Javascript oder siehe Attribute ABOVE und BELOW
        WIDTH=breite_in_pixel            falls kein Ausschnitt definiert wird, so Breite des Layers
                                         falls Ausschnitt per CLIP definiert wird, so Ausschnittbreite > Breite
                                                möglich
```


```
        HEIGHT=hoehe_in_pixel           falls kein Ausschnitt definiert wird, so Höhe des Layers
                                        falls Ausschnitt per CLIP definiert wird, so Ausschnitthöhe > Höhe
                                            möglich
        CLIP="x1,y1,x2,y2" oder "x,y"   Ausschnitt im Layer bzw. auf Layer und seine Umgebung
                                        Ausschnitt kann eigenartigerweise größer als Layerdimension sein
                                        x1,y1   linke obere Ecke des Ausschnitt-Fenster; x ist Spalte
                                        x1 und y1 sind RELATIV zur linken oberen Layerrand-Ecke
                                        x2,y2   rechte untere Ecke des Ausschnitt-Fenster; x ist Spalte
                                        x2 und y2 sind RELATIV zur rechten unteren Layerrand-Ecke
                                        x       Breite in Pixel, kann > WIDTH sein
                                        y       Höhe in Pixel, kann > HEIGHT sein
        SRC="url_oder_dateiname"        im Layer anzuzeigende HTML oder GIF oder JPG-Datei
        BGCOLOR=#rrggbb                 Hintergrund-Farbe
        BACKGROUND="url_oder_datei"     Hintergrund-Grafik-Datei
        VISIBILITY=     "show"          innerhalb Überlappung immer anzeigen
                oder    "hide"          innerhalb Überlappung nicht anzeigen
                oder    "inherit"       nur anzeigen wenn Elternlayer auch angezeigt ist
        ABOVE="logischer_name_des_layers_der_diesen_überlagern_soll"
                                        nicht zusammen mit BELOW und Z-INDEX kodieren
        BELOW="logischer_name_des_layers_der_von_diesem_überlagert_werden_soll"
                                        nicht zusammen mit ABOVE und Z-INDEX kodieren
        Z-INDEX=position_aus_ueberlagerungsfolge
                                        unverschachtelter Layer:
                                            1=unterster ... n=oberster
                                        verschachtelter Layer
                                            wenn >0, so     1=unterster .. n=oberster
                                            wenn <0, so     -1=oberster ..-n=unterster
                                        nicht zusammen mit ABOVE und BELOW kodieren
    >
        html_elemente_des_layer
    </LAYER>

Beispiel:
    <ILAYER> ... </ILAYER>      analog zu <LAYER> ... </LAYER>
```

Erzeugung in Script:

```
var LayerObjekt = new Layer([breite_in_pixel [, zeiger_auf_eltern_layer]]);

Beispiel: var element_aussen = new Layer(300);                          // <LAYER>
          element_aussen.src="url";
          element_aussen.visbilitiy="show";
          element_aussen.document.open();
          element_aussen.document.write('HALLO');
          element_aussen.document.close();
          element_innen = new Layer(100, document. element_aussen);     // <ILAYER>
```

Zugriff:

```
          document. logischer_layer_name.eigenschaft
          document. logischer_layer_name.methode()

          document.layers[index].eigenschaft
          document.layers[index].methode()

                      index:    ab 0
                                muss in [ ] kodiert sein

          Variable laut new Layer()
```

Eigenschaften (ausgewählte):

```
.above                  liefert den Zeiger vom nächsten übergeordneten Layer laut dessen NAME-Attribut
                        liefert nicht zIndex
                        Achtung: Die oberste Ebene ist immer das Browserfenster.
                        Vergleich der Zeiger per
                            if ( id_des_fensters_aus_open ==  document.layer[index].above )
                            { ..........}
                        siehe .zIndex
                        nur lesen

.background             Url einer Grafikdatei für den Hintergrund
                        Pfadangabe möglich
                        automatisches Kacheln
                        "null" für kein Hintergrundbild
                        lesen und schreiben
```

.below
liefert den Zeiger vom nächsten tieferen Layer laut dessen NAME-Attribut
liefert **nicht** zIndex
wenn "null" so kein nächst tieferer Layer vorhanden
siehe .zIndex
nur lesen

.bgColor
Hintergrundfarbe des Layers
"#xxxxxx" z.B. "#FF0000"
vordefiniert z.B. "red"
"null" so Hintergrund transparent, also darunterliegende Ebene(n) schimmert(n) durch
lesen und schreiben

.clip
Diese Eigenschaft dient zur Verwaltung eines Ausschnittes im Layer
entspricht CLIP-Attribut und ist nur über die Untereigenschaften verwendbar:
CLIP="x1,y1,x2,y2"
x1,y1 linke obere Ecke des Ausschnitt-Fenster; x ist Spalte
x1 und y1 sind RELATIV zur linken oberen Layerrand-Ecke
x2,y2 rechte untere Ecke des Ausschnitt-Fenster; x ist Spalte
x2 und y2 sind RELATIV zur rechten unteren Layerrand-Ecke
x1 entspricht .clip.left
y1 entspricht .clip.top
x2 entspricht .clip.right
y2 entspricht .clip.bottom
Hinweis: möglichst nicht kodieren und dafür durch .style.clip zu ersetzen

.clip.bottom
y2 aus CLIP="x1,y1,x2,y2"
lesen und schreiben

.clip.height
Höhe des Ausschnittes in Pixel
y aus CLIP="x,y" oder Differenz aus y2 minus y1 aus CLIP="x1,y1,x2,y2"
.clip.height = .clip.bottom minus .clip.top
lesen und schreiben

.clip.left
x1 aus CLIP="x1,y1,x2,y2"
lesen und schreiben

.clip.right
x2 aus CLIP="x1,y1,x2,y2"
lesen und schreiben

.clip.top
y1 aus CLIP="x1,y1,x2,y2"
lesen und schreiben

.clip.width
Breite des Ausschnittes in Pixel
x aus CLIP="x,y" oder Differenz aus x2 minus x1 aus CLIP="x1,y1,x2,y2"
.clip.width = .clip.right minus .clip.left
lesen und schreiben

.height
Höhe des Layers
lesen und schreiben

.document
Zeiger auf das HTML-Dokument im Layer

.left
X-Koordinate linke obere Layerecke in Pixel, auch negativ
String Ziffernfolge eines Integer und
"absolute" Pixelangabe bezüglich Fenster
"relative" Pixelangaben bezüglich zum den umgebenden HTML-Element
Hinweis: möglichst nicht kodieren und dafür durch STYLE zu ersetzen:
position:absloute;left=....
bzw. position:relative;left=....
lesen und schreiben

.name
enthält logischen Namen des Layers laut NAME-Attribut im Tag <LAYER> bzw. <ILAYER>
nur lesen

.pageX
Abstand in Pixel immer zum linken Rand des Fensters
Hinweis: möglichst nicht kodieren und dafür durch STYLE zu ersetzen:
position:absloute;left=....
lesen und schreiben

.pageY
Abstand in Pixel immer zum oberen Rand des Fensters

Hinweis: möglichst nicht kodieren und dafür durch STYLE zu ersetzen:
position:absloute;top=....
lesen und schreiben

.parentLayer — liefert den Zeiger auf den Eltern-Layer, der diesen Layer enthält.
liefert **nicht** zIndex
Eltern-Layer kann mehrere innere Layer besitzen.
Die oberste Ebene ist immer das Browserfenster.
nur lesen

.siblingAbove — liefert den Zeiger vom nächsten übergeordneten Layer, der innerhalb eines gemeinsamen Eltern-Layers liegt
liefert **nicht** zIndex
ist "null" wenn kein übergeordneter Layer vorhanden ist
nur lesen

.siblingBelow — liefert den Zeiger vom nächsten untergeordneten Layer, der innerhalb eines gemeinsamen Eltern-Layers liegt
liefert **nicht** zIndex
ist "null" wenn kein untergeordneter Layer vorhanden ist
nur lesen

.src — Url desjenigen HTML-Dokumentes, das in dem Layer dargestellt werden soll
lesen und schreiben
siehe .load()

.top — Y-Koordinate linke obere Layerecke in Pixel, auch negativ
String Ziffernfolge eines Integer und
"absolute" Pixelangaben bezüglich Fensters
"relative" Pixelangaben bezüglich zum umgebenden HTML-Element
Hinweis: möglichst nicht kodieren und dafür durch STYLE zu ersetzen:
position:absloute;top=....
bzw. position:relative;top=....
lesen und schreiben

.visibility — Sichtbarkeit des Layers unabhängig von der Verschachtelung
wenn "show" Layer sichtbar
wenn "hide" Layer nicht sichtbar
wenn "inherit" Sichtbarkeit laut Eltern
lesen und schreiben

.width — Breite des Layers
lesen und schreiben

.zIndex — Nummer des Layers in der Folge aller Layer, also **auch die in einer Layerverschachtelung**
Achtung: Die oberste Ebene ist immer das Browserfenster.
Damit gilt: Der höchste Wert indiziert den Layer direkt unterhalb des Fensters und somit den obersten Layer aller Layer
Der kleinste Wert indiziert den Layer, der am tiefsten in der Folge liegt, also am weitesten entfernt zum Browserfenster
dient anstelle von interner Zeigerverwendung per .above und .below
Nummernfolge muss nicht im Abstand von 1 sein
je kleiner die Nummer um so tiefer in der Folge
nur >= 0 (Es kann sein, dass negative Werte toleriert werden.
Dann sind Werte < 0 in der Verschachtelungs-hierarchie immer tiefer als 0 oder positiv.)
wenn 0 so Browserfenster
wenn 1 so der unterste Layer, der am weitesten entfernt ist vom Browserfenster
wenn 1000 so der oberste Layer bei Annahme, dass es keine 1001 oder mehr Layer gibt
Anzahl der Layer - inklusive verschachtelter Layer - ist nur manuell ermittelbar durch mitzählen
(es gibt keine Eigenschaft)
lesen und schreiben

Methoden (ausgewählte):
Hinweis: Sämtliche Wertangaben (außer Zeiger) müssen in " " kodiert sein.

.captureEvent(event) — siehe Eventobjekt des Netscape

.handleEvent("event_id") — siehe Eventobjekt des Netscape

.load(url,x) lädt das HTML-Dokument laut Url in diesen Layer und passt den geladenen Inhalt an die Breite laut x an
Achtung: Wenn x > .clip.width
x > width
so wird der geladene Inhalt abgeschnitten
siehe .src

.moveAbove(layer_instanz) setzt diesen Layer direkt **vor** den Layer laut layer_instanz und wird damit **direkt übergeordneter** Layer von layer_instanz
ob layer_instanz bereits innerhalb eines Layers liegt, ist egal
layer_instanz z.B. laut NAME-Attribut

.moveBelow(layer_instanz) setzt diesen Layer direkt **hinter** den Layer laut layer_instanz und wird damit **direkt untergeordneter** Layer von layer_instanz
ob layer_instanz bereits innerhalb eines Layers liegt, ist egal
layer_instanz z.B. laut NAME-Attribut

.moveBy(x,y) Verschiebung des Layers **um** Pixeldifferenz x und y
x in Pixel, auch negativ
verändert .left um x
y in Pixel, auch negativ
verändert .top um y

Beispiel für Fenster des Browser rausschieben und danach neu anzeigen:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
	var BrowserFenster_AktuelleBreite=2000;  // sollte größer als max. übliche Auflösung sein
	var BrowserFenster_AktuelleHoehe=2000;  // sollte größer als max. übliche Auflösung sein

	function moveWin()
	{
		for (var i = 1; i < BrowserFenster_AktuelleBreite; i++)
		{window.moveBy(1, 1);}

		window.moveBy((-1)* BrowserFenster_AktuelleBreite,(-1) * BrowserFenster_AktuelleHoehe);
	}
//-->
</SCRIPT>
</HEAD>
<BODY onload="moveWin();">
</BODY>
</HTML>
```

.moveTo(x,y) Verschiebung des Layers **auf** neue Position laut x und y
benutzt dabei die Kodierung der Werte von .left und .top:
.left bzw. .top können folgende Zusätze haben, die massgebend für die relative Positionierung per moveTo() sind:
"absolute" Pixelangaben bezüglich Fensters
"relative" Pixelangaben bezüglich zum umgebenden HTML-Element
Hinweis: Nullpunkt liegt in der linken oberen Ecke
x in Pixel, nur positiv
setzt .left auf x
y in Pixel, nur positiv
setzt .top auf y

.moveToAbsolute(x,y) Verschiebung des Layers **auf** neue Position laut x und y immer bezüglich des Browserfensters
Hinweis: Nullpunkt liegt in der linken oberen Ecke
x in Pixel, nur positiv
setzt .pageX auf x
y in Pixel, nur positiv
setzt .pageY auf y

.releaseEvents(event_typ) siehe Eventobjekt des Netscape

.resizeBy(x,y) Änderung der Layerdimension um Pixeldifferenz und ohne Anpassung des Layerinhaltes
Achtung: Bei Verkleinerung kann der Inhalt ausgeblendet werden.
x in Pixel, auch negativ
verändert .width um x
y in Pixel, auch negativ

verändert .height um y

.resizeTo(x,y) Änderung der Layerdimension auf neue Breite und Höhe und ohne Anpassung des Layerinhaltes
Achtung: Bei Verkleinerung kann der Inhalt ausgeblendet werden.
x in Pixel, nur positiv
setzt .width auf x
y in Pixel, nur positiv
setzt .heigth auf y

.routeEvent(event_txp) siehe Eventobjekt des Netscape

Collectionen:

window.document.layers Collection des Netscape (nicht mehr ab Version 6.x)
Feld der Zeiger aller Layer-Objekte im Dokument

Syntax:

document..layers**[**index**]**

index Integer, ab 0
muss in **[ ]** kodiert sein

Eigenschaften:

.length Anzahl der Feldelemente also Feldlänge z.B. bei Collection
Anzahl **nur der unverschachtelten** Layer
Hinweis: Anzahl aller - inklusive verschachtelter - Layer nur durch manuelle Indexverwaltung ermittelbar (es gibt keine Eigenschaft)
Layer-Eigenschaft .zIndex dient zur Referenzierung auch von verschachtelten Layern

Beispiele:

unverschachtelter Layer:

```
<LAYER ID="ID_Layer" STYLE="position: absolute;">Ich bin eine Ebene</LAYER>
```

Layer-Zugriff per Script:

```
document. ID_Layer
oder document.layers["ID_Layer"]
oder document.layers[0]
```

Verwendung eines numerischen Indexes:
Die Indexfolge ist nicht die Folge laut HTML-Kodierung (außer Indexwert 0 = unterster Layer)
dafür z-index-Attribut kodieren und dieses verwenden

Anzahl der obersten Ebenen auf gleichem Level also ohne deren innere Layer:

```
document.layers.length
```

verschachtelte Layer:

```
<LAYER ID="ebene1" STYLE="position: absolute; left: 100; top: 100;">
    Nur ein Test...
    <LAYER ID="ebene2" STYLE="position: absolute; left: 50; top: 50;">
        Jaja!
    </LAYER>
</LAYER>
```

es wird vererbt: Bsp.: Verschiebung des äußeren Layers (Eltern) verschiebt den inneren (Kind), wobei innerer in gleicher relativer Position zum äußeren bleibt wie vor der Verschiebung.

document.layers.length liefert 1

Zugriff auf inneren Layer:

```
document.ebene1.document.ebene2
```

Den Inhalt von Ebenen ändern:

```
document.ebene1.document.open();                        // anzeigen
document.ebene1.document.write("Ein neuer Inhalt");     // füllen
document.ebene1.document.close();                       // und schliessen
                                                        // bleibt aber sichtbar
```

oder

```
with(document.ebene1.document.ebene2)
{
    open();
    write("So geht's auch");
    close();
}
```

Layer dynamisch erzeugen NACH dem kompletten Laden des Dokumentes
z.B. per <BODY onload= .....>

Achtung:
Ein Versuch Ebenen direkt im HEADER per "<SCRIPT>...</SCRIPT>"
zu erzeugen, wird einfach ignoriert !!!!!!

Bsp.:

```
var neueEbene=new Layer(breite_in_Pixel);          // instanzieren

neueEbene.document.open();                         // anzeigen
neueEbene.document.write("test");                  // und füllen
neueEbene.document.close();                        // und schliessen
                                                   // bleibt aber sichtbar !!
```

Inhalt einersLayers:

laden	Aufruf der Methode "load" oder neueEbene.src="seite1.html";
anzeigen	neueEbene.visibility="show";

verschachtelte Layer dynamisch erzeugen

```
var neueEbene=new Layer(breite_in_Pixel);                          // instanzieren
var neueEbene1=new Layer(breite_in_pixel, document.neueEbene);
```

Eigenschaften von Layer lesen und/oder ändern

z.B. X-Position document.neueEbene.left=200; // Pixel
Y-Position document.neueEbene.top= 300; // Pixel

Layer verschieben:

```
Bsp.    document.ebene1.moveBy(50, 20);
oder    document.ebene1.left+=50;   document.ebene1.top+=20;
```

4.3.2.1.3.2.6. window.document.link Objekt des Netscape

Erzeugung unter HTML:

Beispiel:

```
<LINK
    HREF="url"                                  //  protocol://[host_name[:port]/]dateiname
                                                //  protocol://[host_name[:port]/]dateiname#hashtext
                                                //  protocol://[host_name[:port]/]dateiname?serachtext
                                                // Bsp:  http://www.test.com:8888/test.html
        NAME="anker_name_ohne_#"                // ist nicht der logische link_name !!
                                                //      Netscape + Internet Explorer
    oder ID="anker_name_ohne_#"                 // ist    der logische link_name !!
                                                //      nur Internet Explorer ab 4.x
    TARGET="logischer_window_name"
    onClick="eventhandler1"
    onDblClick="eventhandler2"
    onMouseOut="eventhandler3"
    onMouseOver="eventhandler4"
    onMouseDown="eventhandler5"
    onMouseUp="eventhandler6"
>
    link_text_immer_schreiben
</LINK>
```

Zugriff:

document.links**[**index**]**

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

Beispiel für globalen Style per HEAD:

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


```
                    </A>
                </BODY>
```

Eigenschaften (ausgewählte):

.className	Klassenreferenz
.hash	entspricht **#hash**text zum Anspringen eines Ankers hier den Ankernamen mit vorgesetztem # ablegen
.host	entspricht hostname:port
.hostname	entspricht nur hostname
.href	gesamter Url (kompletter Url) zum Anspringen eines Ankers hier den Ankernamen ohne vorgesetztem # ablegen
.name	entspricht NAME
.offsetLeft	Pixelpos bezüglich linken Rand des HTML-Dokumentes
.pathname	entspricht aus url /dateiname bzw. /dateiname**#hash**text bzw. /dateiname**?search**text lesen und schreiben
.port	entspricht port; lesen und schreiben
.protocol	entspricht Protokoll mit Doppelpunkt z.B. "http:" lesen und schreiben
.search	entspricht **?search**text lesen und schreiben
.target	entspricht TARGET kann auch sein _blank _parent _search _self _top lesen und schreiben
.text	enthält den link_text nur lesen
.x	horizontale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes nur lesen
.y	vertikale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes nur lesen

Methoden (ausgewählte):

.blur()	Element den Focus wegnehmen und Event onblur auslösen Der Focus wird nicht automatisch auf irgend ein anderes Element gesetzt !
.focus()	Focus setzen und Focus-Event auslösen nur nach dem kompletten Laden des Dokumentes
.handleEvent()	siehe Eventbehandlung zum NS 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 History-Eintrag zur aktuellen Seite überschreiben (keinen neuen bilden) --> BACK-Button wechselt danach nicht zur überschriebenen Seite, da diese in der History nicht mehr bekannt ist

Collectionen:

window.document.links Collection

Feld der Zeiger aller Objekte mit HREF-Eigenschaft (Attribut) sowie aller AREA-Objekte im Dokument
Elementefolge laut HTML-Coding

Syntax:

```
document.links[index]

        index       Integer, ab 0
                    muss in [ ] kodiert sein
```

Eigenschaften:

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

4.3.2.1.3.2.7. window.document.span Objekt des Netscape

Erzeugung in HTML:

Beispiel: <SPAN STYLE="color: blue">blaues</SPAN>Wort

Das DIV- bzw. SPAN-Objekt ähneln sich sehr stark.. SPAN wird als üblicherweise als Textcontainer benutzt.

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

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

Zugriff:

document.ID_Span.eigenschaft
document.ID_Span.methode()

Beispiel:

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

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

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

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

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

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

4.3.2.1.3.2.8. window.document.style Objekt und window.document.styleSheet Objekt des Netscape (CSS)

Ansatz:

Aufgrund der HTML-Integration von CSS sollte der Browser CSS unterstützen. Ob diese Unterstützung gegenüber den aktuellen Standards zu HTML und CSS vollständig ist, hängt vom Willen des Browserherstellers ab. Letztendlich wird CSS sinnigerweise als grundlegende Komponente und Eigenschaft **aller** Objekte integriert. CSS bietet elementare Vielfalt bei der dynamischen Scriptprogrammierung, ist gewöhnungsbedürftig, aber durch Normung universell auf diverse Objekte anwendbar, die das STYLE-Attribut bzw. die Eigenschaft .style unterstützen.

Es sei darauf hingewiesen, dass

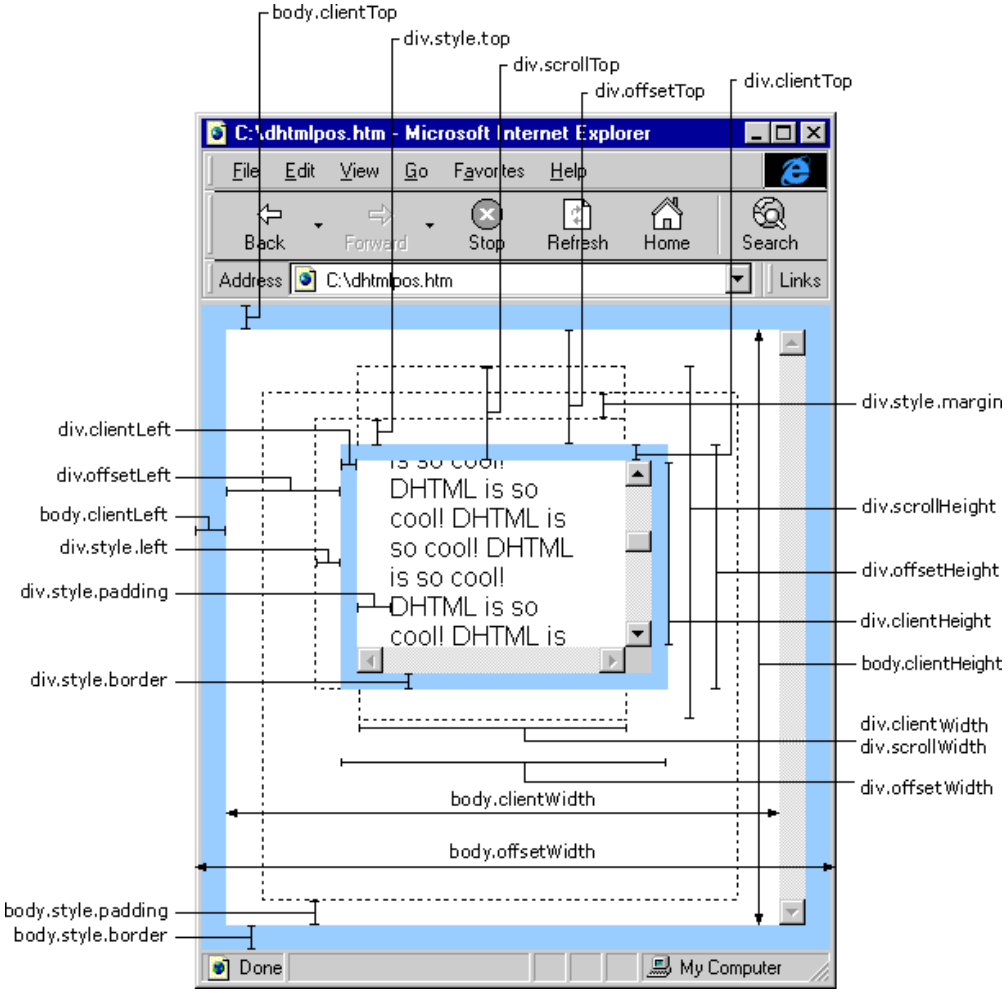
die Verwendung von CSS-Klassen, Kodierung von <STYLE> bzw. dem STYLE-Attribut bzw. der Eigenschaft .style nur **scheinbar** alle synonym sind:

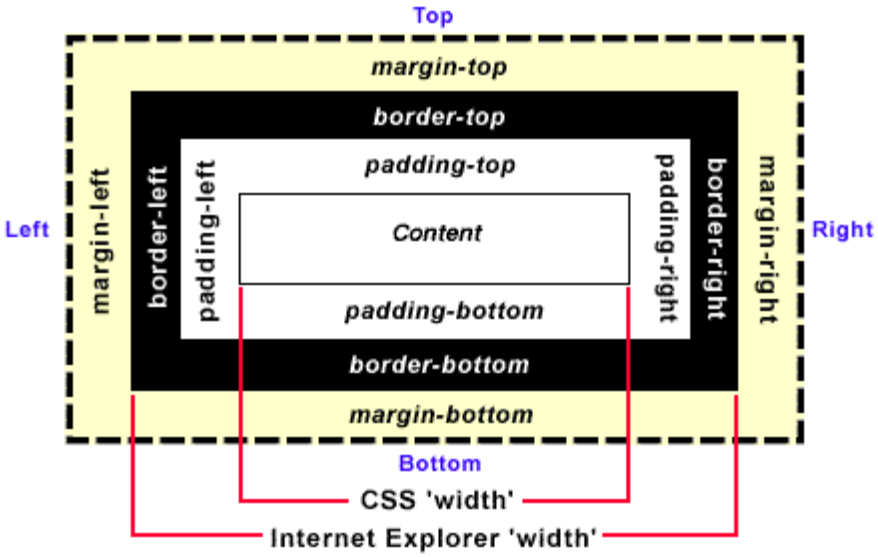
- CSS-Klassen im HEAD können dokumentweit verwendet werden.
- <STYLE>, STYLE-Attribut und .style sind stets HTML-komponentenbezogen.

der Browser diese Formen intern z.T. verschieden (auch bezüglich der Collectionen) verwaltet.

Das Einbinden von Styles aus einer externen Datei ist möglich.

Nachfolgend die Style-Kategorien zum Internet Explorer:





style Objekt und styleSheet Objekt:
Beide Objekte behandeln zwar CSS, unterscheiden sich aber in der Aufgabe:

Das Objekt style repräsentiert CSS eines Objektes im HTML-Dokument
also z.B. das STYLE-Attribut im HTML-Tag
die Eigenschaft .style
alle Styles sind les- und schreibbar

Das Objekt styleSheet repräsentiert CSS im HTML-DOM, also im gesamten Dokument
wird mit der **document.styleSheets Collection verwaltet:**
Feld der Zeiger aller styleSheet Objekte im Dokument
Syntax:
[var ZeigerAufFeld =] document.styleSheets
[var ZeigerAufFeldElement =] document.styleSheets **[**index**]**

index Integer ab 0
muss in **[ ]** kodiert sein

Eigenschaft .length Anzahl der Feldelemente
Integer, ab 1

Nachfolgende Beschreibungen orientieren sich praktischerweise hauptsächlich am Objekt style.

Beispiel: Es sollen DIVs erzeugt werden und deren Objektreferenzen (Zeiger) in einem Zeigerfeld gesammelt werden. Das Zeigerfeld dient der vereinfachten Verwaltung der dynamisch erzeugten DIV's.

```
var DIV_ID="Test_DIV"; //  auch "Otto_DIV" etc.möglich , je nach Wunsch

var DIV_Zeiger=Array(); // sammelt alle ID als Zeichenketten

// DIV's dynamisch erzeugen und Zeiger einsammeln
var Left= 20;            // Abstand vom linken Fensterrand in Pixeln
for (var i=0; i < 3; i++)
{
        Left+=10; //  ab 30 Pixel vom linken Fensterrand
        document.write(         '<DIV ID="' + DIV_ID + i.toString() + ""
                        +       ' STYLE="position:absolute;'
                        +               'left=' + Left + 'px;'
                        +               'top=20px;'
                        +               ""
                        +  '>'
                        +  '</DIV>'
                      );
        eval ('DIV_Zeiger[' + i + '] =' +  DIV_ID + i.toString() );
}

//  DIV's dynamisch auf dem Bildschirm bezüglich dem linken Fensterrand um 20 Pixel
//                      verschieben
Left= 40;
for (i=0; i < 3; i++)
{
        Left+=10; //  ab 50 Pixel vom linken Fensterrand verschieben
        eval('document.all.DIV_Zeiger[' + i + '] .style.left=' + Left);
}
```

Hinweis: Anstelle von i.toString() kann auch nur i kodiert werden, da der Browser
aufgrund von document.write() i automatisch nach String umwandelt.
analog für i.toString() innerhalb eval()

Die Verschiebung erfolgt natürlich mit den Objekten, welche innerhalb von <DIV> .. </DIV>
kodiert wurden. Im Falle von IMG's innerhalb des DIV werden diese mit verschoben.

Vorallem **wegen** der Manipulation von HTML-Objekten werden DIV's verwendet.

Kodierung der Style-Sheets-Angaben:
innerhalb CSS-Klasse: immer mit Bindestrich
als .style-Eigenschaft: ohne Bindestrich aber anstelle dessen die Nachfolgebuchstaben als Großbuchstabe

Bsp: in CSS-Klasse background-color
in Script .style.backgroundColor

Konvertierungsmethode:

Style besitzt für die style-gerechte Umwandlung eines Url-Wertes die Methode url(), welche nicht direkt als Methode des Style-Objektes implementiert ist und damit ohne Punktreferenz kodiert wird.

url(StringWert)	StringWert	Dateiname eines Bildes einer HTC-Datei (siehe Behavior des IE) eines Standard-Behaviors (siehe Behavior des IE) eines Cursors (*.CUR und *.ANI ab IE 6.x) Pfadangabe (relativ oder absolut) mit Dateiname obiger Dateiarten 'none' für nichts (also nicht url() ohne Parameter kodieren !)
	Hinweis: alert (zeiger_auf_objekt.style.cursor);	liefert beim IE den String 'url(cursor_form)' mit cursor_form für den aktuellen **und** gesetzten Cursor.
	style.cursor	ist standardgemäß mit einer Leerkette belegt, solange **kein** Cursor gesetzt wird kann nicht mit url(cursor_form) belegt werden, wenn es sich um eine **standardgemäß im Browser implemtierte** Cursorform handelt wie z.B. 'hand' oder 'normal'. Grund: Diese Cursorformen sind keine Dateien (owohl alert den String 'url(cursor_form)' anzeigt).

Hinweise:

Hinweis zur dynamischen Eigenschaftenveränderung zur Laufzeit:

Die zuletzt während der Laufzeit getätigte Defiintion wird wertmäßig durch den aktuellen Attributwert ersetzt, wenn gleiche Attributnamen / Eigenschaften betroffen sind.

Hinweis zu Style-Begriffen:

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

4.3.2.1.3.2.8.1. Style-Sheet-Deklarationen

Styles innerhalb einer CSS-Klasse immer mit Bindestrich kodieren

Bsp: in CSS-Klasse background-color
in Script .style.backgroundColor

Style-Sheet-Deklarationen im HEAD (Varianten)

Variante 1 für Pseudoklassen:

```
<HEAD>
    <STYLE TYPE="text/css">
    <!--
        @eigenschaften      /* festkodiert ist @
                            /* z.B. @import
    //-->
    </STYLE>
</HEAD>
```

Variante 2 für CSS-Klasse, die im CLASS-Attribut des HTML-Elementes verwendet wird:

```
<HEAD>
    <STYLE TYPE="text/css">
    <!--
        .freier_klassen_bezeichner {eigenschaften_liste}    /* Punkt ist festkodiert
    oder  all.freier_klassen_bezeichner {eigenschaften_liste}
    //-->
    </STYLE>
</HEAD>
<BODY>
    <tag_name CLASS="freier_klassen_bezeichner"> ..... </tag_name>
</BODY>
```

. und **.all** sind synonym

freier_klassen_bezeichner: Bsp: fettkursiv

eigenschaften_liste: Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

Variante 3 für CSS-Klasse, die im ID-Attribut des HTML-Elementes verwendet wird:

```
<HEAD>
    <STYLE TYPE="text/css">
    <!--
```


```
                #freies_id{eigenschaften_liste}    /* festkodiert ist #
        //-->
        </STYLE>
</HEAD>
<BODY>
        <tag_name ID="freies_id"> ..... </tag_name>
</BODY>
```

freies_id: Bsp: fettkursiv
innerhalb STYLE mit vorgesetztem # kodieren !
innerhalb BODY ohne # kodieren !

eigenschaften_liste: Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

Variante 4 für CSS-Klasse, die einem HTML-Tag automatisch und ohne Verwendung des CLASS-Attributes zugeordnet wird, egal wo der Tag im BODY kodiert wurde:

```
<HEAD>
        <STYLE TYPE="text/css">
        <!--
                tag_namen_liste{eigenschaften_liste}
        //-->
        </STYLE>
</HEAD>
<BODY>
        <tag_name_1> ..... </tag_name_1>
        ....
        <tag_name_n> ..... </tag_name_n>

</BODY>
```

tag_namen_liste: mindestens 1 Element (Tag-Bezeichner)
mehrere Elemente per Komma trennen
alle Tags haben die gemeinsame Eigenschaftsliste
Gross-klein bei Tag-Bezeichnern egal
Bsp: h1,h2

eigenschaften_liste: Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

Variante 5 für CSS-Klasse, die einem HTML-Tag mit Pseudoklasse automatisch und ohne Verwendung des CLASS-Attributes zugeordnet wird, egal wo der Tag im BODY kodiert wurde:

```
<HEAD>
        <STYLE TYPE="text/css">
        <!--
                tag_name:schlüsselwort{eigenschaften_liste}   /* Doppelpunkt ist festkodiert
        //-->
        </STYLE>
</HEAD>
<BODY>
        <tag_name> ..... </tag_name>
</BODY>
```

tag_name: schlüsselwort ist Pseudoklasse zum Tag
muss kodiert werden
Bsp: a:link

eigenschaften_liste: Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

Hinweise: Ein gleichnamiges tag-abhängiges Style-Sheet-Format wird in der Darstellung durch das tag-unabhängige ersetzt.
Ein tag-unabhängiges Style-Sheet-Format fügt sich ansonsten zu den tag-abhängigen hinzu.

Variante 6 für CSS-Unterklasse, die im CLASS-Attribut zum entsprechenden HTML-Tag verwendet wird:

```
<HEAD>
        <STYLE TYPE="text/css">
        <!--
                tag_name.unterklasse_name{eigenschaften_liste}      /* Punkt ist festkodiert
        //-->
        </STYLE>
</HEAD>
<BODY>
        <tag_name CLASS="unterklasse_name"> ..... </tag_name>
</BODY>
```

tag_namen.unterklasse_name:	Bsp:	p.normal, muss aber im CLASS-Attribut zum Tag kodiert werden
eigenschaften_liste:	Semikolontrennung muss mit Semikolon enden wenn Blank enthalten, so Liste in " " bzw. ' ' setzen Bsp:	{font-weight:bold;font_style:italic;}

Hinweis: Aufgrund der Kodierung des CLASS-Attributes ist auch Variante 2 verwendbar. Warum Variante 6 existiert, ist unklar.

Beispiele für alle Varianten:
In den Beispielen wurden nur aus Gründen der Übersichtlichkeit Leerzeichen eingefügt.

Beispiel 1

```
<HEAD>
        <STYLE TYPE="text/css">
        <!--
                body{   background-color:#FFFFFF;
                        background-image:url(test.jpg);
                        background-repeat:repeat-y;
                        margin-top: 1cm;
                        margin-left: 2cm;
                        margin-right: 1cm;
                    }
                p{      font-family:serif;
                        font-size: 12pt;
                        text-align:justify;
                  }
                p.zusatz_format_p_tag{text-ident:0.5cm;}
                .zusatz_format_alle_tags{
                                                position:absolute;
                                                top:4cm;
                                                z-index:3;
                                                font-size:40pt;
                                                font-width:bold;
                                                color:blue;
                                                text-align:center;
                                            }
        -->
        </STYLE>
</HEAD>
<BODY>
        <SPAN CLASS=" zusatz_format_alle_tags"> text </SPAN>
        <P CLASS=" zusatz_format_p_tag" text </P>
</BODY>
```

<!-- -->	für Browser kodieren, die CSS nicht können
background-image:	Hintergrundbild test.jpg verwenden Achtung: Möglichst absolute Urls bzw. Pfadangaben verwenden, da eventuell Browser relative Angaben nicht interpretieren kann Bsp.: relativ :url('ordner/test.gif') absolut :url('www.test.de/ordner/test.gif')
background-repeat:	repeat-y Hintergrundbild senkrecht wiederholen repeat-x Hintergrundbild waagerecht wiederholen no-repeat Hintergrundbild nicht wiederholen
p{....}	gilt für JEDES P-Tag ohne CLASS-Attribut zu kodieren
margin-xxxx	Seitenrand
text-align	Textausrichtung

text-ident	Einzug links	
position	Art der position-Angabe folgenden Angaben	
	Bsp: position:absolute	--> Absolutangaben
	top:4cm	hier 4 cm
top	Abstand zum Browserfenster oben	
z-index	Angezeigte Schicht (analog zu LAYER)	
	--> ab 1 = unterste Schicht	
	verwenden für überlappende Textbereiche	
p.zusatz_format_p_tag{ ..}	per CLASS-Attribut anwenden im P-Tag	

Beispiel 2

```
<HTML>
<HEAD>
    <STYLE TYPE="text/css">
    <!--
        .eigene_css_klasse{ .....}
    -->
    </STYLE>
</HEAD>
<BODY>
    <DIV CLASS=eigene_css_klasse>.....</DIV>
    .....
</BODY>
</HTML>
```

Style-Sheet-Deklaration per STYLE-Attribut im HTML-Element:

alle .style-Eigenschaft ohne Bindestrich aber anstelle dessen die Nachfolgebuchstaben als Großbuchstabekodieren

Bsp:	in CSS-Klasse	background-color
	in Script	.style.backgroundColor

Beispiel:

```
<BODY>
    <tag_name STYLE=eigenschaften_liste>..... </tag_name>
</BODY>
```

eigenschaften_liste:	Semikolontrennung
	muss mit Semikolon enden
	wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
	Bsp: {font-weight:bold;font_style:italic;}

Style-Sheet-Deklaration in Script per Eigenschaft .style eines HTML-Elementes:

Beispiel:

```
<DIV ID="ID_Div">Test</DIV>
document.ID_Div.style.height="200px";
```

4.3.2.1.3.2.8.2. Style-Sheet und vordefinierte Werte

Style-Sheet und Farbangaben

browserspezifisch

Bsp.: "yellow"

Hinweis: frei wählbare Farben per RGB-Angaben im Format

#rrggbb

rgb(rrr,ggg,bbb)

Rot-Anteil:			
	hexa	rr	0 bis FF, mit Vornull
	dezimal 0-255	rrr	ohne Vornullen
Grün-Anteil:			
	hexa	gg	0 bis FF, mit Vornull
	dezimal 0-255	ggg	ohne Vornullen
Blau-Anteil:			
	hexa	bb	0 bis FF, mit Vornull
	dezimal 0-255	bbb	ohne Vornullen

Beispiele: #FF01A3

rgb(1,255,10)

Style-Sheet und Dimensionen

pt	Point	=	1/72 inches
pc	Pica	=	12 pt
in	Inch	=	2,54 cm
mm			
cm			

px Pixel

Hinweis: Die kleinste im Browserfenster anzeigbare Pixeleinheit ist 1. Da die Fensterauflösung als Vielfaches von einem **ganzen** Pixel angegeben wird, ist somit z.B. die Pixeleinheit < 1 nicht anzeigbar. Eine numerische Pixeloperation kann in Gleitkomma erfolgen, deren Ergbniss aber zum Rendern als ganze Zahl verwendet wird. Sinnvoll ist also nur die numerische Pixeloperation mit ganzen Zahlen.

Hinweis: Pixel ist nicht in andere Einheiten umrechenbar, da die physische Dimension eines Bildpunktes auf dem Monitor dem eines Pixels entspricht. Welche physische Dimension verwendet wird, hängt vomMonitor ab. Damit lässt sich z.B. die Font-Einheit pt nicht in px umrechnen, es sei denn, man prgrammiert für genau einen Monitortyp, deren physische Pixeldimension z.B. in cm bekannt ist.
Die Verwendung von px als logische Dimension garantiert aber die exakte Umsetzung in die monitor-spezifische physische Dimension (je nach Bildschirmauflösung). Point wird vorrangig in Layouts verwendet, die nicht pixelorientiert sind, sondern z.B. auf Einheit cm basieren (z.B. Textdarstellung). Alternativen: em, ex, %.

Bei einem Monitor mit 0.25 mm pro Bildpunkt mit einer Auflösung von 1024x768:

100 Pixel sind 3 cm

Buchstabe 'W' mit Schriftart

Arial	fett / nicht fett:	15 pt	sind	4 mm Höhe 6 mm Breite
Times New Roman	fett / nicht fett	15 pt	sind	3 mm Höhe 6 mm Breite

em relativ zur Schrifthöhe des Elementes, das per CSS formatiert wird
ex relativ zur Höhe des Buchstaben Grosses X
% Prozentanteil von der Norm des per CSS formatierten Elementes

Style-Sheet- und Dezimalkomma bei numerischen Werten
Dezimalkomma ist der Punkt und NICHT das Komma.

Style-Sheet und Schriften (Font und Style)

vordefinierte Schriftarten:	serif	
	san-serif	
	cursive	
	fantasy	
	monospace	
vordefinierte Style:	italic	entspricht kursiv
	oblique	entspricht kursiv
	normal	entspricht nicht kursiv

4.3.2.1.3.2.8.3. Style-Sheet aus Datei einbinden

Style-Sheet-Schriftdatei laden (*.eot bzw. *.prf)
Microsoft: *.eot
Netscape *.pfr

```
<STYLE TYPE="text/css">
<!--
        @font-face{ eigenschaften_liste; src:url (eot_bzw._prf_datei_liste);}
// -->
</STYLE>
```

eigenschaften_liste: optional
Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

eot_bzw._prf_datei_liste: mindestens 1 Element
Kommatrennung der Elemente

lokale Fontdateien laden
Fontdateien liegen im FONT-Ordner von Windows

```
<STYLE TYPE="text/css">
<!--
        @font-face{ eigenschaften_liste; local (font_datei_liste);}
oder    @font-face{ eigenschaften_liste; format (TrueType);}
```


```
// -->
</STYLE>
```

eigenschaften_liste:	optional Semikolontrennung muss mit Semikolon enden wenn Blank enthalten, so Liste in " " bzw. ' ' setzen Bsp: {font-weight:bold;font_style:italic;}
font_datei_liste:	lokale Fontdateien im FONT-Ordner von Windows mindestens 1 Element Blanktrennung der Elemente
TrueType	es werden nur lokale TrueTyp-Fonts (ttf-Dateien) aus dem FONT-Ordner von Windows verwendet

Style-Sheet-Datei laden (*.css)

per LINK-Tag:

```
<LINK
        REL=stylesheet
        TYPE="text/css"
        HREF="url_oder_css_dateiname"
        [MEDIA="typ_name"]

>
<STYLE TYPE="text/css">
        <!--  weitere Stylesheet-Angaben
        // -->
</STYLE>
```

MEDIA=	optional	
	typ_name:	Name des Ausgabe-Mediums
	"all"	alle Medien
	"screen"	Bildschirm
	"print"	Drucker

Beispiel für Seitenelemente vom Druck ausschließen (ab IE 4.x und NS 6.x):

```
<BODY>
        <LINK REL="stylesheet" MEDIA="print" HREF="print.css">
        <DIV CLASS="keindruck">
                alle Seitenteile, die nicht gedruckt werden sollen
        </DIV>
<BODY>

print.css enthält nur
.keindruck {display:none; }
```

Beispiel für Seitenelemente nur beim Druck anzeigen (ab IE 4.x):

```
<BODY>>
        <LINK REL="stylesheet" MEDIA="screen" HREF="screen.css">
        <DIV CLASS="nurdruck">
                alle Seitenteile, die nur auf Audruck zu sehen sein sollen
        </DIV>
<BODY>
```

per Pseudoklasse @import:

```
<STYLE TYPE="text/css">
<!--
        @import (url_oder_css_datei) typ_liste;
// -->
</STYLE>
```

typ_liste: Liste der Ausgabe-Medien
Kommatrennung
Ausgabe-Mediums können sein

"all"	alle Medien
"screen"	Bildschirm
"print"	Drucker

Bsp: print,screen;

4.3.2.1.3.2.8.4. Style-Sheet und Wertzuweisung an Eigenschaften

eigenschaft: wert; oder eigenschaft=wert;

Hinweis: Pflichtkodierung von Doppelpunkt bzw. Gleichheitszeichen
Semikolon

wenn für wert auto kodiert werden kann, so ist das der Standardwert, falls Style nicht belegt wird

Folge von Eigenschaften möglich: als Liste mit Semikolontrennung
Bsp: eigenschaft1=wert1; .....; eigenschaft_n=wert_n;

wenn Blank in Kodierung enthalten, so alles in " " bzw. ' ' setzen

Bsp: style="font-style**:** italic; color**:**red;"

wenn Eigenschaft mit mehreren Werten belegbar, so
Werte durch Blank trennen
gesamte Liste in " " bzw. ' ' kodieren wegen den Trennblanks

Bsp: "background: url(test.gif) no-repeat middle;"

4.3.2.1.3.2.8.5. Style-Sheet und Ausgabemedien (Pseudoklasse @media)

```
<STYLE TYPE="text/css">
<!--
        @media typ_liste{eigenschaften_liste}
// -->
</STYLE>
```

typ_liste: Liste der Ausgabe-Medien
Kommatrennung
Ausgabe-Mediums können sein

"all"	alle Medien
"screen"	Bildschirm
"print"	Drucker

Bsp: print,screen;

eigenschaften_liste: Semikolontrennung
muss mit Semikolon enden
wenn Blank enthalten, so Liste in " " bzw. ' ' setzen
Bsp: {font-weight:bold;font_style:italic;}

4.3.2.1.3.2.8.6. Style-Sheet und Seiten-Eigenschaften (Pseudoklasse (@page)

@page:first {eigenschaften_liste}	Regel gehört der 1. Seite
@page:footer{eigenschaften_liste}	Regel gehört der Fußnote
@page:header{eigenschaften_liste}	Regel gehört der Kopfnote
@page:left{eigenschaften_liste}	Regel gehört der linken Seite
@page:left:header{eigenschaften_liste}	Regel gehört der Kopfnote Seite links
@page:left:footer{eigenschaften_liste}	Regel gehört der Fußnote Seite links
@page:right{eigenschaften_liste}	Regel gehört der rechten Seite
@page:right:header{eigenschaften_liste}	Regel gehört der Kopfnote Seite rechts
@page:right:footer{eigenschaften_liste}	Regel gehört der Fußnote Seite rechts

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

4.3.2.1.3.2.8.7. Style-Sheet und HTML-Tag-bezogene Eigenschaften

Hinweis: Ist für einen Style der Wert auto kodierbar, so ist das der Standardwert, falls Style nicht belegt wird.

Style-Sheet-Eigenschaften zu <A> und seine Pseudoklassen

a:link{eigenschaften_liste}	Eigenschaften für noch nicht besuchte Links
a:visited {eigenschaften_liste}	Eigenschaften für schon besuchte Links
a:active{eigenschaften_liste}	Eigenschaften für gerade besuchtes Link
a:hover{eigenschaften_liste}	Eigenschaften für Maus über Link , ab IE 4.x nur für Tags mit HREF-Attribut

Style-Sheet-Eigenschaften zu <P> und seine Pseudoklassen

p:first-line{eigenschaften_liste}	Eigenschaften Absatz 1. Zeile
p:first-letter{eigenschaften_liste}	Eigenschaften Absatz 1. Zeile, 1. Zeichen
p:before{content:"freier_text" }	Text vor dem Element einfügen

p:after{content:"freier_text" } Text nach dem Element einfügen

Style-Sheet-Eigenschaften zu <H1> bis <H6>

h1:first-line{eigenschaften_liste} Eigenschaften Überschrift 1. Zeile
h1:first-letter{eigenschaften_liste} Eigenschaften Überschrift 1. Zeile, 1. Zeichen

für H2 bis H6 analog

Beim Internet Explorer 6.0 unter Windows 98 bzw. Windows XP wurde die Darstellung der Überschriften verändert, wenn der jeweilige Standardfont benutzt wird. Empfehlung: Neudefinition der <Hx>-Tags per Style (soweit möglich).

Style-Sheet-Eigenschaften für Tabelle

caption-side:top; Überschrift oberhalb zentriert
caption-side:topleft; Überschrift oberhalb linksbündig
caption-side:topright; Überschrift oberhalb rechtsbündig
caption-side:bottom; Überschrift unterhalb zentriert
caption-side:bottomleft; Überschrift unterhalb linksbündig
caption-side:bottomright; Überschrift unterhalb rechtsbündig

row-span: anzahl_der_zeilen_über_die_sich_die_zelle_ausstrecken_soll;
column-span: anzahl_der_spalten_über_die_sich_die_zelle_ausstrecken_soll;

Style-Sheet und Elemente-Anzeige als Aufzählungsliste (Bullet)

display: list-item; Element in eigenem Absatz UND mit Bullet anzeigen

Style-Sheet-Eigenschaften für Liste (numerisch, Aufzählung)

list-style-type:decimal;		1	2	3 ...
list-style-type:lower-roman;	i	ii	iii ..	
list-style-type:lower-alpha	a	b	c	
list-style-type:upper-roman;	I	II	III ...	
list-style-type:upper-alpha;	A	B	C	

list-style-type:disk; Bullet ist Dateisymbol
list-style-type:circle; Bullet ist rund
list-style-type:square; Bullet ist rechteckig
list-style-type:none;

list-style-position:inside; Listenelement einrücken; ist Standard
list-style-position:outside Listenelement ausrücken

list-style-image: url(bullet_grafik_datei); GIF oder JPG

list-style: liste_aller_obigen_eigenschaften_mit_blank_trennung;

Style-Sheet und Dimension von Elementen

height:wert; Höhe des Objektes
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe
Hinweis: wenn geändert wird, so kann automatischer Umbruch des Inhaltes des HTML-Elementes erfolgen
height:auto;

width:wert; Breite des Objektes
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe
Hinweis: wenn geändert wird, so kann automatischer Umbruch des Inhaltes des HTML-Elementes erfolgen
width:auto;

Style-Sheet und Position von Elementen

position: absolute; Positionsangaben bezüglich Anzeigefenster-Rand, Element ist scrollbar
position: fixed; Positionsangaben bezüglich Anzeigefenster-Rand, Element ist nicht scrollbar
position: relative; Positionsangaben bezüglich Vorgänger-Element
position: static; hebt absolute bzw. fixed bzw. relative auf

top:wert; Abstand zum oben umgebenden Objekt ab IE 5.x
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe
top:auto;
bottom:wert; Abstand zum unten umgebenden Objekt ab IE 5.x
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe

bottom:auto;
left:wert; Abstand zum links umgebenden Objekt ab IE 5.x
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe
left:auto;
right:wert; Abstand zum rechts umgebenden Objekt ab IE 5.x
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe
right:auto;

offset-left: wie left ab IE 5.x
offse-top: wie top ab IE 5.x

pixel-left:wert; absolute X-Position der linken oberen Objekt-Ecke im Pixelsystem, dessen Ursprung (0,0) links oben liegt, also Abstand vom linken Fensterrand
wert nur Integer und ohne Einheit, da automatisch in Pixel interpretiert

pixel-top:wert; absolute Y-Position der linken oberen Objekt-Ecke im Pixelsystem, dessen Ursprung (0,0) links oben liegt, also Abstand vom oberen Fensterrand
wert nur Integer und ohne Einheit, da automatisch in Pixel interpretiert

Style-Sheet und Abstand von HTML-Elementen
top:abstand_oberhalb_in_pixel;
top:auto;

left:abstand_links_in_pixel;
left:auto;

bottom:abstand_unterhalb_in_pixel;
bottom:auto;

right:abstand_rechts_in_pixel;
right:auto;

margin: werte_liste_von_breiten_mit_blank_trennung; Randbreite aller 4 Seiten

wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

wenn nur 1 Wert kodiert: gilt für oben UND unten UND links UND rechts

wenn 2 Werte kodiert: 1. Wert gilt für oben UND unten
2. Wert gilt für links UND rechts

wenn 3 Werte kodiert: 1. Wert gilt für oben
2. Wert gilt für links UND rechts
3. Wert gilt für unten

wenn 4 Werte kodiert: 1. Wert gilt für oben
2. Wert gilt für rechts
3. Wert gilt für unten
4. Wert gilt für links

margin:auto;
margin-top: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite oben
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%
margin-top:auto;
margin-bottom: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite unten
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%
margin-bottom:auto;
margin-left: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite links
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%
margin-left:auto;
margin-right: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite rechts
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%
margin-right:auto;

padding: werte_liste_von_breiten_mit_blank_trennung; Innenabstand aller 4 Seiten

wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

wenn nur 1 Wert kodiert: gilt für oben UND unten UND links UND rechts

wenn 2 Werte kodiert:
1. Wert gilt für oben UND unten
2. Wert gilt für links UND rechts

wenn 3 Werte kodiert:
1. Wert gilt für oben
2. Wert gilt für links UND rechts
3. Wert gilt für unten

wenn 4 Werte kodiert:
1. Wert gilt für oben
2. Wert gilt für rechts
3. Wert gilt für unten
4. Wert gilt für links

padding:auto;

padding-top: werte_liste_obiger_breiten_mit_blank_trennung; Innenabstand oben
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

padding-top:auto;

padding-bottom: werte_liste_obiger_breiten_mit_blank_trennung; Innenabstand unten
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

padding-bottom:auto;

padding-left: werte_liste_obiger_breiten_mit_blank_trennung; Innenabstand links
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

padding-left:auto;

padding-right: werte_liste_obiger_breiten_mit_blank_trennung; Innenabstand rechts
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

padding-right:auto;

Style-Sheet und Überlagerung von Elementen

z-index:auto; jedes neu erzeugte Objekt ist das oberstes
Achtung: Sollte die Dimension des neuen Objektes sich nach der Erzeugung derart verändern, so dass es ein anderes Objekt überlagert, dann ändert sich trotzdem nichts an der Sichtbarkeit

z-index:wert; Schichtnummer des Objektes bei sich irgendwann überlagernden Objekten
(also auch bei Überlagerung, die erst nach deren Erzeugungerfolgt)
Wert Integer ganzzahlig, also auch < 0
je kleiner um so tiefer in der Schicht
je höher um so höher in der Schicht
Sichtbar ist immer die oberste, also höchste Schicht
wenn 2 Objekte mit identischer Schichtnummer, so Sichtbarkeit laut Reihenfolge der Kodierung im Quelltext

Style-Sheet und HTML-Elemente-Anordnung im Dokument

direction:ltr; von links nach rechts; ist Standard
direction:rtl; von rechts nach links

display:block; Element in eigenem Absatz anzeigen
display:inline; Element nicht in eigenem Absatz anzeigen
dsiplay:none; Element nicht anzeigen
display:list-item;

float:left; Element linksbündig; Textfluss rechts
float:right; Element rechtsbündig; Textfluss links
float:none: Standard

clear:left; hebt float:left; auf
clear:right; hebt float:right; auf
clear:both; hebt float:left; UND float:right; auf
clear:none; identisch mit float:none;

Style-Sheet und HTML-Element-Dimension (-Grenzen, -Anzeigebereich)

width: breite_in_pixel;
width:auto;
min-width: minimale_breite_in_pixel;

min-width:auto;
max-width: maximale_breite_in_pixel;
max-width:auto;

height: hoehe_in_pixel;
height:auto;
min-height: minimale_hoehe_in_pixel;
min-height:auto;
max-height: maximale_hoehe_in_pixel;
max-height:auto;

overflow:hidden; wenn Elementgröße > max-width und max-heigt, so wird Element wird auf Maximalwerte begrenzt
overflow:visible; wenn Elementgröße > max-width und max-heigt , so werden Maximalwerte ignoriert

Style-Sheet und HTML-Element-Sichtbarkeit

visibility:hidden;	unsichtbar, aber Platzhalter vorhanden, so dass das HTML-Element im Layout unsichtbar bleibt
visibility:visible;	sichtbar
visibility:inherit;	Sichtbarkeit laut Elternobjekt
display:none;	unsichtbar UND kein Platzhalter möglich, also HTML-Element auch aus dem Layout entfernen
display:block;	sichtbar
display:inline;	sichtbar

Style-Sheet und Hintergrund-Grafik

background-image:url (grafik_datei**);**	Hintergrunddatei GIF oder JPG
background-image:none;	
background-color:#rrggbb;	Hintergrundfarbe
background-color:rgb(rrr,ggg,bbb);	
background-color:vordefinierte_farbe;	
background-repeat:repeat;	Hintergrundgrafik kacheln auf gesamten Hintergrund
background-repeat:repeat-x;	Hintergrundgrafik für 1 Zeile kacheln
background-repeat:repeat-y;	Hintergrundgrafik für 1 Spalte kacheln
background-repeat:no-repeat;	kein Kacheln des Hintergrundbildes
background-attachment:scroll;	Hintergrundgrafik scrollt mit Vordergrund
background-attachment:fixed;	Hintergrundgrafik scrollt nie
background-position:top;	Hintergrundgrafik auf Oberkante Hintergrund
background-position:center;	Hintergrundgrafik zentriert auf Hintergrund
background-position:middle;	Hintergrundgrafik mittig auf Hintergrund
background-position:bottom;	Hintergrundgrafik auf Unterkante Hintergrund
background-position:left;	Hintergrundgrafik linksbündig auf Hintergrund
background-position:right;	Hintergrundgrafik rechtsbündig auf Hintergrund

background: eigenschaften_liste_mit_blank_trennung; alle obigen Eigenschaften kodierbar mit Blanktrennung

Bsp: background: url(back.gif) no-repeat middle; Blank-Trennung !!

-moz-opacity:faktor
Transparenz des Hintergrundbildes
nur NS 6.x
Faktor > 0 und bis 1
Bsp: 0.9 entpricht 90%
Bsp: <IMG STYLE="-moz-opacity:0.9" SRC="test.jpg">

Style-Sheet und Rahmen-Eigenschaften (Border)

border-color:#rrggbb;	Rahmenfarbe aller 4 Seiten
border-color:rgb(rrr,ggg,bbb);	
border-color:vordefinierte_farbe;	
border-top-color:#rrggbb;	Rahmenfarbe oben
border-top-color:rgb(rrr,ggg,bbb);	
border-top-color:vordefinierte_farbe;	
border-bottom-color:#rrggbb;	Rahmenfarbe unten
border-bottom-color:rgb(rrr,ggg,bbb);	
border-bottom-color:vordefinierte_farbe;	
border-left-color:#rrggbb;	Rahmenfarbe links
border-left-color:rgb(rrr,ggg,bbb);	
border-left-color:vordefinierte_farbe;	
border-right-color:#rrggbb;	Rahmenfarbe rechts
border-right-color:rgb(rrr,ggg,bbb);	

border-right-color:vordefinierte_farbe;

border-style:none;	kein Rahmen um das gesamte Element
border-style:dotted;	gepunkteter Rahmen um das gesamt Element
border-style:dashed;	gestrichelter Rahmen um das gesamte Element
border-style:solid;	einfacher durchgezogener Rahmen um das gesamte Element
border-style:double;	doppelter durchgezogener Rahmen um das gesamte Element
border-style:groove;	3D-Effekt 1 um das gesamte Element
border-style:ridge;	3D-Effekt 2 um das gesamte Element
border-style:inset;	3D-Effekt 3 um das gesamte Element
border-style:outset;	3D-Effekt 4 um das gesamte Element
border-top-style:none;	kein Rahmen oben
border-top-style:dotted;	gepunkteter Rahmen oben
border-top-style:dashed;	gestrichelter Rahmen oben
border-top-style:solid;	einfacher durchgezogener Rahmen oben
border-top-style:double;	doppelter durchgezogener Rahmen oben
border-top-style:groove;	3D-Effekt 1 oben
border-top-style:ridge;	3D-Effekt 2 oben
border-top-style:inset;	3D-Effekt 3 oben
border-top-style:outset;	3D-Effekt 4 oben
border-bottom-style:none;	kein Rahmen unten
border-bottom-style:dotted;	gepunkteter Rahmen unten
border-bottom-style:dashed;	gestrichelter Rahmen unten
border-bottom-style:solid;	einfacher durchgezogener Rahmen unten
border-bottom-style:double;	doppelter durchgezogener Rahmen unten
border-bottom-style:groove;	3D-Effekt 1 unten
border-bottom-style:ridge;	3D-Effekt 2 unten
border-bottom-style:inset;	3D-Effekt 3 unten
border-bottom-style:outset;	3D-Effekt 4 unten
border-left-style:none;	kein Rahmen links
border-left-style:dotted;	gepunkteter Rahmen links
border-left-style:dashed;	gestrichelter Rahmen links
border-left-style:solid;	einfacher durchgezogener Rahmen links
border-left-style:double;	doppelter durchgezogener Rahmen links
border-left-style:groove;	3D-Effekt 1 links
border-left-style:ridge;	3D-Effekt 2 links
border-left-style:inset;	3D-Effekt 3 links
border-left-style:outset;	3D-Effekt 4 links
border-right-style:none;	kein Rahmen rechts
border-right-style:dotted;	gepunkteter Rahmen rechts
border-right-style:dashed;	gestrichelter Rahmen rechts
border-right-style:solid;	einfacher durchgezogener Rahmen rechts
border-right-style:double;	doppelter durchgezogener Rahmen rechts
border-right-style:groove;	3D-Effekt 1 rechts
border-right-style:ridge;	3D-Effekt 2 rechts
border-right-style:inset;	3D-Effekt 3 rechts
border-right-style:outset;	3D-Effekt 4 rechts
border-width:wert	Rahmendicke aller 4 Seiten wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
border-width:medium	mittel
border-width:thin	dünn
border-width:thick	dick
border-top-width:wert	Rahmendicke oben, wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
border-top-width:medium	mittel oben
border-top-width:thin	dünn oben
border-top-width:thick	dick oben
border-bottom-width:wert	Rahmendicke unten, wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
border-bottom-width:medium	mittel unten
border-bottom-width:thin	dünn unten
border-bottom-width:thick	dick unten
border-left-width:wert	Rahmendicke links, wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
border-left-width:medium	mittel links
border-left-width:thin	dünn links
border-left-width:thick	dick links
border-right-width:wert	Rahmendicke rechts, wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
border-right-width:medium	mittel rechts
border-right-width:thin	dünn rechts
border-right-width:thick	dick rechts

border-top:eigenschaften_liste; passende Werte-Liste für Rahmen oben, Werte mit Blank trennen
border-bottom: eigenschaften_liste; passende Werte-Liste für Rahmen unten, Werte mit Blank trennen
border-right: eigenschaften_liste; passende Werte-Liste für Rahmen rechts, Werte mit Blank trennen
border-left: eigenschaften_liste; passende Werte-Liste für Rahmen links, Werte mit Blank trennen
border:eigenschaften_liste; Werte-Liste **aller** möglichen Bordereigenschaften, Werte mit Blank trennen

Bsp: border-top: thick inset rgb(192,192,255);

Style-Sheet und Element-Ausschnitt

clip: rect (x1,y1,x2,y2); Ausschnitt
Pixelangaben bezüglich Elementgrenze
Form des Bildaussschnittes: RECHTECKIG
wird eigentlich nur zum Schreiben verwendet
x1,y1 linke obere Ecke
x1 horizontal in Pixel
y1 vertikal in Pixel
x2,y2 rechte untere Ecke
x2 horizontal in Pixel
y2 vertikal in Pixel
x1 bis y2 immer bezüglich links oben im HTML-Element
x1,y1, x2,y2 können den Wert auto haben, der immer die maximale Dimension bewirkt
Bsp. 'rect:(auto,y1,auto,y2)';
wenn alle auf auto, so wird der Bildausschnitt maximiert,was der Standardimension des Elementes entspricht (kein Ausschnitt)

clip:auto; kein Ausschnitt
entspricht clip:rect(auto,auto,auto,auto);

Style-Sheet und Element-Scrolling

Scrolling nötig, wenn
Elementgröße > max-width und max-heigt
aber Maximalwerte eingehalten werden sollen

overflow:scroll; Scrolling immer möglich
overflow:auto;

siehe auch overflow:visible; bzw. overflow:hidden;

Style-Sheet-Eigenschaften für Seitendarstellung im Dokument

margin: werte_liste_von_breiten_mit_blank_trennung; Randbreite aller 4 Seiten

wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

wenn nur 1 Wert kodiert: gilt für oben UND unten UND links UND rechts

wenn 2 Werte kodiert:
1. Wert gilt für oben UND unten
2. Wert gilt für links UND rechts

wenn 3 Werte kodiert:
1. Wert gilt für oben
2. Wert gilt für links UND rechts
3. Wert gilt für unten

wenn 4 Werte kodiert:
1. Wert gilt für oben
2. Wert gilt für rechts
3. Wert gilt für unten
4. Wert gilt für links

margin:auto;

margin-top: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite oben
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

margin-top:auto;

margin-bottom: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite unten
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

margin-bottom:auto;

margin-left: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite links
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

margin-left:auto;

margin-right: werte_liste_obiger_breiten_mit_blank_trennung; Randbreite rechts
wert ist Fliesskommazahl mit Einheitenangabe z.B. pt oder cm
Prozentangabe z.B. 3%

margin-right:auto;

size:seiten_breite; z.B. 20cm
size:seiten_hoehe; z.B. 29cm
size:seiten_breite seiten_hoehe; Blanktrennung der beiden Werte
size:landscape; Querformat
size:portrait; Hochformat
size:auto;

page-break_before:always; immer Seitenumbruch vor aktuellem Element erzeugen
page-break_before:aroid; nie Seitenumbruch vor aktuellem Element erzeugen
page-break_before:left; immer Seitenumbruch vor aktuellem Element erzeugen UND Element dann linksbündig ablegen
page-break_before:right; immer Seitenumbruch vor aktuellem Element erzeugen UND Element dann rechtsbündig ablegen
page-break_before:auto;

page-break_after:always; immer Seitenumbruch nach aktuellem Element erzeugen
page-break_after:aroid; nie Seitenumbruch nach aktuellem Element erzeugen
page-break_after:left; immer Seitenumbruch nach aktuellem Element erzeugen UND Element linksbündig ablegen
page-break_after:right; immer Seitenumbruch nach aktuellem Element erzeugen UND Element rechtsbündig ablegen
page-break_after:auto;

Style-Sheet-Eigenschaften für Text

text-ident:wert; Texteinschub
wert ist Fliesskommazahl mit Einheit z.B. mm oder cm oder pt
Prozent z.B. 3%
wert > 0, so Textzeile einrücken
wert < 0, so Textzeile ausrücken

text-align:left; Text linksbündig
text-align:center; Text zentriert
text-align:right; Text rechtsbündig
text-align:justify; Text im Blocksatz

vertical-align:top; Text auf Oberkante der Umgebung
vertical-align:middle; Text auf Mitte der Umgebung
vertical-align:bottom; Text auf Unterkante der Umgebung
vertical-align:sub; Text tieferstellen
vertical-align:super; Text höher stellen
vertical-align:baseline; Text auf Basislinie des umgebenden Textes mit verschiedenen Schriftarten
vertical-align:text-top; Text auf obere Linie des umgebenden Textes mit verschiedenen Schriftarten
vertical-align:text-bottom; Text auf untere Linie des umgebenden Textes mit verschiedenen Schriftarten

font-size:schriftgroesse_wert; z.B. 2pt oder vordefiniert

line-height:zeilenhoehe_wert; z.B. 2pt* oder vordefiniert

white-space:normal; automatischer Zeilenumbruch einschalten, ab IE 5.x
white-space:pre; manuellen Zeilenumbruch einschalten, ab IE 5.x
white-space:nowrap; kein Zeilenumbruch möglich, ab IE 5.x

color:#rrggbb;
color:rgb(rrr,ggg,bbb);
color:vordefinierte_farbe;

colums:anzahl_der_text_spalten_fuer_textfluss;
column-gap:abstand_der_text_spalten_fuer_textfluss;

column-rule-width:dicke_des_trennstriches_zwischen_den_textspalten_vom_textfluss;
column-rule-color:#rrggbb; oder rgb(rrr,ggg,bbb); Farbe Trennstrich
column-rule-style:none; kein Trennstrich zwischen Textspalten
column-rule-style:dotted; gepunkteter Trennstrich
column-rule-style:dashed; gestrichelter Trennstrich
column-rule-style:solid; einfacher durchgezogener Trennstrich
column-rule-style:double; doppelter durchgezogener Trennstrich
column-rule-style:groove; 3D-Effekt 1
column-rule-style:ridge; 3D-Effekt 2
column-rule-style:inset; 3D-Effekt 3

column-rule-style:outset; 3D-Effekt 4
column-rule:werte_liste_aus_obigen_textfluss_werten_mit_blank_trennung;

word-spacing: abstand_zwischen_woertern_im_text;

word-wrap:normal; kein Wortumbruch, ab IE 5.x
word-wrap:break-word; Wortumbruch, ab IE 5.x

letter-spacing:abstand_zwischen_zeichen_im_text; Fliesskommazahl mit Einheit z.B. mm oder cm oder pt
letter-spacing:normal;

line-height:abstand_zwischen_2_zeilen; Fliesskommazahl mit Einheit z.B. mm oder cm oder pt
line-height:normal;

text-decoration:underline; Text unterstrichen
text-decoration:overline; Text überstrichen
text-decoration:line-through; Text durchgestrichen
text-decoration:blink; Text blinkend
text-decoration:none; Text normal

text-transform:capitalize; Wortanfang ALLER Wörter auf Großbuchstabe
text-transform:uppercase; alle Zeichen nach Großbuchstabe
text-transform:lowercase; alle Zeichen nach Kleinbuchstabe
text-transform:none; Text normal

text-shadow:#rrggbb;
text-shadow:rgb(rrr,ggg,bbb);
text-shadow:vordefinierte_farbe;
text-shadow:none; kein Textschatten

orphans:anzahl_der_VOR_seitenumbruch_zusammenzuhaltender_zeilen; Standard ist 2
entspricht Schusterjunge

widow:anzahl_der_NACH_seitenumbruch_zusammenzuhaltender_zeilen; Standard ist 2
entspricht Hurenkind

Style-Sheet-Eigenschaften für Unicode (Zeichensatz)
unicode-range:U+xxxx-yyyy;

Fragezeichen als Joker innerhalb von xxxx und yyyy verwendbar

Bsp: unicode-range:U+0000-007F; ist ASCII-Zeichensatz
unicode-range:U+0000-00?F; ? steht für 0 bis F --> ist nicht ASCII-Zeichensatz

Style-Sheet-Eigenschaften für Font
font-family: schriftarten_liste_mit_kommatrennung;

es wird **nur** der **ERSTE** auf dem lokalen Rechner **gefundene** Font aus der Liste geladen
wenn Blank vorhanden, so alles in " " bzw. ' ' setzen
vordefinierte Schriftarten: serif
san-serif
cursive
fantasy
monospace

font-style:italic; kursiv
font-style:oblique; kursiv
font-style:normal; nicht kursiv

font-variant:small-caps; kleine Großbuchstaben (Kapitälchen)
font-variant:normal; kein Kapitälchen

font-size:schrift_groesse_wert; Fliesskommazahl mit Einheit z.B. mm oder cm oder pt
Prozent z.B. 3%
font-size:xx-small; winzig
font-size:x-small; sehr klein
font-size:small; klein
font-size:medium; mittel
font-size:large; groß
font-size:x-large; sehr groß
font-size:xx-large; riesig
font-size:smaller; etwas kleiner als normal

font-size:larger;	etwas größer als normal
font-weight:bold;	fett
font-weight:bolder;	extra fett
font-weight:lighter;	dünn
font-weight:normal;	nicht dünn und nicht fett
font-weight:100;	extra dünn
font-weight:200;	
font-weight:300;	
font-weight:400;	
font-weight:500;	medium
font-weight:600;	
font-weight:700;	bold
font-weight:800;	
font-weight:900;	extra fett

font: liste_aller_obiger_eigenschaften_mit_blank_trennung;

Liste darf maximal 6 der nachfolgenden Eigenschaften beinhalten

caption	Zeichnsatz für Objekte mit Überschriften
font-style	
font-variant	
font-weight	
font-size	
line-height	
font-family	
icon	Zeichensatz für Grafik
menu	Zeichensatz in Menüs
message-box	Zeichensatz in Mesages-Boxen
small-caption	Zeichensatz für Control-Elemente
status-bar	Zeichensatz für Stautszeile

Style-Sheet-Eigenschaften für Mauscursor

cursor:auto;	
cursor:default;	je nach Windowseinstellung
cursor:hand;	
cursor:crosshair;	Fadenkreuz
cursor:pointer;	Zeiger
cursor:move;	Kreuz für Beweglichkeit
cursor:n-resize;	Pfeil Nord
cursor:ne-resize;	Pfeil Nord-Ost
cursor:nw-resize;	Pfeil Nord-West
cursor:e-resize;	Pfeil Ost
cursor:se-resize;	Pfeil Süd-Ost
cursor:s-resize;	Pfeil Süd
cursor:sw-resize;	Pfeil Süd-West
cursor:w-resize;	Pfeil West
cursor:text;	| also senkrechter Strich
cursor:wait;	Sanduhr
cursor:help;	Fragezeichen
cursor:url(url_oder_mauscursor_grafik_datei);	GIF oder JPG

Style-Sheet-Eigenschaften für Scrollbalken (Scrollbars)

scrollbar3d-light-color:#rrggbb	Scrollbar-Farbe bei 3D
scrollbar3d-light-color:rgb(rrr,ggg,bbb);	
scrollbar3d-light-color:vordefinierte_farbe;	
scrollbar-arrow-color:#rrggbb;	Scrollbar-Pfeile-Farbe ab IE 5.x
scrollbar-arrow-color:rgb(rrr,ggg,bbb);	
scrollbar-arrow-color:vordefinierte_farbe;	
scrollbar-base-color:#rrggbb;	Scrollbar-Basis-Farbe ab IE 5.x
scrollbar-base-color:rgb(rrr,ggg,bbb);	
scrollbar-base-color:vordefinierte_farbe;	
scrollbar-dark-shadow-color:#rrggbb;	Scrollbar-Farbe des dunklen Schatten ab IE 5.x
scrollbar-dark-shadow-color:rgb(rrr,ggg,bbb);	
scrollbar-dark-shadow-color:vordefinierte_farbe;	
scrollbar-face-color:#rrggbb;	Scrollbar-Face-Farbe ab IE 5.x
scrollbar-face-color:rgb(rrr,ggg,bbb);	
scrollbar-face-color:vordefinierte_farbe;	
scrollbar-highlight-color:#rrggbb;	Scrollbar-Highlight-Farbe ab IE 5.x
scrollbar-highlight-color:rgb(rrr,ggg,bbb);	

scrollbar-highlight-color:vordefinierte_farbe;
scrollbar-shadow-color:#rrggbb; Scrollbar-Farbe des Schatten ab IE 5.x
scrollbar-shadow-color:rgb(rrr,ggg,bbb);
scrollbar-shadow-color:vordefinierte_farbe;

Style-Sheet-Eigenschaften für Zoom eines Elementes

zoom:wert; Objekt vergrößern ab IE 5.x
wert ist Fliesskommazahl als Faktor (1,0 ist normal)
Prozentangabe mit % z.B. 50% (100% ist normal)

zoom:normal;

4.3.2.1.3.2.8.8. Style-Sheet-Beispiele

Beispiel 1

```
<STYLE TYPE="text/css">
<!--
        h1{font-size:24pt;margin-top:1.2cm;margin-left:30px;}
        body {background-color:rgb(51,0,102);}
        p,li {font-size:12pt;line-height:14pt;font-family:Helvetica,Arail;}
        p.normal {fonszie:10pt;color:bllack;}
        p.klein {fontsize:8pt;color:black}
        all.rot {color.red;}    oder .rot {color:red}
                                Rot-Definition für ALLE Tags
                                    Anwendung  z.B. per <P CLASS="normal">text</P>
                                    Anwendung  z.B. per <P CLASS="rot">text</P>
                                    Anwendung  z.B. per <H1 CLASS="rot">text</H1>
        #fett_kursiv {font-weight:bold;font-style:italic;}
                                Anwendung z.B. per <P ID="fett_kursiv">text</P>
        a:link {color:#FF0000;font-weight:bold;}
                                Anwendung z.B. per <A HREF="....." >text</A>
// -->
</STYLE>

Anwendung je nach Tag --> meist innerhalb <BODY>
```

Beispiel 2

```
<BODY>
        <DIV STYLE="background-color:#FF0000;">test</DIV>
        <P>
                text1
                <SPAN   STYLE="color:red;">
                        text2_in_rot;
                </SPAN>
                text3_wie_text1
        </P>
</BODY>
```

4.3.2.1.4. window.event Objekt des Netscape

Instanz aller Ereignisse im Fenster

Ein Ereignis ist ein browserinternes Signal zu einer getätigten Aktion anhand des HTML-Elementes. Das Signal zeigt an, dass eine Aktion erfolgt ist bzw. noch andauert. Typisches Ereignis ist das Mausklicken des Users auf ein HTML-Element: Klickt der User, so wird die Aktion "Klick" per Ereignis "onclick" signalisiert, wenn das HTML-Element klickbar ist, also die Aktion "Klick" unterstützt.

Ein Ereignis kann von einem konkreten HTML-Element bzw. Objekt nur dann ausgelöst werden, wenn es Events überhaupt unterstützt. Nicht alle HTML-Elemente unterstützen Events. Die Event-Unterstützung von HTML-Elementen ist in der Scriptmaschine spezifisch zu jedem HTML-Element genau vordefiniert. Aus dieser Menge der vordefinierten Events zum HTML-Element kann der Programmierer schöpfen und Aktionen des Users zum HTML-Element zulassen oder unterbinden. Dabei gilt die Regel: Was nicht explizit zugelassen wurde, gilt als unterdrückt, falls es keine standardmäßige Zulassung gibt.

Pro Aktionsart existiert eine Eventart. Erfreulicherweise gibt es diverse Aktionen, die von vielen HTML-Elemente unterstützt werden, vorallem eben o.g. Useraktionen. Aber es gibt diverse Aktionen, von denen der User nichts mitbekommt. Die Eventbehandlung dient also nicht ausschliesslich der Aktionsfreudigkeit des Users und Programmierers, sondern ist ein Nebeneffekt in Form der interaktiven Webseite.

Viele Objekte können erst kommunizieren, wenn sie Events erzeugen bzw. ein Signal als Voraussetzung für eine Aktion bekommen. Dabei ist das Wort "Aktion" auch als "Verhalten" zu verstehen., also als Komponente zur Steuerung von HTML-Elementen während der Anzeige des HTML-Dokumentes im Browserfenster. Typisches Event ist das Signal "onload", das ausgelöst wird, wenn das HTML-Dokument **komplett geparst** und falls nötig in das Browserfenster geladen **wurde**. Es gibt diverse Aktionen, die erst **nach** Auslösung von onload zulässig sind. Diese Aktionen betreffen auch die skriptgesteuerte Verwaltung des HTML-Dokumentes zu dessen Laufzeit.

Jedes instanzierte Objekt, das Events unterstützt, nutzt das Eventobjekt, mit dem die Verarbeitung aller unterstützten Events möglich ist (verschiedene Eventarten).

Aufgrund der Verschachtelung von HTML-Elementen müssen Events auch innerhalb der Hierarchie der HTML-Elemente **und** in der Vererbungsfolge verfügbar sein. Events können also durchgereicht werden. In Script wird üblicherweise die Punktnotation für Hierarchieebenen verwendet. Events können nur dann durchgereicht werden, wenn **beide betroffene Ebenen** diese Events unterstützen. Natürlich müssen die verschachtelten HTML-Elemente, welche Events durchreichen (und diese eventuell zuvor auswerten, wobei das Kind anders auf das Event reagieren kann als Eltern, die aber über das Ereignis immer informiert werden **sollten**), das Entstehen von Ereignissen auch überwachen, denn es muss nicht bekannt sein, wann und wo das Ereignis entsteht. Diese Überwachung kann in Verbindung mit Script genau gesteuert werden.

Es existiert pro HTML-Element bzw. Objekt, das Events unterstützt, eine je nach Eventart vordefinierte Standardbehandlung. Aber Events können abweichend von der Standardbehandlung durch private Eventhandler in Script verarbeitet werden. Auch die Eventüberwachung kann in Script genau gesteuert werden: Es muss also nicht sein, dass ein Kind ein Event seinen Eltern mitteilt (darüber entscheidet der Programmierer).

Aufgrund dieser Tatsache sind Script-Methoden der Eventverwaltung nicht im Objekt event implementiert, sonden im Objekt, das das Event erzeugt. Die Implementierung ist objekt-spezifisch aber genormt. Es gibt also nicht viele jedoch wie immer browserspezifische Methoden.

Instanz-Erzeugung in HTML:

Beispiele zur Kodierung von onXXX="..." im HTML-Tag des Objektes, wobei onXXX ein im Objekt implementiertes Event ist, z.B. onclick.

Übliche Kodierung:

```
<HEAD>
        <SCRIPT>
                function InputButton_Event_onclick_Handler()
                {alert("onclick erkannt");}
        </SCRIPT>
</HEAD>
<BODY>
        <INPUT  TYPE=button
                NAME="ID_Button"
                VALUE="Test"
                onclick="InputButton_Event_onclick_Handler()"
        >
</BODY>
```

Alternative 1:

```
<HEAD>
        <SCRIPT>
                function InputButton_Event_onclick_Handler()
                {alert("onclick erkannt");}
        </SCRIPT>
</HEAD>
<BODY>
        <INPUT  TYPE=button
                NAME="ID_Button"
                VALUE="Test"
        >
        <SCRIPT>
                // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
                //      eine im HEAD deklarierte Funktion

                // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
                //      geparst wird, also im HEAD das INPUT-Tag nicht bekannt ist
                ID_Button.onclick =  InputButton_Event_onclick_Handler; //ohne ()
        </SCRIPT>
</BODY>
```

Alternative 2:

```
<BODY>
        <INPUT  TYPE=button
                NAME="ID_Button"
                VALUE="Test"
        >
        <SCRIPT>
                // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
                //      eine im BODY deklarierte Funktion

                // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
```


```
                    //          geparst wird, also im HEAD das INPUT-Tag nicht bekannt ist
                    ID_Button.onclick =  InputButton_Event_onclick_Handler; //ohne ()

                    function ID_Button.onclick()
                    {alert("onclick erkannt");}
            </SCRIPT>
    </BODY>
```

Alterantive 3:

```
    <HEAD>
        <SCRIPT>
            function InputButton_Event_onclick_Handler()
            {alert("onclick erkannt");}
        </SCRIPT>
    </HEAD>
    <BODY>
        <INPUT  TYPE=button
                NAME="ID_Button"
                VALUE="Test"
        >
        <SCRIPT>
            // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
            //          eine im BODY deklarierte Funktion, die eine Funktion im HEAD aktiviert

            // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
            //          geparst wird, also im HEAD das INPUT-Tag nicht bekannt ist
            function ID_Button.onclick()
            {InputButton_Event_onclick_Handler();}
        </SCRIPT>
    </BODY>
```

Events bei sich überlagernden Objekten:

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

4.3.2.1.4.1. Eventarten (Auswahl)

onabort
Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
Tag <IMG>
Objekt image
NS ab 3.x

onblur
Ereignis, das direkt vor dem De-Fokussieren ausgelöst wird
Tag z.B. <FRAME>, <INPUT>, <TEXTAREA>
NS ab 3.x

onchange
Ereignis ausgelöst, wenn Inhalt von Eingabefeld, Textarea, Auswahlliste, Radiobox, Checkbox durch User verändert wurde und **nur** bei Eingabefeld, Textarea, Auswahlliste diese de-fokussiert wurden (bei Radiobox und Checkbox sofort Ereignisauslösung)
Handler hat innerhalb <OPTION> eines <SELECT> keine Wirkung, also nur innerhalb <SELECT> kodieren
Tag z.B. <INPUT>, <SELECT>, <TEXTAREA>
NS ab 3.x

onclick
Ereignis ausgelöst, wenn Maus das Objekt angeklickt hat: 1 Klick = Maustaste drücken und loslassen
im Handler return false; kodieren für Links, Formularelemente, wenn eine Aktion nicht erwünscht ist
Bsp.: bei Link wird dieser nicht ausgeführt, obwohl angeklickt
Tag z.B. <A>, <BODY>, <INPUT>
Objekt document, button, submit, reset, checkbox
NS ab 3.x

ondblclick
Ereignis ausgelöst, wenn Doppelklick auf Objekt durch linke bzw. rechte Maustaste erfolgt
1 Klick = Maustaste drücken und loslassen
Tag fast alle
Objekt link
NS ab 4.x

ondragdrop

NS ermöglicht Drag&Drop per Maus von Dateien und Verknüpfungen auch über Fenster
Ablauf: Klick auf Element und Maus gedrückt lassen,
dann Element ziehen an gewünschte Position,
dann Maus loslassen
Ereignis ausgelöst, wenn eine Datei oder Verknüpfung verschoben wurde
Tag <BODY>
Objekt window
nur NS ab 4.x

onerror

Ereignis ausgelöst, wenn
während des Ladens eines HTML-Dokumentes ein Syntaxfehler auftritt
eines Bildes ein Fehler auftritt
während der Abarbeitung von Scripten ein Runtime-Error auftritt

sämtliche Fehlermeldungen unterbinden per
var RetteOnErrorHandler = window.onerror;
window.onerror = null;

Eventhandler muss wie folgt kodiert werden:

```
function freier_name_fuer_onerror_behandlung
(   error_erklaerung_string,
    url_des_html_dokumentes_als_string,
    zeilen_nr_des_errors_im_html_dokument
)
{
        .....
        return true;        //  nur wenn true geliefert, dann
                                //      wird die browsereigene
                                //      onerror-Behandlung
                            //      unterdrückt
}
```

pro Fehler ein Aufruf des Eventhandlers --> Folge von Fehlern, also Folge von Aufrufen
Tag z.B. <IMG>
Objekt window, img
NS ab 3.x

onfocus

Ereignis ausgelöst, wenn HTML-Element fokussiert wird, also das aktuelle wird
Tag z.B. <BODY>, <DIV>, <INPUT>, <FRAMESET>, <TEXTAREA>
Objekt window
NS ab 3.x

onkeydown

Ereignis ausgelöst für alle Tastenarten
füllt nur beim **ersten** Tastendruck einmalig den ASCII-Code der Taste nach Eigenschaft .which
(auch bei Dauerdruck der Taste nur beim ersten Tastendruck geliefert)
Hinweis: nach String umwandeln per .fromCharMode()
Besonderheit bei Kombination von Steuertaste mit anderer Taste
(z.B. Shift-Taste und B-Taste zugleich für Großbuchstabe B):
Ereignis wird pro Tastendruck aufgerufen, also
1. Mal für Shift-Taste
2. Mal für B-Taste
keydown-Ereignis ruft automatisch das keypress-Ereignis auf, es sei denn, der keydown-Handler
liefert return false;
Tag z.B. <A>, <IMG>, <TEXTAREA>
Objekt document
NS ab 4.x

onkeypress

wird unmittelbar nach dem keydown-Ereignis ausgelöst allerdings nur für
Tastenarten
! @ # $ % ^ & * ()_- + = < [] { } , . / ? \ | ' " ` ~
0 bis 9
a bis z Gross und Klein wird unterschieden !
Enter
Leertaste
dient dem Erkennen von Tastaturkombinationen, die nicht vom Ereignis keydown abgefangen werden
der Aufruf des keypress-Handlers ist **nur** möglich, wenn der keydown-Handler return true; liefert
Tag z.B. <A>, <IMG>, <TEXTAREA>
Objekt document
NS ab 4.x

onkeyup
Ereignis ausgelöst, wenn gedrückte Tastatur-Taste **jeder** Art losgelassen wird
wird automatisch nach Ereignis keypress ausgelöst
besitzt vordefiniertes Schlüsselwort KEYUP für captureEvents
also document.captureEvent(Event.KEYUP);
Tag z.B. <A>, <IMG>, <TEXTAREA>
Objekt document
NS ab 4.x

onload
Ereignis ausgelöst, wenn **vollständig** geladen wurde
HTML-Dokument mit all seinen Elementen jeder Art (BODY-Teil des Dokumentes wurde komplett geparst)
Framset (innerhalb <FRAMESET> kodieren)
Bild
aber bei animiertem Bild (Gif-Datei):
Ereignis load bei **jedem** Bildwechsel der Animation ausgelöst
Erignis muss also für jedes Teil-Bild behandelt werden
DIV
Layer (nicht ab NS-Version 6.x)

Tag z.B. <BODY>, <DIV>, <FRAMESET>, <IMG>
Objekt window, img
NS ab 3.x

onmousedown
Ereignis ausgelöst mit drücken der linken oder rechten Maustaste (mittlere nicht !)
auch in Verbindung mit Umschalt (Shift) und Alt
siehe Event-Eigenschaft .which
return false; unterbindet **immer** die die Ausführung des Eventhandlers
return false bei rechter Maustase auf das Object document: Es wird das Kontextmenü gesperrt.
Tag z.B. <A>, <DIV>, <INPUT TYPE="button">
Objekt document
NS ab 4.x

onmousemove
Ereignis bei **jeder** Mausbewegung ausgelöst
Auswertung für den Bereich eines Objektes durch dessen Eventhandler
Tag z.B. <A>, <DIV>, <IMG>
Objekt document
NS ab 4.x

onmouseout
Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt
Tag z.B. <A>, <DIV>, <IMG>
Objekt document
NS ab 3.x

onmouseover
Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt
Tag z.B. <A>, <DIV>, <IMG>
Objekt document
NS ab 3.x

onmouseup
Ereignis ausgelöst mit Loslassen der linken Maustaste (rechte und mittlere nicht !)
siehe Event-Eigenschaft .which
return false; unterbindet **immer** die die Ausführung des Eventhandlers
Tag z.B. <A>, <DIV>, <INPUT TYPE="button">
Objekt document
NS ab 4.x

onmove
Ereignis ausgelöst, solange ein Fenster verschoben wird
Tag keine
Objekt window
NS ab 4.x

onreset
Ereignis VOR dem Rücksetzen eines Formulars ausgelöst
(Rücksetzen per Reset-Button bzw. Neuladen des Dokumentes)
dient zum Abfangen der Resetaktion des Users und eventueller Unterbindung des Resets z.B. wegen falscher Ausfüllung des Formulars
Reset wird nicht ausgelöst, wenn Handler return false; liefert
Tag <FORM>
Objekt form
NS ab 3.x

onresize
Ereignis mit ausgelöst bei Größenänderung des Fensters per Maus:
Maus auf Fensterrahmen

es erscheint das <-> Symbol
linke Maustaste drücken, gedrückt lassen und ziehen
linke Maus loslassen dann wird Ereignis ausgelöst

Tag keins
Objekt window
NS ab 3.x

onselect
Ereignis ausgelöst wenn sich Textmarkierung ändert
Tag z.B. <INPUT TYPE="text">, <TEXTAREA>
NS ab 3.x

onsubmit
Ereignis VOR dem Abschicken eines Formulars ausgelöst
(Abschicken per Submit-Button)
dient zum Abfangen der Resetaktion des Users und eventueller Unterbindung des Submits z.B. wegen falscher Ausfüllung des Formulars
Submit wird nicht ausgelöst, wenn Handler return false; liefert
Tag <FORM>
Objekt form
NS ab 3.x

onunload
Ereignis ausgelöst, wenn das HTML-Dokument verlassen wird durch
Schliessen Browserfenster
Wechsel zu anderer Seite auch per Browser-Button
window.document.open()
Tag z.B. <BODY>, <FRAMESET>
Objekt window
NS ab 3.x

4.3.2.1.4.2. Eigenschaften (Auswahl)

.data
Zeiger auf Feld der per Urls aller per Drag & Drop auf das HTML-Dokument abgelegten Objekte
NS ab 4.x mit signiertem Script

.height
Höhe in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde
NS ab 4.x

.layerX
Ereignis ist nicht resize:
horizontale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0)
Ereignis ist resize:
neue Breite in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Breite verändert wurde
NS ab 4.x

.layerY
Ereignis ist nicht resize::
vertikale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen
linker oberer Ecke (0,0)
Ereignis ist resize:
neue Höhe in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Höhe verändert wurde
NS ab 4.x

.modifiers
Bitmaske für Zustand der Steuertasten Shift, Alt und Strg
dient zur Ermittlung der Steuertaste per vordefinierter Maske durch bitweise UND-Verknüpfung mit der Bitmaske: Ergibt die Verknüpfung 1, also true, so ist Steuertaste gedrückt worden
vordefinierte Maske für
Alt-Taste "Event.ALT_MASK"
Strg-Taste "Event.CTRL_MASK"
Shift-Taste "Event.SHIFT_MASK"
Bsp: return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0);
// nicht logisches UND (&&) sondern bitweises UND also &
NS ab 4.x

.pageX
horizontale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
NS ab 4.x

.pageY
vertikale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
NS ab 4.x

.screenX
horizontale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes
NS ab 4.x

.screenY vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes
NS ab 4.x

.target Zeiger auf dasjenige HTML-Element, für das das Ereignis ausgelöst wurde
entweder im HTML-Tag definieren per onXXX = " ....."
oder Bezug auf den logischen Namen des HTML-Elementes
if (logischer_name = Ereignis.target) ....
NS ab 4.x

.type enthält die Event-Art z.B. abort
NS ab 4.x

.which enthält bei
gedrückter Maustaste:
1 für linke Taste
2 für mittlere Taste
3 für rechte Taste
Tastatur ASCII-Code der Taste
Hinweis: Umwandlung in String per
String.fromChatCode(TastenKodeASCII)
NS ab 4.x

.width Breite in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde
NS ab 4.x

.x identisch mit .layerX
NS ab 4.x

.y identisch mit .layerY
NS ab 4.x

4.3.2.1.4.3. Methoden

keine
Events können zusätzlich nur durch Methoden anderer Objekte verwaltet werden.

4.3.2.1.4.4. Prinzipen der Eventbehandlung des Netscape

Die Standard-Eventbehandlung wird beim Netscape immer vom window-Objekt selbst realisiert. Damit landen alle Events immer beim obersten Objekt. Eine davon abweichende Eventbehandlung ist zu programmieren.

4.3.2.1.4.4.1. Event des Netscape einer Nicht-Standardbehandlung unterziehen

4.3.2.1.4.4.1.1. Event und Eventhandler dem Objekt window zuordnen (captureEvents(event_liste))

Diese Methode unterbindet die Ereignisregistrierung des Browsers. Damit MUSS das Ereignis von einem Eventhandler verarbeitet werden. Der Eventhandler kann mehrere Ereignisse verarbeiten.

Achtung: In Eventhandlern, die Mausereignisse oder Tastaturereignisse verwalten, sollte kein alert() etc. programmiert werden, da alert() selbst solche Events erzeugt und damit die eigentlichen Events aufhebt. Alternative: Meldungen in der Statuszeile des Fensters anzeigen.

Start der Registrierung des Events durch:

```
window.captureEvents(Event.event_schluesselwort_liste);
```

event_schluesselwort_liste: Folge von Eventbezeichnern mit Trennung durch | also logisches Oder

Bsp. für Eventbezeichner

onXXX	Event.XXX
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS
onload	Event.LOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER
onkeydown	Event.KEYDOWN
onkeypress	Event.KEYPRESS
onkeyup	Event.KEYUP
onresize	Event.RESIZE

Beispiel:

```
function MouseDownHandler(Ereignis)
```


```
{..}

window.captureEvents(Event.MOUSEDOWN);

// Standardeventhandler überschreiben
window.onmouseover=MouseDownHandler;          // Achtung: nicht () kodieren !!
```

Empfehlung: Vor dem Überschreiben des Standardeventhandler diesen retten

```
var RetteHandler = window.onmouseover;
window.onmouseover=MouseDownHandler;
```

Wenn das Ereignis nur einmal bearbeitet und danach wieder der Standardbehandlung zugeordnet werden soll, so muss der Eventhandler vor dem return die Methode .releaseEvents() zum Ereignis aufrufen. Falls der Standardeventhandler gerettet wurde, dann diesen auch rückspeichern.

Wenn das Ereignis **nicht** durch nachgelagerte Eventhandler bzw. Aktionen durch deren automatischen Aufruf bearbeitet werden soll, so muss die Funktion mit return false; enden (sonst return true;) z.B. siehe Event onreset etc. für Formular.

4.3.2.1.4.4.1.2. Event entlang der Eventhierarchie weiterreichen

Es besteht die Möglichkeit, dass ein privater Eventhandler für die Weitergabe des Events die Standardhierarchie benutzt oder einen ausgewählten Eventhandler aufruft.

4.3.2.1.4.4.1.2.1. Event entlang der Nicht-Standard- Eventhierarchie weiterreichen (handleEvent(event_objekt))

Diese Methode ruft den aktuell dem Ereignis laut Parameter event_objekt zugeordneten Eventhandler auf. Diese Methode wird also innerhalb des Programmcodes eines Eventhandlers verwendet, der das Ereignis von einem anderen Eventhandler als Argument bekommen hat.

4.3.2.1.4.4.1.2.2. Event entlang der Standard-Eventhierarchie weiterreichen (routeEvent(ereignis_objekt))

Diese Methode aktiviert das Weiterreichen des Events zum in der Eventhierarchie untergeordneten Eventhandler.

4.3.2.1.4.4.2. Event des Netscape von Nicht-Standardbehandlung wieder der Standardbehandlung unterziehen

4.3.2.1.4.4.2.1. Standard-Eventhandler dem Objekt window zuordnen (releaseEvents(event_liste))

Diese Methode aktiviert die Ereignisregistrierung durch den Browser, ist also der Gegensatz zu. captureEvents(). Es dürfen nur Ereignisse releast werden, die auch gecaptured wurden.

Ende der Registirerung eines Events:

```
window.releaseEvents(Event.event_schluesselwort_liste);
```

event_schluesselwort_liste: Folge von Eventbezeichnern mit Trennung durch | also logisches Oder

Bsp. für Eventbezeichner

onXXX	Event.XXX
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS
onload	Event.LOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER
onkeydown	Event.KEYDOWN
onkeypress	Event.KEYPRESS
onkeyup	Event.KEYUP
onresize	Event.RESIZE

Beispiel:

```
function EinmaligerMouseDownHandler()
{
    .....
    window.releaseEvents(Event.MOUSEDOWN);
}

window.captureEvents(Event.MOUSEDOWN);

window.onmouseover=EinmaligerMouseDownHandler;          // Achtung: nicht () kodieren !!
```

Wenn das Ereignis **nicht** durch nachgelagerte Eventhandler bzw. Aktionen durch deren automatischen Aufruf bearbeitet werden soll, so muss die Funktion mit return false; enden (sonst return true;) z.B. siehe Event onreset etc. für Formular.

4.3.2.1.4.4.2.2. Event entlang der Standard-Eventhierarchie weiterreichen (routeEvent(ereignis_objekt))

Diese Methode aktiviert das Weiterreichen des Events zum in der Eventhierarchie untergeordneten Eventhandler.

4.3.2.1.4.4.3. Eventbehandlung durch eine Fremdseite mit signiertem Script

Eventbehandlung des Netscape für eine Seite mit Frame, in dem ein fremdes HTML-Dokument angezeigt wird:

Es ist möglich, dass die fremde Seite die Ereigniskontrolle derjenigen Seite übernimmt, die das Frame enthält, in dem die Fremdseite angezeigt wird. Dieser Trojanereffekt ist nur dann realisierbar, wenn die Fremdseite ein signiertes Script enthält: Dieses Script muss sich beim Eigentümer der Seite, die den Frame besitzt, das Recht (Privileg) zur Übernahme der Ereigniskontrolle beschaffen. Dazu wird der User, in dessen Browser die Seite mit dem Frame läuft, befragt. Der Netscape hat einen Privileg-Manger integriert.

4.3.2.1.4.4.3.1. Beschaffung der Rechte "UniversalBrowserWrite" bzw. "UniversalBrowserWrite" per Privilegmanger

Die fremde Seite kann z.B. folgenden Code besitzen:

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite");
window.enableExternalCapture();
window.captureEvents(Event.CLICK | Event.MOUSEDOWN);
```

Bei Ausführung des Codes wird der Manager aktiv, um die Rechtegenehmigung durch den User zu beschaffen, da dieser Code (Script) vom User
signiert sein muss.

Der Netscape-Privilegmanager sichert ab, dass der Browseruser gefragt wird, ob er eine Privilegänderung gegenüber den Standard-Privilegien dulden will.

Für ein signiertes Script muss das Privileg "UniversalBrowserWrite" bzw. "UniversalBrowserWrite" vom Privilegmanger angefordert sein.

4.3.2.1.4.4.3.2. Eventbehandlung durch Fremde Seite aktivieren (enableExternalCapture())

Diese Methode aktiviert die Kontrolle einer Fremdseite, die in einem Frame dsrgestellt wird, auf die Seite, die den Frame inne hat. Die Ereignissüberwachung in einem Fenster wird somit aktiv.

Diese Methode benötigt ein signiertes Script.

Das Event muss mit captureEvents() dem Window-Objekt zugeordnet werden.

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

4.3.2.1.4.4.3.3. Eventbehandlung durch Fremde Seite deaktivieren (disableExternalCapture())

Diese Methode deaktiviert die Kontrolle einer Fremdseite, die in einem Frame dargestellt wird, auf die Seite, die den Frame inne hat. Die Ereignissüberwachung in einem Fenster wird somit aktiv.

Diese Methode benötigt ein signiertes Script.

Das Event muss mit captureEvents() dem Window-Objekt zugeordnet werden.

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

4.3.2.1.4.4.4. Beispiel

Es ist zu beachten, dass beim Netscape das Ereignis nicht als Objekt direkt angesprochen werden darf, sondern immer nur über den Parameter zum Eventhandler. Der Bezeichner des Paramateters ist frei wählbar. NS weiss, dass dieser Parameter das Eventobjekt referenziert. Es ist also auch der Bezeichner "event" wählbar.

Beim Internet Explorer muss das Ereignis im Eventhandler immer direkt über das Objekt event (Unterobjekt des Objektes window) behandelt werden. Daher ist für den Eventhandler kein Parameter nötig.

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript">
<!--
        function MouseDownFuerInputButton(Ereignis)
        { ... }

        function OnMouseDownEventWeiterreichen(Ereignis)
        {window.routeEvent(Ereignis);} //  In der Hierarchie weiterreichen

        window.captureEvents(Event.MOUSEDOWN);

        window.onmousedown= OnMouseDownEventWeiterreichen;
                        //  Achtung: ohne () kodieren!!
// -->
</SCRIPT>
</HEAD>
<BODY>
        <FORM>
                <INPUT  TYPE=button
                        VALUE="Weiterreichen"
                        onmousedown= "MouseDownFuerInputButton();"
```


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

Ablauf: Head-Teil: Der Script-Teil im Head wird zuerst abgearbeitet.
Body-Teil: Es wird auf das Input-Button gedrückt.
Wegen captureEvents() wird das Mausereignis ZUERST von
OnMouseDownEventWeiterreichen
bearbeitet. Damit wird auf das Weiterleiten aktiviert. Das untergeordnete Element ist der Input-Button. Damit wird MouseDownFuerInputButton aktiviert.

Hinweis: Dieses Beispiel hinkt, da die Standardbehandlung letztendlich das gleiche bewirkt aber auf anderen Wegen.

4.3.2.1.4.5. Beispiele zur Eventbehandlung des Netscape

4.3.2.1.4.5.1. Formular

Es wird das Click-Event abgefangen und mit einer Funktion bearbeitet.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE ="Javascript">
<!--
        //  Browser feststellen
        var ns = document.layers ? true : false;
        var ie = document.all ? true : false;

        if (ns)
        {
                //  nur für NS: Eventart CLICK wird abgefangen
                window.captureEvents(Event.ONCLICK);
        }

        // soll das Fenster selbst den Click-Eventhandler 1 bekommen, so hier kodieren
        //      window.onclick=click_auswertung_1;
        //      WICHTIG ist, dass das Fenster auch das Event WEITERLEITET !! nach unten

        function click_auswertung_1(Ereignis)
        {
                //  folgende alternative Varianten sind möglich

                // nur für NS:
                //              in der Objekt-Hierarchie weiterleiten
                if (ns)
                {routeEvent(Ereignis);}

                // für NS und IE:
                //              nichts tun
                return true;

                //              im aktuellen Fenster und dessen Dokument das Formular anwählen
                //              und dem Eventhandler des 2. Buttons das Ereignis übergeben,
                //              wobei der Eventhandler des 2. Buttons dieses Ereignis verarbeiten muss
                self.document.logischer_form_name.logischer_button_name_2. handleEvent(Ereignis);

                //              die Auswertung hier kodieren --> bitte kein alert(); !!!
                if (ie)
                {
                        Ereignis=event.button;
                        //  hier das Maus-Ereignis allgmein erfasst
                        //                      0       keine Maustaste gedrückt
                        //                      1       linke Maustaste gedrückt
                        //                      2       rechte Maustaste gedrückt
                        //                      4       mittlere Maustaste gedrückt
                        //                      Kombination aus 1 bis 4 für Maustastenkombination
                        //                              z.B.  3 = linke UND rechte Maustaste gedrückt (1 + 2 = 3)
                        //                                    7 = alle Maustasten gedrückt (1 +2 +3 + 4 = 7)
                        //                      nur Internet Explorer ab 4.x
                        .....
                }
                else
                {
```


```
                        if (ns)
                        {.....}
                }
        }

        function click_auswertung_2(Ereignis)
        { //  analog zu click_auswertung_1}
// -->
</SCRIPT>
</HEAD>
<BODY ....>
        <FORM NAME="logischer_form_name">
                <INPUT          TYPE=button
                                NAME="logischer_button_name_1"
                                onclick="click_auswertung_1();"
                >
                <INPUT          TYPE=button
                                NAME="logischer_button_name_2"
                                onclick="click_auswertung_2();"
                >
        </FORM>
</BODY>
</HTML>
```

4.3.2.1.4.5.2. Tastatur-Eventbehandlung beim Netscape

Der Event-Handler muss als Parameter das gewünschte Event übergeben bekommen

4.3.2.1.4.5.2.1. Tastatur-Eventarten

onkeydown — Ereignis ausgelöst für alle Tastenarten der Tastatur
liefert einmalig den ASCII-Code der Taste nach Eigenschaft .which, wenn Taste das erste Mal gedrückt wird (also auch einmalig bei Dauertastendruck)
Hinweis: nach String umwandeln per var string_name=fromCharMode(..)
Besonderheit bei Kombination Steuertaste und andere Taste
z.B. Shift-Taste und B-Taste zugleich für Großbuchstabe B
Ereignis wird pro Tastendruck aufgerufen, also
1. Mal für Shift-Taste
2. Mal für B-Taste
keydown-Ereignis ruft automatisch das keypress-Ereignis auf, wenn keydown-Handler return false; liefert
.which — ASCII-Code der gedrückten Taste
wenn 0 so keine Taste gedrückt
wobei bei Buchstaben unterschieden wird
Umwandlung in String per String.fromChatCode(TastenKodeASCII)
Zusatzanalyse möglich per Event-Eigenschaft .modifiers:
Bitmaske für Zustand der Steuertasten Shift, Alt und Strg
dient zur Ermittlung der Steuertaste per vordefinierter Maske
durch bitweise UND-Verknüpfung mit der Bitmaske:
ergibt die Verknüpfung 1, also true, so ist Steuertaste gedrückt worden
vordefinierte Maske für
Alt-Taste "Event.ALT_MASK"
Strg-Taste "Event.CTRL_MASK"
Shift-Taste "Event.SHIFT_MASK"
Unterbindung des Aufrufes des keypress-Handlers:
keydown-Handler muss return false; bzw. event.returnValue=false; liefern

onkeypress — wird unmittelbar nach dem keydown-Ereignis ausgelöst allerdings nur für Tastenarten
! @ # $ % ^ & * ( ) _ - + = < [ ] { } , . / ? \ | ' " ` ~
0 bis 9
a bis z mit Unterscheidung Gross oder Klein
Enter
Leertaste
und dient dem Erkennen von Tastaturkombinationen, die nicht vom Ereignis keydown abgefangen werden
Aufruf des keypress-Handlers ist **nur** möglich, wenn der keydown-Handler return true; liefert

onkeyup — Ereignis ausgelöst, wenn gedrückte Tastatur-Taste jeder Art losgelassen wird
wird automatisch nach Ereignis keypress ausgelöst
besitzt vordefiniertes Schlüsselwort KEYUP für captureEvents
also document.captureEvent(Event.KEYUP);

4.3.2.1.4.5.2.2. Tastatur-Eventeigenschaften

.modifiers — Bitmaske für Zustand der Steuertasten Shift, Alt und Strg
dient zur Ermittlung der Steuertaste per vordefinierter Maskedurch bitweise UND-Verknüpfung mit der Bitmaske: ergibt die Verknüpfung 1, also true, so wurde die Steuertaste gedrückt

vordefinierte Maske für
Alt-Taste "Event.ALT_MASK"
Strg-Taste "Event.CTRL_MASK"
Shift-Taste "Event.SHIFT_MASK"

Beispiel:

return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0); // nicht logisches UND && sondern bitweises UND &

.which enthält ASCII-Code der Tastatur-Taste
Umwandlung in String per String.fromChatCode(TastenKodeASCII)

.type enthält die Event-Art z.B. abort

.target Zeiger auf dasjenige HTML-Element, für das das Ereignis ausgelöst wurde
entweder im HTML-Tag definieren per onXXX = " ....."
oder Bezug auf den logischen Namen des HTML-Elementes
if (logischer_name = Ereignis.target) ....

4.3.2.1.4.5.2.3. Beispiel zur Tastatur-Eventbehandlung

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--

var ns = document.layers ? true : false;
var ie = document.all ? true : false;

// Funktionen für Tasten aller Art holen
function NS_IE_TastenKodeHolen_ASCII(Ereignis)
{
    if (ie)
    {return event.keyCode;}
    else
    {
        if (ns)
        {return Ereignis.which;}
    }
}

function NS_IE_TastenKodeHolen_String(TastenKodeASCII)
{
    if (TastenKodeASCII == 0)
    {return "";}
    else
    {return String.fromChatCode(TastenKodeASCII);}
}

function IE_TasteDauerdruck_StatusHolen()
{return event.repeat;} // true, so Taste im Dauerdruck, sonst false

// Funktionen für Shift-Taste
//      liefern true für Shift-Taste gedrücjt; sondt false

function NS_IE_ShiftTaste_StatusHolen(Ereignis)
{
    if (ie)
    {return event.shiftKey;}
    else
    {
        if (ns)
        {return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0); }
    }
}

function IE_ShiftLeftTaste_StatusHolen()
{return event.shiftLeft;}

function IE_ShiftRightTaste_StatusHolen()
{return (event.shiftKey AND (!event.shiftLeft));}

// ********** Funktionen für Alt-Taste analog zu Shift aber für NS mit Event.ALT_MASK

// ********** Funktionen für Strg-Taste analog zu Shift aber für NS mit Event.CTRL_MASK
// -->
</SCRIPT>
```


4.3.2.1.4.5.3. Mouse-Eventbehandlung beim Netscape

Der Event-Handler muss als Parameter das gewünschte Event übergeben bekommen.

4.3.2.1.4.5.3.1. Mouse-Eventarten

onclick	Ereignis ausgelöst, wenn Maus das Objekt angeklickt hat: Taste drücken und loslassen im Handler return false; kodieren für Links, Formularelemente, wenn eine Aktion nicht erwünscht ist
ondblclick	Ereignis ausgelöst, wenn durch linke bzw. rechte Maustaste ein Doppelklick auf Objekt erfolgt 1 Klick = Maustaste drücken und loslassen
onlosecapture	Ereignis ausgelöst, wenn releaseCapture() für mousemove, mouseover und mouseout ausgeführt wurde der User mit der Maus das Browserfenster verlässt
onmousedown	Ereignis ausgelöst mit Drücken der linken oder rechten Maustaste (mittlere nicht !) auch in Verbindung mit Umschalt (Shift) und Alt siehe Event-Eigenschaft .which return false; unterbindet **immer** die die Ausführung des Eventhandlers return false bei rechter Maustase auf Object document wird das Kontextmenü gesperrt
onmousemove	Ereignis bei **jeder** Mausbewegung ausgelöst Auswertung für den Bereich eines Objektes durch dessen Eventhandler
onmouseout	Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt
onmouseover	Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt
onmouseup	Ereignis ausgelöst mit Loslassen der linken Maustaste (rechte und mittlere nicht !) siehe Event-Eigenschaft .which return false; unterbindet **immer** die die Ausführung des Eventhandlers

4.3.2.1.4.5.3.2. Mouse-Event-Eigenschaften

.height	Höhe in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde
.layerX	Ereignis ist nicht resize: horizontale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis ist resize: neue Breite in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Breite verändert wurde
.layerY	Ereignis ist nicht resize: vertikale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis ist resize: neue Höhe in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Höhe verändert wurde
.pageX	horizontale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
.pageY	vertikale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
.screenX	horizontale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes
.screenY	vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes
.target	Zeiger auf dasjenige HTML-Element, für das das Ereignis ausgelöst wurde entweder im HTML-Tag definieren per onXXX = " ....." oder Bezug auf den logischen Namen des HTML-Elementes if (logischer_name = event.scrElement) ....
.type	enthält die Event-Art z.B. abort
.which	enthält bei gedrückter Maustaste: 1 für linke Taste 2 für mittlere Taste 3 für rechte Taste
.width	Breite in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde
.x	identisch mit layerX
.y	identisch mit layerY

4.3.2.1.4.5.4. Lade-Ereignisse beim Netscape

4.3.2.1.4.5.4.1. Lade-Ereignis für Bild

onabort — Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop

onload — Ereignis ausgelöst, wenn Bild vollständig geladen wurde
aber bei animiertem Bild (Gif-Datei):
Ereignis load bei jedem Bildwechsel der Animation ausgelöst
es kann also für jedes Bild das Ereignis behandelt werden

4.3.2.1.4.5.4.2. 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 (nicht mehr ab NS 6.x)

onunload — Ereignis ausgelöst, wenn das HTML-Dokument verlassen wird durch
Schliessen Browserfenster
Wechsel zu anderer Seite auch per Browser-Button
window.document.open()

4.3.2.1.5. window.history Objekt des Netscape

Dieses Objekt ist eigentlich eine Collection als Sammlung der besuchten Webseiten (Verlauf).

Erzeugung:
keine, da vom Browser erzeugt

Zugriff:

window.history**[**index**]**.eigenschaft oder history**[**index**]**.eigenschaft
window.history**[**index**]**.methode() oder history**[**index**]**.methode()

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

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

Eigenschaften (Auswahl):

.current — Zeichenkette mit Url der aktuellen Seite
mit signiertem Script

.length — Anzahl der History-Einträge
nur lesen

.next — Zeichenkette mit Url der nächsten Seite (falls vorhanden)
mit signiertem Script

.previous — Zeichenkette mit Url der vorhergehenden Seite (falls vorhanden)
mit signiertem Script

Methoden (Auswahl):

.back() — vorhergehende Seite laut .previous laden

.forward() — nächste Seite laut next laden

.go(position) — Position der zu ladenen Seite innnerhalb der History
0 entspricht erste geladene Seite
history.length-1 entspricht aktuelle Seite
Bsp.: onClick="window.history.go(0)

4.3.2.1.6. window.location Objekt des Netscape

Dieses Objekt beschreibt die Lage des HTML-Dokumentes auf dem Server sowie dessen Eigenschaften zum HTML-Dokument.

Erzeugung:
keine, da vom Browser erzeugt

Zugriff:

window.eigenschaft oder location.eigenschaft
window.location.methode() oder location.methode()

logischer_window_name.location.eigenschaft
logischer_window_name.location.methode()

Eigenschaften (Auswahl):

.hash — entspricht **#hash**text
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 hier den Ankernamen ohne vergesetztem # ablegen
.pathname	entspricht aus url /dateiname bzw. /dateiname**#hash**text bzw. /dateiname**?search**text lesen und schreiben
.port	entspricht port lesen und schreiben
.protocol	entspricht Protokoll mit Doppelpunkt z.B. "http:" lesen und schreiben
.search	entspricht **?search**text lesen und schreiben

Methoden (Auswahl):

.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

4.3.2.1.7. window.navigator Objekt des Netscape

Container der Informationen über den Browser, Betriebsystem, 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 browsersepzifische Funktionen aufzufrufen. Deren Rückkehrcode liefert den Beweis, ob der gewünschte Browser auch benutzt wird vom User (siehe DOM). Ein typisches Tarnverhalten zeigt der Opera-Browser: Er gibt sich als Internet Explorer aus.

Erzeugung:

keine, da vom Browser erzeugt

Zugriff:

window.navigator.eigenschaft oder navigator.eigenschaft

window.navigator.methode() oder navigator.methode()

logischer_window_name.navigator.eigenschaft
logischer_window_name.navigator.methode()

Eigenschaften (Auswahl):

alle Eigenschaften sind nur lesbar

.appCodeName	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

.productSub	nur NS 6.x
.securityPolicy	Sicherheitseinstellungen vom Netscape nur NS 6.x
.userAgent	Browser-Codename UND -Version Bsp: "Mozilla / 3.0B2 (Win16;I)"
.vendor	nur NS 6.x
.vendorSub	nur NS 6.x

Methoden (Auswahl):

.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

4.3.2.1.7.1. window.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:

zeiger_auf_navigator_objekt.plugins**[**index**]**.eigenschaft
zeiger_auf_navigator_objekt.plugins**[**index**]**.methode()

index: Zeichenkette z.B. "Shockwave"
Integer Index ab 0
muss in **[ ]** kodiert sein

zeiger_auf_navigator_objekt	navigator window.navigator logischer_window_name..navigator

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)	

4.3.2.1.7.2. window.navigator.mimeTypes Collection des Netscape

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 window.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:
zeiger_auf_navigator_objekt.mimeTypes**[**mime_typ**]**.eigenschaft

mime_typ:	Integer, ab 0 String, der als Feldindex dient z.B. "text/html" muss in **[ ]** kodiert sein
zeiger_auf_navigator_objekt	navigator window.navigator logischer_window_name.navigator

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"].enabledPlugin != null)
        {
                document.write('<OBJECT DATA="yzeplin.wrl" WIDTH="400" HEIGHT="300"></OBJECT>');
        }
}
```

Beispiel 4: Suffix prüfen

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

Eigenschaften:

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

.type Mimetyp z.B. "text/html"

Methoden:
keine

4.3.2.1.8. window.screen Objekt des Netscape

Dieses Objekt beschreibt den Bildschirm
Die linke obere Browserfensterecke ist der Usprung des Grafiksystems, also (0,0) mit (x,y)
x für horizontal
y für vertikal

Erzeugung:
keine, da vom Browser erzeugt

Zugriff:
window.screen.eigenschaft oder screen.eigenschaft

logischer_window_name.screen.eigenschaft

Eigenschaften:

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

Methoden:
keine

4.3.2.2. window Objekt des Internet Explorer

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

Das window Objekt hat diverse Zeiger auf andere Objekte:

z.B. beim IE und NS:

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

z.B. beim IE:

.screen	Zeiger auf das Objekt screen

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

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

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

Diese fortlaufenden Änderungen muss der Programmierer in Erfahrung bringen.
Der Programmierer kann sich definitiv nicht auf Verfügbarkeit von Active-X-Controls verlassen und muss damit rechnen, dass seine Webseiten schlagartig nicht mehr komplett laufen weil u.a. Programmcode noch nicht angepasst ist. Ebenfalls muss der Programmierer Varianten von Windows und Patchzustände beachten, die prinzipiell Kostenprobleme verursachen können.
Mit anderen Worten: Wer Microsoft-Komponenten nutzt, muss wissen, was ihm blüht ... siehe nachfolgende Beispiel für Risiken.

Prinzipielle Lizenzprobleme für den Programmierer

Microsoft verlangt Linzensierung von Windows. Bezüglich Windows-Versionen gibt es die Updatestufen z.B. per Servicepacks
Ein Windows mit Servicepack fällt unter die Lizenz des geupdateten Windows.
Ein Windows mit Vorversion zum Servicepack bedarf einer anderen Lizenz.
Will man z.B. den Internet Explorer 7 und 6 parallel testen, benötigt man 2 Windowslizenzen, da beide Versionen nicht parallel installierbar. Dazu kommt, dass es den IE 6 in 2 Versionen gibt: Win SP1 und SP2 (IE 7 nur ab Win SP2).
Für 3 Browserversionen benötigt man 3 Windowslizenzen, will man parallel testen.
Ein Blick auf Browser-Konkurrenzprodukte klärt die Sachlage unschlagbar: Opera ist z.B. parallel installierbar.

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

Abänderungen wegen Sicherheitspatches der jeweiligen Windows-Versionen

Abschaltungen von Active-X-Controls erfolgen auch im Rahmen der Sicherheitspatches zu Windows-Versionen.
Es ist auch möglich, dass wegen Sicherheitslücken abgeschaltet wird und somit Komponenten einer Webseite je nach Windowsversion nicht mehr laufen.

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

Der Absturz ist "read" -Fehler von immer ein und derselben Speicherstelle.
User, die dieses Patch installiert haben, können ab sofort keine IE-Seiten mit codiertem Script mehr ansehen.
Microsoft stellt Abhilfe nach geraumer Zeit zur Verfügung, jedoch spezifisch nach Windows XP-Version:
Patch Q918899 für
Windows XP SP1Download für jedermann bereitgestellt
SP2 nur auf kostenpflichte telefonische Anfrage des Users per Downloadlink bereitgestellt, da Microsoft explizit die User registriert haben will, bei denen das Patchproblem auftritt (User muss sich Telefonnummer besorgen)
Solange also das Patch zum fehlerhatem Patch vom User nicht installiert wird, z.B. weil der User keine Ahnung hat, dass und wo er sich die Telefonnummer von Microsoft besorgen muss bzw. zu besorgen hat, wird der User IE-Seiten mit komprimierten Code dauerhaft nicht nutzen können. (Microsoft-Support ist z.T. nur in Englisch).

Abänderungen wegen Browser-Inkompatibilität

Popupblocker-Fehler

Die Microsoft Browser-Version IE 7 ist nicht abwärtskompatibel bezüglich Popup per window.createPopup()
Popup per window-Objekt ist ein Markenzeichen des IE, das im IE 7 nicht mehr fehlerfrei nutzbar ist.
Der Fehler liegt in der Popup-Blockerverwaltung des IE und wurde mit dem IE 7 implementiert.
Der Fehler tritt nicht auf, wenn ein Fenster per window.open() erzeugt wurde.
Bedingung:
Scripfehleranzeige ist erlaubt im IE 7
Popupblocker ist im IE abgeschaltet
ein aktives Fenster (Register) mit Dokument, dass fortlaufend (rekursiv) genau 1 window.popup per .show()erzeugt.
ein weiteres Fenster (Register) z.B. leere Seite (about:blank)
beide (Register) liegen in einer gemeinsamen IE-Instanz
Ablauf: Wird Focus auf Register der leeren Seite gehalten und wird parallel das Popup per .show() erzeugt, bricht der Browser das Dokument mit .show() ab (Scriptfehler).
Der Popupblocker für die leere Seite verursacht den Programmfehler im Dokument mit .show(). Es wird folgende Meldung angezeigt (in der Informationsleiste):
'Ein Popup wurde geblockt. Klicken Sie hier, um das Popup bzw. weitere Optionen anzuzeigen.'
Die Bedeutung der Meldung laut Microsoft-Hilfe im IE 7:
Der Popupblocker hat ein Popupfenster geblockt. Sie können den Popupblocker deaktivieren oder Popups temporär zulassen, indem Sie auf die Informationsleiste klicken.
Die Realität zur obigen Meldung ist völlig anders:
Linke oder rechte Maus auf die Meldung liefert z.B. Einstellungen darunter
Popupblocker einschalten
weitere Informationen
jedoch keine Möglichkeit wie laut Bedeutung
Damit gilt: Der abgeschaltete Popupblocker ist in Wirklichkeit aktiv.
Pikant: Ein Popup erscheint normalerweise auch über fremde Fenster, die nicht das Popup erzeugt haben (z.B. Fenster

einer Windowsanwendung z.B. einer anderen IE-Instanz)
Der Popupblocker des IE bemeckert aber NUR Webseite, die das Popup erzeugt.
Durch das Abwürgen von Popup wird das Popup natürlich auf und für anderen Seiten nicht relevant; im Falle einer anderen IE-Instanz also auch für diese nicht relevant, obwohl diese Instanz per Popupblocker verwaltet wird.
Der Popupblocker beschneidet die Popup-Reichweite an der Wurzel, ist aber nicht objektorientiert zu den anderen Webseiten (die nicht das Popup erzeugt haben).
Der Popupblocker ist nicht als Filter aufgesetzt sondern reingestrickt worden.

Der Popupblockerfehler verändert die Eventverwaltung:
Es werden u.a. ignoriert
onfocus
onblur
onfocusin
onfocusout
und viele andere, so dass trotz Events z.B. des Body der Popupblockerfehler entsteht.

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

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

focus-Methode beim IE 7

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

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

animierte Gif (mit Timer)

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

Abänderungen wegen Rechtstreitigkeiten von Microsoft mit Fremdanbietern

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

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

Es wurde das Eventmodell gleichzeitig geändert:

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

style.visibility='hidden' wird ignoriert

Die Sprechblase erscheint auch dann, wenn das Control mit style.visibility='hidden' belegt ist, also sich unsichtbar rendert:
Der Sprechblasenrahmen hat genau die Dimension wie die des unsichtbaren Controls. Der Sprechblasenrahmen erscheint also Zusammenhangslos, und der User weiß nicht, warum er klicken soll, wenn er nichts sieht. Vor allem weiß er nicht, WAS er klickt ... ideale Basis für Schadsoftware per Script.

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

Abänderung wegen Abschaltungen

DirectX ist wegen Abschaltung von Active-X--Controls nicht mehr abwärtskompatibel:
Z.B. wurde bei Win XP SP2 Direct Animation aus DirectX schlagartig durch Abschaltung von Bibliotheken dezimiert, die es bei Win XP SP1 aber noch gibt.

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

ActiveX-Controls und Unterstützung/Verbot 20041215

erlaubt sind noch

Tabular Data-Steuerelement{333C7BC4-460F-11D0-BC04-0080C7055A83} Das TDC (Tablular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformatvorliegen, beispielsweise durch Darstellung in einer Tabelle oder Sortierung. Weitere Informationen:• http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp(http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp)

Microsoft Agent Control - Version 2.0 {D45FD31B-5C6E-11D1-9EC1-00C04FD7081F} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kannauch in Webseiten verwendet werden. Weitere Informationen:• http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm(http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm)

Microsoft MSChat-Steuerelement-Objekt 2.0 - 2.5 {D6526FE0-E651-11CF-99CB-00C04FD64497} Dieses Steuerelement wird von Webautoren verwendet, um text- und graphisch basierte Chatgemeinden für Echtzeitkonversationen im Web zu erstellen.

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

verboten sind

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

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

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

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

Internet Explorer

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

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

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

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

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

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

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

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

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

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

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

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

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

Aktive Inhalte im Internet Explorer

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

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

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

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

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

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

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

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

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

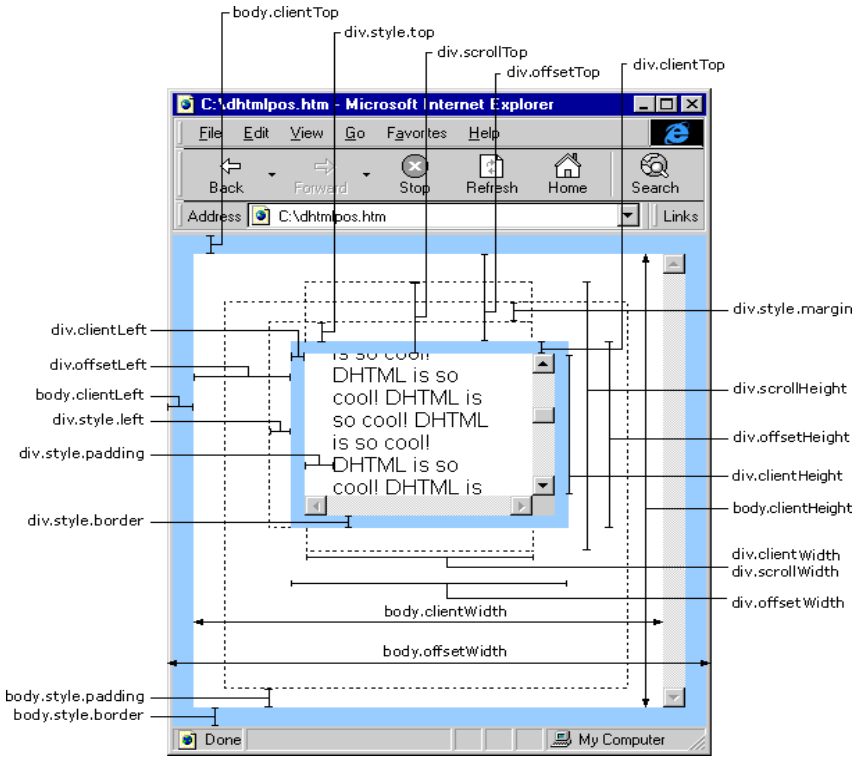
Folgende Metatags sind für den IE 6.x aktiver Inhalt:

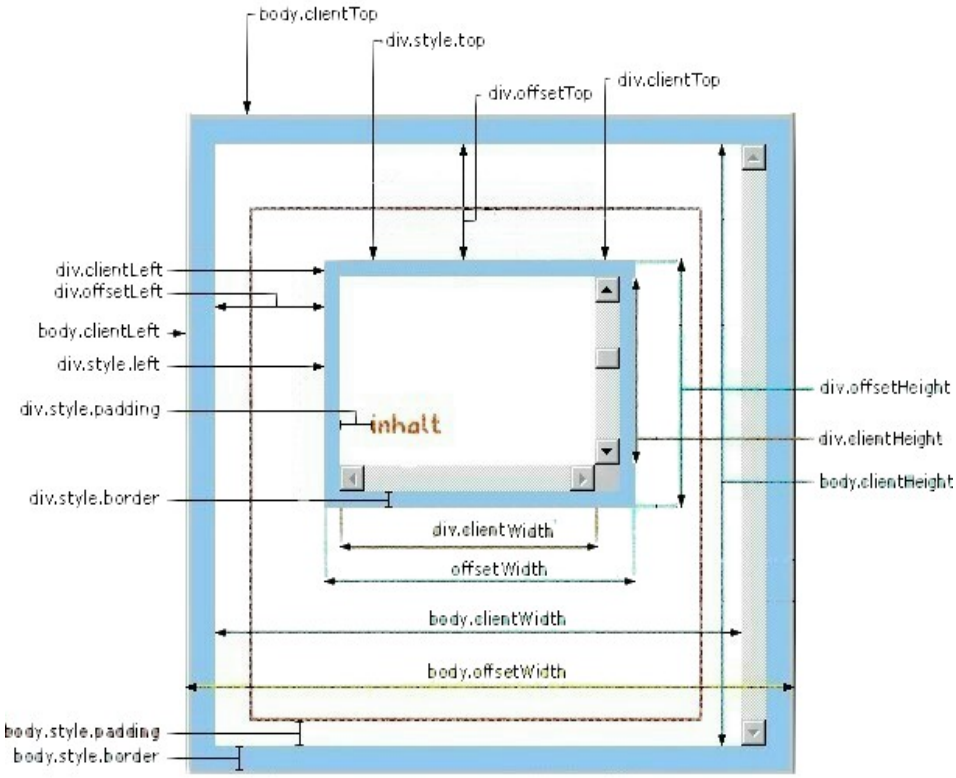
```
<META HTTP-EQUIV="imagetoolbar" CONTENT="no">
```
unterdrückt NICHT IE-Kontextmenü rechte Maus auf Bild

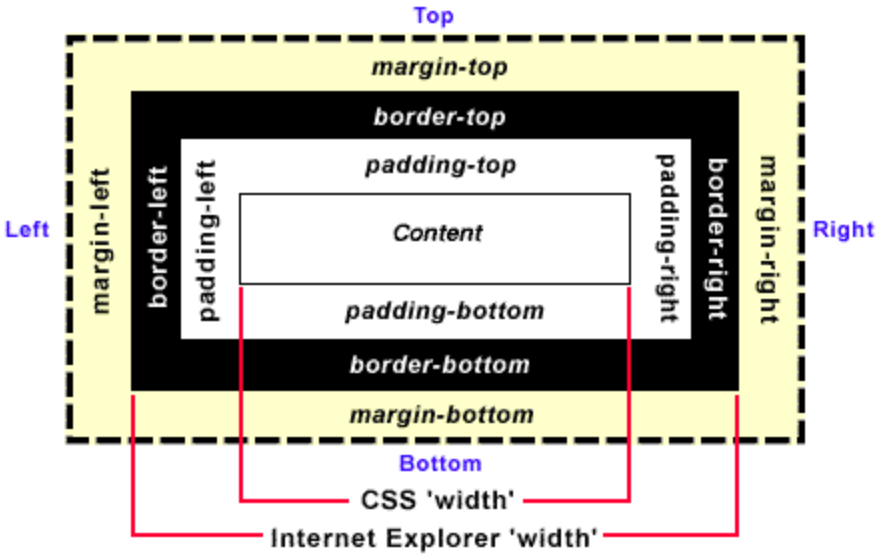
```
<META HTTP-EQUIV="site-enter" CONTENT="revealtrans(duration=0.3, transition=12) ">
<META HTTP-EQUIV="site-exit" CONTENT="revealtrans(duration=0.3, transition=12) ">
```

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

Eigenschaften:
Die Eigenschaften des IE als Grafiken:


body.clientTop
div.style.top
div.scrollTop
div.offsetTop
div.clientTop
C:\dhtmlpos.htm - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search
Address C:\dhtmlpos.htm
Links
DHTML is so cool! DHTML is so cool! DHTML is so cool! DHTML is so cool! DHTML is
Done
My Computer
div.clientLeft
div.offsetLeft
body.clientLeft
div.style.left
div.style.padding
div.style.border
div.style.margin
div.scrollHeight
div.offsetHeight
div.clientHeight
body.clientHeight
div.clientWidth
div.scrollWidth
div.offsetWidth
body.clientWidth
body.offsetWidth
body.style.padding
body.style.border


body.clientTop
div.style.top
div.offsetTop
div.clientTop
div.clientLeft
div.offsetLeft
body.clientLeft
div.style.left
div.style.padding
inhalt
div.style.border
div.offsetHeight
div.clientHeight
body.clientHeight
div.clientWidth
offsetWidth
body.clientWidth
body.offsetWidth
body.style.padding
body.style.border



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

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

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

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

Wert	false so ist Fenster offen true so ist Fenster geschlossen
logischer_window_name	Zeiger laut open()

nur lesen

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

Kette	String
logischer_window_name	Zeiger laut open()

lesen und schreiben

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

Beispiel:

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

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

```
                    FormularDaten.lastName  = FormularElemente.ID_Input2.value;

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

test.htm enthält:

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

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

.dialogHeight           Höhe des Dialogfensters, das mit Methoden
                                .showModalDialog()  bzw. .showModelessDialog()
                        erzeugt wurde
                        ab IE 5.x
                        Syntax:
                                logischer_window_name..dialogHeight [= Kette ]
                                [ var Kette  = ] logischer_window_name..dialogHeight

                                        Kette                   String als numerisches Literal
                                                                        mit kodierter Einheit
                                                                        "cm", "mm", "in", "pt", "pc" oder "px"

                                                                wenn < 100 Pixel  so 100 Pixel automatisch
                                                                        verwendet

                                        logischer_window_name   Zeiger laut open()
                        lesen und schreiben
        Beispiel:
                <HEAD>
                <SCRIPT>
                function ZeigeModalenDialog()
                {
```


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

.dialogLeft — X-Koordinate der linken oberen Ecke des Dialogfensters, das mit Methoden .showModalDialog() bzw. .showModelessDialog() erzeugt wurde

ab IE 5.x

Syntax:

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

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

lesen und schreiben

Beispiel:

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

.dialogTop — Y-Koordinate der linken oberen Ecke des Dialogfensters, das mit Methoden .showModalDialog() bzw. .showModelessDialog() erzeugt wurde

ab IE 5.x

Syntax:

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

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

lesen und schreiben

Beispiel:

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

.dialogWidth — Breite des Dialogfensters, das mit Methoden
.showModalDialog() bzw. .showModelessDialog()
erzeugt wurde
ab IE 5.x
Syntax:
logischer_window_name..dialogWidth [= Kette]
[var Kette =] logischer_window_name..dialogWidth

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

lesen und schreiben

Beispiel:

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

.document — Zeiger auf das Objekt document im Fenster

.event — Zeiger auf das Objekt event im Fenster

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

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

Beispiel:

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


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

.frames — Zeiger auf die Collection document.frames im HTML-Dokument, das im Fenster angezeigt wird

.history — Zeiger auf das Objekt history (Verlauf)

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

Syntax:

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

Wert	Integer, ab 1
logischer_window_name	Zeiger laut open()

nur lesen

.location — Zeiger auf das Objekt location

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

Syntax:

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

Kette	String
logischer_window_name	Zeiger laut open()

lesen und schreiben

Beispiel:

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

.navigator — Zeiger auf das Objekt navigator

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

Syntax:

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

Kette	String	
	"auto"	Default, Browser entscheidet
	"true"	offscreen buffering ein
	"false"	offscreen buffering aus, also direkt sichtbar auf Bildschirm zeichnen
logischer_window_name	Zeiger laut open()	

lesen und schreiben

.opener — Zeiger auf Elternfenster, das **open() enthält** und das Kind-Fenster öffnet, welches in dieser .opener-Eigenschaft den Zeiger auf das Elternfenster enthält
Bezug im geöffneten Fenster auf Instanzen des Aufrufers ist möglich
Bezug des Aufrufers auf geöffnetes Fenster ist möglich
Hinweis: Bei FRAME und IFRAME anstelle opener den Zeiger parent verwenden
Öffnet ein Frame / Iframe ein Fenster, das damit nicht im Framset läuft, so ist der Bezug vom Fenster aus auf das Frameset oder auf einen Frame im Frameset per window.opener.parent möglich, solange opener existiert.

Syntax:

```
logischer_window_name.opener [ =  Zeiger ]
[ var Zeiger = ] logischer_window_name.opener
```

logischer_window_name	Zeiger laut open()

lesen und schreiben

Beispiel:

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


```
}

....

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

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

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

.parent — Zeiger auf das **in der Fensterhierarchie übergeordnete** Fenster, das aber nicht das .open() zum Kindfenster enthalten muss
z.B. Zeiger auf das Fenster, das die FRAMESET-Deklaration enthält, oder das diesem Fenster übergeordnet ist
Test auf Existenz von parent per if (parent != self) ....
Syntax:
[var Zeiger =] logischer_window_name.parent

logischer_window_name	Zeiger laut open()

nur lesen

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

Wert	Ausdruck etc.
logischer_window_name	Zeiger laut open()

lesen und schreiben

.screen — Zeiger auf das Objekt screen

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

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

nur lesen

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

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

nur lesen

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

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

	Kette	String
	logischer_window_name	Zeiger laut open()

lesen und schreiben

Beispiel für Text buchstabenweise in Statuszeile anzeigen:

Der aktuelle Statuszeilentext wird als Teilkette von Position 0 bis zeichen_nr angezeigt, wobei zeichen_nr pro Anzeige um 1 erhöht wird.

```
<HTML>
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var statuszeile         = new Array();
        statuszeile[0]          = "Text0";
        statuszeile[1]          = "Text1";
        statuszeile[2]          = "Text2";

        var statuszeile_nr      = 0;
        var zeichen_nr          = 0;

        function textanzeigen ()
        {
                if (position < statuszeile[statuszeile_nr].length)  // Länge ab 1
                {
                        // nächste Teilkette des aktuellen Statuszeilentextes anzeigen
                        window.status = statuszeile[statuszeile_nr].substring(0, zeichen_nr);

                        // nächste Position
                        position++;
                        setTimeout("textanzeigen()", 100);
                }
                else
                {
                        // aktuelle Statuszeilentext komplett anzeigen
                        window.status = statuszeile[statuszeile_nr];

                        // nächste Statuszeile adressieren
                        statuszeile_nr ++;
                        if (statuszeile_nr >= statuszeile.length)
                        { statuszeile_nr = 0;}

                        // Position 0 einstellen
                        zeichen_nr = 0;

                        // Start der buchstabenweise Anzeige
                        setTimeout("textanzeigen()", 1000);
                }
        }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>
```


Beispiel für automatisch wechselnder Text in der Statuszeile:

```
<HTML>
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var statuszeile = new Array();
        statuszeile[0] = "Text0";
        statuszeile[1] = "Text1";
        statuszeile[2] = "Text2";

        var statuszeile_nr = 0;

        function textanzeigen()
        {
                window.status = statuszeile[statuszeile_nr];
                statuszeile_nr ++;
                if (statuszeile_nr >= statuszeile.length) {statuszeile_nr = 0;}
                setTimeout("textanzeigen()", 1000);
        }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="textanzeigen()">
</BODY>
</HTML>
```

Beispiel für dauerhaftes Scrollen eines Textes in der Statuszeile:

```
<HTML>
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var scrolltext_gesamt = "Dieser Text scrollt!";
        var scrolltext        = "";
        var zahler            = scrolltext_gesamt.length;


        function scrollen()
        {
                if (zahler == scrolltext_gesamt.length)
                {
                        zahler = 0;
                        scrolltext = scrolltext_gesamt;
                }
                else
                {
                        zahler++;  // 1  bis  scrolltext_gesamt.length
                        // Blank vorsetzen, also Kette um 1 Zeichen verlängern
                        scrolltext = " "+scrolltext;

                        // rausgerutschtes Zeichen abschneiden
                        scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
                                                        // Angaben ab 0
                }

                window.status = scrolltext;

                setTimeout("scrollen()", 100);
        }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="scrollen()">
</BODY>
</HTML>
```

Beispiel für einen Scrolltext in der Statusleiste, der angehalten wird, wenn Mauszeiger sich über einen Linkt bewegt:

```
<HTML>
```


```
<HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var scrolltext_gesamt = "Dieser Text scrollt!";
        var scrolltext = "";
        var zahler = scrolltext_gesamt.length;
        var id;

        function scrollen()
        {
                if (zahler >= scrolltext_gesamt.length)
                {
                        zahler = 0;
                        scrolltext = scrolltext_gesamt;
                }
                else
                {
                        zahler++;  // 1  bis  scrolltext_gesamt.length
                        // Blank vorsetzen, also Kette um 1 Zeichen verlängern
                        scrolltext = " "+scrolltext;

                        // rausgerutschtes Zeichen abschneiden
                        scrolltext = scrolltext.substring(0, scrolltext_gesamt.length -1);
                                                        // Angaben ab 0
                }

                window.status = scrolltext;
                id=setTimeout("scrollen()", 100);
        }

        function scrollen_stoppen()
        {clearTimeout(id);}
// -->
</SCRIPT>
</HEAD>

<BODY onLoad="scrollen()">
        Bewegen Sie den Mauspfeil &uuml;ber diesen
        <A      HREF="http://www.test.de"
                onMouseOver="scrollen_stoppen()"
                onMouseOut="scrollen()"
        >
        Link
        </A>
</BODY>
</HTML>
```

Beispiel für Anzeige eines Textes zu einem Hyperlink (HREF) für eine feste Zeitspanne in der Statuszeile:

```
<HTML>
</HEAD>
<SCRIPT  LANGUAGE="JavaScript">
<!--
        var id;
        var anzeige_aktiv = false;
        var anzeige_zeit = 3000;        // 3000 Millisekunden = 3 Sekunden

        function anzeige_starten(href_text)
        {
                if(anzeige_aktiv)
                {clearTimeout(id);}

                window.status = href_text;

                id = setTimeout("anzeige_stoppen()",anzeige_zeit);

                anzeige_aktiv = true;

                return true;            // Wichtig !!!
        }
```


```
		function anzeige_stoppen ()
		{
			anzeige_aktiv = false;

			window.status = "";
		}
//-->
</SCRIPT>
</HEAD>
<BODY>
		<A HREF= ... onMouseOver="javascript:return anzeige_starten('Hinweistext...')">
		...
		</A>
</BODY>
</HTML>
```

Beispiel zur Anzeige eines Formularfeld-Hinweises in der Statuszeile:

```
<HTML>
</HEAD>
		<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
		<!--
			function hinweis_anzeigen(hinweis_text)
			{window.status = hinweis_text;}

			function hinweis_loeschen()
			{window.status = "";}
		// -->
		</SCRIPT>
</HEAD>
<BODY>
		<FORM>
			<INPUT	TYPE=TEXT
				onFocus="hinweis_anzeigen('Hinweis');"
				onBlur="hinweis_loeschen();"
			>
		</FORM>
</BODY>
</HTML>
```

Beispiel für blinkende Anzeige mit Zeitspanne von Text in der Statuszeile:

```
<HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
		var zeitspanne  = 6000;		// 6 Sekunden blinken lassen, danach
						//		Statuszeile löschen
		var anzeigezeit = 500;		// 0,5 Sekunden warten nach Setzen
						//		bzw. Löschen der Statuszeile
		var id_blinken  = null;		// muss auf null initialsiert sein !!

		function blinken_timer_loeschen
		{
			if  (id_blinken != null)
			{
				clearTimeout(id_blinken)};  // blinken stoppen falls aktiv
				id_blinken = null;
			 }
		}

		function blinken_stop()
		{
			blinken_timer_loeschen;
			window.status="";
		}

		function blinken_start(status_text)
		{
			blinken_timer_loeschen;
			blinken(true, status_text);			// Blinken anstossen für
```


```
                                                                    // paralleles Arbeiten per Timer
                        setTimeout("blinken_stop()",zeitspanne);    // warten und danach blinken stoppen
                }

                function blinken(ein_aus, text)
                {
                        if (ein_aus)
                        {       window.status  = text;
                                ein_aus=false;          // im nächsten Aufruf die Statuszeile löschen
                        )
                        else
                        (       window.status = "";
                                ein_aus=true;           // im nächsten Aufruf die Stautuszeile setzen
                        )

                        id_blinken = setTimeout("blinken(" + ein_aus + ")", anzeigezeit)
                                                        // nächsten Aufruf nach Ablauf der anzeigezeit starten
                }
//-->
</SCRIPT>
</HEAD>
<BODY ... onLoad="blinken_start('Ich blinke !')">
</BODY>
</HTML>
```

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

```
[ var Zeiger = ] logischer_window_name.top

        logischer_window_name       Zeiger laut open()
```

nur lesen

Methoden

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

.alert() — Dialogbox erzeugen

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

Syntax:

```
logischer_window_name alert([Kette])

Kette                       String oder Stringausdruck

logischer_window_name       Zeiger laut open()
```

liefert nichts

.attachEvent() — Einschalten des Registrieren eines Events durch Eventhandler
Hinweis: Abschalten mit Methode .detachEvent()
Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider
NICHT verkettet sondern in **Zufallsfolge**, es sei denn
die Handler prüfen ihre Aufruffolge (muss programmiert werden)
Vor dem Neubelegen immer den Standard-Eventhandler retten und diesen
nach .detachEvent() wieder einbinden (siehe Beispiel).
Syntax:

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

        Kette                   String
                                Eventbezeichner

        Zeiger                  auf Eventhandler

        Wert                    true        Einschalten erfolgreich
        false                   Einschalten nicht möglich gewesen
```


```
                                        logischer_window_name        Zeiger laut open()
```

Beispiel 1:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
        function CursorNeu()
        {
                if (event.srcElement == element)
                {
                        normalColor         = style.color;
                        runtimeStyle.color  = "red";
                        runtimeStyle.cursor = "hand";
                }
        }

        function CursorNormal()
        {
                if (event.srcElement == element)
                {
                        runtimeStyle.color   = normalColor;
                        runtimeStyle.cursor = "";
                }
        }

        function EventEntfernen()
        {
                detachEvent ('onmouseover', CursorNeu);    // nicht () kodieren !!!
                window.onmouseover = Rette_Standard_Eventhandler_OnMouseOver;

                detachEvent ('onmouseout', CursorNormal);// nicht () kodieren !!
                window.onmouseout = Rette_Standard_Eventhandler_OnMouseOut;
        }

        var Rette_Standard_Eventhandler_OnMouseOver = window.onmouseover;
        attachEvent ('onmouseover', CursorNeu);

        var Rette_Standard_Eventhandler_OnMouseOut = window.onmouseout;
        attachEvent ('onmouseout', CursorNormal);
</SCRIPT>
```

Beispiel 2:

```
function ResizeHandler()                    // Homepage neu laden
{window.history.go(0);}

// Standardhandler retten
var RetteHandler_Resize = window.onresize;

// und neu belegen
window.onresize = ResizeHandler;            // Homepage neu laden
                                            // ohne () kodieren, damit nicht sofort ausgeführt wird

// und Start des Abfangens vom resize-Event
window.attachEvent('onresize', ResizeHandler);

.........

// irgendwann abschalten der Eventüberwachung
window.detachEvent('onresize', ResizeHandler);

// und Standardhandler wieder einbinden
window.onresize = RetteHandler_Resize;
```

.blur()

Element den Focus wegnehmen und Event onblur auslösen
Der Focus wird **nicht automatisch** auf irgend ein anderes Element gesetzt !
vor IE 5.0 TABINDEX-Attribut muss kodiert sein
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein
Syntax:

```
        logischer_window_name.blur()

                                        logischer_window_name        Zeiger laut open()
```

liefert nichts

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

Syntax:

logischer_window_name .clearInterval(timer_id)

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

liefert nichts

Beispiel 1:

```
var timerID = null;

function timer()
{
	// tue was
}

function starttimer()
{
	if (timerID == null)
	{timerID = setInterval("timer()", 1000);}
}

function stoptimer()
{
	if (timerID != null)
	{
		clearInterval(timerID);
		timerID = null;
	}
}
```

Beispiel 2 für Sekundenbalken:

```
<HTML>
<HEAD>
<STYLE>
	BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
	var timerID = null;
	var Zahler = 0;

	function Init()
	{
		// DIV-Eigenschaften festlegen

		// DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
		ID_Div1.style.setExpression("width","Zahler *10");

		// DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
		ID_Div2.setExpression("innerText","Zahler.toString()");
	}

	function Uhr()
	{
		// Sekunden kumulieren
		Zahler ++;

		// und Anzeige neu berechnen
		document.recalc();
	}

	function Starten()
	{
		if (timerID == null)
		{
			// Start-Button nicht aktivierbar machen
			ID_Button1.disabled = true;
```


```
                                        // Stop-Button aktivierbar machen
                                        ID_Button2.disabled = false;

                                        // Uhr neu starten
                                        timerID = setInterval("Uhr()", 1000);
                                }
                        }

                        function Stoppen()
                        {
                                if (timerID != null)
                                {
                                        clearInterval(timerID);
                                        timerID = null;
                                        ID_Button1.disabled = false;
                                        ID_Button2.disabled = true;
                                }
                        }

                        function ZurueckSetzen()
                        { Zahler = 0; }
                </SCRIPT>
                </HEAD>
                <BODY onload="Init()">
                        <DIV ID="ID_Div1"  STYLE="background-color:lightblue" ></DIV>
                        <DIV ID="ID_Div1"  STYLE="color:hotpink;font-weight:bold"></DIV>
                        <BR>
                        <BUTTON         ID="ID_Button1"
                                        onclick="Starten()"
                        >
                                Start
                        </BUTTON>
                        </BR>
                        <BUTTON         ID="ID_Button2"
                                        DISABLED="true"
                                        onclick="Stoppen()"
                        >
                                Stop
                        </BUTTON>
                        <BR>
                        <BUTTON         ID="ID_Button3"
                                        onclick="ZurueckSetzen()"
                        >
                                Reset
                        </BUTTON>
                        <BR>
                        <P      STYLE="width:200;color:white;background-color:gray">
                                Sekundenbalken
                        </P>
                </BODY>
                </HTML>
```

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

Syntax:

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

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

liefert nichts

.close() Fenster schliessen, das offen ist

Fenster muss nicht explizit mit Methode .open( )Methode erzeugt worden sein

Schliessen des letzten Browserfensters erzeugt immer Dialog-Box-Abfrage

Fenster des Dokumentes schliessen: document.close()

innerhalb von Eventhandler: Immer **window**.close() oder logischer_window_name.close() oder self.close() kodieren

Syntax:

```
logischer_window_name.close()
```


logischer_window_name Zeiger laut open()

liefert nichts

Beispiel:

```
<BODY onclick="window.close();">
    Klick zum Schliessen des Fensters
</BODY>
```

Beispiel für Fenster schliesst sich selbst nach Wartezeit:

```
</HTML>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function fenster_offnen_und_schliessen()
    {

        var fenster;
        fenster=window.open("","Fenster","width=180,height=100");
        fenster.document.write("<H1>Ich schließe mich nach 4 Sekunden</H1>");
        fenster.setTimeout('window.close()',4000);
    }
//-->
</SCRIPT>
</HEAD>
<BODY onload="fenster_offnen_und_schliessen()">
</BODY>
</HTML>
```

Beispiel:

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

....

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

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

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

.confirm() Dialogbox erzeugen mit

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

Syntax:

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

Kette	String oder Stringausdruck
Wert	true für OK false für Abbrechen
logischer_window_name	Zeiger laut open()

.createPopup() Popupfenster ohne irgendwelche Fensterelemente öffnen z.B. für Anzeige von Tooltips
siehe Objekt window.popup

Fenster wird mit der Anzeige per Methode .show() zum aktuellen Fenster
erst geschlossen, wenn **anderes Fenster** aktuell wird
z.B. durch Klicken außerhalb des Popupfensters

ab IE 5.5

Syntax:

[var Zeiger =] logischer_window_name.createPopup()

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

Beispiel

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

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler, wobei Registrierung mit Methode .attachEvent() aktiviert wurde

Achtung: Vor dem Neubelegen des Standard-Eventhandler diesen vor .attachEcent() retten und den Standard-Eventhandler **nach** .detachEvent() wieder einbinden (siehe Beispiel).

Syntax:

logischer_window_name.detachEvent(Kette, Zeiger)

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

liefert nichts

Beispiel 1:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor          = style.color;
            runtimeStyle.color  = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {
        if (event.srcElement == element)
        {
            runtimeStyle.color   = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu);    // nicht () kodieren !!!
        window.onmouseover = Rette_Standard_Eventhandler_OnMouseOver;

        detachEvent ('onmouseout', CursorNormal);// nicht () kodieren !!
        window.onmouseout = Rette_Standard_Eventhandler_OnMouseOut;
    }

    var Rette_Standard_Eventhandler_OnMouseOver = window.onmouseover;
    attachEvent ('onmouseover', CursorNeu);

    var Rette_Standard_Eventhandler_OnMouseOut = window.onmouseout;
    attachEvent ('onmouseout', CursorNormal);
```


```
</SCRIPT>
```

Beispiel 2:

```
function ResizeHandler()                    // Homepage neu laden
{window.history.go(0);}

// Standardhandler retten
var RetteHandler_Resize = window.onresize;

// und neu belegen
window.onresize = ResizeHandler;            // Homepage neu laden
                                            // ohne () kodieren, damit nicht sofort ausgeführt wird

// und Start des Abfangens vom resize-Event
window.attachEvent('onresize', ResizeHandler);

.........

// irgendwann abschalten der Eventüberwachung
window.detachEvent('onresize', ResizeHandler);

// und Standardhandler wieder einbinden
window.onresize = RetteHandler_Resize;
```

.execScript()

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

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

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

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

.focus()

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

```
logischer_window_name.focus()
```

logischer_window_name	Zeiger laut open()

liefert nichts

.moveBy()

linke obere Fenster-Ecke um Pixeldifferenz auf dem Bildschirm verschieben
Fenster ganz aus dem BS verschieben ist möglich
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Bildschirmes
nicht anwendbar bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop und dialogLeft verwenden.
Syntax:

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

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

liefert nichts

Beispiel für Fenster des Browser rausschieben und danach neu anzeigen:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
        var BrowserFenster_AktuelleBreite=2000;  // sollte größer als max. übliche Auflösung sein
        var BrowserFenster_AktuelleHoehe=2000;  // sollte größer als max. übliche Auflösung sein

        function moveWin()
        {
                for (var i = 1; i < BrowserFenster_AktuelleBreite; i++)
                {window.moveBy(1, 1);}

                window.moveBy((-1)* BrowserFenster_AktuelleBreite,(-1) * BrowserFenster_AktuelleHoehe);
        }
//-->
</SCRIPT>
</HEAD>
<BODY onload="moveWin();">
</BODY>
</HTML>
```

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

Syntax:

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

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

liefert nichts

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

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

Kette	String mit Url
logischer_window_name	Zeiger laut open()

liefert nichts

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

.open() neues Fenster als unterstes der aktuellen Fensterhierarchie erzeugen
und dann oberstes sichtbare Fenster öffnen (anzeigen, rendern und als aktuell setzen)
nicht möglich bei Dialog-Fenster oder Popup-Fenster
Hinweis: Zeiger des Fensters, das .open() ausführt, liegt in der Eigenschaft .opener des neu erzeugten Fensters
Syntax:

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

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

werden z.B.
window.open()
anstelle von .open()

logischer_window_name_neues_fenster Zeiger auf das neue Fenster

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

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

null null-Zeiger, also keine Referenzierung möglich

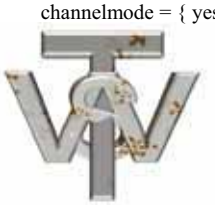
String Titel des Fensters oder vordefinierten Namen
Standard ist "_blank" also ohne Name

vordefinierte Namen:

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

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

channelmode = { yes | no | 1 | 0 }



default ist no bzw. 0
yes oder 1 für Channelleiste anzeigen
(Fenster im Channel-Mode anzeigen)

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

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

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

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

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

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

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

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

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

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

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

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

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

Replace true, so History-Eintrag des Dokumentes, das das neue
Fenster erzeugt, ersetzen durch den Eintrag
des neuen Fensters,
also Zurück zum Dokument, dass das Fenster öffnet,
nicht mehr möglich (**Achtung: Diese Einstellung
ist absolut User-unfreundlich, da der User**

nicht das Button "Zurück" verwenden kann, sondern erst in der Verlauf -Leiste suchen muss, und somit den den Eindruck bekommt, als würde man ihn an die neue Webseite zwanghaft binden wollen ! Das gilt vorallem dann, wenn der User in den Browser-Einstellungen das Öffnen im selben Fenster gewählt hat.)

false, so neuen Histrory-Eintrag zum neuen Fenster erzeugen,
also zurück zum alten Fenster möglich

Beispiel:

```
window.open("Sample.htm",null, "height=200,width=400,status=yes,toolbar=no,menubar=no,location=no");
```

Beispiel:

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

....

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

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

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

.print() ruft Dialog-Box-Druckfenster auf zum Druck des Dokumentes im aktuellen Fenster
entspricht Druckbutton oder Datei-Menü-Drucken
löst folgende Ereignisse aus: onbeforeprint und onafterprint
Syntax:

```
logischer_window_name.print()
```

logischer_window_name Zeiger laut open()

liefert nichts

Beispiel für Verwendung der Ereignisse per Handler

onbeforeprint	Handler blendet Teile der Seite ein/aus, die gedruckt werden sollen
onafterprint	Handler hebt Veränderungen von onbeforeprint wieder auf

Beispiel:

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

.prompt() Eingabefenster erzeugen:

1. Meldungstext anzeigen,
   Eingabezeile vorbelegen und anzeigen,
   OK- bzw. CANCEL-Button anzeigen
2. auf User.Eingabe in die Zeile und anschliessendem Druck auf OK warten

Syntax:

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

Kette1	String freier Text der Meldung
Kette2	String, frei Text der Vorbelegung der Eingabezeile Standard ist "undefined"

	Wert	String oder Integer Ergebnis der User-Eingabe
	logischer_window_name	Zeiger laut open()

.resizeBy() Fenstergröße um Pixeldifferenz verändern
Fenstergröße auf weniger als 100 x 100 Pixel ist nicht möglich
nicht bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop, dialogLeft benutzen
funktioniert nur, wenn open-Merkmal resizable=yes kodiert wurde
Syntax:

logischer_window_name.resizeBy(x,y)

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

liefert nichts

.resizeTo() Fenstergröße neu dimensionieren in Pixel
Fenstergröße auf weniger als 100 x 100 Pixel ist nicht möglich
nicht bei Dialog-Fenster: Dafür dialogHeight, dialogWidth, dialogTop, dialogLeft benutzen
funktioniert nur, wenn open-Merkmal resizable=yes kodiert wurde
Syntax:

logischer_window_name.resizeTo(x, y)

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

liefert nichts

Beispiel für Fensterauflösung ändern:

javascript:window.resizeTo(**640,480**); javascript:window.resizeTo(**800,600**); javascript:window.resizeTo(**1024,768**)

Beispiel für sich aufblasendes Fenster von 100x100 bis auf 640x480

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
        var start_hoehe=100;
        var start_breite=100;
        var max_hoehe=480;
        var max_breite=640;
        var aktuelle_hoehe=start_hoehe;
        var aktuelle_breite=start_breite;
        var y=5;
        var TimerID=null;
        var fenster;

function start()
{
        fenster=window.open("","","scrollbars");

        if ( document.layers  || document.all)
        {
                fenster.resizeTo(start_breite,start_hoehe);
                fenster.moveTo(0,0);
                blasen();
        }
        else
        {alert("Weder Netscape noch Microsoft erkannt !")};
}
```


```
function blasen()
{
        if (aktuelle_hoehe>=max_hoehe)
        {x=0;}

        fenster.resizeBy(5,y);
        aktuelle_hoehe+=5;
        aktuelle_breite+=5;

        if (aktuelle_breite>=max_breite)
        {
                alert("Maximal aufgeblasen !");
                fenster.close();
                x=5;
                TimerID=null;
        }
        else
        {TimerID=setTimeout("blasen()",50);}
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<A      HREF="javascript:start()"
        onMouseOver="javascript:window.status='Oeffne Fenster';return true;"    // Text nach Statuszeile
        onMouseout="javascript:window.status='';"                               // Statuszeile löschen
>Oeffne NEUES Fenster mit 100x100 Pixel und blase es auf  640x480 Pixel !
</A>
</BODY>
</HTML>
```

.scroll()

deprecated und zu ersetzen durch .scrollBy() bzw. .scrollTo()
linke obere Ecke des Dokumentes im Fenster verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters
Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisten etc.
sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters
Syntax:

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

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

liefert nichts

.scrollBy()

ersetzt .scroll()
linke obere Ecke des Dokumentes im Fenster verschieben um Pixeldifferenz bezüglich linker oberer Ecke des Fensters
Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisten etc.
sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde
Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters
Syntax:

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

x	Pixel-Abstand vom linken Fensterrand Integer wenn positiv, so Verschiebung nach rechts wenn negativ, so Verschiebung nach links
y	Pixel-Abstand vom oberen Fensterrand Integer wenn positiv, so Verschiebung nach unten wenn negativ, so Verschiebung nach oben

logischer_window_name Zeiger laut open()

liefert nichts

.scrollTo() ersetzt .scroll()

linke obere Ecke des Dokumentes im Fenster auf Pixelposition bezüglich linker oberer Ecke des Anzeigebereiches des Fensters setzen

Anzeigebereich: innerhalb des Fensterrahmens und abzüglich aller Fenster-Elemente wie Menüleiste, Scrolleisten etc.

sinnvoll, wenn automatische Anzeige von Scrolleisten verhindert wurde

Koordinaten-Ursprung (0,0) liegt in der linken oberen Ecke des Fensters

Syntax:

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

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

liefert nichts

.setActive() Fenster als aktiv setzen (also auch für die Eventdurchreichung aktivieren)

aber **ohne** es zu fokussieren

und **ohne** es scrollbar zu machen

Syntax:

```
logischer_window_name.setActive()
```

logischer_window_name Zeiger laut open()

liefert nichts

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    var ID_Fenster;

    function FensterErzeugen()
    {
        ID_Fenster = window.open( "/test /test.htm",
                                  " ID_Fenster",
                                  "top=10px,left=480px,height=375px,width=200px,resizable=1"
                                );
        this.focus();
    }

    function ButtonAktivieren()
    {window.parent.ID_Fenster.ID_Button.setActive();}
</SCRIPT>
</HEAD>
<BODY onload="FensterErzeugen();">
    <BUTTON ID="ID_Button" onclick="ButtonAktivieren();">
        Button aktivieren
    </BUTTON>
</BODY>
</HTML>
```

.setInterval() endlos-periodischer Aufruf eines Codes mit jeweiligem vorherigen Warten in Millisekunden (Timer)

(getimte Rekursion)

Der mit setInterval() aufgerufene Code wird zyklisch aktiviert, wobei nach dem ERSTEN Aufruf von setInterval() mit der Folgeanweisung **hinter** .setInterval() sofort weitergemacht wird, während die Rekursion parallel läuft. Damit gilt: setInterval() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da sonst diese Anweisungen während der parallelen rekursiven Ereignisüberwachung bereits abgearbeitet werden.

Syntax:

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

Code	auszuführender Code bzw. Zeiger auf Codebaustein vor IE 5.x: String aber kein Zeiger ab IE 5.x: String oder Zeiger Zeiger empfehlenswert, wenn Code in externer Datei

liegt
String empfehlenswert, wenn Code im aktuellen Dokument liegt
Übergabe von Argumenten möglich

Wartezeit	Wartezeit nach deren Ablauf wird Code aktiviert wird Integer in Millisekunden (1000 Millisekunden sind 1 Sekunde)
Sprache	Sprache des Codes (analog zum LANGUAGE-Attribut) String z.B. "JScript", "VBScript", "JavaScript"
TimerID	ID für den Stop der periodischen Aufruffolge mit der Methode .clearInterval() Integer
logischer_window_name	Zeiger laut open()

Beispiel 1

String

```
window.setInterval("EineFunktion()", 5000);
```

Zeiger

```
window.setInterval(EineFunktion, 5000);    // ohne () kodieren
```

Beispiel 2

```
function callback1(){alert("callback1");}

function callback2(){alert("callback2");}

function chooseCallback(Nummer)
{
        switch (Nummer)
        {
                case 0:  return callback1;
                case 1:  return callback2;
                default: return "";
        }
}
var Nummer = 1;
window.setInterval(chooseCallback(Nummer), 5000);
```

Beispiel 3

```
var SekundenZahler=0;
var timer_id=null;

function ZyklischeAktion()
{
        SekundenZahler++;
        window.status= SekundenZahler + " Sekunden ";

        if (      ( SekundenZahler >= 60 )
             &&   ( timer_id != null)
           )
        {window.clearInterval(timer_id);}
}

timer_id=window.setInterval(ZyklischeAktion,1000);
```

Beispiel 4

```
var SekundenZahler=0;
var timer_id=window.setInterval("window.status= SekundenZahler++",1000);
```

Beispiel 5:

```
var timerID = null;

function timer()
{
  // tue was
```

```
}

function starttimer()
{
	if (timerID == null)
	{timerID = setInterval("timer()", 1000);}
}

function stoptimer()
{
	if (timerID != null)
	{
		clearInterval(timerID);
		timerID = null;
	}
}
```

Beispiel 6 für Sekundenbalken:

```
<HTML>
<HEAD>
<STYLE>
	BUTTON {font-size:14;width:150}
</STYLE>
<SCRIPT>
	var timerID = null;
	var Zahler = 0;

	function Init()
	{
		// DIV-Eigenschaften festlegen

		// DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
		ID_Div1.style.setExpression("width","Zahler *10");

		// DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
		ID_Div2.setExpression("innerText","Zahler.toString()");
	}

	function Uhr()
	{
		// Sekunden kumulieren
		Zahler ++;

		// und Anzeige neu berechnen
		document.recalc();
	}

	function Starten()
	{
		if (timerID == null)
		{
			// Start-Button nicht aktivierbar machen
			ID_Button1.disabled = true;

			// Stop-Button aktivierbar machen
			ID_Button2.disabled = false;

			// Uhr neu starten
			timerID = setInterval("Uhr()", 1000);
		}
	}

	function Stoppen()
	{
		if (timerID != null)
		{
			clearInterval(timerID);
			timerID = null;
			ID_Button1.disabled = false;
			ID_Button2.disabled = true;
```


```
                    }
                }

                function ZurueckSetzen()
                { Zahler = 0; }
        </SCRIPT>
        </HEAD>
        <BODY onload="Init()">
                <DIV ID="ID_Div1"  STYLE="background-color:lightblue" ></DIV>
                <DIV ID="ID_Div1"  STYLE="color:hotpink;font-weight:bold"></DIV>
                <BR>
                <BUTTON         ID="ID_Button1"
                                onclick="Starten()"
                >
                        Start
                </BUTTON>
                </BR>
                <BUTTON         ID="ID_Button2"
                                DISABLED="true"
                                onclick="Stoppen()"
                >
                        Stop
                </BUTTON>
                <BR>
                <BUTTON         ID="ID_Button3"
                                onclick="ZurueckSetzen()"
                >
                        Reset
                </BUTTON>
                <BR>
                <P              STYLE="width:200;color:white;background-color:gray">
                        Sekundenbalken
                </P>
        </BODY>
        </HTML>
```

.setTimeout()

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

Der mit setTimeout() aufgerufene Code wird **nicht zyklisch** aktiviert, wobei nach dem Aufruf von setTimeout() mit der Folgeanweisung hinter setTimeout() sofort weitergemacht wird. Damit gilt: setTimeout() kann also nicht für Ereignisüberwachung verwendet werden, wenn nachfolgende Anweisungen das Ereignis verarbeiten sollen, da diese Anweisungen bereits während der Ereignisüberwachung abgearbeitet werden.

Syntax:

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

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

durch.clearTimeout() erfolgen muss
Hinweis: Ist keine Rekursion nötig, soll aber zyklisch aufgerufen werden, dann .setInterval() verwenden

logischer_window_name Zeiger laut open()

Beispiel 1

```
window.setTimeout("alert('Hallo')", 1000);
```

Beispiel 2

```
var Text = "Hallo ";
window.setTimeout(    "alert("
                        + Text
                        + ")",
                        1000
                  );
```

Beispiel 3

```
<SCRIPT>
    function Start(ZeigerAufObjekt)
    {window.setTimeout("Kode(" + ZeigerAufObjekt.id + ")", 3000);}

    function Kode(ID)
    {
        var Zeiger = eval(ID);
        Zeiger.style.display="none";
    }
</SCRIPT>
<INPUT TYPE=button VALUE="Unsichtbar in 3 Sekunden"
    ID="ID_Button" onclick=" Start(this)"
```

Beispiel 4

```
var SekundenZahler=0;
var timeout_id=null;

function ZyklischeAktion()
{
    SekundenZahler++;
    window.status= SekundenZahler + " Sekunden ";

    if (    ( SekundenZahler >= 60 )
         && ( timeou_id != null)
       )
    {window.clearTimeout(timeout_id);}
}

timeout_id=window.setTimeout(ZyklischeAktion,1000);
```

Beispiel 5

```
var timeout_id=window.setTimeout("window.status= 'Es ist 1 Sekunde vergangen !'",1000);
```

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

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

URL String,
URL der Help-Datei

ContextID String oder Integer
Kontext-Identifizierung in der Help-Datei

logischer_window_name Zeiger laut open()

liefert nichts

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

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

URL String als Url des zu ladendenen HTML-Dokumentes

Arguments freie Parameterliste für Werteübergabe Dialog-Fenster
Werteübergabe auch per Felder möglich
Format wie Eigenschaft .dialogArguments
wenn Liste als String, so max 4096 Zeichen in der Liste

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

Wert im Format laut Eigenschaft .returnValue

logischer_window_name Zeiger laut open()

Beispiel 1:

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

    function BoxHoeheErmitteln()
    {
        var OptionsFeld = ID_Formular.ID_Select.options;
        var OptionsFeldIndex = ID_Formular.ID_Select.selectedIndex;
        var OptionsWert = OptionsFeld [OptionsFeldIndex].text;

        // auf Auswahl " Zufallshoehe" prüfen
        if (OptionsWert.indexOf("Zufallshoehe") > -1 )
        {OptionsWert = ErmittleZufallsZahl(document.body.clientHeight);}

        // Features für Boxöffnen ermitteln
```

```
                    var BoxHoeheFeatures ="dialogHeight:" + OptionsWert + "px;";

                    // und liefern
                    return BoxHoeheFeatures;
                }

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

                    window.showModalDialog( "test.htm",
                                            "",
                                            BoxHoeheFeatures
                                          );
                }
            </SCRIPT>
            <FORM NAME="ID_Formular">
                waehle Boxhoehe in Pixel
                <SELECT NAME="ID_Select">
                    <OPTION>Zufallshoehe
                    <OPTION>150
                    <OPTION>200
                    <OPTION>250
                    <OPTION>300
                </SELECT>
                oeffne Modal Dialog Box
                <INPUT TYPE="button" VALUE="Box oeffnen" onclick="BoxOeffnen()">
            </FORM>
```

Beispiel 2:

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

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

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

test.htm enthält:

```
                <HTML>
                <HEAD>
```


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

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

.showModelessDialog() nicht-modales Fenster erzeugen, aber anzeigen nur, wenn Focus geändert wird
verwendbar für Desing von Menüs
Tooltips

Syntax:

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

URL — String als Url des zu ladendenen HTML-Dokumentes

Arguments — freie Parameterliste für Werteübergabe an Dialogbox
Werteübergabe auch per Felder möglich
Format wie Eigenschaft .dialogArguments
wenn Liste als String, so max 4096 Zeichen in der Liste

Features — String als Parameterliste mit Kommatrennung
gesamte Liste in " " setzen bzw. ' '
z.B. "center:yes, status:yes"
Listenelement: option:wert

dialogHeight:hoehe_in_pixel
ab IE 5.x auch Gleitkomma
cm, mm, in, pt, pc, oder px
mindestens 100 Pixel (wenn < 100 so
automatisch 100 verwendet)

dialogLeft:X_koordinate_relativ_linke_obere_Ecke_Screen
(0,0)

dialogTop:Y_koordinate_relativ_linke_obere_Ecke_Screen
(0,0)

dialogWidth:breite_in_pixel
ab IE 5.x auch Gleitkomma
cm, mm, in, pt, pc, oder px

center:{ yes | no | 1 | 0 | on | off }
default ist yes bzw. 1
yes bzw. 1 Dialogbox im Screen zentrieren

dialogHide:{ yes | no | 1 | 0 | on | off }
default ist no bzw. 0
yes bzw. 1 Dialogbox nicht druckbar bzw.
nicht angezeigt in
Druckvorschau

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

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

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

scroll:{ yes | no | 1 | 0 | on | off }

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

Wert im Format laut Eigenschaft .returnValue

logischer_window_name Zeiger laut open()

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    var Vorname="";      // muss  global sein da in Test.htm benutzt

    function VornameAnzeigen()
    {
        showModelessDialog( "Test.htm",                    // Url der Dialog-Box
                            window,
                            "status:false;dialogWidth:300px;dialogHeight:300px"
                          );
    }

    function VornameAktualisieren()          // Funktion wird in Test.htm aufgerufen
    { ID_Span.innerText = Vorname;}

</SCRIPT>
</HEAD>
<BODY>
    <P> aktueller Vorname ist :
        <SPAN ID="ID_Span"
              STYLE="color:red;font-size:24">
        unbekannt
        </SPAN>
    </P>
    <INPUT TYPE="button"
        VALUE="öffnen der Modeless Dialog Box"
        onclick="VornameAnzeigen()">
</BODY>
</HTML>
```

Kode für *Test.htm*, also dem Inhalt der Box

```
<HTML>
<HEAD>
<TITLE>Test.htm</TITLE>
<SCRIPT>
    function VornameAktuellAnzeigen()
    {
        var DialogArgumente = dialogArguments;
        DialogArgumente.Vorname = ID_Input.value;
        DialogArgumente.VornameAktualisieren();
                                        // Aufruf einer Funktion aus
                                        // Elterndokument
    }

    function Init()
    {
        var DialogArgumente = dialogArguments;
        DialogArgumente.Vorname = "unbekannt";
        DialogArgumente.VornameAktualisieren();
                                        // Aufruf einer Funktion aus
                                        // Elterndokument
    }
</SCRIPT>
</HEAD>
<BODY>
    <LABEL FOR="ID_Input" ACCESSKEY="v">
        Gib den
```


```
                    <SPAN STYLE="text-decoration:underline">
                    V
                    </SPAN>ornamen ein :
                </LABEL>
                <INPUT ID= "ID_Input">
                <BR><BR>
                <INPUT VALUE="Anzeige aktueller Vorname und Box offen lassen"
                    TYPE=button
                    onclick=" VornameAktuellAnzeigen();">

                <INPUT VALUE=" Anzeige aktueller Vorname und Box schliessen"
                    TYPE=button
                   onclick=" VornameAktuellAnzeigen();window.close();">

                <INPUT VALUE="Init"
                    TYPE=button
                    onclick=" Init();window.close();"
                >
        </BODY>
        </HTML>
```

4.3.2.2.1. window. clientInformation Objekt des Internet Explorer

ab IE 4.x
beinhaltet die Informationen über den Browser (Client)
Informationen über den Browser (Client) werden microsoft-spezifisch zusätzlich im Behavior clientCaps verwaltet

z.B. per Active-X installierbare Komponenten des IE
der Online-Zustand des Users zum Netzwerk
die Microsoft Virtuelle Maschine für Java

Erzeugung:
durch Browser

Zugriff:

window.clientInformation.eigenschaft oder clientInformation.eigenschaft
window.clientInformation.methode() oder clientInformation.methode()

logischer_window_name.clientInformation.eigenschaft
logischer_window_name.clientInformation.methode()

logischer_window_name Zeiger laut open()

Eigenschaften:

.appCodeName — Codename des Browsers

Syntax:

```
[ var Kette = ] logischer_window_name.clientInformation.appCodeName
```

Kette	String Default ist "Mozilla"
logischer_window_name	Zeiger laut open()

nur lesen

.appMinorVersion — Browser-Versionsnummer: Unternummer der Hauptnummer
z.B. 1 des IE 4.1

Syntax:

```
[ var Wert = ] logischer_window_name.clientInformation.appMinorVersion
```

Wert	Integer Achtung Opera-Browser gibt sich als IE aus !
logischer_window_name	Zeiger laut open()

nur lesen

.appName — Browsername

Syntax:

```
[ var Kette = ] logischer_window_name.clientInformation.appName
```

Kette	String Default ist "Microsoft Internet Explorer" "Netscape" ist der Netscape Navigator. Achtung Opera-Browser gibt sich als IE aus !
logischer_window_name	Zeiger laut open()

.appVersion — Plattform und Browserversion

Aufbau
clientVersion (platform; information; extraInformation)
oder 5.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)

Hinweis: Haupt-Versionsnummer ermittelbar per .parseFloat(navigator.appVersion)
z.B. 4 des IE 4.1
Syntax:
[var Kette =] logischer_window_name.clientInformation.appVersion

Kette String im Format
clientVersion (platform; information; extraInformation)

z.B. "5.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)"

logischer_window_name Zeiger laut open()
nur lesen

.browserLanguage
Browsersprache
Achtung: ab IE 5.x Sprache **immer** laut **regionalen** Einstellungen des Users im **Betriebssystems**
Syntax:
[var Kette =] logischer_window_name.clientInformation.browserLanguage

Kette String
unter 5.x: Sprache des Browsers
ab 5.x: Sprache des Betriebssystemes

logischer_window_name Zeiger laut open()
nur lesen

.cookieEnabled
Cookienutzbarkeit im Browser
Syntax:
[var Wert =] logischer_window_name.clientInformation.cookieEnabled

Wert false Browser unterstützt keine Cookies
true Browser unterstützt Cookies

logischer_window_name Zeiger laut open()
nur lesen

.cpuClass
CPU-Hersteller
Syntax:
[var Kette =] logischer_window_name.clientInformation.cpuClass

Kette String
"x86" Intel
"68K" Motorola
"Alpha" Digital
"PPC" Motorola
"Other" alles andere inklusive Sun SPARC.

logischer_window_name Zeiger laut open()
nur lesen

.onLine
System-Online-Status (nicht Netzwerk-Online-Status !!!)
Syntax:
[var Wert =] logischer_window_name.clientInformation.onLine

Wert true System ist online
false System ist offline

logischer_window_name Zeiger laut open()
nur lesen

Hinweis: Eigenschaft .connectionType des Behavior clientCaps zeigt den Status der genutzten Verbindung an

.platform
Betriebssystem
Syntax:
[var Kette =] logischer_window_name.clientInformation.platform

Kette String

"HP-UX"	HP Unix	
"MacPPC"	Macintosh Power-PC	
"Mac68K"	Macintosh 68K-based Computer	
"SunOS"	Solaris	
"Win32"	Windows 32-Bit	
"Win16"	Windows 16-Bit	
"WinCE"	Windows CE	

logischer_window_name Zeiger laut open()
nur lesen

.systemLanguage Standardsprache des Betriebssystemes
Syntax:
[var Kette =] logischer_window_name.clientInformation.systemLanguage

Kette String
Sprachcode
z.B. "en" oder "de"

logischer_window_name Zeiger laut open()
nur lesen

.userAgent HTTP User-Agent
Syntax:
[var Wert =] logischer_window_name.clientInformation.userAgent

Wert String
z.B. IE 6.0 liefert unter **Windows XP**
"Mozilla/4.0 (compatible; MSIE 6.0; **Windows NT 5.1**)"

logischer_window_name Zeiger laut open()
nur lesen

.userLanguage Sprache des Betriebssystemes laut Usereinstellung
Syntax:
[var Kette =] logischer_window_name.clientInformation.userLanguage

Kette String
Sprachcode
z.B. "en" oder "de"

logischer_window_name Zeiger laut open()
nur lesen

Methoden:

.javaEnabled() Java-Verfügbarkeit
ab IE 6.x
siehe Objekt window.clientInformation
Syntax:
[var Wert =] logischer_window_name.clientInformation.javaEnabled()

Wert true Java ist nutzbar
false Java ist nicht nutzbar

logischer_window_name Zeiger laut open()

Hinweis: Eigenschaft .javaEnabled des Behavior clientCaps zeigt die Verfügbarkeit der **Microsoft Virtuellen Maschine für Java** an

Achtung: Aufgrund eines Streites mit dem Unternehmen Sun entwickelt Microsoft ab IE 6.x die MS VM nicht mehr weiter, hält die VM aber noch per Download verfügbar. Für eine aktuellere Version muss der User selbständig beim Unternehmen Sun eine Linzenz für Java erwerben und die zugehörige Software installieren (inklusive Updates). Es reicht dabei die Run-Time-Version der Virtuellen Java Maschine von Sun. Ob allerdings Sun-Java-Standards in Microsoft-Produkten implementiert werden, ist nicht gesichert.

.taintEnabled() Data Tainting-Verfübarkeit
siehe Objekt window.clientInformation
Syntax:

[var Wert =] logischer_window_name.clientInformation.taintEnabled()

Wert	true	Data tainting ist verfügbar ab IE 6.x
	false	Data tainting is nicht verfügbar
		immer geliefert unter IE 6.x
logischer_window_name	Zeiger laut open()	

4.3.2.2.2. window.clipboardData Objekt des Internet Explorer

Unterstützung des Zugriffes auf vordefinierte Formate für Clipboard-Nutzung
Es wird das System-Clipboard verwendet. Mehrfacheinfügung ist möglich.
ab IE 5.x
siehe auch Objekt event.dataTransfer des IE

Erzeugung:
durch Browser

Zugriff:

window.clipboardData.methode() oder clipboardData.methode()

logischer_window_name. clipboardData.methode()

logischer_window_name Zeiger laut open()

Beispiel für Clipboard-Nutzung ohne Drag&Drop:

```
<HTML>
<HEAD>
<SCRIPT>
	function Init()
	{
		var TextBereich  = document.body.createTextRange();
		TextBereich.findText(ID_Div1.innerText);
		TextBereich.select();  // selektieren
	}

	function VorDemAusschneiden()
	{event.returnValue = false;} // Cut per Menü aktivieren

	function Ausschneiden()
	{
		ID_Div1.innerText = "";
		ID_Input.innerText += window.clipboardData.setData("Text", ID_Div1.innerText);
					// true oder false
					// Clipboard-Daten sind Texttyp
		event.returnValue = false;
	}

	function VorDemEinfuegen()
	{event.returnValue = false;} // Paste per Menü aktivieren

	function Einfuegen()
	{
		ID_Div2.innerText = window.clipboardData.getData("Text");
		event.returnValue = false;
	}
</SCRIPT>
</HEAD>
<BODY onload="Init()">
	<DIV ID=" ID_Div1" onbeforecut="VorDemAusschneiden()" oncut="Ausschneiden()">
		Dieses Text selektieren und ausschneiden
	</DIV>
	<DIV ID="ID_Div2" onbeforepaste="VorDemEinfuegen()" onpaste="Einfuegen()">
		den ausgeschnittenen Text hier einfügen
	</DIV><BR>
	<INPUT ID="ID_Input" TYPE="text" READONLY VALUE="" SIZE="6">
</BODY>
</HTML>
```

Beispiele für Clipboard-Nutzung mit Drag&Drop:

```
<HEAD>
<SCRIPT>
```


```
		function DragStart() // Url eines Ankers als Datenformat festlegen und Daten holen
		{event.dataTransfer.setData("URL", ID_A.href);}

		function DragEnde() // Daten im URL-Format als Textdaten in den Span ablegen
		{ID_Span.innerText = event.dataTransfer.getData("URL");}
</SCRIPT>
</HEAD>
<BODY>
		<A ID="ID_A" HREF="anker"
			onclick="return(false);"
			ondragstart=" DragStart()"
		>
			Testanker
		</A>
		<SPAN ID="ID_Span" ondragenter=" DragEnde()">
			obigen Link auf diese Stelle hierher droppen
		</SPAN>
</BODY>



<HEAD>
<SCRIPT>
		function InitiateDrag()
		{event.dataTransfer.setData("URL", ID_Image.src);}

		function FinishDrag()
		{ID_Span.innerText = event.dataTransfer.getData("URL");}
</SCRIPT>
</HEAD>
<BODY>
		<IMAGE ID="ID_Image" SRC="/test/graphics/test.gif"
					ondragstart="InitiateDrag()">
		<SPAN ID="ID_Span" ondragenter="FinishDrag()"
		>
			Image hierher droppen
		</SPAN>
</BODY>



<HTML>
<HEAD>
<SCRIPT>
		var Wert;

		function TextBereichErzeugen()
		{
			// Text im gesamten Body adressieren
			var TextBereich = document.body.createTextRange();

			// davon den Textteil des Source-Div adressieren
			TextBereich.findText(ID_DivSource.innerText);

			// und diesen selektieren
			TextBereich.select();
		}

		// Ausschneiden aktivieren, da standardgemäß für DIV dekativ ist
		// Ausschneiden bedeutet: Browser verschiebt automatisch in das Clipboard
		function AusschneidenAktivieren()  // ist Eventhandler für onbeforecut
		{event.returnValue = false;}		// Event-Handler-Rückkehrcode

		function Ausschneiden()		// ist Eventhandler für  oncut
		{
			// Clipboard füllen mit Daten vom Texttyp
			//					als Text des Source-Div
			Wert = window.clipboardData.setData("Text", ID_DivSource.innerText);

			// Source-Div Textbereich löschen
			ID_DivSource.innerText = "";
```


```
                    // Rückgabewert vom Clipboardfüllen als Text (true oder false)
                    //          im Text des Input-Button anzeigen
                    ID_Input.innerText += Wert; // Wert mit als Text

                    // Event-Handler-Rückkehrcode
                    event.returnValue = false;
          }
          // Einfügen aktivieren, da standardgemäß für DIV dekativ ist
          // Einfügen bedeutet: Browser fügt aus Clipboard in das Zielobjekt ein
          function EinfuegenAktivieren()   // ist Eventhandler für onbeforepaste
          {event.returnValue = false;}       // Event-Handler-Rückkehrcode

          function Einfuegen()  // ist Eventhandler für onpaste
          {
                    // aus Clipboard Daten vom Texttyp holen und in den Textbereich
                    // des Target-Div ablegen
                    ID_DivTarget.innerText = window.clipboardData.getData("Text");

                    // Event-Handler-Rückkehrcode
                    event.returnValue = false;
          }
</SCRIPT>
</HEAD>
<BODY   onload="TextBereichErzeugen()">
          <DIV ID="ID_DivSource" onbeforecut="AusschneidenAktivieren()"
                                        oncut="Ausschneiden()"
          >
                    Diesen Text mit Maus markieren und ausschneiden
          </DIV>
          <DIV ID="ID_DivTarget" onbeforepaste="EinfuegenAktivieren()"
                                        onpaste="Einfuegen()"
          >
                    An diese Stelle den ausgeschnittenen Text einfügen
          </DIV>
          <BR>
          <INPUT ID="ID_Input" TYPE="text" VALUE="Rückkehrcode = ">
</BODY>
</HTML>



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

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

          // ############## Zielobjekt ################################
          function EventHandlerFuerOnDragEnter
                    // Zielobjekt löst Event aus
                    // Maus über Ziel nach dem Draggen der Daten,
                    //          UND Maustaste ist noch nicht losgelassen
                    //   also Zielobjekt wird mit den Daten betreten
          {
```


```
                        // Daten adressieren
                        var ZwischenAblage = window.event.dataTransfer;

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

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

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

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

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

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

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

Eigenschaften:
keine
Methoden:

.clearData()	Clipboardinhalt löschen Anwendung für Ereignisse ondragstart oder ondrop
.getData()	Clipboard auslesen Anwendung für Ereignisse oncopy oder oncut Handler muss liefern window.event.returnValue=false;
.setData()	Clipboard füllen, also Daten dort ablegen wenn Clipboard nicht leer so immer anhängen

4.3.2.2.3. window.dialogArguments Objekt des Internet Explorer

Dieses Objekt verwaltet die Argumentenübergabe für ein modales oder nicht modales Dialogfenster.
Das Objekt kann folgende Datentypen der Argumente verwalten:

Stringvariable

numerische Variable
Objekt
Feld
Function

Sämtliche Argumente sind im Dialogfenster nur lesbar.
ab IE 5.x

Erzeugung:

Ein Dokument kann den Dialog mit folgenden Methoden erzeugen:

window.showModalDialog() oder .showModalDialog()
bzw. window.showModelessDialog() oder .showModelessDialog()

logischer_window_name.showModalDialog()
bzw. logischer_window_name.showModelessDialog()

logischer_window_name Zeiger laut open()

Diese Methoden erzeugen zum Elternfenster, dass die Methoden aufruft, ein neues Fenster für Dialog (Kind-Fenster).

Zugriff im Kindfenster:

var Zeiger = **window**.dialogArguments

Zeiger.eigenschaft
Zeiger.methode()

Zeiger auf das Objekt
eigenschaft und methode() laut Elternfenster (Elterndokument) also **namensindentische Referenz**

Beispiel:

Dokument, das den Dialog erzeugt (Elterndokument):

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

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

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

Dokument test.htm das den HTML-Container des Dialoges darstellt (Kind-Dokument):

```
<HTML>
<HEAD>
```


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

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

4.3.2.2.4. window.document Objekt des Internet Explorer

Achtung: Das Objekt document.body ist nicht mehr verfügbar und wurde durch das Objekt document.documentElement ersetzt !

document.documentElement ist der Zeiger auf den Body und nicht mehr document.body !

Beispiel zur Abfrage auf CSS1-Konformität und damit zur Verwendung von document.body bzw. document.documentElement

```
function DocTypeAbfrage()
{
        // Annahme: CSS1-Standard, also !DOCTYPE wurde kodiert
         var ZeigerAufDolument=document.documentElement;

         if (document.compatMode=="BackCompat")
         {
                // nicht CSS1-Standard, also wurde kein !DOCTYPE kodiert
                ZeigerAufDolument =document.body;
         }

         return ZeigerAufDolument;
}
```

Hinweise: document.compatMode ist nur lesbar, also !DOCTYPE nicht setzbar
Wert "CSS1Compat" ab IE 6
geliefert wenn !DOCTYPE kodiert wurde
Rendern mit CSS1 Standard (Standard-Compilant-Modus)
Wert "BackCompat" ab IE 3
Geliefert wenn kein !DOCTYPE kodiert wurde
Rendern mit Nicht-kein Standad-Compilant-Modus

Ansatz:
Das Objekt ist das HTML-Dokument, das in Browserfenster geladen und angezeigt wird.
ist der Container für die HTML-Elemente der Webseite z.B. BODY oder DIV
umfasst sämtlichen Code **zwischen** <HTML...> und </HTML>:
Der Code hinter </HTML> (z.B. Script) wird nicht geparst, aber eventuell als Plain-Text angezeigt.
Scriptsteuerung erst ab IE 4.x

Hinweis: Es gibt noch das html Objekt, welches den **HTML-Tag** beschreibt.

Das HTML-Dokument im HTML-DOM:

Das Objekt wird als Hierarchie von Knoten dargestellt, die nicht der Folge der Elemente im HTML-Code entspricht. Ein HTML-Element ist ein Knoten als Kind des Dokumentes.

Das HTML-Element im HTML-DOM:

Das HTML-Element wird in seine Komponenten zerlegt, die Knoten und Kinder des HTML-Elementes sind.
Bsp.: SPAN-Element mit Text: Text ist ein Knoten von SPAN

Zur Referenzierung von HTML-Elementen des Dokumentes sollte im Regelfall **document.all** kodiert werden, damit die Browserperformance steigt (siehe Collection document.all).

Das HTML-Element beim IE und NS:
als HTML-Tag: Der IE und NS unterstützen z.T. verschiedene Tags.

als Objekt: Es gibt Objekte, die im IE **und** NS implementiert sind aber z.T. auf verschiedene Weise
z.B. Objekt document

DOM und Collectionen zum Objekt document:

Zur Verwaltung des Dokumentes existieren Collectionen als Felder, die zugleich als Schnittstelle zum Programmierer dienen. Collectionen des Objektes document werden immer ohne .all kodiert, da sie keine HTML-Elemente darstellen. Der Netscape besitzt teilweise Collectionen, die auch der Internet Explorer bedient.

Objekt document und Browserfenster:

Das HTML-Dokument wird getrennt vom Fensterobjekt verwaltet: Ein Dokument muss zwar in ein Fenster geladen werden, aber das Dokument wird nur dann visualisiert (gerendert), wenn es sichtbare Elemente besitzt. Die Referenzierung des Dokumentes **im** Fenster erfolgt anhand des
logischen Fensternames, der geliefert wird
per Methode .open() zum window Objekt und ein frei festgelegter oder vordefinierter Name sein kann
als Wert des ID-Attributes bei FRAMESET mit FRAMES bzw. IFRAMES (siehe dort).

HTML-Dokument und Druck-Layout im Browserfenster:

Im Internet Explorer ab 5.x lässt sich das Druck-Layout der Seite, also des Dokumentes, bezüglich Kopf- und Fusszeile beeinflussen:

Menüpunkt Datei-Seite einrichten und dort die Angaben editieren.

Die Angaben haben folgende vordefinierte Komponenten:

&w	Platzhalter für den Titel des Fensters, in dem das Dokument angezeigt wird (siehe auch Objekt window)
&u	Platzhalter für die URL des Dokumentes (siehe auch Objekt location)
&d	Platzhalter für das Datum im Kurzformat laut Uhr des PC des Users
&D	Platzhalter für das Datum im Langformat laut Uhr des PC des Users
&t	Platzhalter für die Uhrzeit im 12-Stunden-Format laut Uhr des PC des Users
&T	Platzhalter für die Uhrzeit im 24-Stunden-Format laut Uhr des PC des Users
&p	Platzhalter für die Seitennnummer
&P	Platzhalter für die Gesamtanzahl der Seiten im Dokument
&b	Text zentrieren, der hinter &b kodiert wird
&b&b	Text rechtsbündig setzen, der hinter &b&b kodiert wird
&&	das Zeichen "&"

Das Layout ist in der Seitenansicht sichtbar, wobei die Platzhalter durch konkrete Werte ersetzt sind (als wenn gedruckt werden würde).

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

Folgende Reihenfolge wird beim Parsen eingehalten:

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

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

Hinweis zu den Beschreibungen der Objekte und Collectionen aus der Hierarchie des Dokumentes:

Die Beschreibungen beziehen sich in der Regel aus Gründen der Vereinfachung auf das aktuelle Fenster und dessen (aktuelles) Dokument. sind analog anwendbar für ein per logischem Windownamen adressiertes Dokument.

Erzeugung:
durch den Browser oder per Methode .createDocumentFragment()

Beispiel für Ausdruck des aktuellen Dokumentes:

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

Beispiel für Webseite mit eigenem Icon ab IE 5.x:

```
<LINK    REL="SHORTCUT ICON" HREF="name.ico">
```


name.ico: name ist beliebig, auch mit Pfad

ICO-Datei: 32x32 Pixel mit16 Farben
oder 16x16 Pixel mit 256 Farben
ist BMP-Datei, die zu ICO-Datei konvertiert werden muss
(kann nicht jedes Grafik-Programm, aber Microsoft bietet dafür IconPro an)

Beispiel für Ermittlung der Verweilzeit des Users auf der Webseite:

```
<HTML>
<HEAD>
<SCRIPT>
<!--
        var start_zeit;         // muss global sein

         function start()
        {start_zeit = new Date();}

        function ende()
        {
                var ende_zeit = new Date();
                var wert=ende_zeit.getTime() - start_zeit.getTime();
                wert = wert /1000;      // in Sekunden umrechnen
                alert("Verweilzeit in Sekunden: " + Math.floor(wert).toString());
        }
//-->
</SCRIPT>
</HEAD>

<BODY ... onLoad="start();" onUnload="ende();">
</BODY>
<HTML>
```

Beispiel für Seitenelemente vom Druck ausschließen (ab IE 4.x):

Seitenteile, die nicht gedruckt werden sollen, in <DIV CLASS="keindruck"> und </DIV>
oder in <SPAN CLASS="keindruck"> und </SPAN> einschliessen.

zwischen <HEAD> und </HEAD> einfügen: <LINK REL="stylesheet" MEDIA="print" HREF="print.css">

print.css enthält nur .keindruck {display:none;}

Beispiel für Seitenelemente nur beim Druck anzeigen (ab IE 4.x):

Seitenteile, die nur auf Ausdrucken sichtbar sein solllen, in <DIV CLASS="nurdruck"> und </DIV>
oder in <SPAN CLASS ="nurdruck"> und </SPAN> einschliessen

zwischen <HEAD> und </HEAD> einfügen: <LINK REL ="stylesheet" MEDIA="screen" HREF="screen.css">

screen.css enthält nur .nurdruck {display:none;}

Beispiel für Dokumenten-Hintergrund ein- bzw. ausblenden:

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.2">
<!--
//                      Dokument-Hintergrundfarbe ein- und ausblenden
//                              einblenden beim Dokument laden
//                              ausblenden beim Wechsel zu anderem Dokument, also entladen des aktuellen Dokumentes
//                                      mit Farbveränderung rückwärts gegenüber Laden

//*************************************************************************************************
//
//                      Nachfolgende Variablen muss der Programmierer anpassen
//
//*************************************************************************************************
//      Farbe wird aus Rot- und Gruen- und Blauanteil gebildet
//      Alle Farbanteilwerte gelten von 0 bis 255, alles ganzahlig, wird nicht geprüft
//      Startwerte für Startfarbe des Ein- bzw. Ausblenden, müssen <= Endwerte  sein, wird nicht geprüft
//      Endewerte  für Endefarbe nach Ein- bzw. Ausblenden, müssen >= Startwerte sein, wird nicht geprüft
```


```
//            Farbschritte = Stufen für Ein- bzw. Ausblenden
//                                    je mehr um so langsamer das Ein- bzw. Ausblenden
//                                    nur ganzahlig und maximal die kleinste paarige Differenz der obigen Werte
//                                          (keine Prüfung)

var RotAnteil_Start             = 0;
var GruenAnteil_Start                   = 0;
var BlauAnteil_Start            = 0;
var RotAnteil_Ende              = 255;
var GruenAnteil_Ende            = 255;
var BlauAnteil_Ende             = 255;
var FarbSchritte                = 64;

// ********************************************************************************************
//
//                    Nachfolgenden Code nicht verändern
//
//********************************************************************************************

function DezimalZuHexaString(DezimalerWert)
// liefert String von 2 Ziffern bzw. Grossbuchstaben
// X_DezimalerWert   Gleitkomma
//                                    wenn nicht ganzahlig, so vor der Konvertierung gannzahlig gemacht
//                                    wenn <= 0   so '00' geliefert
//                                    wenn >= 255 so 'FF' geliefert
{
        var HexaZeichenFeld = "0123456789ABCDEF";
        var HexaKette="";

        // Dezimalwert zu ganzahlig
        X_DezimalerWert=Math.floor(X_DezimalerWert);

        if (DezimalerWert < 0)
        {HexaKette="00";}
        else
        {
                if (DezimalerWert > 255)
                {HexaKette="FF";}
                else
                {
                        //            Indexe im HexaZeichenFeld ermitteln
                        var HexaKette_Zeichen1_PositionImHexaZeichenFeld = Math.floor(DezimalerWert / 16);
                        var HexaKette_Zeichen2_PositionImHexaZeichenFeld =
                                DezimalerWert  - (HexaKette_Zeichen1_PositionImHexaZeichenFeld * 16);

                        //            HexaKette bilden
                        HexaKette =
                          HexaZeichenFeld.charAt(HexaKette_Zeichen1_PositionImHexaZeichenFeld)
                        + HexaZeichenFeld.charAt(HexaKette_Zeichen2_PositionImHexaZeichenFeld);
                }
        }

        return HexaKette;
}

function BackgroundEinAusBlenden(RotAnteil_Start,GruenAnteil_Start,BlauAnteil_Start,
                                 RotAnteil_Ende,GruenAnteil_Ende,BlauAnteil_Ende,
                                 FarbSchritte
                                )
{
//            Farbe wird aus Rot- und Gruen- und Blauanteil gebildet
//            Alle Farbanteilwerte gelten von 0 bis 255, alles ganzahlig, wird nicht geprüft
//            Startwerte für Startfarbe des Ein- bzw. Ausblenden, müssen <= Endwerte   sein, wird nicht geprüft
//            Endewerte  für Endefarbe nach Ein- bzw. Ausblenden, müssen >= Startwerte sein, wird nicht geprüft
//            Farbschritte = Stufen für Ein- bzw. Ausblenden
//                                    je mehr um so langsamer das Ein- bzw. Ausblenden
//                                    nur ganzahlig und maximal die kleinste paarige Differenz der obigen Werte
//                                          (keine Prüfung)

        for(var i = 0; i <= FarbSchritte; ++i)
        {
```


```
                        var RotAnteil   = Math.floor(    (RotAnteil_Start   * ((FarbSchritte - i) / FarbSchritte))
                                                       + (RotAnteil_Ende    * (i / FarbSchritte))
                                                     );

                        var GruenAnteil = Math.floor(    (GruenAnteil_Start * ((FarbSchritte - i) / FarbSchritte))
                                                         + (GruenAnteil_Ende  * (i / FarbSchritte))
                                                       );

                        var BlauAnteil  = Math.floor(    (BlauAnteil_Start  * ((FarbSchritte - i) / FarbSchritte))
                                                       + (BlauAnteil_Ende   * (i / FarbSchritte))
                                                     );

                        document.bgColor =      "#"
                                              + DezimalZuHexaString(RotAnteil)
                                              + DezimalZuHexaString(GruenAnteil)
                                              + DezimalZuHexaString(BlauAnteil);
                }
        }

        //              Nachfolgende Funktionen exisitieren NUR, um die globalen Variablen für BODY verfügbar zu machen
        //                      da in der HTML-Body-Anweisung keine Javascript-Variablen akzeptiert werden
        function EinAusBlenden_DokumentLaden()
        {
                BackgroundEinAusBlenden(RotAnteil_Start,GruenAnteil_Start,BlauAnteil_Start,
                                        RotAnteil_Ende,GruenAnteil_Ende,BlauAnteil_Ende,
                                        FarbSchritte
                                       );
        }

        function EinAusBlenden_DokumentEntladen()
        {
                //              ist das Laden rückwärts
                BackgroundEinAusBlenden(RotAnteil_Ende,GruenAnteil_Ende,BlauAnteil_Ende,
                                        RotAnteil_Start,GruenAnteil_Start,BlauAnteil_Start,
                                        FarbSchritte
                                       );
        }
        // -->
        </SCRIPT>
        </HEAD>
        <BODY BGCOLOR=#ffffff      onload="EinAusBlenden_DokumentLaden();"
                                   onunload="EinAusBlenden_DokumentEntladen();"
        >
        </BODY>
        </HTML>
```

Beispiel für Dokumenten-Hintergrund auswählen:

```
        <HTML>
        <HEAD>
        <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.2">
        <!--
        //              Neues Fenster öffnen mit selektierten Hintergrund
        //
        function HintergrundSetzen(bgname)
        {
                var Fenster=window.open("","","width=380,height=380");
                Fenster.focus();
                Fenster.document.open();
                Fenster.document.write(   "<HTML><HEAD></HEAD>"
                                        + "<BODY BACKGROUND="
                                                + ""
                                                + bgname
                                                + ""
                                        + ">"
                                        + "</BODY>"
                                  + "</HTML>");
                Fenster.document.close();
        }
        //-->
        </SCRIPT>
```


```
</HEAD>
<BODY>
        <A HREF="getbackg.htm" onclick="HintergrundSetzen('bg1.jpg');return false">
                <IMG SRC="bg1.jpg" BORDER="0" WIDTH="96" HEIGHT="96">
        </A>
        <A HREF="getbackg.htm" onclick="HintergrundSetzen('bg2.jpg');return false">
                <IMG SRC="bg2.jpg" BORDER="0" WIDTH="96" HEIGHT="96">
        </A>
</BODY>
</HTML>
```

Beispiel für Änderung der Hintergrundfarben:

```
<HTML>
<BODY   bgColor="0000ff"
        onBlur="document.bgColor='ff0000'"
        onFocus="document.bgColor='0000ff'"
>
Dieses Fenster &auml;ndert die Hintergrundfarbe, wenn es nicht mehr aktiv ist.
</BODY>
</HTML>
```

Zugriff:

window.document.eigenschaft	oder	document.eigenschaft	oder	eigenschaft
window.document.methode()	oder	document.methode()	oder	methode()

```
logischer_window_name.document.eigenschaft
logischer_window_name.document.methode()
```

logischer_window_name laut open()

Der **logische** Fenstername muss nur dann kodiert werden, wenn der Bezug nicht auf das Dokument im aktuellen Fenster gehen soll.

Der logische Fenstername stammt aus
der Methode .open() zum window Objekt und kann ein frei festgelegter oder vordefinierter Name sein
aus dem Wert des ID-Attributes bei FRAMESET mit FRAMES.

Die Kodierung des Zeigers von document kann dann entfallen, wenn das aktuelle Dokument referenziert wird.
Allerdings ist von der Kodierung ohne window bzw. ohne window.document abzuraten, denn
es könnten gleichnamige Eigenschaften und Methoden geben, die in anderen Objekten auch implementiert sind
es wird damit die Browserperformance gesenkt.

Beispiel:

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

....

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

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

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

Eigenschaften:

Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:

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

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

Beispiel:

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

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

.bgColor deprecated und durch STYLE-Attribut zu ersetzen
Hintergrundfarbe des Objektes bzw. Dokumentes

.charset Zeichensatz zum Encoden eines Objektes im Dokument

.compatMode Kompatibilität des IE 6.x zu CSS1 bzw. seinen Browservorgängern
siehe style Objekt und Kodierung von !DOCTYPE
ab IE 6.x

Achtung: Das Objekt document.body ist nicht mehr verfügbar und wurde durch das Objekt document.documentElement ersetzt !

document.documentElement ist der Zeiger auf den Body und nicht mehr document.body !

Beispiel zur Abfrage auf CSS1-Konformität und damit zur Verwendung von document.body bzw. document.documentElement

```
function DocTypeAbfrage()
{
	// Annahme: CSS1-Standard, also !DOCTYPE wurde kodiert
	 var ZeigerAufDolument=document.documentElement;

	 if (document.compatMode=="BackCompat")
	 {
		// nicht CSS1-Standard, also wurde kein !DOCTYPE kodiert
		ZeigerAufDolument =document.body;
	 }

	 return ZeigerAufDolument;
}
```

Hinweise: document.compatMode ist nur lesbar, also !DOCTYPE nicht setzbar
Wert "CSS1Compat" ab IE 6
geliefert wenn !DOCTYPE kodiert wurde
Rendern mit CSS1 Standard (Standard-Compilant-Modus)
Wert "BackCompat" ab IE 3
Geliefert wenn kein !DOCTYPE kodiert wurde
Rendern mit Nicht-kein Standad-Compilant-Modus

.cookie Cookie des Dokumentes als Stringwert

Beispiel:

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

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

	function LeseCookieWert(Name)
```


```
// Name des zu lesenden Cookie ist String
// liefert Cookiewert aus unescape() als String
{
	var CookieWert="";   // Annahme: Cookie nicht gefunden oder mit Leerkette als Wert

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

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

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

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

	return CookieWert;
}
</SCRIPT>
```

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

.designMode — Editierbarkeit des Dokumentes (Manipulierbarkeit)
z.B. per Ausführung von Javscript

Syntax:

```
document.designMode [ = Kette ]
[ var Kette = ] document.designMode
```

Kette	String		
		"On"	Dokument kann editiert werden es wird kein Script ausgeführt !!
		"Off" oder "Inherit"	Standard Dokument kann nicht editiert werden Scripte werden ausgeführt "Inherit" bedeutet leider **nicht**: von Eltern geerbt, da Dokument keine Eltern haben muss (z.B. hat Frameset-Dokument Eltern)

lesen und schreiben

.dir — Umflussrichtung
.doctype — Referenz auf Dokumenttyp
.documentElement — Referenz auf Wurzelknoten (Root Node) des Dokumentes liefern

Beispiel:

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


```
</TEXTAREA>
```

Achtung: Das Objekt document.body ist nicht mehr verfügbar und wurde durch das Objekt document.documentElement ersetzt !

document.documentElement ist der Zeiger auf den Body und nicht mehr document.body !

Beispiel zur Abfrage auf CSS1-Konformität und damit zur Verwendung von document.body bzw. document.documentElement

```
function DocTypeAbfrage()
{
        // Annahme: CSS1-Standard, also !DOCTYPE wurde kodiert
         var ZeigerAufDolument=document.documentElement;

         if (document.compatMode=="BackCompat")
         {
                // nicht CSS1-Standard, also wurde kein !DOCTYPE kodiert
                ZeigerAufDolument =document.body;
         }

         return ZeigerAufDolument;
}
```

Hinweise: document.compatMode ist nur lesbar, also !DOCTYPE nicht setzbar
Wert "CSS1Compat" ab IE 6
geliefert wenn !DOCTYPE kodiert wurde
Rendern mit CSS1 Standard (Standard-Compilant-Modus)
Wert "BackCompat" ab IE 3
Geliefert wenn kein !DOCTYPE kodiert wurde
Rendern mit Nicht-kein Standad-Compilant-Modus

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

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

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

Beispiel:

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

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

.fileCreatedDate Datum der Dokumenterstellung

Beispiel 1:

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

Beispiel 2:

```
<SCRIPT>
        window.onload=fnInit;

        function fnInit()
        {
                var AnzahlMillisekundenProTag =86400000;

                var DatumDokumentErzeugung =new Date(document.fileCreatedDate);
```


```
                                        var DatumHeute =new Date();

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

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

.fileModifiedDate	Datum der letzten Dokumentveränderung
.fileSize	Größe des Dokumentes
.implementation	Referenz auf Objekt implementation zum Dokument
.lastModified	Datum der letzten Dokumentveränderung
.linkColor	Farbe von noch nicht benutzten Links im Dokument "#rrggbb" Standard ist "#0000FF" vordefinierter Farbname (browserspezifisch)

.location Zeiger auf das gleichnamige Objekt

Beispiel 1:

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

Beispiel 2:

```
<HEAD>
<SCRIPT>
        function Entfernen()
        {
                // versuche den Text zu entfernen
                try
                {
                        var KindZeigerAufTextImDiv = ID_Div.children(0);
                        ID_Div.removeChild(KindZeigerAufTextImDiv);
                        // Achtung: Der Text ist noch sichtbar !!!!
                }
                // oder fange das Ereignis des bereits entfernten Textes ein
                //                      und behandle das Ereignis
                catch(x)
                {
                        alert(          "Text wurde entfernt !\n"
                                      + "Das Dokument muss neu geladen werden !
                             );
                        document.location.reload();
                }
        }
</SCRIPT>
</HEAD>
<BODY>
        <DIV ID="ID_Div" onclick=" Entfernen()">
                Klick, um diesen Text zu entfernen !
        </DIV>
```


```
</BODY>
```

.parentWindow — Referenz auf Elternfenster des Dokumentes

.protocol — Protokoll-Teil einer Url inklusive http und ftp liefern
nur lesen bei Objekten document
img
location

Beispiel:

location.protocol liefert z.B. http: und immer inklusive dem Doppelpunkt

.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

Beispiel:

```
alert(document.body.readyState);
```

.referrer — Url der direkt vorhergehenden Seite der aktuellen Seite
aktuelle Seite muss durch Link angesprungen sein
Eingabe in Adresszeile oder per Menü Datei-Öffnen etc. nicht verwendet
ist Leerkette, wenn aktuelle Seite nicht durch Link von der vorherigen aktiviert wurde

.title — Fenstertitel des Dokumentes
siehe Objekt document

Beispiel für permanenten Fenstertitel-Wechsel:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    var titel_texte_feld =  [
                            "Titel 1",
                            "Titel 2",
                            "Titel 3",
                            "Titel 4"
                            ];

    var wechsel_tempo = 2500;       // Zeit in ms zwischen zwei Schritten
    var zahler = 0;
    var id;

    function start()
    {
        titel_texte_feld[titel_texte_feld.length] = document.title;
        // beim ersten Aufruf wird der Original-Titel mit gespeichert
        // Länge ab 1, aber Index ab 0
        //    Länge == Index +1, an desssen Position der Original-Titel abgelegt wird

        window.setTimeout("wechseln()", wechsel_tempo);
    }

    function wechseln()
    {
        document.title = titel_texte_feld [zahler];
        zahler ++;

        if(zahler >= texte.length)
        {zahler = 0}

        id = window.setTimeout("wechseln()", wechsel_tempo);
    }
//-->
</SCRIPT>
```

Beispiel für Fenstertitelzeile mit scrollendem Text:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
//
```


```
//		In der aktuellen Fenster-Titelzeile einen freien Text durch endloses Scrollen von lins nach rechts vorsetzen

//*************************************************************************************************
//
//			nachfolgende Variablen sind vom Programmierer zu setzen
//
//*************************************************************************************************
var VorsetzText_Wert		= "Freien Text vorsetzen durch scrollen und warten ";
var ScrollGeschwindigkeit	= "150";	// in Millisekunden, immer > 0, wird nicht geprüft
var DummyLeerzeichenAnzahl	= 10;		// immer > 0 , wird nicht geprüft
						// Die Leerzeichen werden nicht sichtbar angezeigt, aber gescrollt.
						//		So entsteht eine Wartezeit (pro Leerzeichen laut
						//		ScrollGeschwindigkeit)
						//		für die vollständige Anzeige von VorsetzText NACH dem
						//		vollständigem Vorsetzen per Scrollen von links nach
						//		rechts, ehe der Scroller wieder von vorn beginnt

//*************************************************************************************************
//
//			nachfolgender Code darf nicht verändert werden
//			wird anstelle von onLoad innerhalb BODY verwendet
//
//*************************************************************************************************
//		rechtsbündige Auffüllung mit DummyLeerzeichen
for (i=0; i <=DummyLeerzeichenAnzahl; i++)
{VorsetzText_Wert+=" ";}

//		Länge NACH Auffüllung ermitteln
var VorsetzText_Laenge=VorsetzText_Wert.length;

//		globale Variablen für EndlosScrollen(), die dort laufend manipuliert werden,
//			also hier initialisiert werden müssen
var Zahler="0";
var VorsetzText="";
//-->
</SCRIPT>
</HEAD>

<BODY>
<SCRIPT LANGUAGE="JavaScript1.2">

//*************************************************************************************************
//
//			nachfolgender Code darf nicht verändert werden
//
//*************************************************************************************************

function EndlosScrollen()
{
	// endloses Scrollen

	if (		(document.all)
			|| (document.getElementById)
	  )
	{
			// VorsetzText um nächsten Buchstaben aus dem VorsetzText_Wert erweitern
			// (einschliesslich DummyLeerzeichen)
			VorsetzText+=VorsetzText_Wert.charAt(Zahler);	// charAt ab Null

			// Fenster-Titelzeile neu belegen
			document.title=VorsetzText;

			Zahler++;
			if (Zahler==VorsetzText_Laenge)
			{
				Zahler="0";
				VorsetzText="";
			}

			setTimeout("EndlosScrollen()",ScrollGeschwindigkeit);
	}
```


```
}

window.onload=EndlosScrollen; // ohne () kodieren, damit nicht sofort sondern beim Laden ausgeführt wird
//-->
</SCRIPT>
</BODY>
</HTML>
```

.uniqueID — durch den Browser automatisch-genriertes ID des Objektes
Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde
kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

Beispiel:

```
<PUBLIC:ATTACH EVENT="onload" FOR="window" ONEVENT="init()"/>
<SCRIPT LANGUAGE="JScript">
    function init()
    {
        var newTextAreaID = element.document.uniqueID;
        element.document.body.insertAdjacentHTML (  "beforeEnd",
                                                    "<P><TEXTAREA STYLE='height: 200 ;"
                                                  + "width: 350' ID=" + newTextAreaID
                                                  + "></TEXTAREA></P>"
                                                 );
    }
</SCRIPT>
```

.URL — Url des Dokumentes
Achtung: Gross-und kleinschreibung wird unterschieden !!!
ist identisch mit location.href

.URLUnencoded — Url des Dokumentes ohne Zeichenencoding

.vlinkColor — Farbe bereits geklickter Links

.XMLDocument — Referenz auf XML-Dokument (XML-DOM)

.XSLDocument — Referenz auf den obersten Knoten des XSL-Dokumentes (Style-Sheet-Dokument)

Methoden:

.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)

Beispiel:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor         = style.color;
            runtimeStyle.color  = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {
        if (event.srcElement == element)
        {
            runtimeStyle.color   = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu);          // nicht () kodieren !!!
        detachEvent ('onmouseout', CursorNormal);// nicht () kodieren !!
    }

    attachEvent ('onmouseover', CursorNeu);
    attachEvent ('onmouseout', CursorNormal);
</SCRIPT>
```


.clear() Dokument löschen

.close() Ausgabe-Datenstream schliessen und Anzeige des Dokumentes beenden
schliessen einer mit .open() erzeugten Dokument-Instanz

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
	function ErzeugeZurLaufZeit()
	{
		// Dokumentinstanz erzeugen
		var ID_Dokument = document.open("text/html", "replace");

		// Dokumentinhalt festlegen und anzeigen
		ID_Dokument.write("<HTML><BODY>Hallo</BODY></HTML>");

		// Dokumentinstanz schliessen
		ID_Dokument.close();
	}
</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE="button" onclick=" ErzeugeZurLaufZeit();">
</BODY>
</HTML>
```

Beispiel:

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

....

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

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

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

.createAttribute() ein Attribut im Dokument erzeugen und Referenz auf das erzeugte Attribut liefern
Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!
DOM nicht geändert

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
	function Hinzufuegen()
	{
		// Attribut mit Wert erzeugen
		var ZeigerAufAttribut = document.createAttribute("title");
		ZeigerAufAttribut.value = "Tooltip-Text ";

		// Attribut als Feldelement anhängen
		var ZeigerAufFeld = ID_Div.attributes;
		ZeigerAufFeld.setNamedItem(ZeigerAufAttribut);
	}
```


```
		</SCRIPT>
		</HEAD>
		<BODY onload=" Hinzufuegen();">
			<DIV ID="ID_Div">Es wird ein Tooltip-Text hinzugefuegt</DIV>
		</BODY>
		</HTML>
```

.createComment() Comment-Objekt (Kommentar-Objekt) im Dokument erzeugen und Referenz liefern
verwendbar anstelle von direkt im HTML- bzw. Script-Quellcode kodiertem Kommentar
DOM wird geändert, da das Dokument erweitert wird

Beispiel:

```
var Kommentar = document.createComment("Das ist ein Kommentar");
```

.createDocumentFragment() erzeugt neues Dokument

.createElement() HTML-Objekt (Tags) im Dokument erzeugen und Referenz liefern
Achtung: Erzeugtes Objekt muss in DOM noch per Methode .insertBefore() bzw..appenChild() eingereiht werden.
Hinweis: Attribute mit der Methode .createAttribute() erzeugen
DOM wird geändert

Beispiel 1:

```
<SCRIPT>
	function Erzeugen()
	{
		ID_Span.innerHTML="";
		var FeldDerZeigerAufOption = ID_Select.options[ID_Select.selectedIndex];

		if(FeldDerZeigerAufOption.text.length>0)
		{
			var ZeigerAufElement=
				document.createElement(FeldDerZeigerAufOption.text);
			eval(	" ZeigerAufElement."
				+ FeldDerZeigerAufOption.value
				+ "='"
				+ ID_Input.value
				+ "'"
			    );

			if(FeldDerZeigerAufOption.text=="A")
			{ ZeigerAufElement.href="javascript:alert('A link.')"; }
		}
		ID_Span.appendChild(ZeigerAufElement);
	}
</SCRIPT>
<SELECT ID="ID_Select" onchange="Erzeugen()">
	<OPTION VALUE="innerText">A
	<OPTION VALUE="value">&lt;INPUT TYPE="button"&gt;
</SELECT>
<INPUT TYPE="text" ID="ID_Input" VALUE="Beispiel Text">
<SPAN ID="ID_Span" ></SPAN>
```

Beispiel 2:

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

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


```
                    ONCLICK=" Erzeuge()"
                    VALUE="Zwei Radio Buttons erzeugen">
                <BR>
                <INPUT  TYPE="BUTTON"
                    ONCLICK="alert(document.body.outerHTML)"
                    VALUE="Click um HTML zu sehen">
                <BODY>
        </HTML>
```

.createEventObject() Event-Objekt im Dokument erzeugen nur für Methode .fireEvent()

Beispiel:

```
var EventObjekt = document.createEventObject();
document.fireEvent("onclick", EventObjekt);
```

.createStyleSheet() styleSheet Objekt im Dokument erzeugen

Beispiel:

```
document.createStyleSheet('styles.css');
```

.createTextNode() Text-Objekt im Dokument erzeugen
nur Plain-Text, also keine HTML-Tags

Beispiel:

```
<SCRIPT>
    function TextElementAendern()
    {
        var ZeigerAufTextElement = document.createTextNode("Neuer Text");
        var ZeigerAufSpanInhalt = ID_Span.childNodes(0);
        ZeigerAufSpanInhalt.replaceNode(ZeigerAufTextElement);
    }
</SCRIPT>
<SPAN ID = "ID_Span" onclick=" TextElementAendern()">
    Original Text
</SPAN>
```

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler
wobei Registrierung mit Methode .attachEvent() aktiviert wurde
Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das **nicht** behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)

Beispiel:

```
<PUBLIC:ATTACH EVENT="ondetach" ONEVENT=" EventEntfernen()"/>

<SCRIPT LANGUAGE="JScript">
    function CursorNeu()
    {
        if (event.srcElement == element)
        {
            normalColor         = style.color;
            runtimeStyle.color  = "red";
            runtimeStyle.cursor = "hand";
        }
    }

    function CursorNormal()
    {
        if (event.srcElement == element)
        {
            runtimeStyle.color   = normalColor;
            runtimeStyle.cursor = "";
        }
    }

    function EventEntfernen()
    {
        detachEvent ('onmouseover', CursorNeu);          // nicht () kodieren !!!
        detachEvent ('onmouseout', CursorNormal);// nicht () kodieren !!
    }

    attachEvent ('onmouseover', CursorNeu);
    attachEvent ('onmouseout', CursorNormal);
</SCRIPT>
```


.elementFromPoint() liefert Referenz auf Objekt an Pixelpos relativ zum Fenster
Fenster linke obere Ecke (0,0)
Objekt muss Mausevents unterstützen

.execCommand() Kommando ausführen z.B. im aktuellen Dokument
in aktueller Selektion
im aktuellen Bereich
erst nach dem kompletten Laden des Dokumentes zulässig
Hinweis: Selektion = Markierung z.B. von Textbereich (Block)
Control = Element zur Steuerung analog zum HTML-Element (Tag)
Input-Control = Element mit Eingabeeigenschaft

Beispiel 1:

```
<HTML>
<BODY>
<H1 UNSELECTABLE="on">Demo</H1>
<SCRIPT>
function AddLink()
{
var SelektierterText = document.selection.createRange();

if (!SelektierterText =="")
{
// Link erzeugen
document.execCommand("CreateLink");

if (SelektierterText.parentElement().tagName == "A")
{

// markierten Text mit Eltern-Url ersetzen
SelektierterText.parentElement().innerText=
SelektierterText.parentElement().href;

// Vordergrundfarbe setzen im Dokument
document.execCommand(
"ForeColor","false","#FF0033");
}
}
else
{alert("Bitte im blauen Text selektieren !");
}
</SCRIPT>
<P UNSELECTABLE="on">
Selektiere (markiere) im nachfolgenden blauen Text die Stelle mit
dem Text MARKIERE_MICH.<BR>
Danach auf das Button klicken.<BR>
Anstelle von MARKIERE_MICH im blauen Text erscheint dort
nun eine Url.
</P>
<P STYLE="color=#3366CC">
Meine beliebeste Webseite MARKIERE_MICH bitte besuchen !
</P>
<BUTTON onclick="AddLink()" UNSELECTABLE="on">Klick</BUTTON>
</BODY>
</HTML>
```

Beispiel 2:

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

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

function HandlerFuerOnMove ()
{
ID_Span1.innerHTML = event.srcElement.offsetLeft;
```


```
                    ID_Span2.innerHTML = event.srcElement.offsetTop;
                }

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

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

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

.focus()

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

Beispiel:

```
<BODY onload="document.body.focus();>
```

.getElementById()

Referenz auf das im Dokument ZUERST gefundene Objekt laut ID (analog zum ID-Attribut) liefern
Achtung: Objekte, die kein ID besitzen, werden nicht erfasst !
Für Verwaltung per NAME (analog zum NAME-Attribut):
siehe Methode getElementsByName()
Für Verwaltung per Tag-Name
siehe Methode .getElementsByTagName()
wenn mehrere Elemente mit ein und demselben ID, so das ERSTE Element von diesen referenziert
Eine Referenzierung einer Collection der Objekte mit gemeinsamen ID ist leider nicht möglich. Deswegen der strenge Hinweis:
In der Regel werden ID vom Programmierer objektweise getrennt vergeben, es sei denn, man will bewusst eine Gruppe von Objekten (z.B. RadioBox) verwaltbar machen und kennt diese Objekte. Die maschinelle Analyse eines fremden Dokumentes mit der Methode .getElementById() ist nicht möglich.
DOM nicht geändert

Beispiel:

```
<SCRIPT>
    function Referenziere()
    {
        var Zeiger =document.getElementById("ID_Div");
    }
</SCRIPT>
<DIV ID="ID_Div">Test</DIV>
<INPUT TYPE="button" VALUE=" Referenziere" onclick=" Referenziere()">
```

.getElementsByName()

Referenz auf ein Feld (Collection) aller im Dokument befindlichen Objekte mit gemeinsamen NAME (analog zum Attribut NAME) liefern
Achtung: Objekte, die kein NAME besitzen, werden nicht erfasst !
Für Verwaltung per ID (analog zum ID-Attribut):
siehe Methode getElementsById()
Für Verwaltung per Tag-Name
siehe Methode .getElementsByTagName()
DOM nicht geändert

Beispiel:

```
<SCRIPT>
    function Referenziere()
    {
        var FeldDerInput =document.getElementsByName("GemeinsamerName");
    }
</SCRIPT>
<INPUT TYPE="text" NAME="GemeinsamerName">
<INPUT TYPE="text" NAME="GemeinsamerName">
<INPUT TYPE="button" NAME="Button" VALUE=" Referenziere" onclick=" Referenziere()">
```

.getElementsByTagName() Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc.

Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection)

Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst !

Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementsById()

Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()

DOM nicht geändert

Beispiel 1:

```
var DIV_KinderZeigerFeld = document.body.getElementsByTagName("DIV");
```

Hinweis: entspricht var DIV_KinderZeigerFeld = document.body.all.tags("DIV");

Beispiel 2:

```
<SCRIPT>
    var Feld_Span = ID_DivEltern.getElementsByTagName("SPAN");
                                    // alle SPAN-Kinder referenzieren
</SCRIPT>
<DIV ID="ID_DivEltern">
    <SPAN>
        Span-Kind von ID_DivEltern
        <DIV>
            Div-Kind vonDivEltern-Span
            <SPAN>
            Span-Kind vonDivEltern-Span-Div
            </SPAN>
        </DIV>
    </SPAN>
</DIV>
```

Beispiel 3:

```
<SCRIPT>
    function Anzeige()
    {
        var ZeigerAufOnClickEventQuelle=event.srcElement;
        var Feld =
            ZeigerAufOnClickEventQuelle.parentElement.getElementsByTagName("LI");
        alert(      "Anzahl LI : "
                + Feld.length
                + "\nErster Eintrag: "
                + Feld [0].childNodes[0].nodeValue
            );
    }
</SCRIPT>
<UL onclick="Anzeige()">
    <LI>Menuepunkt 1
    <UL>
        <LI> Menuepunkt 1.1
            <OL>
                <LI> Menuepunkt 1 1.1
                <LI> Menuepunkt 1 1.2
            </OL>
        <LI> Menuepunkt 1.2
        <LI> Menuepunkt 1.3
    </UL>
    <LI> Menuepunkt 2
    <UL>
        <LI> Menuepunkt 2.1
```


```
                    <LI> Menuepunkt 2.3
                </UL>
                <LI> Menuepunkt 3
            </UL>
```

.hasFocus()
Objekt im Focus-Zustand
true Objekt hat den Focus
false Objekt hat keinen Focus

.mergeAttributes()
alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen
Attribute sind: HTML
Events
Styles
ab IE 5.01 auch ID, NAME
Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
DOM wird geändert

Beispiel:

```
<SCRIPT>
    function Mischen()
    { ID_SPAN.children[1].mergeAttributes(ID_SPAN.children[0]);}
</SCRIPT>
<SPAN ID=ID_SPAN>
    <DIV    ID="ID_Div_Quelle"
            ATTRIBUTE1="true"
            ATTRIBUTE2="true"
            onclick="alert('click');"
            onmouseover="this.style.color='#0000FF';"
            onmouseout="this.style.color='#000000';"
    >
            Quell<B>Div</B>
    </DIV>
    <DIV    ID="ID_Div_Ziel">
            Ziel-Div
    <DIV>
</SPAN>

<INPUT  TYPE="button" VALUE=" Mischen" onclick="Mischen()">
```

.open()
Dokument instanzieren und mit/ohne Fenster öffnen und Referenz liefern auf Dokument
Ausgabe-Datenstream öffnen und Anzeige des Dokumentes
Instanzieren ohne .open(): siehe .write() und .writeln()
Hinweis: neues Fenster erzeugen als unterstes der aktuellen Fensterhierarchie
und dann Fenster öffnen (anzeigen, rendern)
Syntax:
für Öffnen mit Fenster:

```
[ var ZeigerAufWindow =] document.open(
                [URL] [, Name] [, Features] [, Replace])
```

für Öffnen im aktuellen Fenster

```
[ var ZeigerAufDokument =] document..open([mime_typ] [,Replace])
```

für Öffnen im beliebigen Fenster

```
[ var ZeigerAufDokument =] window_instanz.document..open([mime_typ] [,Replace])
```

URL String mit Multipurpose Internet Mail Extensions (MIME) Typ
Standard: MIME-Typ "text/html"
nur "text/html" zulässig

Name String Dokumentname für Verlauf (History)
wenn kodiert, so immer das aktuelle Dokument durch das Neue ersetzt im Verlauf
wenn nicht kodiert, so neues Dokument angehängt an History

physischer Window-Name: identisch mit Wert laut Attribut
TARGET z.B. im Link
dient der Referenz
auch null kodierbar für keine Referenz
Vordefinierte Namen:
"_blank" Standard : ohne Name
"_media" Dokument laut Url wird in den HTML-Content-Bereich der Media Bar geladen

ab IE 6.x
"_parent" Dokument laut Url wird in den Elternframe geladen wird.
wenn kein Frame so wirkunglos
"_search" Dokument laut Url wird in das Browser-Search-Fenster geladen
ab IE 5.x
"_self" Dokument laut Url wird in das neue Fenster geladen
"_top" Dokument laut Url wird in das oberste Fenster geladen
identisch mit _self, da neues Fenster das oberste wird

mime_typ Datentyp des Dokumentes
meistens **"text/html"** HTML-Dokument
oder **"text/plain"** Normaltext-Dokument

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

alwaysLowered = { yes | no }
yes: Fenster öffnen und permanent als unterstes in der Fensterhierarchie belassen
nur per signiertem Script
beachte .setZOptions()
nur NS

alwaysRaised = { yes | no}
yes: Fenster öffnen und permanent als oberstes in der Fensterhierarchie belassen
nur per signiertem Script
beachte .setZOptions()
nur NS

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

dependent = { yes | no }
yes: Fenster an den .opener bezüglich Fensterschliessen koppeln: auch schliessen, wenn .opener geschlossen wird
sowie geöffnetes Fenster nicht in der Taskleiste von Windows anzeigen
nur NS

directories = { yes | no | 1 | 0 }
default ist yes bzw. 1

yes oder 1 Directory Buttons anzeigen

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

height = Pixelwert, Fensterhöhe
bei IE : mindestens 100 (wenn < 100, so auf
100 automatisch gesetzt)
bei NS: nur mit signiertem Script < 100
ohne signiertem Script: wenn < 100,
so auf 100 automatisch gesetzt

hotkeys = { yes | no}
yes: alle sicherheitsrelevanten
Tastenkombinationen
verwendbar machen
no: nur noch ALT+F4 funktioniert
(schliessen des Fensters)
nur NS

innerHeight= anzeigebereich_höhe_in_pixel
ohne signiertes Script: >= 100
Anzeigebereich = Fenster abzüglich
Leisten, Scrollbars etc.
nur NS

innerWidth= anzeigebereich_breite_in_pixel
ohne signiertes Script: >= 100
Anzeigebereich = Fenster abzüglich
Leisten, Scrollbars etc.
nur NS

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

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

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

outerHeight= fenster_höhe_in_pixel
ohne signiertes Script: >= 100
Fenster = Anzeigebereich sowie Leisten,
Scrollbars etc.
nur NS

outerWidth= fenster_breite_in_pixel
ohne signiertes Script: >= 100
Fenster = Anzeigebereich sowie Leisten,
Scrollbars etc.
nur NS

personalbar = { yes | no }
yes: persönliche Toolbar anzeigen
nur NS

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

bei NS beachte .setResizable()

screenX= horizontale_pixel_pos des Fensters
bezüglich dem Bildschirm,
dessen Ursprung (0,0) in der
linken oberen Ecke liegt
nur NS

screenY= vertikale_pixel_pos des Fensters
bezüglich dem Bildschirm,
dessen Ursprung (0,0) in der
linken oberen Ecke liegt
nur NS

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

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

titlebar = { yes | no | 1 | 0 }
default ist yes bzw. 1
yes bzw. 1 für Titelzeile anzeigen
Titel werden immer angezeigt bei Dialog-Box
bei NS: nur mit signiertem Script setzbar

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

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

width = Pixelwert, Fensterbreite,
bei IE : mindestens 100 (wenn < 100, so auf
100 automatisch gesetzt)
bei NS: nur mit signiertem Script < = 100
ohne signiertem Script: wenn < 100,
so auf 100 automatisch gesetzt

z-lock = { yes | no}
yes: Fenster kann kein anderes
überlagern
nur per signiertem Script
beachte .setZOptions()
nur NS

Replace true, so History-Eintrag des Dokumentes, in dem das neue
Fenster erzeugt wird, ersetzen durch den Eintrag
des neuen Fensters, also Zurück zum Dokument,
dass das Fenster öffnet, nicht möglich
false, so neuen History-Eintrag zum neuen Fenster
erzeugen, also zurück zum alten Fenster möglich

ZeigerAufWindow Referenz (ID) auf das neue Fenster mit Dokument
z.B. für Methode window.close()

ZeigerAufDokument
Referenz (ID) auf das neue Dokument
z.B. für Methode ZeigerAufDokument.close()

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    function ErzeugeZurLaufZeit()
```


```
                {
                        // Dokumentinstanz erzeugen
                        var ID_Dokument = document.open("text/html", "replace");

                        // Dokumentinhalt festlegen und anzeigen
                        ID_Dokument.write("<HTML><BODY>Hallo</BODY></HTML>");

                        // Dokumentinstanz schliessen
                        ID_Dokument.close();
                }
        </SCRIPT>
        </HEAD>
        <BODY>
        <INPUT TYPE="button" onclick=" ErzeugeZurLaufZeit();">
        </BODY>
        </HTML>
```

Beispiel:

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

        ....

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

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

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

.queryCommandEnabled()	prüfen ob Kommando ausführbar ist
.queryCommandIndeterm()	prüfen ob Kommando-Status bestimmbar ist oder nicht
.queryCommandState()	Status des aktuellen Kommando ermitteln: ob ausgeführt wurde oder nicht
.queryCommandSupported()	prüfen ob Kommando im aktuellen Bereich unterstützt wird
.queryCommandValue()	Wert eines Kommandos liefern

.recalc()	dynamischen Eigenschaften des Dokumentes neu berechnen Hinweis: andere Objekte, die Eigenschaften des Dokumentes nutzen, werden auch neu berechnet, wenn Eigenschaften nicht in einem Berechungsausdruck vorliegen

Beispiel 1:

```
        <HTML>
        <HEAD>
        <STYLE>
                BUTTON {font-size:14;width:150}
        </STYLE>
        <SCRIPT>
                var timerID = null;
                var seconds = 0;

                function init()
                {
                        ID_Div1.style.setExpression("width","seconds*10");
                        ID_Div2.setExpression("innerText","seconds.toString()");
                }

                function timer()
```


```
		{
			seconds++;
			document.recalc();
		}

		function starttimer()
		{
			if (timerID == null)
			{
				timerID = setInterval("timer()", 1000);
				ID_Button1.disabled = true;
				ID_Button2.disabled = false;
			}
		}

		function stoptimer()
		{
			if (timerID != null)
			{
				clearInterval(timerID);
				timerID = null;
				ID_Button1.disabled = false;
				ID_Button2.disabled = true;
			}
		}

		function resettimer()
		{seconds = 0;}
	</SCRIPT>
	</HEAD>
	<BODY onload="init()">
		<DIV ID="ID_Div1" STYLE ="background-color:lightblue" ></DIV>
		<DIV ID="ID_Div2"  STYLE="color:hotpink;font-weight:bold"></DIV>
		<BR>
		<BUTTON ID="ID_Button1"                    onclick="starttimer()">Start Timer</BUTTON></BR>
		<BUTTON ID="ID_Button2" DISABLED="true"  onclick="stoptimer()">Stop  Timer</BUTTON><BR>
		<BUTTON                                    onclick="resettimer()">Reset Timer</BUTTON><BR>
	</BODY>
	</HTML>
```

Beispiel 2 für Sekundenbalken:

```
	<HTML>
	<HEAD>
	<STYLE>
		BUTTON {font-size:14;width:150}
	</STYLE>
	<SCRIPT>
		var timerID = null;
		var Zahler = 0;

		function Init()
		{
			// DIV-Eigenschaften festlegen

			// DIV-Breite je nach Sekundenanzahl, also dynamische DIV-Breite als Balken
			ID_Div1.style.setExpression("width","Zahler *10");

			// DIV-Inhalt als Sekundentext, der permanent aktualisiert wird
			ID_Div2.setExpression("innerText","Zahler.toString()");
		}

		function Uhr()
		{
			// Sekunden kumulieren
			Zahler ++;

			// und Anzeige neu berechnen
			document.recalc();
		}
```


```
		function Starten()
		{
			if (timerID == null)
			{
				// Start-Button nicht aktivierbar machen
				ID_Button1.disabled = true;

				// Stop-Button aktivierbar machen
				ID_Button2.disabled = false;

				// Uhr neu starten
				timerID = setInterval("Uhr()", 1000);
			}
		}

		function Stoppen()
		{
			if (timerID != null)
			{
				clearInterval(timerID);
				timerID = null;
				ID_Button1.disabled = false;
				ID_Button2.disabled = true;
			}
		}

		function ZurueckSetzen()
		{ Zahler = 0; }
	</SCRIPT>
	</HEAD>
	<BODY onload="Init()">
		<DIV ID="ID_Div1"  STYLE="background-color:lightblue" ></DIV>
		<DIV ID="ID_Div1"  STYLE="color:hotpink;font-weight:bold"></DIV>
		<BR>
		<BUTTON		ID="ID_Button1"
				onclick="Starten()"
		>
			Start
		</BUTTON>
		</BR>
		<BUTTON		ID="ID_Button2"
				DISABLED="true"
				onclick="Stoppen()"
		>
			Stop
		</BUTTON>
		<BR>
		<BUTTON		ID="ID_Button3"
				onclick="ZurueckSetzen()"
		>
			Reset
		</BUTTON>
		<BR>
		<P		STYLE="width:200;color:white;background-color:gray">
			Sekundenbalken
		</P>
	</BODY>
	</HTML>
```

Beispiel 3 für Sound mit Sekundenanzeige:

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

		var PixelBreiteProBalkenErweiterung = 10;

		// ++++++++++++ Browser-Typ ermitteln
```


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

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

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

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

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

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

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

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

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

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

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

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

                // Millisekunden warten und danach die Funktion SoundAbspielen() neu aufrufen
                SoundTimeoutID = setTimeout(
                                "SoundAbspielen()",
```


```
                                                SoundObjekt.SoundFileDauerInMillisekundeSekunden
                                                        );
                        }
                        else
                        {
                                // Dieser Zweig wird erst mit dem 2. Aufruf der Funktion abgearbeitet

                                // Sound zu Ende
                                SoundObjekt.SoundFileBeendet=true;

                                // Sekundenzähler stoppen
                                SekundenZaehlen_Stop();

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

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

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

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

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

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

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

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

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


```
                                                + '</DIV>'
                                                + '<BR>'
                                        );

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

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

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

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

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

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

Beispiel:

```
<BODY   onload="ID_Div.setCapture();"
        onclick="document.releaseCapture();"
>
        <DIV ID="ID_Div"
                onmousemove="ID_Textarea.value = event.clientX + event.clientY;"
                onlosecapture="alert(event.srcElement.id + ' hat keine Mausueberwachung mehr');"
        >
                <TEXTAREA ID="ID_Textarea" COLS=2>Test</TEXTAREA>
        </DIV>
</BODY>
```

.setActive() Objekt für die Eventdurchreichung aktivieren
aber ohne es zu fokussieren
und ohne es scrollbar zu machen

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
        var ID_Fenster;

        function FensterErzeugen()
        {
                ID_Fenster = window.open( "/test /test.htm",
                                          " ID_Fenster",
                                          "top=10px,left=480px,height=375px,width=200px,resizable=1"
                                        );
                this.focus();
        }

        function ButtonAktivieren()
        {window.parent.ID_Fenster.ID_Button.setActive();}
</SCRIPT>
</HEAD>
<BODY onload="FensterErzeugen();">
```


```
                    <BUTTON ID="ID_Button" onclick="ButtonAktivieren();">
                        Button aktivieren
                    </BUTTON>
                </BODY>
                </HTML>
```

.write() Text bzw. HTML-Ausdruck in das Dokument ausgeben und anzeigen bzw. sofort parsen

Hinweis:

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

Plain-Text
HTML-Text
Script

.writeln() Text bzw. HTML-Ausdruck in das Dokument ausgeben und anzeigen bzw. sofort parsen

Hinweis:

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

nach der Anzeige einen Zeilenvorschub anzeigen
Plain-Text
HTML-Text
Script

4.3.2.2.4.1. window.document.all Collection des Internet Explorer

(dokumentbezogene Verwaltung von HTML-Elementen)

Die Collection ist

das Feld der Zeiger **aller HTML**-Elemente des Dokumentes inklusive der HTML-Tags HTML, HEAD, TITLE etc.

nicht das Feld aller Knoten im DOM (siehe Beschreibung vom DOM), weil die Folge der HTML-Elemente im HTML-Code zum Dokument **verschieden** ist zur Folge in der gesamten **und** dokument-eigenen DOM-Hierarchie.

dient zur Verwaltung aller

IE-üblichen und IE-speziellen Eigenschaften und Methoden der Objekte vom Objekt document und natürlich der des Dokumentes selbst

Eigenschaften und Methoden und von Objekten außerhalb der Hierarchie des Objektes document

wird **nicht** vom Netscape erkannt

Unterschied der Kodierung document und document.all:

mit oder ohne .all werden kodiert:

IE-**übliche** Eigenschaften und Methoden (also die des Objektes document), da diese vererbt sind

Eigenschaften und Methoden von Objekten außerhalb der Hierarchie des Objektes document

mit .all werden kodiert:

IE-**spezielle** Eigenschaften und Methoden (also die der Objekte innerhalb der Hierarchie des Objektes document)

Empfehlung: Es sollte beim IE <u>immer</u> document.all kodiert werden, egal um welches HTML-Objekt es sich innerhalb des Dokumentes handelt, also auch bei der Referenzierung über den Wert des ID- bzw. NAME-Attributes. Anders formuliert: Es sollte immer der <u>absolute Pfad ab Objekt document</u> kodiert werden, wobei der Pfad bei Mehrfachverwendung in einem Zeiger liegen sollte (höhere Browser-Performance).

IE-übliche bzw. -spezielle Objekte:

IE-**übliche** Eigenschaften und Methoden gehören dem Objekt document
werden an die Objekte des Objektes document vererbt
können wie folgt kodiert werden:

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

oder gehören Objekten außerhalb der Hierarchie des Objektes document
werden z.T. vom Netscape unterstützt
können wie folgt kodiert werden: document.object.eigenschaft

document.object.methode()
mit object als Zeiger auf
das HTML-Element
z.B. laut ID-Attribut

IE-**spezielle** Eigenschaften und Methoden gehören den Objekten des Objektes document
liegen innerhalb der Hierarchie des Objektes document
müssen mit .all kodiert werden: document.all.eigenschaft
document.all.methode()

Falls es zulässig ist, dass ein Objekt des Objektes document als Kind eines anderen Objektes instanziert werden kann, dann ist die Kodierung natürlich um die Referenz auf das Elternobjekt zu erweitern, es sei denn, man verwendet den Wert des ID-Attributes des Kindes als eindeutigen Zeiger .

Es ist möglich, dass **sämtliche** Eigenschaften und Methoden, deren Kodierung **kein .all** benötigen (also IE-**übliche** Eigenschaften und Methoden bzw. Eigenschaften und Methoden von Objekten außerhalb der Hierarchie des Objektes document) **auch vom Netscape** unterstützt werden (falls das betreffende HTML-Element unterstützt wird).

Falls der Netscape auch Objekte des IE unterstützt, so kann es trotzdem sein, dass deren Eigenschaften und Methoden im NS bzw. IE z.T. **verschieden** implementiert sind (inklusive Syntax).

Empfehlung: Es sollte beim IE immer document.all kodiert werden, egal um welches HTML-Objekt es sich innerhalb des Dokumentes handelt, also auch bei der Referenzierung über den Wert des ID- bzw. NAME-Attributes. Anders formuliert: Es sollte immer der absolute Pfad ab Objekt document kodiert werden, wobei der Pfad bei Mehrfachverwendung in einem Zeiger liegen sollte (höhere Browser-Performance).

Erzeugung:
durch Browser
Syntax:
[var ZeigerAufFeld =] document.all
[var ZeigerAufFeldElement =] document.all**[**Index [, SubIndex]**]**

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

SubIndex optional
nur kodieren wenn Index ein String ist
Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()

Eigenschaften:

.length	Anzahl der Feldelemente also Feldlänge

Methoden:

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

4.3.2.2.4.2. HTML-Elemente übergreifende Verwaltung im HTML-Dokument

4.3.2.2.4.2.1. Verwaltung mehrerer HTML-Elemente gleicher Art im HTML-Dokument (Auswahl)

Die Verwaltung erfolgt z.T. über Collectionen als Zeigerfelder auf Instanzen gleichartiger HTTML-Elemente als Objekte.

4.3.2.2.4.2.1.1. window.document.anchors Collection des Internet Explorer (HTML-Element A (Anker))

Feld der Zeiger aller Anker im Dokument
Feldelementefolge laut HTML-Coding
Erzeugung:
durch den Browser

Beispiel für Anker:

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


```
        anker_text
</A>
```

Hinweis zu HREF= :

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

zu Anspringen eines Ankers: den Wert laut NAME ablegen in
window.location.hashohne vorgesetzten #
window.location.href mit vorgesetzen #

Syntax:

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

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

```
ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()

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

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

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

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

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

                                writeln("</BODY></HTML>");
                                close();
```


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

Eigenschaften:

.length	Anzahl der Feldelemente also Feldlänge

Methoden:

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

f4.3.2.2.4.2.1.2. window.document.applets Collection des Internt Explorer

Feld der Zeiger aller Applet-Objekte im Dokument
Elementefolge laut HTML-Coding
Applet muss alle Eigenschaften und Methoden, die im HTML-Dokument benutzt werden sollen, als **public** deklarieren

Erzeugung:

durch den Browser

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

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

Syntax:

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

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

```
        ZeigerAufFeldElement.eigenschaft
        ZeigerAufFeldElement.methode()

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

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


Eigenschaften:

.length	Anzahl der Feldelemente also Feldlänge

Methoden:

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

4.3.2.2.4.2.1.3. window.document.body.timeAll Collection des Internet Explorer

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

Erzeugung:
durch den Browser

Syntax:

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

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

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

ZeigerAufFeldElement.eigenschaft
ZeigerAufFeldElement.methode()

Eigenschaften:

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

Methoden:

.item()	Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern außer bei Formular mit <INPUT TYPE=image ...> da dafür die children-Collection verwendet werden muss !!!

4.3.2.2.4.2.1.4. window.document.embeds Collection des Internet Explorer (vermutlich auch im IE 6.x)

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

Achtung: Ab IE 6.x werden keine Plugins mehr unterstützt, damit ist die plugins Collection hinfällig. Inwieweit diese Collectionen noch implementiert ist oder bleibt, ist durch den Programmierer zu prüfen. Ab dem IE 6.x muss jedes Plugin, das in das Dokument eingebunden werden soll, durch ein ActiveX-Control implementiert werden. Plugins, wie sie beim Netscape exisitieren, laufen unter dem IE ab 6.x nicht mehr.
Es ist zu vermuten, dass im IE ab 6.x diese Collection nur eine Dummy-Funktion hat, also existiert, aber nie angefasst wird.

Diese Collection ist ein Alias für die plugins Collection **nur** bezüglich von Plugins (und nicht anderen einbettbaren Objekten), wobei letztere nur der Kompatibilität mit anderen plugin-fähigen Browsern dient. Alle eingebettenten Objekte haben in document.embeds ihren Zeiger (inklusive Plugins, Ausnahme: siehe tiefer).

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

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

Erzeugung:
durch Browser

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

```
<EMBED
        SRC="url"
        TYPE="mime_typ"             // z.B. "image/gif"
        PLUGINSPAGE="plugin_url"
        HEIGHT="hoehe_plugin_objekt"
        WIDTH="breite_plugin_objekt"
        NAME="ID_Embed"
        HIDDEN="true oder false"    // true so Objekt unsichtbar
```


```
>
    <PARAM Name="parameter_name"  VALUE=parameter_wert>
    .....
</EMBED>
```

Syntax:

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

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

```
ZeigerAufFeldElement.eingenschaft
ZeigerAufFeldElement.methode()

beim IE:
    document.all.ID_Embed.eigenschaft
    document.all.ID_Embed.methode()

beim NS:
    document.ID_Embed.eigenschaft
    document.ID_Embed.methode()
```

Eigenschaften beim Interner Explorer:

.length	Anzahl der Feldelemente also Feldlänge

Methoden beim Internet Explorer:

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

4.3.2.2.4.2.1.5. window.document.form.elements Collection des Internet Explorer

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

Syntax:

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

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

Eigenschaften:

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

Methoden:

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

4.3.2.2.4.2.1.6. window.document.forms Collection des Internet Explorer

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

Syntax:

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

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

Eigenschaften:

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

Methoden:

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

4.3.2.2.4.2.1.7. window.document.frames Collection des Internet Explorer

Feld der Zeiger aller Frames (beim IE inklusive der inline-Frames, also IFRAMES)
Elementefolge laut HTML-Coding
Element ist eine Referenz auf das Fenster mit Frame-Eigenschaften

Syntax:

Achtung: Für eine Referenz auf das Frame-Objekt ohne diese Collection ist immer document.all zu kodieren.
Bei Fenstererzeugung bitte jedem Fenster seinen eigenen Namen zuweisen (Name nicht doppelt verwenden)

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

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

Beispiel für Inhalte zweier Frames tauschen:

Kodierung im Dokument, dass die beiden Frames definiert !

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

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

Beispiel für Inhalte beliebig vieler Frames als Ring tauschen:

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

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

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

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

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

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

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

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

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

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

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

Eigenschaften:

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

Methoden:

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>
da dafür die children-Collection verwendet werden muss !!!

.namedItem() Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern

4.3.2.2.4.2.1.8. window.document.images Collection des Internet Explorer

Feld der Zeiger aller img Objekte
Elementefolge laut HTML-Koding
Achtung: Per new Image() erzeugte Bilder sind **nicht** Bestandteil der Collection !!
Diese Collection umfasst **nicht** das Objekt input.image, da für dieses Objekt die childrenCollection verwendet werden muss !

Syntax:

[var ZeigerAufFeld =] document.images
[var ZeigerAufFeldElement =] document.images **[**Index [, SubIndex] **]**

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

SubIndex optional
nur kodieren wenn Index ein String ist
Integer als Unterindex also Unterelement eines Elementes

Eigenschaften:

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

Methoden:

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

4.3.2.2.4.2.1.9. window.document.links Collection des Internet Explorer (HTML-Element mit HREF-Attribut)

Feld der Zeiger aller Objekte mit HREF-Eigenschaft (Attribut) sowie aller AREA-Objekte im Dokument,
wobei **alle** diese Objekte das Attribut NAME und oder ID besitzen **müssen**, um in dieser Collection referenziert zu sein.
Elementefolge laut HTML-Coding

Syntax:

[var ZeigerAufFeld =] document.links
[var ZeigerAufFeldElement =] document.links**[**Index**]**

Index	Integer ab 0 muss in **[ ]** kodiert sein

Eigenschaften:

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

Methoden:

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

4.3.2.2.4.2.1.10. map.areas Collection des Internet Explorer

Zeigerfeld aller area Objekte eines map Objektes (siehe auch dort)
Erzeugung eines Elementes kann nur erfolgen durch die Methoden
.createElement() mit anschliessendem .add()
oder .insertAdjacentHTML()

Syntax:

[var ZeigerAufFeld =] zeiger_auf_map_objekt.areas
[var ZeigerAufFeldElement =] zeiger_auf_map_objekt.areas**[**Index , SubIndex**]**

zeiger_auf_map_objekt	laut ID-Attribut
Index	Integer ab 0 oder String z.B. laut ID-Attribut muss in **[ ]** kodiert sein
SubIndex	optional Unterindex also Unterelement eines Elementes nur kodieren wenn Index ein String ist Integer
ZeigerAufFeld	ist null, wenn keine area Objekte vorhanden
ZeigerAufFeldElement	ist null, wenn Feldelement nicht vorhanden

Eigenschaften:

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

Methoden:

.add()	ein bereits erzeugtes Element einer Collection hinzufügen hinzufügen erst nach dem kompletten Laden des Dokumentes Element erzeugen per Methode .createElement()
.item()	Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern außer bei Formular mit <INPUT TYPE=image ...> da dafür die children-Collection verwendet werden muss !!!
.namedItem()	Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern

.remove() ein Element entfernen aus einer Collection
.tags() Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern
siehe tags Collection des DOM
.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

4.3.2.2.4.2.1.11. window.document.select.options Collection des Internet Explorer

Eine Option ist über dieses Feld der Zeiger aller Optionen anwählbar:
Die Zeiger aller Optionen werden in der Collection document.select.options gesammelt.
Ab IE 5.5 werden option Objekte zusätzlich auch über die Collection **document.all** referenzierbar.
Eine Option aus dem Feld löschen, erfolgt durch Zuweisung des Null-Zeiger (null). Zugleich wird automatisch das Feld kondensiert, also alle Lücken werden entfernt.
Zum Hinzufügen eines option Objektes müssen das select und option Objekt in einem **gemeinsamen** Fenster liegen.

Syntax:

[var ZeigerAufFeld =] document.zeiger_auf_select_objekt.options
[var ZeigerAufFeldElement =] document.zeiger_auf_select_objekt.options [Index **[**, SubIndex]**]**

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

SubIndex optional
nur kodieren wenn Index ein String ist
Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement
ist null, wenn Feldelement nicht vorhanden
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection **aus diesen** Script-Objekten geliefert

zeiger_auf_select_objekt z.B. laut ID-Attribut

Beispiel:

```
var Feld = document.all.tags("SELECT");

if (Feld.length>0)
{
	for (var i=0; i < Feld[0].options.length; i++)
	{
		alert(	"Element " + i
			+ " :  mit internem Text = "	+ Feld[0].options(i).text
			+ " und angezeigtem Wert = "	+ Feld[0].options(i).value);
	}
}
```

Eigenschaften:

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

Methoden:

.add() ein bereits erzeugtes Element einer Collection hinzufügen
hinzufügen erst nach dem kompletten Laden des Dokumentes
Element erzeugen per Methode .createElement()
.item() Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>
da dafür die children-Collection verwendet werden muss !!!
.namedItem() Referenz auf Eintrag (FeldElement) anhand des Namen
(analog zu ID oder NAME-Attribut) liefern
.remove() ein Element aus einer Collection entfernen
.tags() Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern
siehe tags Collection des DOM
.urns() Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

4.3.2.2.4.2.1.12. window.document.selection.controlrange Collection des Internet Explorer

Feld der Zeiger aller Control-Elemente, die in der **aktiven** Selektion per document.selection Objekt markiert wurden
nur für body Objekt
ein Objekt controlrange existiert nicht
ab IE 5.x

Syntax:

[var ZeigerAufFeld =] document.selection.controlrange
[var ZeigerAufFeldElement =] document.selection.controlrange **[**Index**]**

Index Integer ab 0
oder String Name oder ID des Elementes
(analog zu NAME und ID-Attribut)

muss in **[ ]** kodiert sein

ZeigerAufFeldElement
ist null, wenn Feldelement nicht vorhanden
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection
aus diesen Script-Objekten geliefert

Eigenschaften:

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

Methoden:

.add()	ein bereits erzeugtes Element einer Collection hinzufügen hinzufügen erst nach dem kompletten Laden des Dokumentes Element erzeugen per Methode .createElement()
.execCommand()	Kommando ausführen z.B. im aktuellen Dokument in aktueller Selektion im aktuellen Bereich erst nach dem kompletten Laden des Dokumentes zulässig Hinweis: Selektion = Markierung z.B. von Textbereich (Block) Control = Element zur Steuerung analog zum HTML-Element (Tag) Input-Control = Element mit Eingabeeigenschaft
.item()	Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern außer bei Formular mit <INPUT TYPE=image ...> da dafür die children-Collection verwendet werden muss !!!
.queryCommandEnabled()	prüfen ob Kommando ausführbar ist
.queryCommandIndeterm()	prüfen ob Kommando-Status bestimmbar ist oder nicht
.queryCommandState()	Status des aktuellen Kommando ermitteln: ob ausgeführt wurde oder nicht
.queryCommandSupported()	prüfen ob Kommando im aktuellen Bereich unterstützt wird
.queryCommandValue()	Wert eines Kommandos liefern
.remove()	ein Element aus einer Collection entfernen
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Selektion eines Textranges (Textbereich, Objekt textrange) oder ControlRange (Control-Elemente) nur unter Windows 32-Bit siehe Objekt document.selection Methoden .createTextRange() .createControlRange() und .createRange()

4.3.2.2.4.2.1.13. window.document.selection.textrange Collection des Internet Explorer

Feld der Zeiger aller textrange Objekte der **aktiven** Selektion
ab IE 5.5
Elternobjekte mit Textrange sind
body Objekt
button Objekt
textarea Objekt
input text Objekt
selection Objekt (nur wenn ein Text selektiert wurde (mit oder ohne HTML))
wobei diese weitere HTML-Elemente enthalten können, die ebenfalls Textbereiche besitzen können

Syntax:

[var ZeigerAufFeld =] document.selection.textrange
[var ZeigerAufFeldElement =] document.selection.textrange**[**Index [, SubIndex] **]**

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

SubIndex optional
nur kodieren wenn Index ein String ist
Integer als Unterindex also Unterelement eines Elementes

ZeigerAufFeldElement
ist null, wenn Feldelement nicht vorhanden
wenn gleichnamige Script-Objekte vorhanden, so wird ein Zeiger auf die Collection
aus diesen Script-Objekten geliefert

Eigenschaften:

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

Methoden:

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

4.3.2.2.4.2.1.14. window.document.TextRange Objekt des Internet Explorer (Textbereich im Dokument)

Objekt des gesamten Plaintextes eines Objektes (Textbereich, Textrange)
basiert auf dem Objekt TextNode
Elternobjekte mit Textrange sind alle Objekte, die eine der nachfolgend beschriebenen Methoden besitzen
z.B. body Objekt
button Objekt
textarea Objekt
input text Objekt
selection Objekt (nur wenn ein Text selektiert wurde (mit oder ohne HTML))
können weitere HTML-Elemente enthalten, die ebenfalls Textbereiche besitzen können

Textbereich kann in Texteile zerlegt sein: Jeder Teil ist ein Element mit Plaintext.
hat seinen Bereichrahmen (siehe Objekt TextRange.TextRectangle und Collection TextRange.TextRectangle)

Erzeugung:
in HMTL innerhalb der Tag-Begrenzer muss Text kodiert sein
z.B. leerer DIV hat keinen Textbereich
kann nachträglich einen Textbereich per Script erhalten

Beispiel:
```
<HTML>
<BODY>
<H1>Hallo</H1>
<CENTER><H2> und willommen ! <H2></CENTER>
</BODY>
</HTML>
```

Textrange ist erzeugt worden im BODY-Teil
und enthält nur Plain-Text also
"Hallo und willkommen !"
hat genau einen Zeiger auf die Textrange-Element

Textrange-Element ist z.B. das Wort "Hallo" des H1-Elementes im Container BODY

per Script:

.createTextRange() Textbereich erzeugen
Syntax:
[var Zeiger =] object.createTextRange()

Zeiger auf Range
null wenn TextRange nicht erzeugbar

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
    var FeldAllerButtonElemente coll = document.all.tags("BUTTON");

    if (    (FeldAllerButtonElemente !=null )
         && (FeldAllerButtonElemente.length>0)
       )
    {
        var ZeigerAufRange = FeldAllerButtonElemente[0].createTextRange();
        ZeigerAufRange.text = "Clicked";
    }
</SCRIPT>
```

Beispiel 2:

```
function ErzeugeUndSelektiereTextRange()
{
    var TextBereich = document.body.createTextRange();
    TextBereich.findText("Testtext");
    TextBereich.select();
}
```

.createRange() Zeiger auf einen Universal-Bereich erzeugen per document.selection Objekt des Dokumentes
Universal-Bereich kann enthalten:
Text (document.selection.textrange Collection
textrange Objekt)
oder Control-Element(e) (document.selection.controlRange Collection)
siehe auch Methoden .createControlRange() .createTextRange() und .createRangeCollection()
Eigenschaft .type
Syntax:
[var Zeiger =] document.selection.createRange()

.createRangeCollection() document.selection.textrange Collection erzeugen per document.selection Objekt des Dokumentes
nur wenn der Browser multible Selektion unterstützt, so mehr als 1 Feld-Element textrange Objekt
vorhanden bei Mehrfach-Selektion durch User
siehe auch textrange Objekt

Methoden .createRange() .createControlRange() und .createTextRange()

Syntax:

[var Zeiger =] = document.selection.createRangeCollection()

Eigenschaften:

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

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

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

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

.htmlText HMTL-Text im Textbereich
nicht den Plaintext-Anteil liefern
per textrange Objekt
nur unter Windows 32-Bit

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.text Plain-Text im Textbereich
per textrange Objekt
Dokument muss komplett geladen sein
nur unter Windows 32-Bit

Methoden:

.collapse() Textbereich-Zeichen-Zeiger auf Anfang oder Ende des Textbereiches setzten
Hinweis: Zeiger nur auf Plain-Text-Zeichen
Bsp.: <BODY><P><B>abc
Zeiger kann wandern von VOR a bis HINTER c
VOR a entspricht Start des Bereiches
mit Zeigerwert 0
HINTER c entspricht Ende des Bereiches
mit Zeigerwert 3
per textrange Objekt
nur unter Windows 32-Bit

.compareEndPoints() Vergleich der Textbereich-Zeichen-Zeiger von 2 Textbereichen
Hinweis: Zeiger nur auf Plain-Text-Zeichen
Bsp.: <BODY><P><B>abc
Zeiger kann wandern von VOR a bis HINTER c
VOR a entspricht Start des Bereiches
mit Zeigerwert 0
HINTER c entspricht Ende des Bereiches
mit Zeigerwert 3
per textrange Objekt
nur unter Windows 32-Bit

.duplicate() Zeiger auf eine Duplikat-Instanz eines Textbereiches liefern
per textrange Objekt
nur unter Windows 32-Bit
Prüfung eines Textbereichen auf Kopie eines anderen Textbereiches per Methode .inRange()
bzw. .isEqual()

.execCommand() Kommando ausführen z.B. im aktuellen Dokument
in aktueller Selektion
im aktuellen Bereich
erst nach dem kompletten Laden des Dokumentes zulässig
Hinweis: Selektion = Markierung z.B. von Textbereich (Block)
Control = Element zur Steuerung analog zum HTML-Element (Tag)
Input-Control = Element mit Eingabeeigenschaft

.expand() Plain-Text als Teil eines Textbereiches derart ausdehen, dass er komplett die Dimension des Elternobjektes (Container-Objektes) einnimt
per textrange Objekt
nur unter Windows 32-Bit

.findText() Text im gesamten Textbereich suchen
per textrange Objekt
nur unter Windows 32-Bit
für Nutzung einer Textmarke siehe Methoden .getBookmark() und .moveToBookmark()

.getBookmark() Textmarke (Bookmark) im Textbereich setzen
Textmarke kann mit Methode .moveToBookmark() anpositioniert werden
per textrange Objekt

	nur unter Windows 32-Bit
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle
.inRange()	prüfen ob 2 Textbereiche sich einschliessen z.B. bei verschachtelten DIV's per textrange Objekt nur unter Windows 32-Bit
.isEqual()	Auf Identität zweier Textbereiche prüfen per textrange Objekt
.move()	Textlbereich-Zeichen-Zeiger bewegen bezüglich aktueller Position im Textbereich per textrange Objekt nur unter Windows 32-Bit
.moveEnd()	Textbereich-Ende neu setzen (für den Textbereich-Zeiger) bezüglich aktueller Position im Textbereich per textrange Objekt nur unter Windows 32-Bit
.moveStart()	Textbereich-Anfang neu setzen (für den Textbereich-Zeiger) bezüglich aktueller Position im Textbereich per textrange Objekt nur unter Windows 32-Bit
.moveToBookmark()	Textbereich-Zeichen-Zeiger auf eine Textmarke (Bookmark) im Textbreich positionieren Textmarke wurde gesetzt per Methode .getBookmark() per textrange Objekt nur unter Windows 32-Bit
.moveToElementText()	Textbereich in ein Element/Objekt bewegen, das Text enthalten darf per textrange Objekt nur unter Windows 32-Bit
.moveToPoint()	Inhalt des Textbereiches um eine Pixelspanne verschieben (Offset) relativ zur linken oberen Fensterecke nach Verschiebung kann Textbereich leer sein per textrange Objekt nur unter Windows 32-Bit
.parentElement()	Zeiger auf das Elternelement (Container) des Textbereiches liefern Hinweis bei Elementverschachtelung: es wird das direkt um den Textbereich liegende Element referenziert per textrange Objekt nur unter Windows 32-Bit
.pasteHTML()	Textbereich-Inhalt durch Text ersetzen oder leeren Textbereich füllen Text kann auch HTML enthalten Tabelle nur mit kompletten HTML-Code einfügbar, also z.B. keine einzelne Zelle Achtung: Wenn HTML-Code im Text enthalten ist, muss das Elternobjekt auch diesen ansich unterstützen Bsp.: textArea erlaubt kein HTML-Code HTML-Code wird geparst per textrange Objekt nur unter Windows 32-Bit
.queryCommandEnabled()	prüfen ob Kommando ausführbar ist
.queryCommandIndeterm()	prüfen ob Kommando-Status bestimmbar ist oder nicht
.queryCommandState()	Status des aktuellen Kommando ermitteln: ob ausgeführt wurde oder nicht
.queryCommandSupported()	prüfen ob Kommando im aktuellen Bereich unterstützt wird
.queryCommandValue()	Wert eines Kommandos liefern
.scrollIntoView()	Objekt derart scrollen, dass es im Fenster für User sichtbar wird Objekt muss an sich schon renderbar sein
.select()	Objekt selektieren z.B. Textbereich: wird hervorgehoben ControlRange: es wird Rechteck-Rahmen erzeugt nur unter Windows 32-Bit
.setEndPoint()	Textbereichanfang bzw. –ende von 2 Textbereichen synchronisieren per textrange Objekt nur unter Windows 32-Bit

4.3.2.2.4.2.1.14.1. window.document.TextRange.TextRectangle Collection des Internet Explorer

ab IE 5.x
Feld der TextRectangle-Objekte im HTML-Element (Objekt) mit Plain-Text

TextRectangle-Objekt: Rahmen der Positionierung einer einzelnen Textzeile im Plain-Text
Positionierung bezüglich Koordinatensystem des HTML-Dokumentes

Beispiel: Ein DIV enthält Plain-Text:
Jede im Browser dargestellte Zeile (auch jedes einzelne <BR>) besitzt einen eigenes Rechteck als
Rahmen der Positionierung innerhalb des DIV.
Damit lassen sich Textelemente per Rahmen innerhalb des Dokumentes positionieren.

Achtung: Nach einer Änderung der Fensterdimension (resize) muss die Collection neu erzeugt werden (ist zu programmieren !).

Syntax:

[var FeldZeiger =] zeiger_auf_textrange.getClientRects();

[var FeldElementZeiger =] FeldZeiger**[**Index**]**;

Index: Integer und ab 0
muss in **[ ]** kodiert sein

Zugriff:

FeldZeiger.eigenschaft
FeldZeiger.methode()

Eigenschaften:

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

Methoden:

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>
da dafür die children-Collection verwendet werden muss !!!

.namedItem() Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern

4.3.2.2.4.2.1.14.2. window.document.TextRange.TextRectangle Objekt des Internet Explorer

ab IE 5.x
TextRectangle-Objekt: Rahmen der Positionierung einer einzelnen Textzeile im Plain-Text (Positionierung bezüglich Koordinantensystem des HTML-Dokumentes)
instanziert als Element der Collection TextRange.TextRectangle

Beispiel: Ein DIV enthält Plain-Text:
Jede im Browser dargestellte Zeile (auch jedes einzelne <BR>) besitzt einen eigenes Rechteck als Rahmen der Positionierung innerhalb des DIV.
Damit lassen sich Textelemente per Rahmen innerhalb des Dokumentes positionieren.

Achtung: Nach einer Änderung der Fensterdimension (resize) muss die Collection neu erzeugt werden (ist zu programmieren !).

Syntax:

[var Zeiger =] zeiger_auf_textrectangle.getBoundingClientRect();

zeiger_auf_textrectangle ist Element der Collection TextRange.TextRectangle

[var Zeiger =] zeiger_auf_textrectangle_collection**[**Index**]**.getBoundingClientRect();

Index Integer, ab 0
muss in **[ ]** kodiert

Beispiel:

```
<HEAD>
<SCRIPT>
    var ZeigerAufTextRectangleCollection;
    var Index =0;

    function Anzeigen(Zeiger)  // Zeiger auf den einzigsten DIV mit Text also mit Textbereich
    {
        // aktuelle Collection der Rectangle von Div0  referenzieren:
        //      Das ist nötig nach resize des Fensters, da resize leider nicht automatisch erkannt
        //          wird !
        ZeigerAufTextRectangleCollection = Zeiger.getClientRects();

        // Anzahl der Elemente ermitteln, also Anzahl der im Browser dargestellte Zeilen,
        //                  wobei es egal ist, ob diese per <BR> erzeugt wurden oder nicht
        //          Hinweis: Zeilenlänge hängt auch von der Fensterbreite ab
        //                  (automatischer Umbruch)
        // Jede Zeile besitzt ihr eigenes Rectangle
        AnzahlTextRectangle     = ZeigerAufTextRectangleCollection.length;

        // aktuellen Index prüfen ob letzte Zeile bereits erreicht wurde
        if (Index > AnzahlTextRectangle  -1)  // Index ab 1, Anzahl ab 1
        {
            // es wurde die letzte Zeile erreicht

            // Div2 unsichtbar machen also entfärben
            //   Hinweis: Div2 liegt auf dem Rectangle von Div0
            ID_Div2.style.display="none";

                    // rücksetzen des Index, also mit erster Zeile weitermachen
```


```
                                        Index = 0;
                                }

                                // Rechteck der aktuellen Zeile ermitteln unter Beachtung eines eventuellen Scrollens
                                var PosRechts       = ZeigerAufTextRectangleCollection [Index].right + ID_Body.scrollLeft;
                                var PosLinks        = ZeigerAufTextRectangleCollection [Index].left  + ID_Body.scrollLeft;
                                var PosOben1        = ZeigerAufTextRectangleCollection [Index].top   + ID_Body.scrollTop;

                                // und Div1 auf die Zeile positionieren, also Zeile einfärben
                                ID_Div1.style.top     = PosOben1;
                                ID_Div1.style.width   = (PosRechts  - PosLinks) - 5;
                                ID_Div1.style.display = 'inline';

                                // aktuelle Position des Rechteckes von DIV0 ermitteln unter Beachtung eines eventuellen
                                //          Scrollens
                                PosRechts           = Zeiger.getBoundingClientRect().right  + ID_Body.scrollLeft;
                                PosLinks            = Zeiger.getBoundingClientRect().left   + ID_Body.scrollLeft;
                                var PosOben2        = Zeiger.getBoundingClientRect().top    + ID_Body.scrollTop;

                                // und Div2 überlagern positionieren
                                ID_Div2.style.top     = PosOben2;
                                ID_Div2.style.width   = (PosRechts  - PosLinks) - 5;
                                ID_Div2.style.height  = PosOben1 - PosOben2;

                                // aber nur einfärben, wenn mindestens 1 Zeile bereits eingefärbt wurde
                                if (Index  > 0){ ID_Div2.style.display = 'inline';}

                                // Rectangle der nächsten Zeile einstellen
                                Index++;
                        }
                </SCRIPT>
                </HEAD>
                <BODY ID="ID_Body">
                        <DIV    ID="ID_Div0"
                                onclick=" Anzeigen(this)"
                        >
                                klicke
                                <BR>
                                ABCDEFGHIJKLMNOPQRSTUVWXYZ
                                12345678901234567890123456789012345678901234567890
                                ABCDEFGHIJKLMNOPQRSTUVWXYZ
                                12345678901234567890123456789012345678901234567890
                                ABCDEFGHIJKLMNOPQRSTUVWXYZ
                                12345678901234567890123456789012345678901234567890
                                ABCDEFGHIJKLMNOPQRSTUVWXYZ
                                12345678901234567890123456789012345678901234567890
                        </DIV>
                        <DIV    ID="ID_Div1"
                                STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:yellow; display:none"
                        >
                        </DIV>
                        <DIV    ID="ID_Div2"
                                STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:beige; display:none"
                        >
                        </DIV>
                </BODY>
```

Zugriff:

Zeiger.eigenschaft

Eigenschaften:

sind les- und schreibbar

.bottom	untere Pixelposition des Rechteckes um ein Objekt
.left	linke Pixelposition des Rechteckes um ein Objekt
.right	rechte Pixelposition des Rechteckes um ein Objekt
.top	obere Pixelposition des Rechteckes um ein Objekt

Methoden:

keine

4.3.2.2.4.2.2. Verwaltung mehrerer HTML-Elemente gleicher oder verschiedener Arten im HTML-Dokument

4.3.2.2.4.2.2.1. allgemeine HTML-Element bezogene Verwaltung

Nachfolgend werden Collectionen beschrieben, die der **allgemeinen HTML-Element-bezogenen** Verwaltung dienen.