window.document.body Objekt des Internet Explorer

Rumpf des HTML-Dokumentes
Im Dokument gibt es genau 1 Rumpf also genau 1 Body Objekt

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

Fakt 1:

Die **ERSTE** Erzeugung eines HTML-Tags bewirkt das **automatische Öffnen eines neuen** HTML-Dokumentes, wenn das aktuelle Dokument **bereits komplett geladen**, also das Ereignis onload **bereits** ausgelöst wurde. Letzteres ist immer dann der Fall, wenn der BODY-Teil des Dokumentes komplett 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.

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

Fakt 2:

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

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

Erzeugung in HTML:
Beispiel:

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

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

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

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

Zugriff:

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

Beispiel 1 für Ausdruck des aktuellen Dokumentes:

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

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

```
<BODY LEFTMARGIN=0 TOPMARGIN=0>
```

Hinweis: Beim Nescape muss kodiert werden <BODY MARGINWIDTH=0 MARGINHEIGH=0>

Beispiel 3 für Sound mit Sekundenanzeige:

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

```
var PixelBreiteProBalkenErweiterung = 10;

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

// ++++++++++++ Routinen der Sekundenzählung
var SekundenZahler              = 0;
var 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>
```

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

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

.innerHTML ruft intern .createElement() auf

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

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

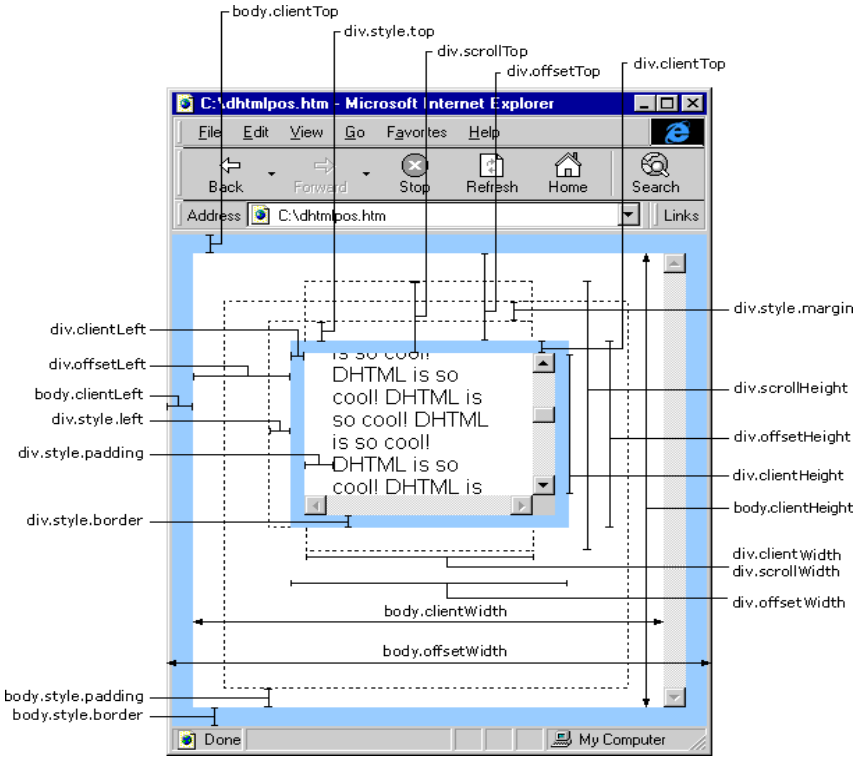
1. Element-Bezeichner
2. CLASS-Attribut mit Bezeichner aus Klassendeklaration im HEAD
3. ID-Attribut
4. STYLE-Attribut mit Style-Werten (nicht mit Bezeichner aus Style-Deklaration im HEAD)

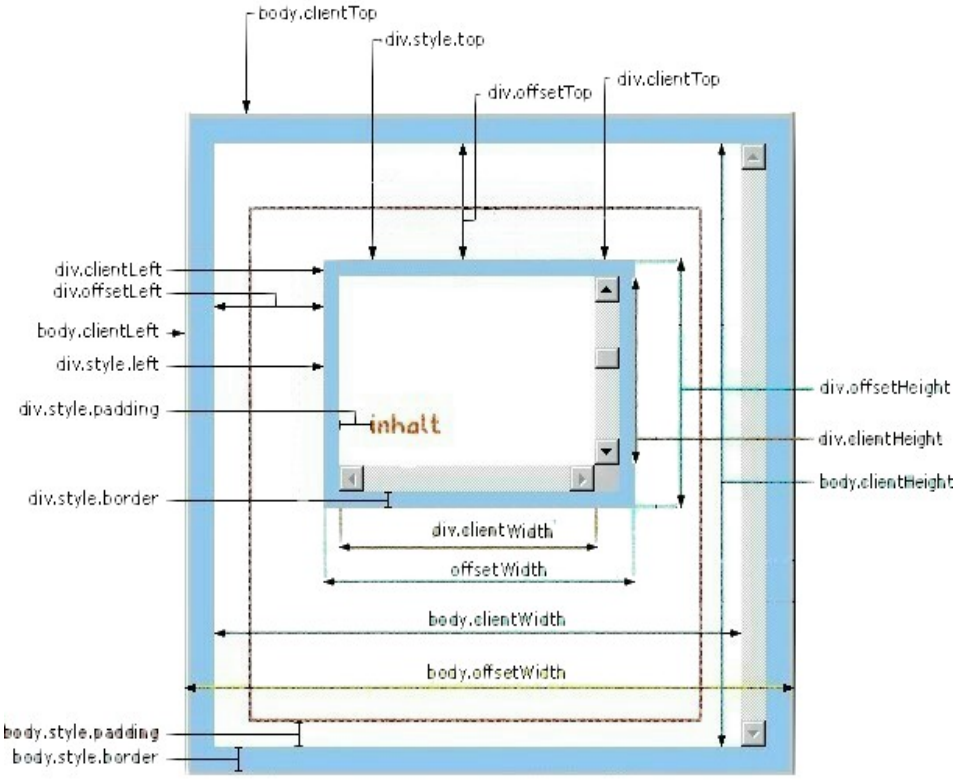
Es gilt: Wenn gleiche Bezeichner verwendet, so nur Werte des **zuletzt** geparsten Bezeichners verwendet !
Die CLASS-Deklaration aus dem HEAD des Dokumentes wird wertmäßig durch die Style-Deklaration per Attribut des HTML-Elementes überschrieben, wenn gleiche Style-Eigenschaften betroffen sind (ansonsten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

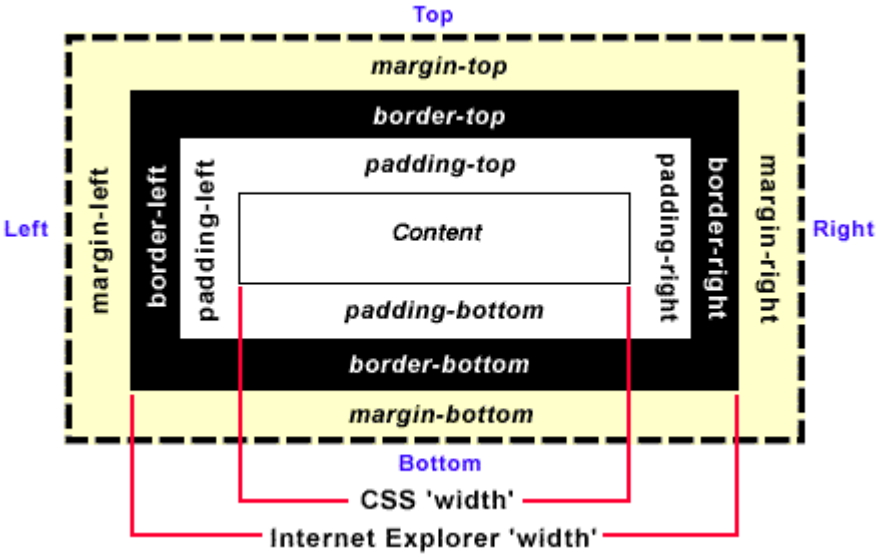
Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:

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

Eigenschaften des IE als Grafiken







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

Eigenschaften

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

.aLink Farbe eines ALinks

ATOMICSELECTION Selektierbarkeit einstellen

.background Hintergrundbild

.bgColor deprecated und durch STYLE-Attribut zu ersetzen
Hintergrundfarbe des Objektes bzw. Dokumentes

.bgProperties Scroll-Eigenschaften des Hintergrundbildes beim Dokumentscrollen

.blockDirection Umfluss um ein Objekt

.bottomMargin Bottom Margin in Pixel

.className Klassenreferenz

.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt
ohne Rahmen
ohne Scrollbalken

.clientLeft Abstand in Pixel zum linken Rand des Fensters

.clientTop Abstand in Pixel zum oberen Rand des Fensters

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

.contentEditable Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat)
Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.

.dir Umflussrichtung

.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.hideFocus Focussierbarkeit

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

.innerHTML Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches
.lang Sprache für Anzeige von Sonderzeichen etc.
.language Sprache für Script festlegen
.leftMargin Left Margin in Pixel des Dokumentes
.link Farbe eines Links
.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType Knotentyp laut attributes Collection
.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)
.noWrap Wortumbruch einstellen
.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (offsetParent)
.offsetParent Referenz der Eltern
für Nutzung von offsetHeigt, offserLeft, offsetTop und offsetWidth
.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offestParent)
.offsetWidth X-Koordinate der rechen unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (offestParent)
.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.ownerDocument Referenz auf das document Objekt zu dem der Koten gehört, also in dem der Knoten erzeugt wurde
.parentElement Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit Textbereich des Elternobjektes referenzieren
.previousSibling Referenz auf das Vorgängerkind
.readyState aktueller Status des Objektes beim Füllen des Objektes mit Daten
.rightMargin Right Margin in Pixel des Documentes
.scopeName Namensraum laut XMLNS-Attribut
.scroll Scrollbar-Anzeige ein/aus
vor IE 6.x nur BODY-Objekt
ab IE 6.x alle HTML-Elemente wenn DOCTYPE im Kopf des Dokumentes kodiert
.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex Index des Objektes in der Collection document.all
STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tabIndex Index des Elementes in der Tab-Tasten-Folge
für Anspringen des Dokumentes
Anspringen verbunden mit Focus erhalten
--> Ereignisse werden ausgelöst !!
unter IE 5.x
onblur, onfocus
ab IE 5.x
onblur, onfocus, onkeydown, onkeypress, onkeyup
Anspringen default per TAB-Taste
für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA

Anspringen default nicht per TAB-Taste
für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName Tag-Bezeichner des Objektes
.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.text Textfarbe (Vordergrundfarbe)
.timeStartRule deprecated
Startpunkt der Timeline
.title Tooltip-Text bei Mouse over über Objekt
.topMargin Top Margin in Pixel des Dokumentes
.uniqueID durch den Browser automatisch-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
UNSELECTABLE Selektierbarkeit z.B. mit Maus
wird nicht an Kinder vererbt

bei Wechsel: vorhergehende vorhandene Selektion wird nicht verändert
.vLink Farbe eine VLINK

Methoden

.addBehavior() DHTML-Verhaltenseigenschaft einem Element hinzufügen
Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig).
ab IE 5.x bis unter IE 5.5

.appendChild() Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern
DOM wird geändert
Zeiger wird zugleich immer am Ende der Collection childNodes angehangen
Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehangen wird UND Ende-Tag (falls vorhanden) des Knoten geparst wurde

.applyElement() Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern
DOM wird geändert
Element kann selbst Kinder haben
Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparst wurde
Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !

.attachEvent() Einschalten des Registrieren eines Events durch Eventhandler
Hinweis: Abschalten mit Methode .detachEvent()
Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider NICHT verkettet sondern in **Zufallsfolge**, es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)

.blur() Element den Focus wegnehmen und Event onblur auslösen
Der Focus wird nicht automatisch auf irgend ein anderes Element gesetzt !
vor IE 5.0 TABINDEX-Attribut muss kodiert sein
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein

.clearAttributes() alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute
Script-erzeugte Attribute nicht entfernbar
DOM wird geändert

.click() simuliert einen Klick auf das Element und löst onclick-Event aus
manipuliert nicht den Focus

.cloneNode() Objekt klonen und Referenz des erzeugten Klone liefern
DOM wird nicht geändert, da Klone nicht in DOM eingebunden wird (reines Neu-Instanzieren eines DOM-Objektes im Hauptspeicher außerhalb des DOM)

.componentFromPoint() Layout-Komponente eines Objektes ermitteln, die an einer Koordinate liegt
auch für CSS-Layout
onmouseover-Event hat **nicht die Pixelgenauigkeit** wie die Angaben der Methode .componentFromPoint()
also wenn Event erzeugt, muss die Maus noch lange nicht genau die obengenannte Pixelpos erreicht haben
Overbereich der Maus ist mehr als 1 Pixel gross
beachte Einstellungen der Maus zur Cursorgeschwindigkeit etc.

.contains() prüfen ob Element innerhalb eines Elementes liegt, also ob das innere, eingeschlossene Element Eltern (Eltern-Objekt, Container) hat und somit ein Kind-Objekt ist
DOM nicht geändert

.createControlRange() Selektionsbereich erzeugen
es kann nur genau 1 Range pro Zeitpunkt existieren: wenn einer vorhanden, so diesen überschreiben

.createTextRange() Textbereich erzeugen

.detachEvent() Abschalten des Registrieren eines Events durch Eventhandler
wobei Registrierung mit Methode .attachEvent() aktiviert wurde
Abschalten = ordnet dem Window-Objekt das Event laut Parameter event_bezeichner zu, das **nicht** behandelt werden soll, falls es für das Window-Objekt auftritt (also Standardbehandlung aktiv)

.doScroll() Click auf Scrollbar simulieren
Achung: Scrollelemente müssen bereits vorhanden sein !

.dragDrop() prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element

.fireEvent() ein Event auslösen

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

.getAdjacentText() Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann
Text kann HTML-Tags enthalten, muss aber nicht
DOM nicht geändert

.getAttribute() Wert eines per HTML erzeugten Attributes liefern
DOM nicht geändert

.getAttributeNode() Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft.
Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht
Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt
Wert des Attributes wird somit über die Referenz laut DOM erreichbar
DOM nicht geändert

.getBoundingClientRect() Referenz auf TextRectangle-Objekt im Element holen

.getClientRects() Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster
Feld mit Index als Integer ab 0
pro Eintrag ein Rectangle

.getElementsByTagName() Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc.
Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection)
Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst !
Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementsById()
Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
DOM nicht geändert

.getExpression() Wert einer Style-Eigenschaft anhand des Ausdruckes berechnen und liefern
Style-Eigenschaft ist per Methoden expression() oder setExpression() zu definieren
DOM wird nicht verändert (nur Werteveränderung), aber das Dokument-Layout (nach dem eventuellen expliziten Dokument-Refresh)

.hasChildNodes() prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
DOM nicht geändert

.insertAdjacentElement() Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann
wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM
nur nach dem kompletten Laden des Dokumentes möglich
DOM wird geändert

.insertAdjacentHTML() HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann
nur nach dem kompletten Laden des Dokumentes möglich
HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen **nicht** ausgeführt
eingefügter Code wird **nur** dann sofort geparst und ausgeführt, wenn syntaktisch korrekt ist
bei Script-Code: <SCRIPT DEFER .....> muss kodiert werden
DOM wird geändert

.insertAdjacentText() Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann
nur nach dem kompletten Laden des Dokumentes
DOM wird geändert

.insertBefore() Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern
einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein
Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten
Sichtbarkeit erst wenn Ende-Tag geparst wurde
DOM wird geändert

.mergeAttributes() alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen
Attribute sind: HTML
Events
Styles
ab IE 5.01 auch ID, NAME
Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
DOM wird geändert

.normalize() Normalisierung des DOM zur Erreichung einer konsistenten Struktur
Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen

.pause() deprecated
auf Timeline pausieren

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

.removeAttribute() entfernen eines per HTML erzeugten Attributes
Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute!
per Methode .createAttribute() erzeugte Attribute werden nicht erfasst
DOM wird geändert

.removeAttributeNode() entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
DOM wird geändert

.removeBehavior() per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
DOM wird geändert

.removeChild() Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
Sichtbarkeit erst, wenn Ende-Tag geparst wurde, also das Dokument neu geladen wurde
DOM wird geändert

.removeExpression() Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient.
Ausdruck muss mit der Methode .setExpression() gesetzt worden sein
DOM wird nicht geändert

.removeNode()	Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern Sichtbarkeit erst wenn Ende-Tag geparst wurde DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.replaceChild()	Kind-Objekt ersetzen durch ein Objekt ersetzende Objekt muss per Methode .createElement() erzeugt worden sein Sichtbarkeit erst wenn Ende-Tag geparst wurde DOM wird geändert
.replaceNode()	Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern sichtbar erst mit parsen des Endetags DOM wird geändert
.resume()	deprecated Pause auf der Timeline beenden
.setActive()	Objekt für die Eventdurchreichung aktivieren aber ohne es zu fokussieren und ohne es scrollbar zu machen
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt DOM wird nur bei Erzeugung geändert
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparst DOM wird geändert

Events

onactivate Fires when the object is set as the active element.
onafterprint Fires on the object immediately after its associated document prints or previews for printing.
onbeforeactivate Fires immediately before the object is set as the active element.
onbeforecut Fires on the source object before the selection is deleted from the document.
onbeforedeactivate Fires immediately before the activeElement is changed from the current object to another object in the parent document.
onbeforeeditfocus Fires before an object contained in an editable element enters a UI-activated state or when an editable container object is control selected.
onbeforepaste Fires on the target object before the selection is pasted from the system clipboard to the document.
onbeforeprint Fires on the object before its associated document prints or previews for printing.
onbeforeunload Fires prior to a page being unloaded.
onclick Fires when the user clicks the left mouse button on the object.
oncontextmenu Fires when the user clicks the right mouse button in the client area, opening the context menu.
oncontrolselect Fires when the user is about to make a control selection of the object.
oncut Fires on the source element when the object or selection is removed from the document and added to the system clipboard.
ondblclick Fires when the user double-clicks the object.
ondeactivate Fires when the activeElement is changed from the current object to another object in the parent document.
ondrag Fires on the source object continuously during a drag operation.
ondragend Fires on the source object when the user releases the mouse at the close of a drag operation.
ondragenter Fires on the target element when the user drags the object to a valid drop target.
ondragleave Fires on the target object when the user moves the mouse out of a valid drop target during a drag operation.
ondragover Fires on the target element continuously while the user drags the object over a valid drop target.
ondragstart Fires on the source object when the user starts to drag a text selection or selected object.
ondrop Fires on the target object when the mouse button is released during a drag-and-drop operation.
onfilterchange Fires when a visual filter changes state or completes a transition.
onfocusin Fires for an element just prior to setting focus on that element.
onfocusout Fires for the current element with focus immediately after moving focus to another element.
onkeydown Fires when the user presses a key.
onkeypress Fires when the user presses an alphanumeric key.
onkeyup Fires when the user releases a key.
onload Fires immediately after the browser loads the object.
onlosecapture Fires when the object loses the mouse capture.
onmousedown Fires when the user clicks the object with either mouse button.
onmouseenter Fires when the user moves the mouse pointer into the object.

onmouseleave Fires when the user moves the mouse pointer outside the boundaries of the object.
onmousemove Fires when the user moves the mouse over the object.
onmouseout Fires when the user moves the mouse pointer outside the boundaries of the object.
onmouseover Fires when the user moves the mouse pointer into the object.
onmouseup Fires when the user releases a mouse button while the mouse is over the object.
onmousewheel Fires when the wheel button is rotated.
onmove Fires when the object moves.
onmoveend Fires when the object stops moving.
onmovestart Fires when the object starts to move.
onpaste Fires on the target object when the user pastes data, transferring the data from the system clipboard to the document.
onpropertychange Fires when a property changes on the object.
onreadystatechange Fires when the state of the object has changed.
onresizeend Fires when the user finishes changing the dimensions of the object in a control selection.
onresizestart Fires when the user begins to change the dimensions of the object in a control selection.
onscroll Fires when the user repositions the scroll box in the scroll bar on the object.
onselect Fires when the current selection changes.
onselectstart Fires when the object is being selected.
onunload Fires immediately before the object is unloaded.