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:**

| | |
|---|---|
| .innerHTML | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag |
| .innerText | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag |
| .length | Anzahl der Feldelemente also Feldlänge z.B. bei Collection |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag<br>wirksam mit parsen des Ende-Tag<br>nur nach kompletten Einlesen des Dokumentes nutzbar |
| .outerText | Referenz auf den gesamten Plain-Text im Objekt<br>nur nach kompletten einlesen des Dokumentes nutzbar |
| .sourceIndex | Index des Objektes in der Collection document.all |

**Methoden:**

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

#### 4.3.2.2.4.3.42. table Objekt des Internet Explorer

Objekt einer Tabelle (HTML-Element TABLE)

##### *4.3.2.2.4.3.42.1. Erzeugung der Tabelle*

*4.3.2.2.4.3.42.1.1. Erzeugung in HTML*

Die Tabelle kann folgende Tabellenelemente (Tags) besitzen:
CAPTION,
COL,
COLGROUP,
TBODY,
TD,
TFOOT,
TH,
THEAD
TR

wobei in der Tabelle maximal
1 THEAD
1 TFOOT
1 CAPTION (Überschrift)
auftauchen kann

TBODY-Tag nicht kodiert werden muss, wenn keine Fehlzuordung zu THEAD **und** kein TFOOT möglich ist

Beispiel:

```
<TABLE BORDER=1 WIDTH=80% BGCOLOR="gray">
<THEAD BGCOLOR="blue">
    <TR>
        <TH>Titel Spalte 1</TH>
        <TH>Titel Spalte 2</TH>
    </TR>
</THEAD>
<TBODY BGCOLOR="yellow">
    <TR>
        <TD>Zeile 1, Spalte 1 </TD>
        <TD>Zeile 1, Spalte 2 </TD>
    </TR>
    <TR>
        <TD>Zeile 2, Spalte 1</TD>
        <TD>zeile 2, Spalte 2</TD>
    </TR>
</TBODY>
```

```
<TFOOT BGCOLOR="green">
        <TR>
                <TD COLSPAN="4">TFOOT</TD>
        </TR>
</TFOOT>
<CAPTION VALIGN="BOTTOM" STYLE="font-size=10;">
        Caption
</CAPTION>
</TABLE>
```

Tabelle wird erst angezeigt, wenn das Dokument komplett geladen ist und das Event onload ausgelöst wurden.

*4.3.2.2.4.3.42.1.2. Erzeugung in JScript*

IE 4.x
Die Erzeugung und Manipulation der Tabelle sind in Script erst möglich, wenn das Dokument komplett geladen und das Event onload ausgelöst wurden. Veränderungen der Tabellenelemente per Style werden sofort sichtbar. Veränderungen der Tabelle bezüglich Art und Anzahl der Tabellenelemente werden z.T. erst nach Aufruf der Methode .refresh() sichtbar.

#### *4.3.2.2.4.3.42.2. Daten der Tabelle*

*4.3.2.2.4.3.42.2.1. Datenbereitsstellung*

*4.3.2.2.4.3.42.2.1.1. in HTML*

Die Daten sind bereits in der Kodierung der Tabelle implementiert und damit von statischer Natur.
Die Berechnung des Layouts der Tabelle erfolgt einmalig.
Alternativ erfolgt die Erzeugung eines HTML-Codes mit Daten z.B. per PHP auf Basis eines Datenbankservers.

*4.3.2.2.4.3.42.2.1.2. in JScript*

Die Daten sind während und nach der Erzeugung der Tabelle manipulierbar und damit von dynamischer Natur.
Die Berechnung des Layouts der Tabelle erfolgt automatisch .

*4.3.2.2.4.3.42.2.1.3. per Active-X-Control*

Für den Internet Explorer existiert eine dynamische Datenbereitstellung über das ActiveX-Control TDC (Tabular Data Container) mit dem Class-ID "clsid:333C7BC4-460F-11D0-BC04-0080C7055A83" .

TDC liest eine Textdatei, deren Inhalt bestimmten Regeln enstprechen muss, und stellt die Daten aus der Textdatei in der Tabelle dar. Die Verwaltung der Textdatei, also deren Daten, ist sehr einfach.

Nachteilig beim TDC ist, dass die Textdatei im Browser-Cache liegen muss und damit vom User manipulierbar ist. TDC unterstützt keine Verschlüsselung der Text-Datei.

TDC wird an anderer Stelle in dieser Dokumentation beschrieben. Leider ist TDC kein HTML-Standard.

Das Tabellen-Objekt besitzt bereits Eigenschaften und Methoden, die vom TDC benutzt werden.

*4.3.2.2.4.3.42.2.2. Dateneinbindung*

*4.3.2.2.4.3.42.2.2.1. in HTML*

Die Daten sind bereits in der Kodierung der Tabelle implementiert und damit einmalig eingebunden. Alternativ erfolgt die Erzeugung eines HTML-Codes z.B. per PHP auf Basis eines Datenbankservers.

*4.3.2.2.4.3.42.2.2.2. in JScript*

Die Daten sind während und nach der Erzeugung der Tabelle manipulierbar.
Die Berechnung des Layouts der Tabelle erfolgt automatisch .

*4.3.2.2.4.3.42.2.2.3. per Active-X-Control*

Die Daten werden einmalig eingebunden.

#### *4.3.2.2.4.3.42.3. Tabellen-Objektmodell (TOM) in JScript*

Für die dynamische Verwaltung einer Tabelle wird ein eigenes Tabellenmodel (TOM) verwendet, welches aber mit dem HTML-DOM (DOM) kommuniziert. Tabellenelemente sind also im DOM und TOM verfügbar und müssen Eigenschaften und Methoden zum DOM und TOM implementiert haben.

Die Tabelle ist der Container (Eltern) für ihre Elemente.

Die Eigenschaften und Methoden der Tabelle im TOM werden in der Regel unter Beibehaltung des Bezeichners durch die Eigenschaften und Methoden des Tabellenelementes im TOM überschrieben und **nicht** vererbt.

Im TOM sind auch die Methoden implementiert, die das Füllen der Tabelle mit den Tabellenelemente (außer TBODY) zulassen.
Alle Tabellenlemente TBODY müssen
entweder in HTML kodiert sein
oder über die Methoden des DOM erzeugt werden.
TOM ist nicht in der Lage, TBODY-Elemente zu erzeugen.

Der Aufwand in der DHTML-Programmierung ist wesentlich höher dafür abstrakter als die pure HTML-Kodierung der Tabelle im Dokument. Aber letztere lässt keine dynamische Struktur- und Datenverwaltung zu.

#### *4.3.2.2.4.3.42.4. Methoden des DOM und TOM zur Verwaltung der Tabellenelemente (Übersicht)*

Zur physischen Veränderung der Tabelle und ihrer Elemente sind folgende Methoden im TOM und DOM implementiert:



Methoden der Tabelle TABLE als Objekt:

| | |
|---|---|
| .createTHead() | THEAD erzeugen |
| .deleteTHead() | THEAD löschen |
| .createTFoot() | TFOOT erzeugen. |
| .deleteTFoot() | TFOOT löschen |
| .createCaption() | CAPTIOIN erzeugen. |
| .deleteCaption() | CAPTION löschen |
| .insertRow() | Zeile erzeugen |
| .deleteRow() | Zeile löschen |
| .moveRow() | Zeilen austauschen |

Methoden des Tabellenkörpers TBODY als Objekt:

| | |
|---|---|
| .insertRow() | Zeile erzeugen |
| .deleteTHead() | THEAD löschen |
| .deleteTFoot() | TFOOT löschen |
| .deleteRow() | Zeile löschen |
| .moveRow() | Zeilen austauschen |

Methoden des Tabellekopfes THEAD als Objekt:

| | |
|---|---|
| .insertRow() | Zeile erzeugen |
| .deleteRow() | Zeile löschen |
| .moveRow() | Zeilen austauschen |

Methoden des Tabellenfusses TFOOT als Objekt:

| | |
|---|---|
| .insertRow() | Zeile erzeugen |
| .deleteRow() | Zeile löschen |
| .moveRow() | Zeilen austauschen |

Methoden der Tabellenzeile TR als Objekt:

| | |
|---|---|
| .deleteCell() | Zelle löschen in Zeile, Methode von TR |
| .insertCell() | Zelle erzeugen in Zeile, Methode von TR |

Die Erzeugung von TBODY ist im TOM nicht implementiert. Dafür ist **nur** DOM zu verwenden.

Die Erzeugung /Löschung von Tabellelementen bewirkt Änderung im DOM.und in den betroffenen Collectionen des TOM.

#### *4.3.2.2.4.3.42.5. Dynamische Struktur und Daten einer Tabelle per JScript*

##### *4.3.2.2.4.3.42.5.1. Strukturerzeugung der Tabelle*

###### *4.3.2.2.4.3.42.5.1.1. in HTML*

Wird die HTML-Kodierung einer Tabelle im BODY-Teil des Dokumentes grwünscht, so ist zu beachten:

Im HTML-Code muss eine leere Tabelle aus folgenden Tags kodiert werden:
TABLE-Tag
**allen** TBODY-Tags
Für die Tags sind ID-Attribute zu kodieren.

Die Erzeugung der anderen Tabellenelemente erfolgt per Script, das aktiviert wird nach dem kompletten Laden des Dokumentes und damit dem Laden der leeren Tabelle (onload="..." kodieren im BODY-Tag)

Beispiel für Erzeugung einer Tabelle in Script per HTML, TOM und DOM:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
        function TabelleFuellen()
        {
                var Zeile;
                var Zelle;

                // +++++   Tabellenrahmen und Rahmenfarbe ++++++++++++++++++++++++++++++++++++
                ID_Tabelle.border    = 1;
                ID_Tabelle.bgColor  = "magenta";

                // +++++   TabellenKopf füllen +++++++++++++++++++++++++++++++++++++++++++++++
                // -----       erzeugen
                var TabellenKopf      = ID_Tabelle.createTHead();

                // -----       Hintergrundfarbe
```



```
		TabellenKopf.bgColor = "blue";

		// -----	mit 1 Zeile
		Zeile			= TabellenKopf.insertRow();

		//		Zeile mit 1 Zellenfüllen
		Zelle			= Zeile.insertCell();
		Zelle.align		= "center";
		Zelle.style.fontWeight = "bold";
		Zelle.innerText		= "Tabellenkopf Zelle";

		// +++++	Tabellenbody 0 füllen ++++++++++++++++++++++++++++++++++++++++++++++++
		//			wurde per HTML-Kodierung im BODY erzeugt
		// -----	Hintergrundfarbe
		ID_TBody0.bgColor = "green";

		//-----		2 Zeilen mit je 1 Zelle erzeugen
		for ( var ZeilenZahler =0 ; ZeilenZahler <2; ZeilenZahler ++)
		{
			// -----	Zeile erzeugen
			Zeile = ID_TBody0.insertRow();

			// -----	Zelle in der Zeile erzeugen
			Zelle = Zeile.insertCell();

			//		und füllen
			Zelle.innerText = "TBODY 0 Zeile " + ZeilenZahler + " Zelle"
		}

		// +++++	Tabellenbody 1 füllen ++++++++++++++++++++++++++++++++++++++++++++++++
		//			wurde per HTML-Kodierung im BODY erzeugt
		// -----	Hintergrundfarbe
		ID_TBody1.bgColor = "yellow";

		//-----		2 Zeilen mit je 1 Zelle erzeugen
		for ( var ZeilenZahler =0 ; ZeilenZahler <2; ZeilenZahler ++)
		{
			// -----	Zeile erzeugen
			Zeile = ID_TBody1.insertRow();

			// -----	Zelle in der Zeile erzeugen
			Zelle = Zeile.insertCell();

			//		und füllen
			Zelle.innerText = "TBODY 1 Zeile " + ZeilenZahler + " Zelle"
		}

		// +++++	TabellenFuss  füllen ++++++++++++++++++++++++++++++++++++++++++++++++
		// -----	erzeugen
		var TabellenFuss	= ID_Tabelle.createTFoot();

		// -----	mit Hintergrundfarbe
		TabellenFuss.bgColor = "brown";

		// -----	mit 1 Zeile
		Zeile			= TabellenFuss.insertRow();

		//		Zeile mit 1 Zellenfüllen
		Zelle			= Zeile.insertCell();
		Zelle.align		= "center";
		Zelle.style.fontWeight = "bold";
		Zelle.innerText		= "Tabellenfuss Zelle";
		Zelle.colSpan		= "1";
		Zelle.id		= "ZelleImTabellenFuss";

		// +++++	TabellenCapiton füllen ++++++++++++++++++++++++++++++++++++++++++++++++
		// -----	erzeugen
		var TabellenCaption	= ID_Tabelle.createCaption();

		// ----- und füllen
		TabellenCaption.align = "bottom";
		TabellenCaption.style.fontSize = "10";
		TabellenCaption.innerText = "TabelleCaption"
	}
```



```
		function ZelleImTabellenFussAendern()
		{ZelleImTabellenFuss.innerText = "Tabellenfuss Zelle neu"}
	</SCRIPT>
	</HEAD>
	<BODY onload="TabelleFuellen();">
		<! --- Tabelle leer mit allen TBODY erzeugen, aber komplett ohne Daten, da sonst nicht manipulierbar !!! //--->
		<TABLE  ID="ID_Tabelle"
			BORDER
			BGCOLOR="gray"
		>
			<TBODY ID="ID_TBody0"></TBODY>
			<TBODY ID="ID_TBody1"></TBODY>
		</TABLE>
		<BR>
		Tabelle mit HTML und TOM erzeugt
		<BR>
		<BUTTON onclick="ZelleImTabellenFussAendern();">Zelle im Tabellenfuss aendern</BUTTON>
	<BODY>
	<HTML>
```

#### *4.3.2.2.4.3.42.5.1.2. per Methoden des DOM*

Wird die HTML-Kodierung einer Tabelle **nicht** im BODY-Teil des Dokumentes gewünscht, so ist zu beachten:

Es sind sind alle Elemente (inklusvie der Tabelle) per Methode .createElement() leer zu erzeugen und dann per Methode .appendChild() in der richtigen Reihenfolge in das DOM einzubinden.

Es sollte im BODY-Teil des Dokumentes ein leerer HTML-Container z.B. DIV mit ID-Attribut kodiert sein. In diesen Container erfolgt die Einbindung der Tabelle und damit in das Dokument. Der Container kann frei positioniert sein (und damit die Tabelle). Alternativ ist auch das Einbinden direkt in den BODY-Teil möglich, wobei das ID-Attribut im BODY-Tag verwendet wird. Der BODY-Teil des Dokumentes kann aber nicht positioniert werden (abgesehen vom Scrollen des Dokumentes im Anzeigefenster).

Die Erzeugung **aller** Tabellenelemente erfolgt per Script, das aktiviert wird nach dem kompletten Laden des Dokumentes (onload="..." kodieren im BODY-Tag)

Beispiel für Erzeugung einer Tabelle in Script per DOM und TOM aber ohne HTML:

```
	<HTML>
	<HEAD>
	<SCRIPT LANGUAGE="JScript">
		function TabelleFuellen()
		{
			var Zeile;
			var Zelle;

			// +++++   Tabelle erzeugen ++++++++++++++++++++++++++++++++++++++++++++++++++++
			var Tabelle		= document.createElement("TABLE");

			// ----- Tabellenrahmen und Rahmenfarbe
			Tabelle.border		=1;
			Tabelle.bgColor		="magenta";

			// +++++   Tabellenelemente  erzeugen ++++++++++++++++++++++++++++++++++++++++++
			var TabellenKopf	= document.createElement("THEAD");
			var Tabellenbody0	= document.createElement("TBODY");
			var Tabellenbody1	= document.createElement("TBODY");
			var TabellenFuss	= document.createElement("TFOOT");
			var TabellenCaption	= document.createElement("CAPTION");

			// +++++   Tabelle zusammenbauen ++++++++++++++++++++++++++++++++++++++++++++++
			Tabelle.appendChild(TabellenKopf);
			Tabelle.appendChild(Tabellenbody0);
			Tabelle.appendChild(Tabellenbody1);
			Tabelle.appendChild(TabellenFuss);
			Tabelle.appendChild(TabellenCaption);

			// +++++   TabellenKopf füllen ++++++++++++++++++++++++++++++++++++++++++++++++

			// -----	mit Hintergrundfarbe
			TabellenKopf.bgColor = "blue";

			// -----	mit 1 Zeile
			Zeile			= document.createElement("TR");  // erzeugt automatisch Start- und Ende-Tag
			TabellenKopf.appendChild(Zeile);
```



```
		//		Zeile mit 1 Zellenfüllen
		Zelle			= Zeile.insertCell();
		Zelle.align		= "center";
		Zelle.style.fontWeight = "bold";
		Zelle.innerText		= "Tabellenkopf Zelle";

		// +++++	Tabellenbody 0 füllen ++++++++++++++++++++++++++++++++++++++++++++++++

		// -----	Hintergrundfarbe
		Tabellenbody0.bgColor = "green";

		//-----		2 Zeilen mit je 1 Zelle erzeugen
		for ( var ZeilenZahler =0 ; ZeilenZahler <2; ZeilenZahler ++)
		{
			// -----	Zeile erzeugen
			Zeile = document.createElement("TR");  // erzeugt automatisch Start- und Ende-Tag
			Tabellenbody0.appendChild(Zeile);

			// -----	Zelle in der Zeile erzeugen
			Zelle = document.createElement("TD"); // erzeugt automatisch Start- und Ende-Tag
			Zeile.appendChild(Zelle);

			//		und füllen
			Zelle.innerText = "TBODY 0 Zeile " + ZeilenZahler + " Zelle"
		}

		// +++++	Tabellenbody 1 füllen ++++++++++++++++++++++++++++++++++++++++++++++++

		// -----	Hintergrundfarbe
		Tabellenbody1.bgColor = "yellow";

		//-----		2 Zeilen mit je 1 Zelle erzeugen
		for ( var ZeilenZahler =0 ; ZeilenZahler <2; ZeilenZahler ++)
		{
			// -----	Zeile erzeugen
			Zeile = document.createElement("TR");  // erzeugt automatisch Start- und Ende-Tag
			Tabellenbody1.appendChild(Zeile);

			// -----	Zelle in der Zeile erzeugen
			Zelle = document.createElement("TD"); // erzeugt automatisch Start- und Ende-Tag
			Zeile.appendChild(Zelle);

			//		und füllen
			Zelle.innerText = "TBODY 1 Zeile " + ZeilenZahler + " Zelle"
		}

		// +++++	TabellenFuss  füllen ++++++++++++++++++++++++++++++++++++++++++++++++

		// -----	mit Hintergrundfarbe
		TabellenFuss.bgColor = "brown";

		// -----	mit 1 Zeile
		Zeile			= document.createElement("TR");  // erzeugt automatisch Start- und Ende-Tag
		TabellenFuss.appendChild(Zeile);

		//		Zeile mit 1 Zellenfüllen
		Zelle			= document.createElement("TD"); // erzeugt automatisch Start- und Ende-Tag
		Zeiles.appendChild(Zelle);

		Zelle.align		= "center";
		Zelle.style.fontWeight = "bold";
		Zelle.innerText		= "Tabellenfuss Zelle";
		Zelle.colSpan		= "1";
		Zelle.id		= "ZelleImTabellenFuss";

		// +++++	TabellenCapiton füllen ++++++++++++++++++++++++++++++++++++++++++++++++
		TabellenCaption.align = "bottom";
		TabellenCaption.style.fontSize = "10";
		TabellenCaption.innerText = "TabelleCaption"

		// +++++	Tabelle in den DIV einbinden ++++++++++++++++++++++++++++++++++++++++++
		ID_Div.appendChild(Tabelle);
	}
```



```
            function ZelleImTabellenFussAendern()
            {ZelleImTabellenFuss.innerText = "Tabellenfuss Zelle neu"}
</SCRIPT>
</HEAD>
<BODY onload="TabelleFuellen();">
            <! --- Tabelle wird im DIV erzeugt wobei das ID des DIV verwendet wird , um die Tabelle einzubinden //--->
            <DIV ID="ID_Div"></DIV>
            <BR>
            Tabelle mit DOM und TOM erzeugt
            <BR>
            <BUTTON onclick="ZelleImTabellenFussAendern();">Zelle im Tabellenfuss aendern</BUTTON>
<BODY>
<HTML>
```

#### *4.3.2.2.4.3.42.5.2. Datenerzeugung mit der Strukturbildung und zur Laufzeit*

Die dynamische Struktur muss per Script erzeugt worden sein. Bei der Erzeugung der Struktur können bereits Daten implementiert werden.

Die dynamische Änderung von Daten einer Tabelle kann zur Laufzeit nur durch den Bezug auf die jeweilige Zelle in der Zeile erfolgen.
Folgende Eigenschaften müsse n daher für eine Zelle in der Struktur kodiert worden sein:

| | |
|---|---|
| .id | für den Bezug auf die Zelle<br>immer kodieren<br>alternativ: Zugriff über die Collectionen rows und cells |
| .innerText | für den Textinhalt einer Zelle (Plain-Text)<br>alternativ auch .innerHTML |
| .innerHTML | für den HTML-Inhalt einer Zelle<br>alternativ auch .innerText<br>immer notwendig zur Erzeugung von TH |

**Nur bei einer Zelle** sind die Eigenschaften .innerHTML bzw. .innerText. schreibbar.

Beispiel Kalender:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
<!--
// Kalender aktueller Monat
// nur für IE

// ##### änderbare Variablen ###########################################################################

// +++++ Position des Kalenders ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
var Kalender_Top=20;
var Kalender_Left=20;

// +++++ Farben und Schrift-Layout ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// Schriften: Es sollten nur Fonts verwendet werden, die auf jedem Windows-PC installiert sind

// ----- Kalenderkopf ----------------------------------------------------------------

var Kalender_Kopf_SchriftArt='Times New Roman, Arial';
var Kalender_Kopf_SchriftGroesse= 5;
var Kalender_Kopf_SchriftFarbe='#CCEEFF';
var Kalender_Kopf_HintergrundFarbe='#3A6EA5'; // identisch mit BGCOLOR des BODY
var Kalender_SpaltenKopf_SchriftFarbe='#AAFFFF';

// ----- Tag -------------------------------------------------------------------------

var Tag_SchriftArt='Times New Roman, Verdana,Arial';
var Tag_SchriftGroesse=4;
var Tag_SchriftFarbe='#CCCCCC';
var Tag_HintergrundFarbe=Kalender_Kopf_HintergrundFarbe;
var Tag_Sonntag_SchriftFarbe='#FFDD00';
var Tag_Heute_SchriftFarbe='#FFFFFF';
var Tag_Heute_HintergrundFarbe='#1A4E85';
var Tag_Heute_Sonntag_SchriftFarbe=Tag_Heute_SchriftFarbe;
var Tag_Heute_Sonntag_HintergrundFarbe=Tag_Heute_HintergrundFarbe;

// +++++ Timer ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// Timer für Kalender zur Erkennung von Tages-, Monats- und Jahreswechsel verwenden
```



```
var Kalender_Timer_Verwenden=false; // true für verwenden, false für nicht verwenden

// Timer in Millisekunden für Aktualisierung des Kalenders
//   nur verwendet wenn Kalender_Timer_Verwenden auf true
//              Wert so hoch wie möglich setzen damit durch den Timer wenig Ressourcen genutzt werden
var Kalender_TimerWert=5000;


// +++++ Monatsnamen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

var MonatsNamenFeld = new Array
(
 'Januar',
 'Februar',
 'M&auml;rz',
 'April',
 'Mai',
 'Juni',
 'Juli',
 'August',
 'September',
 'Oktober',
 'November',
 'Dezember'
);

// +++++ Tagnamen kurz ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

var TagKurzNamenFeld = new Array // Woche beginnt mit Montag
(
 'Mo',
 'Di',
 'Mi',
 'Do',
 'Fr',
 'Sa',
 'So'
);

// ##### interne Variablen ##################################################################
//      alle Variablen möglichst Wert-Typ-gerecht initialisieren (bei Zeiger kein "nil" oder
//            "null" verwenden, also Zeiger ohne init).

var TagKurzNamenFeld_Laenge=TagKurzNamenFeld.length;

// Zeit- und Datum-Berechnungen
var Zeit_Jetzt;
var Zeit_Jetzt_Kette='';
var Zeit_Jetzt_Tag_Numerisch=0;
var Zeit_Jetzt_Monat_Numerisch=0;
var Zeit_Jetzt_Jahr_Numerisch=0;
var Zeit_Jetzt_Stunden_Numerisch=0;
var Zeit_Jetzt_Minuten_Numerisch=0;
var Zeit_Jetzt_Sekunden_Numerisch=0;
var Zeit_Vormonat;
var Zeit_VormonatErsterTag;

// Kalender-Anzeige: Globale Variablen, da Anzeige per Timer gesteuert werden kann
//                   und somit nicht pro Aufruf die Variablen neu erzeugt werden müssen.
var Anzeige_Kalender_TagesZahler=0;
var Anzeige_Kalender_HTMLCode='';
var Anzeige_Kalender_Kette='';
var Anzeige_Kalender_Zahler_TR=0;
var Anzeige_Kalender_Zahler_TD=0;
var Anzeige_Kalender_Tag_Kette1='';
var Anzeige_Kalender_Tag_Kette2='';
var Anzeige_Kalender_Tag_Kette3='';
var Anzeige_Kalender_Sonntag_Kette1='';
var Anzeige_Kalender_Sonntag_Kette2='';
var Anzeige_Kalender_Sonntag_Kette3='';

// ##### Funktionen der Zeit- und Datum-Berechnungen ##########################################

function Zeit_AktuelleAngabenHolen()
```



```
{
 Zeit_Jetzt                          = new Date();
 Zeit_Jetzt_Tag_Numerisch            = Zeit_Jetzt.getDate();
 Zeit_Jetzt_Monat_Numerisch          = Zeit_Jetzt.getMonth() + 1;
 Zeit_Jetzt_Jahr_Numerisch           = Zeit_Jetzt.getYear();
 Zeit_Jetzt_Stunden_Numerisch        = Zeit_Jetzt.getHours();
 Zeit_Jetzt_Minuten_Numerisch        = Zeit_Jetzt.getMinutes();
 Zeit_Jetzt_Sekunden_Numerisch       = Zeit_Jetzt.getSeconds();
 Zeit_Vormonat                       = new Date(Zeit_Jetzt_Jahr_Numerisch,Zeit_Jetzt_Monat_Numerisch-1,1);
 Zeit_VormonatErsterTag              = Zeit_Vormonat.getDay();

 if(Zeit_Jetzt_Jahr_Numerisch < 100)
 {Zeit_Jetzt_Jahr_Numerisch+=1900;}

 if(Zeit_Jetzt_Jahr_Numerisch < 100)
 {Zeit_Jetzt_Jahr_Numerisch+=1900;}
}

function Zeit_NumerischNachString_MitVornull(NumerischerWert)
{
 Zeit_Jetzt_Kette=NumerischerWert.toString();

 // auf Vornull prüfen
 if (NumerischerWert < 10)
 {Zeit_Jetzt_Kette='0' + Zeit_Jetzt_Kette;}

 return Zeit_Jetzt_Kette;
}

// ##### Funktionen des Kalenders ###########################################################

// +++++ TR erzeugen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

function Anzeige_Kalender_HTMLCode_TR_Erzeugen(
                                    FarbeBackground,
                                    BreiteAlsString,
                                    HoeheAlsString,
                                    Ausrichtung_Horizontal,
                                    Ausrichtung_Vertikal,
                                    SpaltenSpannweiteAlsString,
                                    SchriftGroesse,
                                    SchriftFarbe,
                                    SchriftArt,
                                    Kette
                                                        )
// erweitert Anzeige_Kalender_HTMLCode
{
 // +++++ TR erzeugen: Beginn-Tag
 Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + '<TR>';

 // +++++ TD (Zelle) erzeugen

 // ----- Tag-Beginn
 Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + '<TD';

 // ----- Attribut Background
 if (FarbeBackground != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' BGCOLOR="' + FarbeBackground + '";}

 // ----- Attribut Breite
 if (BreiteAlsString != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' WIDTH=' + BreiteAlsString;}

 // ----- Attribut Höhe
 if (HoeheAlsString != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' HEIGHT=' + HoeheAlsString;}

 // ----- Attribut Ausrichtung horizontal
 if (Ausrichtung_Horizontal != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' ALIGN=' + Ausrichtung_Horizontal;}

 // ----- Attribut Ausrichtung vertikal
 if (Ausrichtung_Vertikal != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' VALIGN=' + Ausrichtung_Vertikal;}
```



```
 // ----- Attribut Spannweite
 if (SpaltenSpannweiteAlsString != '')
 {Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + ' COLSPAN=' + SpaltenSpannweiteAlsString;}

 // ----- Tag-Ende
 Anzeige_Kalender_HTMLCode=  Anzeige_Kalender_HTMLCode + '>';

 // ----- Inhalt der Zelle als Font mit Text
 Anzeige_Kalender_HTMLCode=
                      Anzeige_Kalender_HTMLCode
                    +  '<FONT SIZE='  + SchriftGroesse
                    +       ' COLOR=' + SchriftFarbe
                    +       ' FACE="' + SchriftArt + '"'
                    +  '>'
                    +   Kette
                    +  '</FONT>'
  // ----- Ende-Tag
 Anzeige_Kalender_HTMLCode=  Anzeige_Kalender_HTMLCode + '</TD>'

 // +++++ TR erzeugen: Ende-Tag
 Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode + '</TR>';
}

// +++++ TD erzeugen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

function Anzeige_Kalender_HTMLCode_TD_Erzeugen(
                                  Inhalt,
                                  HintergrundFarbe,
                                  SchriftFarbe,
                                  SchriftGroesse,
                                  SchriftArt,
                                  BorderColor
                                                      )
// erweitert Anzeige_Kalender_HTMLCode
// alle Variablen alle global, damit nicht pro Funktionsaufruf die Variablen neu lokal erzeugt werden.
{
 // +++++ Inhalt auf FETT setzen
 Inhalt='<B>' + Inhalt + '</B>';

 // +++++ TD erzeugen
 //       Inhalt der TD ist eine Tabelle
 Anzeige_Kalender_HTMLCode=   Anzeige_Kalender_HTMLCode
                    + '<TD BGCOLOR="' + BorderColor + '">'
                    +  '<TABLE BORDER=0'
                    +              ' CELLSPACING=0'
                    +              ' CELLPADING=5'
                    +              ' WIDTH=100%'
                    +              ' HEIGHT=100%'
                    +  '>'

 Anzeige_Kalender_HTMLCode_TR_Erzeugen(HintergrundFarbe,'30','30','center','middle','',
                          SchriftGroesse,SchriftFarbe,SchriftArt,Inhalt);

 Anzeige_Kalender_HTMLCode= Anzeige_Kalender_HTMLCode
                    +  '</TABLE>'
                    + '</TD>';
}

// +++++ Tag (nicht Sonntag) TD erzeugen ++++++++++++++++++++++++++++++++++++++++++++++++

function Anzeige_Kalender_Tag_HTMLCode_TD_Erzeugen(Inhalt,Heute)
// erweitert Anzeige_Kalender_HTMLCode
// Heute true, so Tag = heute
// alle Variablen alle global, damit nicht pro Funktionsaufruf die Variablen neu lokal erzeugt werden.
{
 // Annahme: Tag ist nicht heute
 Anzeige_Kalender_Tag_Kette1=Tag_HintergrundFarbe;
 Anzeige_Kalender_Tag_Kette2=Tag_SchriftFarbe;
 Anzeige_Kalender_Tag_Kette3=Kalender_Kopf_HintergrundFarbe;

 // prüfen auf Tag == heute
 if (Heute)
 {
```



```
 Anzeige_Kalender_Tag_Kette1=Tag_Heute_HintergrundFarbe;
 Anzeige_Kalender_Tag_Kette2=Tag_Heute_SchriftFarbe;
 Anzeige_Kalender_Tag_Kette3=Tag_SchriftFarbe;
 }

 Anzeige_Kalender_HTMLCode_TD_Erzeugen(
                            Inhalt,
                            Anzeige_Kalender_Tag_Kette1,
                            Anzeige_Kalender_Tag_Kette2,
                            Tag_SchriftGroesse,
                            Tag_SchriftArt,
                            Anzeige_Kalender_Tag_Kette3
                                                    );
}

// +++++ Sonntag  TD erzeugen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

function Anzeige_Kalender_SonnTag_HTMLCode_TD_Erzeugen(Inhalt,Sonntag)
// erweitert Anzeige_Kalender_HTMLCode
// Sonntag true, so Tag = Sonntag
// alle Variablen alle global, damit nicht pro Funktionsaufruf die Variablen neu lokal erzeugt werden.
{
 // Annahme: Tag ist nicht Sonntag
 Anzeige_Kalender_Sonntag_Kette1=Tag_HintergrundFarbe;
 Anzeige_Kalender_Sonntag_Kette2=Tag_Sonntag_SchriftFarbe;
 Anzeige_Kalender_Sonntag_Kette3=Kalender_Kopf_HintergrundFarbe;

 // prüfen auf Tag == Sonntag
 if (Sonntag)
 {
  Anzeige_Kalender_Sonntag_Kette1=Tag_Heute_Sonntag_HintergrundFarbe;
  Anzeige_Kalender_Sonntag_Kette2=Tag_Heute_Sonntag_SchriftFarbe;
  Anzeige_Kalender_Sonntag_Kette3=Tag_Sonntag_SchriftFarbe;
 }

 Anzeige_Kalender_HTMLCode_TD_Erzeugen(
                            Inhalt,
                            Anzeige_Kalender_Sonntag_Kette1,
                            Anzeige_Kalender_Sonntag_Kette2,
                            Tag_SchriftGroesse,
                            Tag_SchriftArt,
                            Anzeige_Kalender_Sonntag_Kette3
                                                    );
}

// +++++ Kalender erzeugen und anzeigen ++++++++++++++++++++++++++++++++++++++++++++++++++++++

function Anzeige_Kalender()
// periodischer Aufruf per Timer NUR zur Erkennung des Tages- oder Monats- oder Jahreswechsel, falls
//   genau zu diesen Zeitpunkten der Kalender aktiv ist.
// Es spart Ressourcen, wenn Timer nicht benutzt wird !!
// Kalender_Timer_Verwenden auf false für Timer nicht verwenden
//                      auf true  für Timer verwenden
//
// alle Variablen alle global, damit nicht pro Funktionsaufruf die Variablen neu lokal erzeugt werden.
{
 // +++++ aktuelle Zeitangaben holen
 Zeit_AktuelleAngabenHolen();  // füllt
                               //   Zeit_Jetzt
                               //   Zeit_Jetzt_Tag_Numerisch
                               //   Zeit_Jetzt_Monat_Numerisch
                               //   Zeit_Jetzt_Jahr_Numerisch
                               //   Zeit_Jetzt_Stunden_Numerisch
                               //   Zeit_Jetzt_Minuten_Numerisch
                               //   Zeit_Jetzt_Sekunden_Numerisch
                               //   Zeit_Vormonat
                               //   Zeit_VormonatErsterTag

 // +++++ Start- und Endwert der Tage für Erzeugung des Kalenders holen

 // ----- Erster Tag des Vormonats als Startwert
 var Start = Zeit_VormonatErsterTag;
 if(Start > 0)
 {Start--;}   // minus 1
```



```
 else
 {Start = 6;} //

// ----- Endewert für Monatstage ermitteln
// Annahme: Monat hat 31 Tage
var Stop  = 31;

// für alle Monate mit 30 Tagen korrigieren
if(    (Zeit_Jetzt_Monat_Numerisch==4)
    || (Zeit_Jetzt_Monat_Numerisch==6)
    || (Zeit_Jetzt_Monat_Numerisch==9)
    || (Zeit_Jetzt_Monat_Numerisch==11)
  )
{--Stop;}

// für Februar korrigieren: 29 oder 28 Tage, also Schaltjahr
if(Zeit_Jetzt_Monat_Numerisch==2)
{
 // Februar
 // Annahme: Kein Schaltjahr, also 28 Tage
 Stop=28;

 // Schaltjahr mit 29 Tagen ermitteln
 if ((Zeit_Jetzt_Jahr_Numerisch%4) == 0)
 {Stop++;}  // 29

 if ( (Zeit_Jetzt_Jahr_Numerisch%100) == 0)
 {Stop--;} // wieder 28

 if ((Zeit_Jetzt_Jahr_Numerisch%400) ==0 )
 {Stop++;} // 29
}

// +++++ HTML-Code des Kalendes als Tabelle erzeugen

// +++++ äusserste Tabelle also die des Kalenders insgesamt
Anzeige_Kalender_HTMLCode=
                        '<TABLE ID="TABLE_Kalender"'
                      +              ' BORDER=0'
                      +              ' CELLPADDING=1'
                      +              ' CELLSPACING=0'
                      +              ' STYLE="position:absolute;'
                      +                        'left:'  + Kalender_Left + ';'
                      +                        'top:'   + Kalender_Top
                      +                       ""
                      + '>';

// +++++ 1. TR des Kalenders als Anzeige aktueller Monat und aktuelles Jahr in Fettschrift

// -----  Text der 1. TR ermitteln: aktueller Monat und aktuelles Jahr in Fettschrift
Anzeige_Kalender_Kette='<B>';
Anzeige_Kalender_Kette=MonatsNamenFeld[Zeit_Jetzt_Monat_Numerisch-1]; // Monatsname als String
Anzeige_Kalender_Kette+=' ';
Anzeige_Kalender_Kette+=Zeit_Jetzt_Jahr_Numerisch.toString(); // Jahr zu String
Anzeige_Kalender_Kette+='</B>';

// ----- Abstand zum Nachfolger
Anzeige_Kalender_Kette+='<BR>';
Anzeige_Kalender_Kette+='<BR>';

// ----- und erzeugen
Anzeige_Kalender_HTMLCode_TR_Erzeugen(
                            Kalender_Kopf_HintergrundFarbe,'','','center','middle','7',
                             Kalender_Kopf_SchriftGroesse,
                             Kalender_Kopf_SchriftFarbe,
                             Kalender_Kopf_SchriftArt,
                             Anzeige_Kalender_Kette
                                                  );

// +++++ 2. TR Spaltenkopf der Monatstage; pro Spalte ein TD
Anzeige_Kalender_HTMLCode+='<TR>';

// ----- Tage der Woche abklappern: Woche beginnt mit Montag
for(Anzeige_Kalender_TagesZahler=0;
```



```
    Anzeige_Kalender_TagesZahler < TagKurzNamenFeld_Laenge;
    Anzeige_Kalender_TagesZahler++
   )
 {
 // ----- Text als Kurzname des Tages holen
 Anzeige_Kalender_Kette=TagKurzNamenFeld[Anzeige_Kalender_TagesZahler];

 // ----- und Text erzeugen
 Anzeige_Kalender_HTMLCode_TD_Erzeugen(
                                      Anzeige_Kalender_Kette,
                                      Kalender_Kopf_HintergrundFarbe,
                                              Kalender_SpaltenKopf_SchriftFarbe,
                                              Tag_SchriftGroesse,
                                              Kalender_Kopf_SchriftArt,
                                              Kalender_Kopf_HintergrundFarbe
                                                  );
}

Anzeige_Kalender_HTMLCode+='</TR>'; // Ende der Wocheneinteilung in 7 Tage (Spaltenköpfe)

// +++++ ab 3. TR erfolgt die Wochenauflistung
//       pro Woche 1 Zeile
//       pro Zeile 1 TR
//       ab 3. TR im gesamten Kalender

// ----- init des Tageszähler des Monats
Anzeige_Kalender_TagesZahler = 1;

// ----- Wochen abklappern also Zeilen (TR)
for(Anzeige_Kalender_Zahler_TR=0;
      Anzeige_Kalender_Zahler_TR < 6;
      Anzeige_Kalender_Zahler_TR++
   )
 {
 // - - - aktuelle Woche (Zeile) erzeugen
 Anzeige_Kalender_HTMLCode+='<TR>';

 // - - - pro Spalte 1 TD erzeugen: Nur die ersten 6 Tage der Woche, OHNE Sonntag !
 for(Anzeige_Kalender_Zahler_TD=0;
       Anzeige_Kalender_Zahler_TD < 6; // ohne Sonntag !
       Anzeige_Kalender_Zahler_TD++
    )
 {
 // - - - prüfen ob 1. TR-Zeile noch nicht abgeklappert wurde
 if(       (Anzeige_Kalender_Zahler_TR == 0)
     && (Anzeige_Kalender_Zahler_TD < Start)
   )
 {
 // 1. TR  und nur 1. TD
 Anzeige_Kalender_Tag_HTMLCode_TD_Erzeugen(' ',false);
 }
 else
 {
 // 1. TR ab 2.TD oder ab 2.TR

 // - - - prüfen ob alle Tage bereits abgeklappert wurden
 if (Anzeige_Kalender_TagesZahler > Stop)
 {
  // alle Tage sind abgeklapppert
  Anzeige_Kalender_Tag_HTMLCode_TD_Erzeugen(' ',false);
 }
 else
 {
  // 1. TR ab 2.TD oder ab 2.TR
  // es sind noch Tage abzuklappern

  // - - - prüfen ob aktueller Tag der Tag von JETZT ist, also Heute ist
  if(Anzeige_Kalender_TagesZahler==Zeit_Jetzt_Tag_Numerisch)
  {
   // 1. TR ab 2.TD oder ab 2.TR
   // es sind noch Tage abzuklappern
   // aktueller Tag ist heute

   Anzeige_Kalender_Tag_HTMLCode_TD_Erzeugen(Anzeige_Kalender_TagesZahler,true);
```



```
    }
    else
    {
     // 1. TR ab 2.TD oder ab 2.TR
     // es sind noch Tage abzuklappern
     // aktueller Tag ist nicht heute
     Anzeige_Kalender_Tag_HTMLCode_TD_Erzeugen(Anzeige_Kalender_TagesZahler,false);
    }

    // - - - Tageszähler erhöhen für nächsten Tag als neuen aktuellen
    Anzeige_Kalender_TagesZahler++;
   }
  }
 }

 // - - - ALLE TD der ersten 6 Tage der aktuellen Woche (Zeile) wurden erzeugt
 //       offen ist der 7. Tag, also der Sonntag (falls vorhanden)

 //      prüfen ob alle Tage abgeklappert wurden, also ob es keinen Sonntag gibt
 if(Anzeige_Kalender_TagesZahler > Stop)
 {
  // keine Tage mehr abzuklappern, kein Sonntag mehr da
  Anzeige_Kalender_SonnTag_HTMLCode_TD_Erzeugen(' ',false);
 }
 else
 {
  // - - - Es sind noch Tage abzuklappern, also 7. Tag = Sonntag vorhanden

  // - - - prüfen ob der aktuelle Tag der von heute ist,
  //             wenn ja, dann ist heute Sonntag
  if(Anzeige_Kalender_TagesZahler==Zeit_Jetzt_Tag_Numerisch)
  {
   // heute ist Sonntag, also den Tag mit anderem Hintergrund erzeugen
   //    als alle anderen Sonntage
   Anzeige_Kalender_SonnTag_HTMLCode_TD_Erzeugen(Anzeige_Kalender_TagesZahler,true);
  }
  else
  {
   // heute ist nicht Sonntag, also den Tag dem normalen Hintergrund für Sonntage erzeugen
   Anzeige_Kalender_SonnTag_HTMLCode_TD_Erzeugen(Anzeige_Kalender_TagesZahler,false);
  }

  // - - - Tageszähler erhöhen für nächsten Tag als neuen aktuellen
  Anzeige_Kalender_TagesZahler++;
 }

 // - - - aktuelle Woche also TD beenden
 Anzeige_Kalender_HTMLCode+='</TR>';
}

// +++++ Ende des Kalenders
Anzeige_Kalender_HTMLCode+='</TABLE>';

// +++++ und Kalenderdaten neu rendern
document.all.ID_Kalender.innerHTML = Anzeige_Kalender_HTMLCode;

// +++++ prüfen ob der Timer zur Erkennung der Tages-, Monats- und Jahreswechsel
//          verwendet werden soll
if (Kalender_Timer_Verwenden)
 {window.setTimeout('Anzeige_Kalender()',5000);}
}

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

<BODY BGCOLOR=#3A6EA5>
<!-- BODY mit identischer Hintergrundfarbe wie laut Kalender_Kopf_HintergrundFarbe -->

<DIV ID="ID_Kalender"></DIV>
<!--   Der DIV wird nach seiner Erzeugung erst gefüllt: per .innerHTML -->

<SCRIPT LANGUAGE="JScript">
<!--
```



```
Anzeige_Kalender();
// -->
</SCRIPT>
</BODY>
</HTML>
```

#### 4.3.2.2.4.3.42.6. Dynamische Veränderung einer Tabelle in JScript

Die Manipulation der Tabelle ist erst möglich, wenn das Dokument komplett geladen und das Event onload ausgelöst wurden.

##### 4.3.2.2.4.3.42.6.1. Datenveränderung per JScript

auf Basis einer in Script kodierten Struktur und zur Laufzeit des Dokumentes, das die Tabelle besitzt
nur in der jeweiligen Zelle einer Zeile (siehe oben)
**Nur bei einer Zelle** sind die Eigenschaften .innerHTML bzw. .innerText. schreibbar.

##### 4.3.2.2.4.3.42.6.2. Layoutveränderung per Style

auf Basis einer in Script kodierten Struktur und zur Laufzeit des Dokumentes, das die Tabelle besitzt
werden am schnellsten und automatisch gerendert
Tabellen-Refresh nicht nötig
alle Tabellenelement wie die Tabelle selbst besitzen das STYLE-Attribut
siehe sonst oben

##### 4.3.2.2.4.3.42.6.3. Strukturveränderung per JScript

auf Basis einer in Script kodierten Struktur und zur Laufzeit des Dokumentes, das die Tabelle besitzt
siehe oben

#### 4.3.2.2.4.3.42.7. Eigenschaften der Tabelle

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .background | Hintergrundbild eines Objektes |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .border | Rahmendicke in Pixel |
| .borderColor | Borderfarbe (Rahmenfarbe) |
| .borderColorDark | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>dunkle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border) |
| .borderColorLight | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>helle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft boder) |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .caption | Zeiger auf das Objekt table.caption<br>es darf nur 1 CAPTION zur Tabelle existieren |
| .cellPadding | Abstand zwischen Zellrahmen und dem Inhalt der Zelle |
| .cellSpacing | Abstand zwischen den Zellen |

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
    <TR>
        <TD>Zelle 1</TD>
        <TD>Zelle 2</TD>
    </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.cellSpacing=20">20 </BUTTON>
<BUTTON onclick="ID_Tabelle.cellSpacing=5">5 </BUTTON>
```

| | |
|---|---|
| .className | Klassenreferenz, Klassenname |
| .clientHeight | Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt |
| .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 |
| .cols | Anzahl der Spalten in der Tabelle<br>wenn belelgt so wird Tabelle schneller gerendert |

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
    <TR>
        <TD>Zelle 1</TD>
        <TD>Zelle 2</TD>
    </TR>
</TABLE>
<BUTTON onclick="alert(ID_Tabelle.cols);">Anzahl</BUTTON>
```

| | |
|---|---|
| .dataPageSize | Anzahl der sichtbaren Datensätze auf einer Tabellenseite<br>Anzahl der Sätze pro Dataset-Anzeige |

Beispiel:

Datensätze liegen in der Datei adress.txt
Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument



hat folgenden Inhalt:

| | |
|---|---|
| 1. Satz | **vorname**;**name**;**telefon** |
| ab 2. Satz | Guericke;"Otto, von";0815 |
| | Willmann;Theo;1234 |
| | Xantippe;Isa;5678 |
| | Arktin;Richard;1055 |

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
	var SortierRichtung = true;

	function Sortieren(FeldBezeichner)
	{
		// Sortierrichtung festlegen
		var Kette = "-";	// Annahme
		if (SortierRichtung) {Kette = "+";}

		// nächste Sortierung umgekehrt
		SortierRichtung = !SortierRichtung;

		// sortieren
		ID_Datenbank.Sort= Kette + FeldBezeichner;

		// und das Ergebnis anzeigen
		ID_Datenbank.Reset();
	}

	function vorwaerts(AnzahlSaetze)
	{
		// nächste Seite anzeigen
		document.all.ID_Tabelle.nextPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  !=
					ID_Datenbank.recordset.RecordCount)
			{ID_Datenbank.recordset.MoveNext();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==  ID_Datenbank.recordset.RecordCount)
		{alert("Letzter Datensatz erreicht!");}
	}

	function rueckwaerts(AnzahlSaetze)
	{
		// vorhergehende Seite anzeigen
		document.all.ID_Tabelle.previousPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  > 1)
			{ID_Datenbank.recordset.MovePrevious();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==1)
		{alert("Erster Datensatz erreicht!");}
	}
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
	CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
	<PARAM NAME ="DataURL"    VALUE="adress.txt">
	<PARAM NAME ="UseHeader"  VALUE="True">
	<PARAM NAME ="FieldDelim"  VALUE=";">
</OBJECT>

<TABLE  ID="ID_Tabelle"
```



```
                    DATASRC=#ID_Datenbank
                    DATAPAGESIZE=1
                >
                    <THEAD>
                        <TR>
                            <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
                            <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
                            <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
                        </TR>
                    </THEAD>
                    <TBODY>
                        <TR>
                            <TD><DIV DATAFLD="vorname"></DIV></TD>
                            <TD><DIV DATAFLD="name"></DIV></TD>
                            <TD><DIV DATAFLD="telefon"></DIV></TD>
                        </TR>
                    </TBODY>
                </TABLE>
                <BR>
                <INPUT      TYPE="button"
                            VALUE="Zurueck"
                            onclick=rueckwaerts(1)"
                >                                       <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->

                <INPUT      TYPE="button"
                            VALUE="Vorwaerts"
                            onclick=vorwaerts(1)"
                >                                       <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->
                </BODY>
                </HTML>
```

.dataSrc — Datenquelle als Anker festlegen

.dir — Umflussrichtung

.disabled — Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.end — Objektaktivitäten laut Eigenschaft .timeAction beenden

.firstChild — Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

.frame — Art des Rahmens um eine Tabelle
"void" Standard, kein Rahmen
"above" Rahmen oberhalb
"below" Rahmen unterhalb
"border" Rahmen auf allen Seiten
"box" Rahmen auf allen Seiten
"hsides" Rahmen oben und unten
"lhs" Rahmen links
"rhs" Rahmen rechts
"vsides" Rahmen links und rechts

.hasMedia — Objekt ist HTML-Media-Objekt

.height — Höhe des Objektes in Pixel

.hideFocus — Focussierbarkeit

.id — Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.
lesen erfolgt anhand der Angaben im Quelltext und laut Lage des Objektes

.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
nur Plain-Text also keine HTML-Elemente und kein Script
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem
des Elternobjektes (.offsetParent)

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem
des Elternobjektes (.offsetParent)

.offsetParent Referenz der Eltern
für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem
des Elternobjektes (.offestParent)

.offsetWidth X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem
des Elternobjektes (.offestParent)

.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound

.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar
schreiben: wenn Bereich gefüllt so komplett überschreiben
Plain-Text
HTML-Elemente möglich
nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.

.outerText Referenz auf den gesamten Plain-Text im Objekt
nur nach kompletten einlesen des Dokumentes nutzbar
nur Plain-Text
schreiben: immer komplett überschreiben
nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR

.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
"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

.rules Art der sichtbaren inneren Rahmen einer Tabelle
Art der sichtbaren Rahmen zwischen den Tabellenelementen
Art des sichtbaren Rahmens um Tabelle
"all" Rahmen um alle Zeilen und Spalten
"cols" Trennlinie zwischen allen Spalten
"groups" horizontale Trennlinie zwischen allen THEAD, TBODY's und TFOOT
vertikale Trennline zwischen allen COLGROUP
"none" keine Rahmen und Trennlinien zwischen Tabellenelementen
"rows" Trennlinie zwischen allen Zeilen
"" keine Rahmen generell (auch nicht um Tabelle)

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
<THEAD>
```



```
<TR>
<TD>Kopf Zelle 1</TD>
<TD>Kopf Zelle 2</TD>
</TR>
</THEAD>
<TBODY>
<TR>
<TD>Body Zeile 1 Zelle 1</TD>
<TD>Body Zeile 1 Zelle 2</TD>
</TR>
<TR>
<TD>Body Zeile 2 Zelle 1</TD>
<TD>Body Zeile 2 Zelle 2</TD>
</TR>
<TR>
<TD>Body Zeile 3 Zelle 1</TD>
<TD>Body Zeile 3 Zelle 2</TD>
</TR>
</TBODY>
<TFOOT>
<TR>
<TD>Fuss Zelle 1</TD>
<TD>Fuss Zelle 2</TD>
</TR>
</TFOOT>
</TABLE>
<BUTTON onclick='ID_Tabelle.rules="";'>keine Rahmen</BUTTON>
<BUTTON onclick='ID_Tabelle.rules="none";'>none</BUTTON>
<BUTTON onclick='ID_Tabelle.rules="all";'>all</BUTTON>
<BUTTON onclick='ID_Tabelle.rules="cols";'>cols</BUTTON>
<BUTTON onclick='ID_Tabelle.rules="groups";'>groups</BUTTON>
<BUTTON onclick='ID_Tabelle.rules="rows";'>rows</BUTTON>
```

| | |
|---|---|
| .scopeName | Namensraum laut XMLNS-Attribut |
| .scrollHeight | Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes |
| .scrollLeft | Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollTop | Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .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)<br>Hinweis: für Scripting ist das Style-Objekt zu nutzen |
| .summary | Kommentar in einer Tabelle, der nicht gerendert wird |

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10 SUMMARY="Kommentar">
<TR>
<TD>Zelle 1</TD>
<TD>Zelle 2</TD>
</TR>
</TABLE>
```

| | |
|---|---|
| .syncMaster | Synchronisierung der Animation des im Container liegenden Elementes auf Timeline |
| .systemBitrate | wird hier nicht erklärt |
| .systemCaptions | wird hier nicht erklärt |
| .systemLanguage | Sprache festlegen für das Objekt |
| .systemOverdubOrSubtitle | wird hier nicht erklärt |
| .tabIndex | Index des Elementes in der Tab-Tasten-Folge |
| .tagName | Tag-Bezeichner des Objektes |
| .tagUrn | Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut |
| .tFoot | Zeiger auf das Objekt table.tFoot |
| .tHead | Zeiger auf das Objekt table.tHead |
| .timeContainer | Typ der Timeline des Objektes<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .title | Tooltip-Text bei Mouse over über Objekt |
| .uniqueID | durch den Browser automatisch-genriertes ID des Objektes<br>Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde<br>kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden |
| UNSELECTABLE | Selektionsfähigkeit eines Objektes |
| .width | Breite des Objektes in Pixel |

#### *4.3.2.2.4.3.42.8. Methoden der Tabelle*

| | |
|---|---|
| .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

.createCaption() leeres CAPTION in einer Tabelle erzeugen **und** einbinden
es darf nur 1 CAPTION zur Tabelle existieren

.createTFoot() leeres TFOOT in einer Tabelle erzeugen **und** einbinden
es darf nur 1 TFOOT zur Tabelle existieren

.createTHead() leeres THEAD in einer Tabelle erzeugen **und** einbinden
es darf nur 1 THEAD zur Tabelle existieren

.deleteCaption() CAPTION löschen aus Tabelle
es darf nur 1 CAPTION zur Tabelle existieren

.deleteRow() Zeile löschen aus Tabelle
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.deleteTFoot() TFOOT der Tabelle löschen
es darf nur 1 TFOOT zur Tabelle existieren
siehe Objekt table
siehe Objekt table.tBody

.deleteTHead() THEAD der Tabelle löschen
es darf nur 1 THEAD zur Tabelle existieren
siehe Objekt table
siehe Objekt table.tBody

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

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

.fireEvent() ein Event auslösen

.firstPage() erste Seite des Datasets in Tabelle anzeigen
Eigenschaft .dataPageSize muss belegt worden sein

Beispiel:



Datensätze liegen in der Datei adress.txt
Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument hat folgenden Inhalt:

| | |
|---|---|
| 1. Satz | **vorname**;**name**;**telefon** |
| ab 2. Satz | Guericke;"Otto, von";0815 |
| | Willmann;Theo;1234 |
| | Xantippe;Isa;5678 |
| | Arktin;Richard;1055 |

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
	var SortierRichtung = true;

	function Sortieren(FeldBezeichner)
	{
		// Sortierrichtung festlegen
		var Kette = "-";	// Annahme
		if (SortierRichtung) {Kette = "+";}

		// nächste Sortierung umgekehrt
		SortierRichtung = !SortierRichtung;

		// sortieren
		ID_Datenbank.Sort= Kette + FeldBezeichner;

		// und das Ergebnis anzeigen
		ID_Datenbank.Reset();
	}

	function vorwaerts(AnzahlSaetze)
	{
		// nächste Seite anzeigen
		document.all.ID_Tabelle.nextPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  !=
					ID_Datenbank.recordset.RecordCount)
			{ID_Datenbank.recordset.MoveNext();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==  ID_Datenbank.recordset.RecordCount)
		{alert("Letzter Datensatz erreicht!");}
	}

	function rueckwaerts(AnzahlSaetze)
	{
		// vorhergehende Seite anzeigen
		document.all.ID_Tabelle.previousPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  > 1)
			{ID_Datenbank.recordset.MovePrevious();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==1)
		{alert("Erster Datensatz erreicht!");}
	}
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
	CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
	<PARAM NAME ="DataURL"    VALUE="adress.txt">
	<PARAM NAME ="UseHeader"  VALUE="True">
	<PARAM NAME ="FieldDelim"  VALUE=";">
</OBJECT>
```



```
<TABLE  ID="ID_Tabelle"
        DATASRC=#ID_Datenbank
        DATAPAGESIZE=1
>
        <THEAD>
                <TR>
                        <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
                        <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
                        <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
                </TR>
        </THEAD>
        <TBODY>
                <TR>
                        <TD><DIV DATAFLD="vorname"></DIV></TD>
                        <TD><DIV DATAFLD="name"></DIV></TD>
                        <TD><DIV DATAFLD="telefon"></DIV></TD>
                </TR>
        </TBODY>
</TABLE>
<BR>
<INPUT          TYPE="button"
                VALUE="Zurueck"
                onclick=rueckwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->

<INPUT          TYPE="button"
                VALUE="Vorwaerts"
                onclick=vorwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->
</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

.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

.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

.insertRow() Zeile in die Tabelle einfügen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.lastPage() letzte Seite des Datasets in Tabelle anzeigen
Eigenschaft .dataPageSize muss belegt worden sein

Beispiel:

Datensätze liegen in der Datei adress.txt

Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument
hat folgenden Inhalt:

| | |
|---|---|
| 1. Satz | **vorname**;**name**;**telefon** |
| ab 2. Satz | Guericke;"Otto, von";0815 |
| | Willmann;Theo;1234 |
| | Xantippe;Isa;5678 |
| | Arktin;Richard;1055 |

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
	var SortierRichtung = true;

	function Sortieren(FeldBezeichner)
	{
		// Sortierrichtung festlegen
		var Kette = "-";	// Annahme
		if (SortierRichtung) {Kette = "+";}

		// nächste Sortierung umgekehrt
		SortierRichtung = !SortierRichtung;

		// sortieren
		ID_Datenbank.Sort= Kette + FeldBezeichner;

		// und das Ergebnis anzeigen
		ID_Datenbank.Reset();
	}

	function vorwaerts(AnzahlSaetze)
	{
		// nächste Seite anzeigen
		document.all.ID_Tabelle.nextPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  !=
					ID_Datenbank.recordset.RecordCount)
			{ID_Datenbank.recordset.MoveNext();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==  ID_Datenbank.recordset.RecordCount)
		{alert("Letzter Datensatz erreicht!");}
	}

	function rueckwaerts(AnzahlSaetze)
	{
		// vorhergehende Seite anzeigen
		document.all.ID_Tabelle.previousPage();

		// und Satzzeiger korrigieren
		for (var i = 0; i < AnzahlSaetze; i++)
		{
			if (ID_Datenbank.recordset.AbsolutePosition  > 1)
			{ID_Datenbank.recordset.MovePrevious();}
		}

		if (ID_Datenbank.recordset.AbsolutePosition  ==1)
		{alert("Erster Datensatz erreicht!");}
	}
```



```
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
        CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
        <PARAM NAME ="DataURL"    VALUE="adress.txt">
        <PARAM NAME ="UseHeader"  VALUE="True">
        <PARAM NAME ="FieldDelim"  VALUE=";">
</OBJECT>

<TABLE  ID="ID_Tabelle"
        DATASRC=#ID_Datenbank
        DATAPAGESIZE=1
>
        <THEAD>
                <TR>
                        <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
                        <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
                        <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
                </TR>
        </THEAD>
        <TBODY>
                <TR>
                        <TD><DIV DATAFLD="vorname"></DIV></TD>
                        <TD><DIV DATAFLD="name"></DIV></TD>
                        <TD><DIV DATAFLD="telefon"></DIV></TD>
                </TR>
        </TBODY>
</TABLE>
<BR>
<INPUT          TYPE="button"
                VALUE="Zurueck"
                onclick=rueckwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->

<INPUT          TYPE="button"
                VALUE="Vorwaerts"
                onclick=vorwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->
</BODY>
</HTML>
```

.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

.moveRow() 2 Zeilen in der Tabelle austauschen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
        <TR>
                <TD>Body Zeile 1 Zelle 1</TD>
                <TD>Body Zeile 1 Zelle 2</TD>
        </TR>
        <TR>
                <TD>Body Zeile 2 Zelle 1</TD>
                <TD>Body Zeile 2 Zelle 2</TD>
        </TR>
        <TR>
                <TD>Body Zeile 3 Zelle 1</TD>
                <TD>Body Zeile 3 Zelle 2</TD>
        </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.moveRow(0,1);">verschieben Zeile 0 auf  Zeile 1</BUTTON>
```

.nextPage() nächste Seite des Datasets in Tabelle anzeigen
Eigenschaft .dataPageSize muss belegt worden sein

Beispiel:



Datensätze liegen in der Datei adress.txt
Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument hat folgenden Inhalt:

| | |
|---|---|
| 1. Satz | **vorname**;**name**;**telefon** |
| ab 2. Satz | Guericke;"Otto, von";0815 |
| | Willmann;Theo;1234 |
| | Xantippe;Isa;5678 |
| | Arktin;Richard;1055 |

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
        var SortierRichtung = true;

        function Sortieren(FeldBezeichner)
        {
                // Sortierrichtung festlegen
                var Kette = "-";        // Annahme
                if (SortierRichtung) {Kette = "+";}

                // nächste Sortierung umgekehrt
                SortierRichtung = !SortierRichtung;

                // sortieren
                ID_Datenbank.Sort= Kette + FeldBezeichner;

                // und das Ergebnis anzeigen
                ID_Datenbank.Reset();
        }

        function vorwaerts(AnzahlSaetze)
        {
                // nächste Seite anzeigen
                document.all.ID_Tabelle.nextPage();

                // und Satzzeiger korrigieren
                for (var i = 0; i < AnzahlSaetze; i++)
                {
                        if (ID_Datenbank.recordset.AbsolutePosition  !=
                                        ID_Datenbank.recordset.RecordCount)
                        {ID_Datenbank.recordset.MoveNext();}
                }

                if (ID_Datenbank.recordset.AbsolutePosition  ==  ID_Datenbank.recordset.RecordCount)
                {alert("Letzter Datensatz erreicht!");}
        }

        function rueckwaerts(AnzahlSaetze)
        {
                // vorhergehende Seite anzeigen
                document.all.ID_Tabelle.previousPage();

                // und Satzzeiger korrigieren
                for (var i = 0; i < AnzahlSaetze; i++)
                {
                        if (ID_Datenbank.recordset.AbsolutePosition  > 1)
                        {ID_Datenbank.recordset.MovePrevious();}
                }

                if (ID_Datenbank.recordset.AbsolutePosition  ==1)
                {alert("Erster Datensatz erreicht!");}
        }
// -->
</SCRIPT>
</HEAD>
<BODY>
<OBJECT ID="ID_Datenbank"
        CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
>
        <PARAM NAME ="DataURL"    VALUE="adress.txt">
        <PARAM NAME ="UseHeader"  VALUE="True">
        <PARAM NAME ="FieldDelim"  VALUE=";">
</OBJECT>
```



```
<TABLE  ID="ID_Tabelle"
        DATASRC=#ID_Datenbank
        DATAPAGESIZE=1
>
        <THEAD>
                <TR>
                        <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
                        <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
                        <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
                </TR>
        </THEAD>
        <TBODY>
                <TR>
                        <TD><DIV DATAFLD="vorname"></DIV></TD>
                        <TD><DIV DATAFLD="name"></DIV></TD>
                        <TD><DIV DATAFLD="telefon"></DIV></TD>
                </TR>
        </TBODY>
</TABLE>
<BR>
<INPUT          TYPE="button"
                VALUE="Zurueck"
                onclick=rueckwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->

<INPUT          TYPE="button"
                VALUE="Vorwaerts"
                onclick=vorwaerts(1)"
>                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->
</BODY>
</HTML>
```

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

.previousPage() vorhergehende Seite des Datasets in Tabelle anzeigen
Eigenschaft .dataPageSize muss belegt worden sein

Beispiel:

Datensätze liegen in der Datei adress.txt

Datei adress.txt liegt im gleichen Verzeichnis wie nachfolgendes HTML-Dokument
hat folgenden Inhalt:

| | |
|---|---|
| 1. Satz | **vorname**;**name**;**telefon** |
| ab 2. Satz | Guericke;"Otto, von";0815 |
| | Willmann;Theo;1234 |
| | Xantippe;Isa;5678 |
| | Arktin;Richard;1055 |

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript" LANGUAGE="JScript">
<!--
        var SortierRichtung = true;

        function Sortieren(FeldBezeichner)
        {
                // Sortierrichtung festlegen
                var Kette = "-";        // Annahme
                if (SortierRichtung) {Kette = "+";}

                // nächste Sortierung umgekehrt
                SortierRichtung = !SortierRichtung;

                // sortieren
                ID_Datenbank.Sort= Kette + FeldBezeichner;

                // und das Ergebnis anzeigen
                ID_Datenbank.Reset();
        }

        function vorwaerts(AnzahlSaetze)
        {
                // nächste Seite anzeigen
                document.all.ID_Tabelle.nextPage();

                // und Satzzeiger korrigieren
```



```
                        for (var i = 0; i < AnzahlSaetze; i++)
                        {
                                if (ID_Datenbank.recordset.AbsolutePosition  !=
                                                        ID_Datenbank.recordset.RecordCount)
                                {ID_Datenbank.recordset.MoveNext();}
                        }

                        if (ID_Datenbank.recordset.AbsolutePosition  ==  ID_Datenbank.recordset.RecordCount)
                        {alert("Letzter Datensatz erreicht!");}
                }

                function rueckwaerts(AnzahlSaetze)
                {
                        // vorhergehende Seite anzeigen
                        document.all.ID_Tabelle.previousPage();

                        // und Satzzeiger korrigieren
                        for (var i = 0; i < AnzahlSaetze; i++)
                        {
                                if (ID_Datenbank.recordset.AbsolutePosition  > 1)
                                {ID_Datenbank.recordset.MovePrevious();}
                        }

                        if (ID_Datenbank.recordset.AbsolutePosition  ==1)
                        {alert("Erster Datensatz erreicht!");}
                }
        // -->
        </SCRIPT>
        </HEAD>
        <BODY>
        <OBJECT ID="ID_Datenbank"
                CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
        >
                <PARAM NAME ="DataURL"    VALUE="adress.txt">
                <PARAM NAME ="UseHeader"  VALUE="True">
                <PARAM NAME ="FieldDelim"  VALUE=";">
        </OBJECT>

        <TABLE  ID="ID_Tabelle"
                DATASRC=#ID_Datenbank
                DATAPAGESIZE=1
        >
                <THEAD>
                        <TR>
                                <TH><A HREF="JavaScript:Sortieren('vorname')">Vorname </A></TH>
                                <TH><A HREF="JavaScript:Sortieren('name')">Name</A></TH>
                                <TH><A HREF="JavaScript:Sortieren('telefon')">Telefon</A></TH>
                        </TR>
                </THEAD>
                <TBODY>
                        <TR>
                                <TD><DIV DATAFLD="vorname"></DIV></TD>
                                <TD><DIV DATAFLD="name"></DIV></TD>
                                <TD><DIV DATAFLD="telefon"></DIV></TD>
                        </TR>
                </TBODY>
        </TABLE>
        <BR>
        <INPUT          TYPE="button"
                        VALUE="Zurueck"
                        onclick=rueckwaerts(1)"
        >                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->

        <INPUT          TYPE="button"
                        VALUE="Vorwaerts"
                        onclick=vorwaerts(1)"
        >                                               <!-- Anzahl der Sätze laut TABLE DATAPAGESIZE=1 //->
        </BODY>
        </HTML>
```

.refresh() Anzeige der Tabelle neu erzeugen
Änderungen an der Tabelle sichtbar machen z.B. wenn Tabelle in Anzahl Zeilen bzw. Spalten manipuliert wurde
siehe Objekt table

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

#### *4.3.2.2.4.3.42.9. table.caption Objekt des Internet Explorer*

Tabellenüberschrift
Es kann nur 1 CAPTION für eine Tabelle kodiert werden
Beispiel:

```
<TABLE>
    <CAPTION VALIGN=BOTTOM>
        Ueberschrift
    </CAPTION>
    ...
</TABLE>
```

**Eigenschaften:**

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .background | Hintergrundbild eines Objektes |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction |



siehe Objekt currTimeState und Behavior .style.time2

.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann

.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf

.className Klassenreferenz, Klassenname

.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt

.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

.dir Umflussrichtung

.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.end Objektaktivitäten laut Eigenschaft .timeAction beenden

.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

.hasMedia Objekt ist HTML-Media-Objekt

.hideFocus Focussierbarkeit

.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag
nur Plain-Text also keine HTML-Elemente und kein Script
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetParent Referenz der Eltern
für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.offsetWidth X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)



| | |
|---|---|
| .onOffBehavior | deprecated ab IE 5.x<br>Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag<br>wirksam mit parsen des Ende-Tag<br>nur nach kompletten Einlesen des Dokumentes nutzbar<br>schreiben: wenn Bereich gefüllt so komplett überschreiben<br>Plain-Text<br>HTML-Elemente möglich<br>nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR. |
| .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<br>"uninitialized" Objekt ist nicht initialisiert<br>"loading" Objekt ist nicht intialisiert aber lädt gerade Daten zur Initialisierung<br>"loaded" Objekt hat alle Daten komplett geladen und ist komplett intitialisiert<br>"interactive" Objekt kann vom User interaktiv verwendet werden zum Füllen mit Daten<br>"complete" Object hat alle Daten geladen und ist mit diesen komplett initialisiert |
| .scopeName | Namensraum laut XMLNS-Attribut |
| .scrollHeight | Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes |
| .scrollLeft | Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollTop | Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .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)<br>Hinweis: für Scripting ist das Style-Objekt zu nutzen |
| .syncMaster | Synchronisierung der Animation des im Container liegenden Elementes auf Timeline |
| .systemBitrate | wird hier nicht erklärt |
| .systemCaptions | wird hier nicht erklärt |
| .systemLanguage | Sprache festlegen für das Objekt |
| .systemOverdubOrSubtitle | wird hier nicht erklärt |
| .tabIndex | Index des Elementes in der Tab-Tasten-Folge |
| .tagName | Tag-Bezeichner des Objektes |
| .tagUrn | Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut |
| .timeContainer | Typ der Timeline des Objektes<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .title | Tooltip-Text bei Mouse over über Objekt |
| .uniqueID | durch den Browser automatisch-genriertes ID des Objektes<br>Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde<br>kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden |
| UNSELECTABLE | Selektionsfähigkeit eines Objektes |
| .vAlign | Lage der CAPTION einer Tabelle<br>es darf nur 1 CAPTION zur Tabelle existieren<br>nach Änderung ist ein Tabellen-Refresh per Methode .refresh() notwendig<br>"top" Überschrift am Kopf der Tabelle<br>Standard<br>"bottom" Überschrift am Fuss der Tabelle |

**Methoden:**

| | |
|---|---|
| .addBehavior() | DHTML-Verhaltenseigenschaft einem Element hinzufügen<br>Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig).<br>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<br>DOM wird geändert<br>Zeiger wird zugleich immer am Ende der Collection childNodes angehangen<br>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<br>DOM wird geändert<br>Element kann selbst Kinder haben<br>Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparst wurde<br>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

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

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

.fireEvent() ein Event auslösen

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

#### *4.3.2.2.4.3.42.10. table.col Objekt des Internet Explorer*

Spalte einer Tabelle
kann innerhalb von COLGROUP kodiert sein:

COL **erbt keine** gleichnamige Eigenschaften von COLGROUP
überschreibt gleichnamige Eigenschaften von COLGROUP

Beispiel:

```
<TABLE BORDER="2">
    <COL SPAN="2" STYLE="color:red">
    <COL STYLE="color:blue">
    .....
</TABLE>
```

**Eigenschaften:**

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste
.align Ausrichtung
ATOMICSELECTION Selektierbarkeit des Objektes einstellen
.bgColor deprecated und durch STYLE-Attribut zu ersetzen
.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf
.className Klassenreferenz, Klassenname
.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt
.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
.dir Umflussrichtung
.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich
.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.
.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.
.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich
.isMultiLine Mehrzeiligkeit des Objektinhaltes
.isTextEdit Erzeugbarkeit eines Textbereiches
.lang Sprache für Anzeige von Sonderzeichen etc.
.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.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
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden
.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)



.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetParent Referenz der Eltern
für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.offsetWidth X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound

.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar
schreiben: wenn Bereich gefüllt so komplett überschreiben
Plain-Text
HTML-Elemente möglich
nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.

.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

| | |
|---|---|
| "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 |

.scopeName Namensraum laut XMLNS-Attribut

.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes

.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes

.sourceIndex Index des Objektes in der Collection document.all

.span Anzahl der Spalten innerhalb einer Spaltengruppe einer Tabelle

Beispiel:

```
<TABLE BORDER="2">
    <COLGROUP SPAN="3" STYLE="color:green;background:black">
        <COL SPAN="2" STYLE="color:red">
    </COLGROUP>
....
</TABLE>
```

STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen

.tagName Tag-Bezeichner des Objektes

.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut

.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

.vAlign vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes

| | |
|---|---|
| "middle" | zentriert, Standard |
| "baseline" | Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes |
| "bottom" | am Fuss |
| "top" | am Kopf |

.width Breite des Objektes in Pixel
Integer in Pixels für absolute Breite, >=0
bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel
String Ziffernfolge eines Integer mit nachfolgendem % für Breite als Anteil der Breite des Elternobjektes
z.B. 10%

**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)

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

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

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

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

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

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

.fireEvent() ein Event auslösen

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

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

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

.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

.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

.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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein

.setAttribute() Wert von vorhandenem Attribut setzen
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
DOM wird nur bei Erzeugung geändert

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

.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

#### *4.3.2.2.4.3.42.11. table.colGroup Objekt des Internet Explorer*

Objekt zur Gruppierung von Spalten einer Tabelle
COL kann innerhalb von COLGROUP kodiert sein:

COL **erbt keine** gleichnamige Eigenschaften von COLGROUP
überschreibt gleichnamige Eigenschaften von COLGROUP
Bsp: Es sollte das SPAN-Attribut innerhalb von COL kodiert werden, wenn SPAN in COLGROUP mit einem anderen Wert hat als dem Standardwert des SPAN-Attributes von COL kodiert wurde.

Beispiel 1:

```
<TABLE BORDER="2" RULES="groups">
```



```
                <COLGROUP SPAN="2" STYLE="color:red">
                </COLGROUP>
                <COLGROUP STYLE="color:blue">
                </COLGROUP>
        ....
        </TABLE>
```

Beispiel 2:

```
        <TABLE BORDER="2">
                <COLGROUP SPAN="3" STYLE="color:green;background:black">
                        <COL SPAN="2" STYLE="color:red">
                </COLGROUP>
        ....
        </TABLE>
```

**Eigenschaften:**

| | |
|---|---|
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .className | Klassenreferenz, Klassenname |
| .clientHeight | Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt |
| .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 |
| .dir | Umflussrichtung |
| .firstChild | Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes |
| .id | Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)<br>ID-Attribut in HTML: Wert ist String alphanumerisch<br>muss mit Buchstaben beginnen<br>Unterstrich _ verwendbar<br>kann in " " bzw. ' ' kodiert werden, muss aber nicht<br>Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).<br>Zeiger aus ID bilden var Zeiger = eval(object.id);<br>Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen. |
| .innerHTML | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag<br>Plain-Text<br>HTML-Elemente je nach Objekt möglich<br>Script: muss mit DEFER-Attribut kodiert sein<br>schreiben: wenn Bereich nicht leer, so komplett überschreiben<br>wenn Bereich leer so einfügen<br>nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR. |
| .isContentEditable | Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) |
| .isDisabled | Interaktionsfähigkeit<br>nur wenn sichtbar so User-Interaktion möglich |
| .isMultiLine | Mehrzeiligkeit des Objektinhaltes |
| .isTextEdit | Erzeugbarkeit eines Textbereiches |
| .lang | Sprache für Anzeige von Sonderzeichen etc. |
| .lastChild | Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes |
| .nextSibling | Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes |
| .nodeName | String als Name des Kindes (Knoten, Node, Element)<br>also TAG-Bezeichner, Attribut-Name; #text für Anker |
| .nodeType | Knotentyp laut attributes Collection<br>1 für Element-Knoten.<br>3 für Textknoten.<br>oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden |
| .nodeValue | Knotenwert (Wert des Kindes, Node, Elementes)<br>nur für Text- und Attribut-Elemente<br>nicht für Element-Knoten (Knotentyp 1) |
| .offsetHeight | Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent) |
| .offsetLeft | X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent) |
| .offsetParent | Referenz der Eltern<br>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 rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)
.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar
schreiben: wenn Bereich gefüllt so komplett überschreiben
Plain-Text
HTML-Elemente möglich
nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.
.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

| | |
|---|---|
| "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 |

.scopeName Namensraum laut XMLNS-Attribut
.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex Index des Objektes in der Collection document.all
.span Anzahl der Spalten innerhalb einer Spaltengruppe einer Tabelle

Beispiel:

```
<TABLE BORDER="2">
    <COLGROUP SPAN="3" STYLE="color:green;background:black">
        <COL SPAN="2" STYLE="color:red">
    </COLGROUP>
....
</TABLE>
```

STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen
.tagName Tag-Bezeichner des Objektes
.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.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
.vAlign vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes

| | |
|---|---|
| "middle" | zentriert, Standard |
| "baseline" | Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes |
| "bottom" | am Fuss |
| "top" | am Kopf |

.width Breite des Objektes in Pixel
Integer in Pixels für absolute Breite, >=0
bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel
String Ziffernfolge eines Integer mit nachfolgendem % für Breite als Anteil der Breite des Elternobjektes
z.B. 10%

**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)

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

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

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

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

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

.fireEvent() ein Event auslösen

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

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

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

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

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

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

.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

.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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein

.setAttribute() Wert von vorhandenem Attribut setzen
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
DOM wird nur bei Erzeugung geändert

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

.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

#### *4.3.2.2.4.3.42.12. table.rows Collection des Internet Explorer*

referenziert alle Zeilen der Tabelle, egal wo diese liegen (THEAD etc.), in der Reihenfolge der Zeilen in der Tabelle
siehe auch Collection table.tBody.rows
table.tFoot.rows
table.tHead.rows
siehe auch Objekt table.tr und dort die Eigenschaften .rowIndex und .sectionRowIndex

**Syntax:**

[ var ZeigerAufFeld = ] zeiger_auf_tabelle.rows
[ var ZeigerAufFeldElement = ] zeiger_auf_tabelle.rows**[**Index**]**

Index Integer, ab 0
muss in **[ ]** kodiert werden

zeiger_auf_tabelle laut ID-Attribut

Beispiel:

```
var CollectionRows                  = zeiger_auf_tabelle.rows;
var AnzahlElementeInDer CollectionRows  = CollectionRows.length;
```



```
for (var i = 0; i < AnzahlElementeInDer CollectionRows; i ++)
{CollectionRows [i].style.fontWeight = "bold";}
```

**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<br>außer bei Formular mit <INPUT TYPE=image ...><br>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<br>siehe tags Collection des DOM |
| .urns() | Referenz auf Feld aller Elemente mit gemeinsamer URN liefern |

#### *4.3.2.2.4.3.42.13. table.rows.cells Collection des Internet Explorer*

referenziert alle Zellen genau einer Zeile laut Reihenfolge der Zellen in der Zeile
Jede Zeile ist eine Element in der Collection table.rows
hat eine eigene Collection table.rows.cells.
siehe auch Collection table.tBody.rows.cells
table.tFoot.rows.cells
table.tHead.rows.cells
siehe auch Objekt table.tr und dort die Eigenschaften .rowIndex und .sectionRowIndex

**Syntax:**

[ var ZeigerAufFeld = ] zeiger_auf_tabelle.rows**[**Index1**]**.cells
[ var ZeigerAufFeldElement = ] zeiger_auf_tabelle.rows**[**Index1**]**.cells**[**Index2**]**

Index1 Integer, ab 0
muss in **[ ]** kodiert werden

Index2 Integer, ab 0
muss in **[ ]** kodiert werden

zeiger_auf_tabelle laut ID-Attribut

[ var ZeigerAufFeld = ] zeiger_auf_zeile.cells
[ var ZeigerAufFeldElement = ] zeiger_auf_zeile.cells**[**Index**]**

Index Integer, ab 0
muss in **[ ]** kodiert werden

zeiger_auf_zeile laut ID-Attribut bzw. laut Erzeugung

Beispiel:

```
var CollectionRows                    = zeiger_auf_tabelle.rows;
var AnzahlElementeInDer CollectionRows  = CollectionRows.length;

for (var i = 0; i < AnzahlElementeInDer CollectionRows; i ++)
{
    var AktuelleZeile                 = CollectionRows [i];
    var AnzahlZellenInAktuelleZeile       = AktuelleZeile.length;

    for (var j = 0; j < AnzahlZellenInAktuelleZeile; j ++)
    {
        var AktuelleZelleDerZeile = AktuelleZeile.cells[j];

        AktuelleZelleDerZeile.style.width = "20";
    }
}
```

**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<br>außer bei Formular mit <INPUT TYPE=image ...><br>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<br>siehe tags Collection des DOM |
| .urns() | Referenz auf Feld aller Elemente mit gemeinsamer URN liefern |

#### *4.3.2.2.4.3.42.14. table.tBody Objekt des Internet Explorer*

Tabellenkörper TBODY
kann folgende innere Elemente als Kinder haben: TD, TH, TR
TBODY muss nicht kodiert werden, wenn keine Fehlzuordung zu THEAD **und** kein TFOOT möglich ist



Beispiel:

```
<TABLE>
    <THEAD>
    <TR>
        <TD>Head</TD>
    </TR>
    </THEAD>
    <TBODY>
    <TR>
        <TD>Body</TD>
    </TR>
    </TBODY>
</TABLE>
```

**Eigenschaften:**

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .background | Hintergrundbild eines Objektes |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .className | Klassenreferenz, Klassenname |
| .clientHeight | Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt |
| .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 |
| .dir | Umflussrichtung |
| .end | Objektaktivitäten laut Eigenschaft .timeAction beenden |
| .firstChild | Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes |
| .hasMedia | Objekt ist HTML-Media-Objekt |
| .hideFocus | Focussierbarkeit |
| .id | Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)<br>ID-Attribut in HTML: Wert ist String alphanumerisch<br>muss mit Buchstaben beginnen<br>Unterstrich _ verwendbar<br>kann in " " bzw. ' ' kodiert werden, muss aber nicht<br>Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).<br>Zeiger aus ID bilden var Zeiger = eval(object.id);<br>Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen. |
| .innerHTML | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag<br>Plain-Text<br>HTML-Elemente je nach Objekt möglich<br>Script: muss mit DEFER-Attribut kodiert sein<br>schreiben: wenn Bereich nicht leer, so komplett überschreiben<br>wenn Bereich leer so einfügen<br>nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR. |
| .innerText | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag<br>nur Plain-Text also keine HTML-Elemente und kein Script<br>schreiben: wenn Bereich nicht leer, so komplett überschreiben<br>wenn Bereich leer so einfügen<br>nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR |
| .isContentEditable | Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) |
| .isDisabled | Interaktionsfähigkeit<br>nur wenn sichtbar so User-Interaktion möglich |
| .isMultiLine | Mehrzeiligkeit des Objektinhaltes |
| .isTextEdit | Erzeugbarkeit eines Textbereiches |
| .lang | Sprache für Anzeige von Sonderzeichen etc. |
| .language | Sprache für Script festlegen |
| .lastChild | Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes |
| .nextSibling | Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes |
| .nodeName | String als Name des Kindes (Knoten, Node, Element) |



| | |
|---|---|
| | also TAG-Bezeichner, Attribut-Name; #text für Anker |
| .nodeType | Knotentyp laut attributes Collection |
| | 1 für Element-Knoten. |
| | 3 für Textknoten. |
| | oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden |
| .nodeValue | Knotenwert (Wert des Kindes, Node, Elementes) |
| | nur für Text- und Attribut-Elemente |
| | nicht für Element-Knoten (Knotentyp 1) |
| .offsetHeight | Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem |
| | des Elternobjektes (.offsetParent) |
| .offsetLeft | X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem |
| | des Elternobjektes (.offsetParent) |
| .offsetParent | Referenz der Eltern |
| | für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth |
| .offsetTop | Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem |
| | des Elternobjektes (.offestParent) |
| .offsetWidth | X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem |
| | des Elternobjektes (.offestParent) |
| .onOffBehavior | deprecated ab IE 5.x |
| | Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag |
| | wirksam mit parsen des Ende-Tag |
| | nur nach kompletten Einlesen des Dokumentes nutzbar |
| | schreiben: wenn Bereich gefüllt so komplett überschreiben |
| | Plain-Text |
| | HTML-Elemente möglich |
| | nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, |
| | THEAD, TR. |
| .outerText | Referenz auf den gesamten Plain-Text im Objekt |
| | nur nach kompletten einlesen des Dokumentes nutzbar |
| | nur Plain-Text |
| | schreiben: immer komplett überschreiben |
| | nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR |
| .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 |
| | "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 |
| .scopeName | Namensraum laut XMLNS-Attribut |
| .scrollHeight | Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren |
| | sichtbaren Rand des Umgebungsobjektes |
| .scrollLeft | Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes |
| | nur nutzbar nach dem kompletten Laden des Dokumentes |
| | immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollTop | Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes |
| | nur nutzbar nach dem kompletten Laden des Dokumentes |
| | immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollWidth | Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten |
| | sichtbaren Rand des Umgebungsobjektes |
| .sourceIndex | Index des Objektes in der Collection document.all |
| STYLE | direkt im HTML-Element kodierter Style (Inline-Style) |
| | Hinweis: für Scripting ist das Style-Objekt zu nutzen |
| .syncMaster | Synchronisierung der Animation des im Container liegenden Elementes auf Timeline |
| .systemBitrate | wird hier nicht erklärt |
| .systemCaptions | wird hier nicht erklärt |
| .systemLanguage | Sprache festlegen für das Objekt |
| .systemOverdubOrSubtitle | wird hier nicht erklärt |
| .tabIndex | Index des Elementes in der Tab-Tasten-Folge |
| .tagName | Tag-Bezeichner des Objektes |
| .tagUrn | Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut |
| .timeContainer | Typ der Timeline des Objektes |
| | siehe Objekt currTimeState und Behavior .style.time2 |
| .title | Tooltip-Text bei Mouse over über Objekt |
| .uniqueID | durch den Browser automatisch-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 | Selektionsfähigkeit eines Objektes |
| .vAlign | vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes |



| | | |
|---|---|---|
| | "middle" | zentriert, Standard |
| | "baseline" | Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes |
| | "bottom" | am Fuss |
| | "top" | am Kopf |

**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 |
| .deleteRow() | Zeile löschen aus Tabelle |
| | siehe Objekt table |
| | siehe Objekt table.tBody |
| | siehe Objekt table.tHead |
| | siehe Objekt table.tFoot |
| .deleteTFoot() | TFOOT der Tabelle löschen |
| | es darf nur 1 TFOOT zur Tabelle existieren |
| | siehe Objekt table |
| | siehe Objekt table.tBody |
| .deleteTHead() | THEAD der Tabelle löschen |
| | es darf nur 1 THEAD zur Tabelle existieren |
| | siehe Objekt table |
| | siehe Objekt table.tBody |
| .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) |
| .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

.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

.insertRow() Zeile in die Tabelle einfügen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.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

.moveRow() 2 Zeilen in der Tabelle austauschen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
 <TR>
  <TD>Body Zeile 1 Zelle 1</TD>
  <TD>Body Zeile 1 Zelle 2</TD>
 </TR>
 <TR>
  <TD>Body Zeile 2 Zelle 1</TD>
  <TD>Body Zeile 2 Zelle 2</TD>
 </TR>
 <TR>
  <TD>Body Zeile 3 Zelle 1</TD>
  <TD>Body Zeile 3 Zelle 2</TD>
 </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.moveRow(0,1);">verschieben Zeile 0 auf Zeile 1</BUTTON>
```

.normalize() Normalisierung des DOM zur Erreichung einer konsistenten Struktur



| | |
|---|---|
| | Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen |
| .releaseCapture() | Maus-Überwachung ausschalten für ein Objekt<br>Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.<br>Hinweis: einschalten per Methode .setCapture() |
| .removeAttribute() | entfernen eines per HTML erzeugten Attributes<br>Achtung: Der Browser unterscheidet zwischen HTML-erzeugte oder mit dieser Methode erzeugte Attribute! per Methode .createAttribute() erzeugte Attribute werden nicht erfasst<br>DOM wird geändert |
| .removeAttributeNode() | entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern<br>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)<br>DOM wird geändert |
| .removeChild() | Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern<br>Sichtbarkeit erst, wenn Ende-Tag geparst wurde, also das Dokument neu geladen wurde<br>DOM wird geändert |
| .removeExpression() | Ausdruck entfernen, der für die Berechnung des Wertes einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient.<br>Ausdruck muss mit der Methode .setExpression() gesetzt worden sein<br>DOM wird nicht geändert |
| .removeNode() | Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern<br>Sichtbarkeit erst wenn Ende-Tag geparst wurde<br>DOM wird geändert |
| .replaceAdjacentText() | Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern<br>DOM wird nicht geändert |
| .replaceChild() | Kind-Objekt ersetzen durch ein Objekt<br>ersetzende Objekt muss per Methode .createElement() erzeugt worden sein<br>Sichtbarkeit erst wenn Ende-Tag geparst wurde<br>DOM wird geändert |
| .replaceNode() | Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern<br>sichtbar erst mit parsen des Endetags<br>DOM wird geändert |
| .scrollIntoView() | Objekt derart scrollen, dass es im Fenster für User sichtbar wird<br>Objekt muss an sich schon renderbar sein |
| .setActive() | Objekt für die Eventdurchreichung aktivieren<br>aber ohne es zu fokussieren<br>und ohne es scrollbar zu machen |
| .setAttribute() | Wert von vorhandenem Attribut setzen<br>wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt<br>DOM wird nur bei Erzeugung geändert |
| .setAttributeNode() | Attribut einem Knoten zuweisen und Referenz liefern<br>DOM wird geändert |
| .setCapture() | Maus-Überwachung einschalten für ein Objekt<br>Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout.<br>ab IE 5.5<br>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<br>Ausdruck nur als Script kodierbar<br>DOM wird nicht geändert |
| .swapNode() | Positionen von 2 Knoten im DOM tauschen (Zeigertausch)<br>nur sichtbar wenn Endetag geparst<br>DOM wird geändert |

#### *4.3.2.2.4.3.42.14.1. table.tBody.rows Collection des Internet Explorer*

referenziert alle Zeilen der Tabelle im TBODY
laut Reihenfolge der Zeilen im TBODY

ist eine interne Collection, die nur über die Eigenschaft .sectionRowIndex des Objektes table.tr durch Lesen ansprechbar ist (siehe dort):
Diese Eigenschaft liefert den Index der Zeile in dieser Collection.

wird nur erzeugt, wenn TBODY mit Zeilen erzeugt wurde (z.B. per TBODY-Tag)

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)

#### *4.3.2.2.4.3.42.14.2. table.tBody.rows.cells Collection des Internet Explorer*

referenziert alle Zellen genau **einer** Zeile aus TBODY
laut Reihenfolge der Zellen in der Zeile



ist eine interne Collection, die nur über die Collection table.tBody.rows ansprechbar ist, welche die Zeile im TBODY indirekt referenziert
(siehe dort)

wird nur erzeugt, wenn TBODY mit Zeilen und Zellen erzeugt wurde (z.B. per TBODY-Tag)

Jede Zeile ist ein Element in der Collection table.tBody.rows
hat eine eigene Collection table.tBody.rows.cells.

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)
siehe auch Collection table.rows .cells (Indexe aller Zellen einer Zeile der Tabelle, egal wo die Zeile liegt)

#### *4.3.2.2.4.3.42.15. table.tBodies Collection des Internet Explorer*

referenziert alle TBODY der Tabelle (also nicht THEAD und TFOOT) in der Reihenfolge der TBODY-Elemente in der Tabelle.
Die Erzeugung von TBODY ist im TOM **nicht implementiert**. Dafür ist DOM zu verwenden.

**Syntax:**

[ var ZeigerAufFeld = ] zeiger_auf_tabelle.tBodies
[ var ZeigerAufFeldElement = ] zeiger_auf_tabelle.tBodies**[**Index**]**

Index Integer, ab 0
muss in **[ ]** kodiert werden

zeiger_auf_tabelle laut ID-Attribut

**Eigenschaften:**

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

**Methoden:**

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des
Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>
da dafür die children-Collection verwendet werden muss !

.namedItem() Referenz auf Eintrag (FeldElement) anhand des Namen
(analog zu ID oder NAME-Attribut) liefern

.tags() Referenz auf Feld aller HTML-Elemente mit gemeinsamen Tag-Namen liefern
siehe tags Collection des DOM

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

#### *4.3.2.2.4.3.42.16. table.tFoot Objekt des Internet Explorer*

Tabellenfuss TFOOT
es kann maximal nur 1 TFOOT existieren
kann folgende innere Elemente als Kinder haben: TD, TH, TR
Beispiel:

```
<TABLE>
<TBODY>
    <TR>
        <TD>Koerper</TD>
    </TR>
</TBODY>
<TFOOT>
    <TR>
        <TD>Fuss</TD>
    </TR>
</TFOOT>
</TABLE>
```

**Eigenschaften:**

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste
.align Ausrichtung
ATOMICSELECTION Selektierbarkeit des Objektes einstellen
.begin Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction
siehe Objekt currTimeState und Behavior .style.time2
.bgColor deprecated und durch STYLE-Attribut zu ersetzen
.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf
.className Klassenreferenz, Klassenname
.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt
.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
.dir Umflussrichtung
.end Objektaktivitäten laut Eigenschaft .timeAction beenden
.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia Objekt ist HTML-Media-Objekt
.hideFocus Focussierbarkeit
.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen



Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht

Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).

Zeiger aus ID bilden var Zeiger = eval(object.id);

Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag
nur Plain-Text also keine HTML-Elemente und kein Script
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetParent Referenz der Eltern
für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.offsetWidth X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound

.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar
schreiben: wenn Bereich gefüllt so komplett überschreiben
Plain-Text
HTML-Elemente möglich
nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.

.outerText Referenz auf den gesamten Plain-Text im Objekt
nur nach kompletten einlesen des Dokumentes nutzbar
nur Plain-Text
schreiben: immer komplett überschreiben
nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR

.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
"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
.scopeName Namensraum laut XMLNS-Attribut
.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex Index des Objektes in der Collection document.all
STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster Synchronisierung der Animation des im Container liegenden Elementes auf Timeline
.systemBitrate wird hier nicht erklärt
.systemCaptions wird hier nicht erklärt
.systemLanguage Sprache festlegen für das Objekt
.systemOverdubOrSubtitle wird hier nicht erklärt
.tabIndex Index des Elementes in der Tab-Tasten-Folge
.tagName Tag-Bezeichner des Objektes
.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer Typ der Timeline des Objektes
siehe Objekt currTimeState und Behavior .style.time2
.title Tooltip-Text bei Mouse over über Objekt
.uniqueID durch den Browser automatisch-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 Selektionsfähigkeit eines Objektes
.vAlign vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes
"middle" zentriert, Standard
"baseline" Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes
"bottom" am Fuss
"top" am Kopf

**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

.deleteRow() Zeile löschen aus Tabelle
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.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)

.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

.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

.insertRow() Zeile in die Tabelle einfügen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead



siehe Objekt table.tFoot

.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

.moveRow() 2 Zeilen in der Tabelle austauschen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
    <TR>
        <TD>Body Zeile 1 Zelle 1</TD>
        <TD>Body Zeile 1 Zelle 2</TD>
    </TR>
    <TR>
        <TD>Body Zeile 2 Zelle 1</TD>
        <TD>Body Zeile 2 Zelle 2</TD>
    </TR>
    <TR>
        <TD>Body Zeile 3 Zelle 1</TD>
        <TD>Body Zeile 3 Zelle 2</TD>
    </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.moveRow(0,1);">verschieben Zeile 0 auf Zeile 1</BUTTON>
```

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

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

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

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

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

.removeChild() Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
Sichtbarkeit erst, wenn Ende-Tag 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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein

.setActive() Objekt für die Eventdurchreichung aktivieren
aber ohne es zu fokussieren
und ohne es scrollbar zu machen

.setAttribute() Wert von vorhandenem Attribut setzen
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
DOM wird nur bei Erzeugung geändert

.setAttributeNode() Attribut einem Knoten zuweisen und Referenz liefern



DOM wird geändert

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

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

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

#### *4.3.2.2.4.3.42.16.1. table.tFoot.rows Collection des Internet Explorer*

referenziert alle Zeilen der Tabelle im TFOOT
laut Reihenfolge der Zeilen im TFOOT

ist eine interne Collection, die nur über die Eigenschaft .sectionRowIndex des Objektes table.tr durch Lesen ansprechbar ist (siehe dort):
Diese Eigenschaft liefert den Index der Zeile in dieser Collection.

wird nur erzeugt, wenn TFOOT mit Zeilen erzeugt wurde (z.B. per TFOOT-Tag)

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)

#### *4.3.2.2.4.3.42.16.2. table.tFoot.rows.cells Collection des Internet Explorer*

referenziert alle Zellen genau **einer** Zeile aus TFOOT
laut Reihenfolge der Zellen in der Zeile

ist eine interne Collection, die nur über die Collection table.tFoot.rows ansprechbar ist, welche die Zeile im TFOOT indirekt referenziert (siehe dort)

wird nur erzeugt, wenn TFOOT mit Zeilen und Zellen erzeugt wurde (z.B. per TFOOT-Tag)

Jede Zeile ist eine Element in der Collection table.tFoot.rows
hat eine eigene Collection table.tFoot.rows.cells.

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)
siehe auch Collection table.rows .cells (Indexe aller Zellen einer Zeile der Tabelle, egal wo die Zeile liegt)

### *4.3.2.2.4.3.42.17. table.tHead Objekt des Internet Explorer*

Tabellenkopf THEAD
es kann maximal nur 1 THEAD existieren
kann folgende innere Elemente als Kinder haben: TD, TH, TR
Beispiel:

```
<TABLE>
<THEAD>
    <TR>
        <TD>Kopf</TD>
    </TR>
</THEAD>
<TBODY>
    <TR>
        <TD>Body</TD>
    </TR>
</TBODY>
</TABLE>
```

**Eigenschaften:**

.accessKey Tastaturzugriff auf ein Objekt per Alt + Taste
.align Ausrichtung
ATOMICSELECTION Selektierbarkeit des Objektes einstellen
.begin Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction
siehe Objekt currTimeState und Behavior .style.time2
.bgColor deprecated und durch STYLE-Attribut zu ersetzen
.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf
.className Klassenreferenz, Klassenname
.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt
.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
.dir Umflussrichtung



| | |
|---|---|
| .end | Objektaktivitäten laut Eigenschaft .timeAction beenden |
| .firstChild | Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes |
| .hasMedia | Objekt ist HTML-Media-Objekt |
| .hideFocus | Focussierbarkeit |
| .id | Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)<br>ID-Attribut in HTML: Wert ist String alphanumerisch<br>muss mit Buchstaben beginnen<br>Unterstrich _ verwendbar<br>kann in " " bzw. ' ' kodiert werden, muss aber nicht<br>Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).<br>Zeiger aus ID bilden var Zeiger = eval(object.id);<br>Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen. |
| .innerHTML | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag<br>Plain-Text<br>HTML-Elemente je nach Objekt möglich<br>Script: muss mit DEFER-Attribut kodiert sein<br>schreiben: wenn Bereich nicht leer, so komplett überschreiben<br>wenn Bereich leer so einfügen<br>nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR. |
| .innerText | Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag<br>ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B><br>dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,<br>also nach dem Laden des Objektes<br>aber wirksam erst mit parsen des HTML-Endetag<br>nur Plain-Text also keine HTML-Elemente und kein Script<br>schreiben: wenn Bereich nicht leer, so komplett überschreiben<br>wenn Bereich leer so einfügen<br>nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR |
| .isContentEditable | Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) |
| .isDisabled | Interaktionsfähigkeit<br>nur wenn sichtbar so User-Interaktion möglich |
| .isMultiLine | Mehrzeiligkeit des Objektinhaltes |
| .isTextEdit | Erzeugbarkeit eines Textbereiches |
| .lang | Sprache für Anzeige von Sonderzeichen etc. |
| .language | Sprache für Script festlegen |
| .lastChild | Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes |
| .nextSibling | Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes |
| .nodeName | String als Name des Kindes (Knoten, Node, Element)<br>also TAG-Bezeichner, Attribut-Name; #text für Anker |
| .nodeType | Knotentyp laut attributes Collection<br>1 für Element-Knoten.<br>3 für Textknoten.<br>oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden |
| .nodeValue | Knotenwert (Wert des Kindes, Node, Elementes)<br>nur für Text- und Attribut-Elemente<br>nicht für Element-Knoten (Knotentyp 1) |
| .offsetHeight | Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent) |
| .offsetLeft | X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent) |
| .offsetParent | Referenz der Eltern<br>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 rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent) |
| .onOffBehavior | deprecated ab IE 5.x<br>Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag<br>wirksam mit parsen des Ende-Tag<br>nur nach kompletten Einlesen des Dokumentes nutzbar<br>schreiben: wenn Bereich gefüllt so komplett überschreiben<br>Plain-Text<br>HTML-Elemente möglich<br>nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR. |
| .outerText | Referenz auf den gesamten Plain-Text im Objekt |



nur nach kompletten einlesen des Dokumentes nutzbar
nur Plain-Text
schreiben: immer komplett überschreiben
nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR

.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

| | |
|---|---|
| "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 |

.scopeName Namensraum laut XMLNS-Attribut
.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind
.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex Index des Objektes in der Collection document.all
STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen
.syncMaster Synchronisierung der Animation des im Container liegenden Elementes auf Timeline
.systemBitrate wird hier nicht erklärt
.systemCaptions wird hier nicht erklärt
.systemLanguage Sprache festlegen für das Objekt
.systemOverdubOrSubtitle wird hier nicht erklärt
.tabIndex Index des Elementes in der Tab-Tasten-Folge
.tagName Tag-Bezeichner des Objektes
.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.timeContainer Typ der Timeline des Objektes
siehe Objekt currTimeState und Behavior .style.time2
.title Tooltip-Text bei Mouse over über Objekt
.uniqueID durch den Browser automatisch-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 Selektionsfähigkeit eines Objektes
.vAlign vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes

| | |
|---|---|
| "middle" | zentriert, Standard |
| "baseline" | Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes |
| "bottom" | am Fuss |
| "top" | am Kopf |

**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

.deleteRow() Zeile löschen aus Tabelle
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.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)

.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

.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

.insertRow() Zeile in die Tabelle einfügen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

.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

.moveRow() 2 Zeilen in der Tabelle austauschen
siehe Objekt table
siehe Objekt table.tBody
siehe Objekt table.tHead
siehe Objekt table.tFoot

Beispiel:

```
<TABLE ID="ID_Tabelle" BORDER CELLSPACING=10>
 <TR>
  <TD>Body Zeile 1 Zelle 1</TD>
  <TD>Body Zeile 1 Zelle 2</TD>
 </TR>
 <TR>
  <TD>Body Zeile 2 Zelle 1</TD>
  <TD>Body Zeile 2 Zelle 2</TD>
 </TR>
 <TR>
  <TD>Body Zeile 3 Zelle 1</TD>
  <TD>Body Zeile 3 Zelle 2</TD>
 </TR>
</TABLE>
<BUTTON onclick="ID_Tabelle.moveRow(0,1);">verschieben Zeile 0 auf Zeile 1</BUTTON>
```

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

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

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

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

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

.removeChild() Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
Sichtbarkeit erst, wenn Ende-Tag 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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein



| | |
|---|---|
| .setActive() | Objekt für die Eventdurchreichung aktivieren<br>aber ohne es zu fokussieren<br>und ohne es scrollbar zu machen |
| .setAttribute() | Wert von vorhandenem Attribut setzen<br>wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt<br>DOM wird nur bei Erzeugung geändert |
| .setAttributeNode() | Attribut einem Knoten zuweisen und Referenz liefern<br>DOM wird geändert |
| .setCapture() | Maus-Überwachung einschalten für ein Objekt<br>Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,<br>onmouseover und onmouseout.<br>ab IE 5.5<br>Hinweis: ausschalten per Methode .releaseCapture() |
| .setExpression() | Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-<br>Eigenschaft als Objektreferenz der Form<br>objekt.style.eigenschaft.<br>dient<br>Ausdruck nur als Script kodierbar<br>DOM wird nicht geändert |
| .swapNode() | Positionen von 2 Knoten im DOM tauschen (Zeigertausch)<br>nur sichtbar wenn Endetag geparst<br>DOM wird geändert |

#### *4.3.2.2.4.3.42.17.1. table.tHead.rows Collection des Internet Explorer*

referenziert alle Zeilen der Tabelle im THEAD
laut Reihenfolge der Zeilen im THEAD

ist eine interne Collection, die nur über die Eigenschaft .sectionRowIndex des Objektes table.tr durch Lesen ansprechbar ist (siehe dort):
Diese Eigenschaft liefert den Index der Zeile in dieser Collection.

wird nur erzeugt, wenn THEAD mit Zeilen erzeugt wurde (z.B. per THEAD-Tag)

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)

#### *4.3.2.2.4.3.42.17.2. table.tHead.rows.cells Collection des Internet Explorer*

referenziert alle Zellen genau **einer** Zeile aus THEAD
laut Reihenfolge der Zellen in der Zeile

ist eine interne Collection, die nur über die Collection table.tHead.rows ansprechbar ist, welche die Zeile im THEAD indirekt referenziert (siehe dort)

wird nur erzeugt, wenn THEAD mit Zeilen und Zellen erzeugt wurde (z.B. per THEAD-Tag)

Jede Zeile ist eine Element in der Collection table.tHead.rows
hat eine eigene Collection table.tHead.rows.cells.

siehe auch Collection table.rows (Indexe aller Zeilen der Tabelle, egal wo sie liegen)
siehe auch Collection table.rows .cells (Indexe aller Zellen einer Zeile der Tabelle, egal wo die Zeile liegt)

### *4.3.2.2.4.3.42.18. table.tr Objekt des Internet Explorer*

Tabellenzeile TR
enthält alle Zellen der Zeile
Die Zeilen innerhalb der Tabelle werden in der Collection table.rows referenziert, wobei der Index der Collection die Zeilennummer ab 0 darstellt.
pro Zeile eine eigene Collection table.rows.cells
Beispiel 1:

```
<TABLE>
    <TR>
        <TD>Zelle1</TD>
    </TR>
    <TR>
        <TD>Zelle2</TD>
    </TR>
</TABLE>
```

Beispiel 2:

```
<SCRIPT>
    function ZeilenErzeugen()
    {
        // +++++   Collection aller Tabellenzeilen adressieren
        var TabellenZeilenCollection = ID_Tabelle.rows;

        // +++++   Zeile 1 erzeugen durch anhängen
        // -----   aktuelle Anzahl der Tabellenzeilen ermitteln
```



```
                        var AnzahlTabellenZeilen = TabellenZeilenCollection.length;
                        // -----    Zeile 1 in Tabelle einbinden sowie der Collection anhängen
                        var TabellenZeile1 = ID_Tabelle.insertRow(AnzahlTabellenZeilen);

                        // +++++   Zeile 2 erzeugen durch anhängen
                        // -----    aktuelle Anzahl der Tabellenzeilen ermitteln
                        AnzahlTabellenZeilen = TabellenZeilenCollection.length;
                        //-----    Zeile 2 in Tabelle einbinden sowie der Collection anhängen
                        var TabellenZeile2= ID_Tabelle.insertRow(AnzahlTabellenZeilen);

                        // +++++   Zeile 1 mit 2 Zellen versorgen
                        //  -----    Collection der Zellen adressieren
                        var TabellenZeile_ZellenCollection = TabellenZeile1.cells;

                        // -----    Zelle 1 in die Tabellenzeile 1 einfügen und der Collection anhängen
                        var AktuellerIndexDerZelle    = TabellenZeile1.rowIndex;
                        var AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        var AktuelleAnzahlZellen      = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile1_Zelle1     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // -----    Zelle 2 in die Tabellenzeile 1einfügen und der Collection anhängen
                        AktuellerIndexDerZelle    = TabellenZeile1.rowIndex;
                        AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen              = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile1_Zelle2     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // +++++   Zeile 2 mit 2 Zellen versorgen
                        //  -----    Collection der Zellen adressieren
                        var TabellenZeile_ZellenCollection = TabellenZeile2.cells;

                        // -----    Zelle 1 in die Tabellenzeile 2 einfügen und der Collection anhängen
                        AktuellerIndexDerZelle    = TabellenZeile2.rowIndex;
                        AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen              = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle1     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // -----    Zelle 2 in die Tabellenzeile 2einfügen und der Collection anhängen
                        AktuellerIndexDerZelle    = TabellenZeile2.rowIndex;
                        AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen              = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle2     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // alle Zellen mit Daten versorgen, die sofort angezeigt werden (kein Tabellen-Refresh nötig)
                        TabellenZeile1_Zelle1.innerHTML="<B>Zeile1 Zelle1</B>";
                        TabellenZeile1_Zelle2.innerHTML="<B>Zeile1 Zelle2</B>";
                        TabellenZeile2_Zelle1.innerHTML="<B>Zeile2 Zelle1</B>";
                        TabellenZeile2_Zelle2.innerHTML="<B>Zeile2 Zelle2</B>";
                }
        </SCRIPT>
        <INPUT TYPE="button" VALUE="Zeilen erzeugen" onclick="ZeilenErzeugen();">
        <TABLE ID="ID_Tabelle" BORDER=1>
        </TABLE>
```

**Eigenschaften:**

Folgende Eigenschaften der Zeile liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index des Zeile im DOM |
| .rowIndex | Index der Zeile bezüglich Tabelle |
| .sectionRowIndex | Index der Zeile bezüglich Tabellenelement wie THEAD, TBODY bzw. TFOOT<br>nur für TD, nicht für TH |

Folgende Eigenschaften der Zelle liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index der Zelle im DOM |
| .cellIndex Index | Index der Zelle innerhalb der Zeile |

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .borderColor | Borderfarbe (Rahmenfarbe)<br>wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"<br>Eigenschaft .border mit Wert 0<br>#rrggbb<br>vordefinierter Farbname (browserspezifisch) |



.borderColorDark deprecated und durch Eigenschaft .borderColor zu ersetzen
dunkle Farbe des 3D-Rahmens eines Objektes
zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border)

.borderColorLight deprecated und durch Eigenschaft .borderColor zu ersetzen
helle Farbe des 3D-Rahmens eines Objektes
zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft boder)

.canHaveChildren prüfen ob Kind möglich ist, also ob Objekt Parent sein kann

.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf

.className Klassenreferenz, Klassenname

.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt

.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

.dir Umflussrichtung

.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich
true Element nicht interaktionsfähig
false Default, Element ist interaktionfähig

.end Objektaktivitäten laut Eigenschaft .timeAction beenden

.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

.hasMedia Objekt ist HTML-Media-Objekt

.height Höhe des Objektes in Pixel

.hideFocus Focussierbarkeit

.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag
nur Plain-Text also keine HTML-Elemente und kein Script
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)



| | |
|---|---|
| .offsetLeft | X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent) |
| .offsetParent | Referenz der Eltern<br>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 rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent) |
| .onOffBehavior | deprecated ab IE 5.x<br>Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag<br>wirksam mit parsen des Ende-Tag<br>nur nach kompletten Einlesen des Dokumentes nutzbar<br>schreiben: wenn Bereich gefüllt so komplett überschreiben<br>Plain-Text<br>HTML-Elemente möglich<br>nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR. |
| .outerText | Referenz auf den gesamten Plain-Text im Objekt<br>nur nach kompletten einlesen des Dokumentes nutzbar<br>nur Plain-Text<br>schreiben: immer komplett überschreiben<br>nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR |
| .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<br>"uninitialized" Objekt ist nicht initialisiert<br>"loading" Objekt ist nicht intialisiert aber lädt gerade Daten zur Initialisierung<br>"loaded" Objekt hat alle Daten komplett geladen und ist komplett intitialisiert<br>"interactive" Objekt kann vom User interaktiv verwendet werden zum Füllen mit Daten<br>"complete" Object hat alle Daten geladen und ist mit diesen komplett initialisiert |
| .rowIndex | Index der Tabellenzeile in der Collection table.rows |
| .scopeName | Namensraum laut XMLNS-Attribut |
| .scrollHeight | Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes |
| .scrollLeft | Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollTop | Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollWidth | Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes |
| .sectionRowIndex | Index der Tabellenzeile in der Collection<br>table.tBody.rows<br>bzw. table.tFoot.rows<br>bzw. table.tHead.rows<br>Zeile muss im **existierenden** Tabellenteil TOBDY bzw. TFOOT bzw. THEAD liegen |
| .sourceIndex | Index des Objektes in der Collection document.all |
| STYLE | direkt im HTML-Element kodierter Style (Inline-Style)<br>Hinweis: für Scripting ist das Style-Objekt zu nutzen |
| .syncMaster | Synchronisierung der Animation des im Container liegenden Elementes auf Timeline |
| .systemBitrate | wird hier nicht erklärt |
| .systemCaptions | wird hier nicht erklärt |
| .systemLanguage | Sprache festlegen für das Objekt |
| .systemOverdubOrSubtitle | wird hier nicht erklärt |
| .tabIndex | Index des Elementes in der Tab-Tasten-Folge |
| .tagName | Tag-Bezeichner des Objektes |
| .tagUrn | Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut |
| .timeContainer | Typ der Timeline des Objektes<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .title | Tooltip-Text bei Mouse over über Objekt |
| .uniqueID | durch den Browser automatisch-genriertes ID des Objektes<br>Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde<br>kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden |
| .vAlign | vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes<br>"middle" zentriert, Standard<br>"baseline" Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes<br>"bottom" am Fuss<br>"top" am Kopf |



.width — Breite des Objektes in Pixel
Integer in Pixels für absolute Breite, >=0
bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel
String Ziffernfolge eines Integer mit nachfolgendem % für Breite als
Anteil der Breite des Elternobjektes
z.B. 10%

**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

.deleteCell() — Zelle in die Zeile einer Tabelle löschen

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

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

.fireEvent() — ein Event auslösen

.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

.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

.insertCell() Zelle TD in die Zeile einer Tabelle einfügen
Zelle TH **nicht direkt** erzeugbar: Es muss .innerHTML mit <B>-Tag gefüllt werden

Beispiel :

```
var Zeile = zeiger_auf_tabelle.insertRow();
var Zelle = Zeile.insertCell();
Zelle.innerHTML = " <I>Test </I>";
```

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

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

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

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

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

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

.removeChild() Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
Sichtbarkeit erst, wenn Ende-Tag 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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein

.setActive() Objekt für die Eventdurchreichung aktivieren
aber ohne es zu fokussieren
und ohne es scrollbar zu machen

.setAttribute() Wert von vorhandenem Attribut setzen
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
DOM wird nur bei Erzeugung geändert

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

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

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

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

#### *4.3.2.2.4.3.42.18.1. table.tr.td Objekt des Internet Explorer*

Zelle TD einer Tabellenzeile
enthält die Daten
Beispiel 1:

```
<TABLE>
    <TR>
        <TD>Zelle1</TD>
    </TR>
    <TR>
        <TD>Zelle2</TD>
    </TR>
</TABLE>
```

Beispiel 2:

```
<SCRIPT>
    function ZeilenErzeugen()
    {
        // +++++   Collection aller Tabellenzeilen adressieren
        var TabellenZeilenCollection = ID_Tabelle.rows;

        // +++++   Zeile 1 erzeugen durch anhängen
        // -----   aktuelle Anzahl der Tabellenzeilen ermitteln
        var AnzahlTabellenZeilen = TabellenZeilenCollection.length;
        // -----   Zeile 1 in Tabelle einbinden sowie der Collection anhängen
        var TabellenZeile1 = ID_Tabelle.insertRow(AnzahlTabellenZeilen);

        // +++++   Zeile 2 erzeugen durch anhängen
        // -----   aktuelle Anzahl der Tabellenzeilen ermitteln
        AnzahlTabellenZeilen = TabellenZeilenCollection.length;
        //-----   Zeile 2 in Tabelle einbinden sowie der Collection anhängen
        var TabellenZeile2= ID_Tabelle.insertRow(AnzahlTabellenZeilen);

        // +++++   Zeile 1 mit 2 Zellen versorgen
        //  -----   Collection der Zellen adressieren
        var TabellenZeile_ZellenCollection = TabellenZeile1.cells;

        // -----   Zelle 1 in die Tabellenzeile 1 einfügen und der Collection anhängen
        var AktuellerIndexDerZelle    = TabellenZeile1.rowIndex;
        var AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
        var AktuelleAnzahlZellen      = TabellenZeile_ZellenCollection.length;
        var TabellenZeile1_Zelle1     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

        // -----   Zelle 2 in die Tabellenzeile 1einfügen und der Collection anhängen
        AktuellerIndexDerZelle        = TabellenZeile1.rowIndex;
        AktuelleZelle                 = TabellenZeilenCollection(AktuellerIndexDerZelle);
```



```
                        AktuelleAnzahlZellen                    = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile1_Zelle2       = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // +++++   Zeile 2 mit 2 Zellen versorgen
                        //  -----      Collection der Zellen adressieren
                        var TabellenZeile_ZellenCollection = TabellenZeile2.cells;

                        // -----      Zelle 1 in die Tabellenzeile 2 einfügen und der Collection anhängen
                        AktuellerIndexDerZelle          = TabellenZeile2.rowIndex;
                        AktuelleZelle                   = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen                    = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle1       = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // -----      Zelle 2 in die Tabellenzeile 2einfügen und der Collection anhängen
                        AktuellerIndexDerZelle          = TabellenZeile2.rowIndex;
                        AktuelleZelle                   = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen                    = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle2       = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // alle Zellen mit Daten versorgen, die sofort angezeigt werden (kein Tabellen-Refresh nötig)
                        TabellenZeile1_Zelle1.innerHTML="Zeile1 Zelle1";
                        TabellenZeile1_Zelle2.innerHTML="Zeile1 Zelle2";
                        TabellenZeile2_Zelle1.innerHTML="Zeile2 Zelle1";
                        TabellenZeile2_Zelle2.innerHTML="Zeile2 Zelle2";
                }
        </SCRIPT>
        <INPUT TYPE="button" VALUE="Zeilen erzeugen" onclick="ZeilenErzeugen();">
        <TABLE ID="ID_Tabelle" BORDER=1>
        </TABLE>
```

**Eigenschaften:**

Eigenschaften .innerText und .innerHTML sind les- **und** schreibbar
Eigenschaft .style verfügbar

Folgende Eigenschaften der Zelle liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index der Zelle im DOM |
| .cellIndex Index | Index der Zelle innerhalb der Zeile |

Folgende Eigenschaften der Zeile liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index der Zeile im DOM |
| .rowIndex | Index der Zeile bezüglich Tabelle |
| .sectionRowIndex | Index der Zeile bezüglich Tabellenelement wie THEAD, TBODY bzw. TFOOT<br>nur für TD, nicht für TH |

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .background | Hintergrundbild eines Objektes |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .borderColor | Borderfarbe (Rahmenfarbe)<br>wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"<br>Eigenschaft .border mit Wert 0<br>#rrggbb<br>vordefinierter Farbname (browserspezifisch) |
| .borderColorDark | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>dunkle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border) |
| .borderColorLight | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>helle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft boder) |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .cellIndex | Index der Zelle in der Collection table.rows.cells<br>siehe Objekt table.tr.td<br>siehe Objekt table.tr.th |
| .className | Klassenreferenz, Klassenname |
| .clientHeight | Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt |
| .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 |
| .colSpan | Anzahl der Spalten einer Zelle in einer Tabelle<br>siehe Objekt table.tr.td<br>siehe Objekt table.tr.th |



.dir Umflussrichtung
.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich
true Element nicht interaktionsfähig
false Default, Element ist interaktionfähig
.end Objektaktivitäten laut Eigenschaft .timeAction beenden
.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia Objekt ist HTML-Media-Objekt
.height Höhe des Objektes in Pixel
.hideFocus Focussierbarkeit
.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);
Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.
.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
Plain-Text
HTML-Elemente je nach Objekt möglich
Script: muss mit DEFER-Attribut kodiert sein
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.
.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>
dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag
nur Plain-Text also keine HTML-Elemente und kein Script
schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen
nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR
.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)
.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich
.isMultiLine Mehrzeiligkeit des Objektinhaltes
.isTextEdit Erzeugbarkeit eines Textbereiches
.lang Sprache für Anzeige von Sonderzeichen etc.
.language Sprache für Script festlegen
.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType Knotentyp laut attributes Collection
1 für Element-Knoten.
3 für Textknoten.
oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden
.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)
.noWrap Wortumbruch einstellen
false Default.
Browser bricht den Text automatisch um
true Browser bricht den Text nicht um
.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 rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)
.onOffBehavior deprecated ab IE 5.x



Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound

.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar
schreiben: wenn Bereich gefüllt so komplett überschreiben
Plain-Text
HTML-Elemente möglich
nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR.

.outerText Referenz auf den gesamten Plain-Text im Objekt
nur nach kompletten einlesen des Dokumentes nutzbar
nur Plain-Text
schreiben: immer komplett überschreiben
nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR

.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
"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

.rowSpan Anzahl der Zeilen einer Zelle in einer Tabelle
siehe Objekt table.tr.td
siehe Objekt table.tr.th

.scope Gruppierung, in der eine Zelle liegt
siehe Objekt table.tr.td
siehe Objekt table.tr.th
"row" Zelle enthält Header-Informationen zur Zeile
"col" Zelle enthält Header-Informationen zur Spalte
"rowgroup" wie "row" aber zur Gruppe aus Zeilen
"colgroup" wie "col" aber zur Gruppe aus Spalten

.scopeName Namensraum laut XMLNS-Attribut

.scrollHeight Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes

.scrollLeft Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

.scrollTop Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes
immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind

.scrollWidth Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes

.sectionRowIndex Index der Tabellenzeile in der Collection
table.tBody.rows
bzw. table.tFoot.rows
bzw. table.tHead.rows
Zeile muss im **existierenden** Tabellenteil TOBDY bzw. TFOOT bzw. THEAD liegen

.sourceIndex Index des Objektes in der Collection document.all

STYLE direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen

.syncMaster Synchronisierung der Animation des im Container liegenden Elementes auf Timeline

.systemBitrate wird hier nicht erklärt

.systemCaptions wird hier nicht erklärt

.systemLanguage Sprache festlegen für das Objekt

.systemOverdubOrSubtitle wird hier nicht erklärt

.tabIndex Index des Elementes in der Tab-Tasten-Folge

.tagName Tag-Bezeichner des Objektes

.tagUrn Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut

.timeContainer Typ der Timeline des Objektes
siehe Objekt currTimeState und Behavior .style.time2

.title Tooltip-Text bei Mouse over über Objekt

.uniqueID durch den Browser automatisch-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 Selektionsfähigkeit eines Objektes
"off" Default. selektierbar
"on" nicht selektierbar

.vAlign vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes
"middle" zentriert, Standard
"baseline" Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des



**benachbarten** Objektes

"bottom" am Fuss
"top" am Kopf

.width Breite des Objektes in Pixel
Integer in Pixels für absolute Breite, >=0
bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel
String Ziffernfolge eines Integer mit nachfolgendem % für Breite als
Anteil der Breite des Elternobjektes
z.B. 10%

**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

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

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

.fireEvent() ein Event auslösen

.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

.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

.scrollIntoView() Objekt derart scrollen, dass es im Fenster für User sichtbar wird
Objekt muss an sich schon renderbar sein

.setActive() Objekt für die Eventdurchreichung aktivieren
aber ohne es zu fokussieren
und ohne es scrollbar zu machen

.setAttribute() Wert von vorhandenem Attribut setzen
wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt
DOM wird nur bei Erzeugung geändert

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

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

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

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

#### *4.3.2.2.4.3.42.18.2. table.tr.th Objekt des Internet Explorer*

Zelle TH einer Tabellenzeile
enthält keine Daten
dient nur der Spaltenüberschrift (Inhalt der Zelle) in automatischer Fettdarstellung
als Objekt nicht per Script erzeugbar:
Erzeugung nur als TD-Zelle, wobei .innerHTML mit <B>-Tag gefüllt werden muss

Beispiel 1:

```
<TABLE>
    <TR>
        <TH>Zelle1 fett</TH>
    </TR>
    <TR>
        <TH>Zelle2 fett</TH>
    </TR>
</TABLE>
```

Beispiel 2:

```
<SCRIPT>
    function ZeilenErzeugen()
    {
        // +++++   Collection aller Tabellenzeilen adressieren
        var TabellenZeilenCollection = ID_Tabelle.rows;

        // +++++   Zeile 1 erzeugen durch anhängen
        // -----    aktuelle Anzahl der Tabellenzeilen ermitteln
        var AnzahlTabellenZeilen = TabellenZeilenCollection.length;
        // -----    Zeile 1 in Tabelle einbinden sowie der Collection anhängen
        var TabellenZeile1 = ID_Tabelle.insertRow(AnzahlTabellenZeilen);

        // +++++   Zeile 2 erzeugen durch anhängen
        // -----    aktuelle Anzahl der Tabellenzeilen ermitteln
        AnzahlTabellenZeilen = TabellenZeilenCollection.length;
        //-----    Zeile 2 in Tabelle einbinden sowie der Collection anhängen
        var TabellenZeile2= ID_Tabelle.insertRow(AnzahlTabellenZeilen);

        // +++++   Zeile 1 mit 2 Zellen versorgen
        //  -----    Collection der Zellen adressieren
        var TabellenZeile_ZellenCollection = TabellenZeile1.cells;
```



```
                        // -----    Zelle 1 in die Tabellenzeile 1 einfügen und der Collection anhängen
                        var AktuellerIndexDerZelle    = TabellenZeile1.rowIndex;
                        var AktuelleZelle             = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        var AktuelleAnzahlZellen      = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile1_Zelle1     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // -----    Zelle 2 in die Tabellenzeile 1einfügen und der Collection anhängen
                        AktuellerIndexDerZelle        = TabellenZeile1.rowIndex;
                        AktuelleZelle                 = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen                  = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile1_Zelle2     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // +++++  Zeile 2 mit 2 Zellen versorgen
                        //  -----    Collection der Zellen adressieren
                        var TabellenZeile_ZellenCollection = TabellenZeile2.cells;

                        // -----    Zelle 1 in die Tabellenzeile 2 einfügen und der Collection anhängen
                        AktuellerIndexDerZelle        = TabellenZeile2.rowIndex;
                        AktuelleZelle                 = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen                  = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle1     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // -----    Zelle 2 in die Tabellenzeile 2einfügen und der Collection anhängen
                        AktuellerIndexDerZelle        = TabellenZeile2.rowIndex;
                        AktuelleZelle                 = TabellenZeilenCollection(AktuellerIndexDerZelle);
                        AktuelleAnzahlZellen                  = TabellenZeile_ZellenCollection.length;
                        var TabellenZeile2_Zelle2     = AktuelleZelle.insertCell(AktuelleAnzahlZellen);

                        // alle Zellen mit Daten versorgen, die sofort angezeigt werden (kein Tabellen-Refresh nötig)
                        TabellenZeile1_Zelle1.innerHTML="<B>Zeile1 Zelle1</B>";
                        TabellenZeile1_Zelle2.innerHTML="<B>Zeile1 Zelle2</B>";
                        TabellenZeile2_Zelle1.innerHTML="<B>Zeile2 Zelle1</B>";
                        TabellenZeile2_Zelle2.innerHTML="<B>Zeile2 Zelle2</B>";
                }
        </SCRIPT>
        <INPUT TYPE="button" VALUE="Zeilen erzeugen" onclick="ZeilenErzeugen();">
        <TABLE ID="ID_Tabelle" BORDER=1>
        </TABLE>
```

**Eigenschaften:**

Eigenschaften .innerText und .innerHTML sind les- **und** schreibbar
Eigenschaft .style verfügbar

Folgende Eigenschaften der Zelle liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index der Zelle im DOM |
| .cellIndex Index | Index der Zelle innerhalb der Zeile |

Folgende Eigenschaften der Zeile liefern Indexe im DOM bzw. TOM:

| | |
|---|---|
| .sourceIndex | Index der Zeile im DOM |
| .rowIndex | Index der Zeile bezüglich Tabelle |
| .sectionRowIndex | Index der Zeile bezüglich Tabellenelement wie THEAD, TBODY bzw. TFOOT<br>nur für TD, nicht für TH |

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste |
| .align | Ausrichtung |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .background | Hintergrundbild eines Objektes |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .bgColor | deprecated und durch STYLE-Attribut zu ersetzen |
| .borderColor | Borderfarbe (Rahmenfarbe)<br>wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"<br>Eigenschaft .border mit Wert 0<br>#rrggbb<br>vordefinierter Farbname (browserspezifisch) |
| .borderColorDark | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>dunkle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft border) |
| .borderColorLight | deprecated und durch Eigenschaft .borderColor zu ersetzen<br>helle Farbe des 3D-Rahmens eines Objektes<br>zu verwenden mit Eigenschaft .border (nicht Style-Eigenschaft boder) |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .cellIndex | Index der Zelle in der Collection table.rows.cells<br>siehe Objekt table.tr.td |



siehe Objekt table.tr.th

.className Klassenreferenz, Klassenname

.clientHeight Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt

.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

.colSpan Anzahl der Spalten einer Zelle in einer Tabelle

siehe Objekt table.tr.td

siehe Objekt table.tr.th

.dir Umflussrichtung

.end Objektaktivitäten laut Eigenschaft .timeAction beenden

.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

.hasMedia Objekt ist HTML-Media-Objekt

.height Höhe des Objektes in Pixel

.hideFocus Focussierbarkeit

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

ID-Attribut in HTML: Wert ist String alphanumerisch
muss mit Buchstaben beginnen
Unterstrich _ verwendbar
kann in " " bzw. ' ' kodiert werden, muss aber nicht

Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).

Zeiger aus ID bilden var Zeiger = eval(object.id);

Die Verwendung des ID-Attributes erhöht die Performance des Browsers und ist zu empfehlen.

.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

Plain-Text

HTML-Elemente je nach Objekt möglich

Script: muss mit DEFER-Attribut kodiert sein

schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen

nur lesen bei COL, COLGROUP, FRAMESET, HTML, **STYLE**, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.

.innerText Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag
ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B>

dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes,
also nach dem Laden des Objektes
aber wirksam erst mit parsen des HTML-Endetag

nur Plain-Text also keine HTML-Elemente und kein Script

schreiben: wenn Bereich nicht leer, so komplett überschreiben
wenn Bereich leer so einfügen

nur lesen bei HTML, TABLE, TBODY, TFOOT, THEAD, TR

.isContentEditable Editierbarkeit des Objekt-Content (auch wenn kein Layout hat)

.isDisabled Interaktionsfähigkeit

nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection

1 für Element-Knoten.

3 für Textknoten.

oder null (**nicht** 0 !!) wenn Knoten nicht vorhanden

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.noWrap Wortumbruch einstellen

false Default.
Browser bricht den Text automatisch um

true Browser bricht den Text nicht um

.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 rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent) |
| .onOffBehavior | deprecated ab IE 5.x<br>Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound |
| .outerHTML | Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag<br>wirksam mit parsen des Ende-Tag<br>nur nach kompletten Einlesen des Dokumentes nutzbar<br>schreiben: wenn Bereich gefüllt so komplett überschreiben<br>Plain-Text<br>HTML-Elemente möglich<br>nur lesen bei CAPTION, COL, COLGROUP, FRAMESET, HTML, TBODY, TD, TFOOT, TH, THEAD, TR. |
| .outerText | Referenz auf den gesamten Plain-Text im Objekt<br>nur nach kompletten einlesen des Dokumentes nutzbar<br>nur Plain-Text<br>schreiben: immer komplett überschreiben<br>nur lesen bei HTML, THEAD, TBODY, TFOOT, TD, TH, TR |
| .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<br>"uninitialized" Objekt ist nicht initialisiert<br>"loading" Objekt ist nicht intialisiert aber lädt gerade Daten zur Initialisierung<br>"loaded" Objekt hat alle Daten komplett geladen und ist komplett intitialisiert<br>"interactive" Objekt kann vom User interaktiv verwendet werden zum Füllen mit Daten<br>"complete" Object hat alle Daten geladen und ist mit diesen komplett initialisiert |
| .rowSpan | Anzahl der Zeilen einer Zelle in einer Tabelle<br>siehe Objekt table.tr.td<br>siehe Objekt table.tr.th |
| .scope | Gruppierung, in der eine Zelle liegt<br>siehe Objekt table.tr.td<br>siehe Objekt table.tr.th<br>"row" Zelle enthält Header-Informationen zur Zeile<br>"col" Zelle enthält Header-Informationen zur Spalte<br>"rowgroup" wie "row" aber zur Gruppe aus Zeilen<br>"colgroup" wie "col" aber zur Gruppe aus Spalten |
| .scopeName | Namensraum laut XMLNS-Attribut |
| .scrollHeight | Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes |
| .scrollLeft | Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .scrollTop | Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes<br>nur nutzbar nach dem kompletten Laden des Dokumentes<br>immer 0 wenn Scrollbars im Objekt nicht vorhanden bzw. nicht erlaubt sind |
| .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)<br>Hinweis: für Scripting ist das Style-Objekt zu nutzen |
| .syncMaster | Synchronisierung der Animation des im Container liegenden Elementes auf Timeline |
| .systemBitrate | wird hier nicht erklärt |
| .systemCaptions | wird hier nicht erklärt |
| .systemLanguage | Sprache festlegen für das Objekt |
| .systemOverdubOrSubtitle | wird hier nicht erklärt |
| .tabIndex | Index des Elementes in der Tab-Tasten-Folge |
| .tagName | Tag-Bezeichner des Objektes |
| .tagUrn | Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut |
| .timeContainer | Typ der Timeline des Objektes<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .title | Tooltip-Text bei Mouse over über Objekt |
| .uniqueID | durch den Browser automatisch-genriertes ID des Objektes<br>Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde<br>kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden |
| .vAlign | vertikale Ausrichtung des Inhaltes z.B. Text eines Objektes<br>"middle" zentriert, Standard<br>"baseline" Basislinie der 1. Zeile eines Textes im Objekt ausrichten an der Basislinie des **benachbarten** Objektes<br>"bottom" am Fuss<br>"top" am Kopf |



.width — Breite des Objektes in Pixel
Integer in Pixels für absolute Breite, >=0
bei Tabelle-Zelle und Spalte als Elternobjekt: maximal 32750 Pixel
String Ziffernfolge eines Integer mit nachfolgendem % für Breite als
Anteil der Breite des Elternobjektes
z.B. 10%

**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

.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)

.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

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

#### 4.3.2.2.4.3.43. textarea Objekt des Internet Explorer

Objekt des mehrzeiligen Text-Eingabe-Controls (in HTML der Tag TEXTAREA)

Beispiel 1:

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

<SCRIPT>
    function Anzeige(ZeigerAufTextarea)
    {
        var FeldAllterTextArea = document.all.tags("TEXTAREA");

        // prüfen ob mindestens 1 TEXTAREA-Element im Body vorliegt
        if (FeldAllterTextArea != null)
        {
            var TextBereich = ZeigerAufTextarea.createTextRange();

            // prüfen ob Textbereich erzeugt wurde
            if (TextBereich != null)
            {alert(".boundingHeight = " + TextBereich.boundingHeight); }
        }
    }
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS=100 ROWS=2 onclick="Anzeige(this)">
    .....
</TEXTAREA>
```

Beispiel 2:

```
<SCRIPT>
    function HoleHTML()
    {ID_Textarea.value= document.documentElement.innerHTML;}
</SCRIPT>
<TEXTAREA ID="ID_Textarea" COLS = 50 ROWS = 10>
</TEXTAREA>
```

Beispiel 3:



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

Beispiel 4:

```
<SPAN UNSELECTABLE="on" >
	Dieser Text kann nicht selektiert werden
	<TEXTAREA WRAP="PHYSICAL" ROWS="5"
			STYLE="font-weight: bold;"
	>
		Dieser Text kann selektiert werden
	</TEXTAREA>
	Dieser Text kann nicht selektiert werden
</SPAN>
```

Beispiel 5:

```
<HEAD>
<SCRIPT>
	function ScrolleSeiteRechts()
	{document.body.doScroll("scrollbarPageRight");}

	function ScrolleDown()
	{ID_Textaerea.doScroll("scrollbarDown");}

	function ScrolleSeiteDown()
	{ID_Textaerea.doScroll("scrollbarPageDown");}
</SCRIPT>
</HEAD>
<BODY>
	<BUTTON onclick=" ScrolleSeiteRechts()">
		scrolle Seite rechts
	</BUTTON>
	<BR>
	<BUTTON
		onclick=" ScrolleDown()"
		ondblclick=" ScrolleSeiteDown()"
	>
		Texarea: Click = scrolle down Doppelklick = scrolle Seite down
	</BUTTON>
	<BR>
	<BR>
	<TEXTAREA ID="ID_Textaerea" >
		ABCDEFGHIJKLMNOPQRSTUVWXYZ
		12345678901234567890123456789012345678901
		ABCDEFGHIJKLMNOPQRSTUVWXYZ
		12345678901234567890123456789012345678901
		ABCDEFGHIJKLMNOPQRSTUVWXYZ
		12345678901234567890123456789012345678901
		ABCDEFGHIJKLMNOPQRSTUVWXYZ
		12345678901234567890123456789012345678901
	</TEXTAREA>
</BODY>
```



Beispiel 6:

```
<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>

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

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

**Eigenschaften:**

| | |
|---|---|
| .accessKey | Tastaturzugriff auf ein Objekt per Alt + Taste<br>bei Ausführung der Tastenkombination wird<br>das Sprungziel wird focussiert<br>die Sprungquelle defocussiert<br>das Focus-Ereignis ausgelöst<br>vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt |
| ATOMICSELECTION | Selektierbarkeit des Objektes einstellen |
| .begin | Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction<br>siehe Objekt currTimeState und Behavior .style.time2 |
| .canHaveChildren | prüfen ob Kind möglich ist, also ob Objekt Parent sein kann |
| .canHaveHTML | prüfen ob Objekt HTML-Tags enthalten darf |
| .className | Klassenreferenz, Klassenname |
| .clientHeight | Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt<br>ohne Rahmen<br>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<br>ohne Rahmen<br>ohne Scrollbalken |
| .cols | Anzahl der sichtbaren Zeichen in einer Zeile der TEXTAREA<br>Hinweis: Es können mehr Zeichen in der TEXTAREA enthalten sein als sichtbar<br>siehe .rows für Mehrzeiligkeit<br>.wrap für Wortumbruch<br>Scrollleisten sind erzeugbar |
| .contentEditable | Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat)<br>Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc. |
| .dataFld | Datenquelle-Name vergeben (ID) |
| .dataSrc | Datenquelle als Anker festlegen |
| .defaultValue | Vorbelegung der TEXTAREA |



sichtbare Auswirkungen nur bei
Instanzierung des Objektes
Formular-Reset

.dir Umflussrichtung

.disabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.end Objektaktivitäten laut Eigenschaft .timeAction beenden
ab IE 6.x
alternativ: Eigenschaft .dur
siehe Objekt currTimeState und Behavior .style.time2

.firstChild Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes

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

.hasMedia Objekt ist HTML-Media-Objekt
siehe Objekt currTimeState und Behavior .style.time2

.hideFocus Focussierbarkeit

.id Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
Zeiger aus ID bilden var Zeiger = eval(object.id);

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

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

.isDisabled Interaktionsfähigkeit
nur wenn sichtbar so User-Interaktion möglich

.isMultiLine Mehrzeiligkeit des Objektinhaltes

.isTextEdit Erzeugbarkeit eines Textbereiches

.lang Sprache für Anzeige von Sonderzeichen etc.

.language Sprache für Script festlegen

.lastChild Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes

.name Name des Objektes (nicht ID !!!)
muss beim Formular für alle zu sendenden Felder kodiert sein !!
Element darf nicht per Methode .createElement() erzeugt worden sein

.nextSibling Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes

.nodeName String als Name des Kindes (Knoten, Node, Element)
also TAG-Bezeichner, Attribut-Name; #text für Anker

.nodeType Knotentyp laut attributes Collection

.nodeValue Knotenwert (Wert des Kindes, Node, Elementes)
nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

.offsetHeight Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetLeft X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)

.offsetParent Referenz der Eltern
für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth

.offsetTop Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.offsetWidth X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)

.onOffBehavior deprecated ab IE 5.x
Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound

.outerHTML Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag
wirksam mit parsen des Ende-Tag
nur nach kompletten Einlesen des Dokumentes nutzbar

.outerText Referenz auf den gesamten Plain-Text im Objekt
nur nach kompletten einlesen des Dokumentes nutzbar

.ownerDocument Referenz auf das document Objekt zu dem der 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

.readOnly Editierbarkeit eines Text-Controls
Editierung bedeutet Focuserhalt !
false Default
Objekt ist **nicht** read-only
also ist es editierbar
kann also Focus erhalten
true Objekt ist read-only



also ist es nicht editierbar
kann kein also Focus erhalten

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

.recordNumber — Datenquelle-Satznummer eines Datenfeldes

.rows — Anzahl der sichtbaren Zeilen der TEXTAREA

.scopeName — Namensraum laut XMLNS-Attribut

.scrollHeight — Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes

.scrollLeft — Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes

.scrollTop — Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes
nur nutzbar nach dem kompletten Laden des Dokumentes

.scrollWidth — Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes

.sourceIndex — Index des Objektes in der Collection document.all

.status — Selektionsstatus eines Control-Elementes
Control-Element: kann durch User interaktiv verändert werden
wird bei Ceckbox-Control auch verändert durch Eigenschaft .indeterminate
bei Statusveränderung wird Event onpropertychange erzeugt
false Default außer bei textArea Objekt
Control ist nicht selektiert
true Control ist selektiert
null Control ist nicht initialisiert
Default bei textArea Objekt

STYLE — direkt im HTML-Element kodierter Style (Inline-Style)
Hinweis: für Scripting ist das Style-Objekt zu nutzen

.syncMaster — Synchronisierung der Animation des im Container liegenden Elementes auf Timeline
(Container ist Master in der Synchronisierung), wobei ein Master **nur** genau ein zu synchronisierendes Element haben darf (Eineindeutigkeit).
nur sinvoll bei Zwangssynchronisierung (siehe .syncBehavior)
ersetzt die Eigenschaft clockSource, da diese deprecated ist und nicht mehr verwendet werden darf !
siehe Objekt currTimeState und Behavior .style.time2
siehe .syncTolerance und .syncBehavior

.systemBitrate — wird hier nicht erklärt

.systemCaptions — wird hier nicht erklärt

.systemLanguage — Sprache festlegen für das Objekt

.systemOverdubOrSubtitle — wird hier nicht erklärt

.tabIndex — Index des Elementes in der Tab-Tasten-Folge
für Anspringen des Dokumentes
Anspringen verbunden mit Focus erhalten
--> Ereignisse werden ausgelöst !!
unter IE 5.x
onblur, onfocus
ab IE 5.x
onblur, onfocus, onkeydown, onkeypress, onkeyup
Anspringen default per TAB-Taste
für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA
Anspringen default nicht per TAB-Taste
für APPLET, DIV, FRAMESET, SPAN, TABLE, TD

.tagName — Tag-Bezeichner des Objektes

.tagUrn — Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut

.timeContainer — Typ der Timeline des Objektes
siehe Objekt currTimeState und Behavior .style.time2

.title — Tooltip-Text bei Mouse over über Objekt

.type — Type des TEXTAREA-Control

.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 — Selektionsfähigkeit eines Objektes

.value — Wert eines Objekt-Attributes
Hinweis: Wert eines Elementes ist z.T. über das VALUE-Attribut definierbar

.wrap — Wortumbruch der TEXTAREA
"soft" Standard
Wortumbruch aktiv
im Formular wird nicht gesendet:
Zeilenumbruch
Zeilenvorschub

"hard" Wortumbruch aktiv
im Formular wird gesendet:
Zeilenumbruch
Zeilenvorschub

"off" Wortumbruch nicht aktiv



**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

.createTextRange() Textbereich erzeugen

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();
}
```

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



| | | |
|---|---|---|
| "scrollbarDown" | oder "down" | Down scroll Pfeil |
| "scrollbarHThumb" | | horizontaler Scrollbalken |
| "scrollbarLeft" | oder "left" | Left scroll Pfeil |
| "scrollbarPageDown" | oder "pageDown" | Page-down Scrollbalken |
| "scrollbarPageLeft" | oder "pageLeft" | Page-left Scrollbalken |
| "scrollbarPageRight" | oder "pageRight" | Page-right Scrollbalken |
| "scrollbarPageUp" | oder "pageUp" | Page-up Scrollbalken |
| "scrollbarRight" | oder "right" | Right scroll Pfeil |
| "scrollbarUp" | oder "up" | Up scroll Pfeil |
| "scrollbarVThumb" | | vertikaler Scrollbalken |

Beispiel:

```
<HEAD>
<SCRIPT>
    function ScrolleSeiteRechts()
    {document.body.doScroll("scrollbarPageRight");}

    function ScrolleDown()
    {ID_Textaerea.doScroll("scrollbarDown");}

    function ScrolleSeiteDown()
    {ID_Textaerea.doScroll("scrollbarPageDown");}
</SCRIPT>
</HEAD>
<BODY>
    <BUTTON onclick=" ScrolleSeiteRechts()">
        scrolle Seite rechts
    </BUTTON>
    <BR>
    <BUTTON
        onclick=" ScrolleDown()"
        ondblclick=" ScrolleSeiteDown()"
    >
        Texarea: Click = scrolle down Doppelklick = scrolle Seite down
    </BUTTON>
    <BR>
    <BR>
    <TEXTAREA ID="ID_Textaerea" >
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        12345678901234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        12345678901234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        12345678901234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        12345678901234567890123456789012345678901234567890
    </TEXTAREA>
</BODY>
```

| | |
|---|---|
| .dragDrop() | prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element |
| .fireEvent() | ein Event auslösen |
| .focus() | Focus setzen und Focus-Event auslösen<br>nur nach dem kompletten Laden des Dokumentes<br>vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen |
| .getAdjacentText() | Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann<br>Text kann HTML-Tags enthalten, muss aber nicht<br>DOM nicht geändert |
| .getAttribute() | Wert eines per HTML erzeugten Attributes liefern<br>DOM nicht geändert |
| .getAttributeNode() | Referenz auf Eigenschaft des attribute-Objektes liefern, also Zeiger auf attribute.name Eigenschaft.<br>Eigenschaft kann mit HTML-Anweisung erzeugt worden sein, muss aber nicht<br>Eigenschaft ist selbst ein Knoten in der Attribute-Objekt-Hierarchie zum Objekt<br>Wert des Attributes wird somit über die Referenz laut DOM erreichbar<br>DOM nicht geändert |
| .getBoundingClientRect() | Referenz auf TextRectangle-Objekt im Element holen |
| .getClientRects() | Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster<br>Feld mit Index als Integer ab 0<br>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.<br>Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection)<br>Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst !<br>Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementsById()<br>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

.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

.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

Beispiel:

```
<SCRIPT LANGUAGE="JScript" FOR=document EVENT=onclick >
    var TextBereich = document.body.createTextRange();
    TextBereich.moveToPoint(window.event.x, window.event.y);
    TextBereich.expand("word");
    TextBereich.select();
</SCRIPT>
```

.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

#### 4.3.2.2.4.3.44. 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<br>Objekt muss an sich schon renderbar sein |
| .select() | Objekt selektieren<br>z.B. Textbereich: wird hervorgehoben<br>ControlRange: es wird Rechteck-Rahmen erzeugt<br>nur unter Windows 32-Bit |
| .setEndPoint() | Textbereichanfang bzw. –ende von 2 Textbereichen synchronisieren<br>per textrange Objekt<br>nur unter Windows 32-Bit |

#### *4.3.2.2.4.3.44.1. 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 Koordinantensystem 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<br>außer bei Formular mit <INPUT TYPE=image ...><br>da dafür die children-Collection verwendet werden muss !!! |
| .namedItem() | Referenz auf Eintrag (FeldElement) anhand des Namen<br>(analog zu ID oder NAME-Attribut) liefern |

#### *4.3.2.2.4.3.44.2. 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

#### 4.3.2.2.5. window.event Objekt

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.

Ein HTML-Objekt kann natürlich nur dann Events auslösen, wenn es instanziert ist. Beispiel für einen DIV: Wird dem DIV über die Eigenschaft .innerHTML nur ein Leerzeichen zugewiesen ( zeiger_auf_div.innerHTML=' '; mit zeiger laut ID-Attribut-Wert), so gilt der DIV für die Eventerzeugung z.B. onmousedown als **nicht instanziert**. Desweiteren darf der DIV per Style-Eigenschaft style.visibility mit Wert 'hidden' **nicht ausgeblendet** sein.

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).

Per JScript kann ein Eventobjekt nur für das Objekt document erzeugt werden.

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.

Standardgemäß gilt beim Internet Explorer: Kinder reichen Events zu den Eltern hoch und das bis zum BODY, also der obersten Dokumentenebene. Die Eltern haben standardgemäß immer das letzte Wort: Will ein Kind ein Event auslösen, so kann es das. Aber die Eltern entscheiden, ob das Event erhalten bleibt. Besonders gilt das für Body (document.body).

Beispiel: Wird per

```
document.body.onclick=OnClickHandler; // ohne () kodieren, damit nicht sofort ausgeführt wird
document.body.attachEvent('onclick',OnClickHandler);
```

ein Eventhandler instanziert, so kann das Kind zwar **sein** onclick (eigener Handler muss im Kind implementiert sein) auslösen und entsprechende Reaktionen verursachen, aber die Eltern, also document.body, können innerhalb OnClickHandler() das Ereignis onclick sperren, so dass die Reaktionen des Kindes aufgehoben werden. Diese Reaktionen sind also nur solange wirksam, bis document.body das Event onclick erreicht hat. Fatal wird das bei Bildschirmausgaben des Kindes aufgrund onclick: Diese Ausgaben werden schlichtweg durch die Eltern aufgehoben (keine Anzeige mehr im Dokument).