table Objekt des Internet Explorer

table Objekt des Internet Explorer 1
1. Erzeugung der Tabelle 1
1.1. Erzeugung in HTML 1
1.2. Erzeugung in JScript 2
2. Daten der Tabelle 3
2.1. Datenbereitsstellung 3
2.1.1. in HTML *3*
2.1.2. in JScript *3*
2.1.3. per Active-X-Control *3*
2.2. Dateneinbindung 3
2.2.1. in HTML *3*
2.2.2. in JScript *3*
2.2.3. per Active-X-Control *3*
3. Tabellen-Objektmodell (TOM) in JScript 4
4. Methoden des DOM und TOM zur Verwaltung der Tabellenelemente 5
5. Dynamische Struktur und Daten einer Tabelle per JScript 6
5.1. Strukturerzeugung der Tabelle 6
5.1.1. in HTML *6*
5.1.2. per Methoden des DOM *8*
5.2. Datenerzeugung mit der Strukturbildung und zur Laufzeit 10
6. Dynamische Veränderung einer Tabelle in JScript 19
6.1. Datenveränderung per JScript 19
6.2. Layoutveränderung per Style 19
6.3. Strukturveränderung per JScript 19
7. Eigenschaften der Tabelle 20
8. Methoden der Tabelle 25
9. table.caption Objekt des Internet Explorer 34
10. table.col Objekt des Internet Explorer 39
11. table.colGroup Objekt des Internet Explorer 43
12. table.rows Collection des Internet Explorer 47
13. table.rows.cells Collection des Internet Explorer 48
14. table.tBody Objekt des Internet Explorer 49
14.1. table.tBody.rows Collection des Internet Explorer 53
14.2. table.tBody.rows.cells Collection des Internet Explorer 54
15. table.tBodies Collection des Internet Explorer 55
16. table.tFoot Objekt des Internet Explorer 56
16.1. table.tFoot.rows Collection des Internet Explorer 60
16.2. table.tFoot.rows.cells Collection des Internet Explorer 61
17. table.tHead Objekt des Internet Explorer 62
17.1. table.tHead.rows Collection des Internet Explorer 66
17.2. table.tHead.rows.cells Collection des Internet Explorer 67
18. table.tr Objekt des Internet Explorer 68
18.1. table.tr.td Objekt des Internet Explorer 74
18.2. table.tr.th Objekt des Internet Explorer 81

TABLE Objekt einer Tabelle (HTML-Element TABLE)

1. Erzeugung der Tabelle

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.

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.

2. Daten der Tabelle

2.1. Datenbereitsstellung

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.

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 .

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.

2.2. Dateneinbindung

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.

2.2.2. in JScript

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

2.2.3. per Active-X-Control

Die Daten werden einmalig eingebunden.

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. Methoden des DOM und TOM zur Verwaltung der Tabellenelemente

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.

5. Dynamische Struktur und Daten einer Tabelle per JScript

5.1. Strukturerzeugung der Tabelle

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>
```

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>
```

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 immer kodieren alternativ: Zugriff über die Collectionen rows und cells
.innerText	für den Textinhalt einer Zelle (Plain-Text) alternativ auch .innerHTML
.innerHTML	für den HTML-Inhalt einer Zelle alternativ auch .innerText 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>
```


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.

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.

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

6.3. Strukturveränderung per JScript

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

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
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
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
.caption Zeiger auf das Objekt table.caption
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
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
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 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
.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 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
.width	Breite des Objektes in Pixel

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 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

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 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
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	Lage der CAPTION einer Tabelle es darf nur 1 CAPTION zur Tabelle existieren nach Änderung ist ein Tabellen-Refresh per Methode .refresh() notwendig "top" Überschrift am Kopf der Tabelle

Standard
"bottom" Überschrift am Fuss der Tabelle

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

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

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) 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 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

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
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

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
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

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 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

.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
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

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)

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 eine 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)

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

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

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)

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)

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) 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

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)

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)

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 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 siehe Objekt currTimeState und Behavior .style.time2
.bgColor	deprecated und durch STYLE-Attribut zu ersetzen
.borderColor	Borderfarbe (Rahmenfarbe) wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no" Eigenschaft .border mit Wert 0 #rrggbb 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
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

.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
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

.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

.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

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 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 siehe Objekt currTimeState und Behavior .style.time2
.bgColor	deprecated und durch STYLE-Attribut zu ersetzen
.borderColor	Borderfarbe (Rahmenfarbe) wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no" Eigenschaft .border mit Wert 0 #rrggbb 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
.cellIndex	Index der Zelle in der Collection table.rows.cells 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
.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

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 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 siehe Objekt currTimeState und Behavior .style.time2
.bgColor	deprecated und durch STYLE-Attribut zu ersetzen
.borderColor	Borderfarbe (Rahmenfarbe) wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no" Eigenschaft .border mit Wert 0 #rrggbb 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
.cellIndex	Index der Zelle in der Collection table.rows.cells 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
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

.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

.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)

.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

"#default#behaviorName....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
#text....22, 35, 39, 43, 50, 57, 63, 70, 76, 83
.accessKey....20, 34, 39, 49, 56, 62, 69, 75, 82
.addBehavior()....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
.align....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.appendChild()....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
.applyElement()....25, 36, 41, 45, 51, 58, 64, 71, 77, 84
.attachEvent()....25, 36, 41, 45, 51, 58, 64, 71, 72, 78, 84, 85
.background....20, 34, 49, 75, 82
.begin....20, 34, 49, 56, 62, 69, 75, 82
.bgColor....20, 39, 43, 49, 56, 62, 69, 75, 82
.blur()....25, 36, 51, 58, 64, 71, 78, 84
.border....20
.borderColor....20, 69, 75, 82
.borderColorDark....20, 69, 75, 82
.borderColorLight....20, 69, 75, 82
.canHaveChildren....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.caption....20
.cellIndex....75, 82
.cellPadding....20
.cellSpacing....20
.className....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.clearAttributes()....25, 36, 41, 45, 51, 58, 64, 71, 78, 85
.click()....25, 36, 51, 58, 64, 71, 78, 85
.clientHeight....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.clientLeft....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.clientTop....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.clientWidth....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
.cloneNode()....25, 36, 41, 45, 51, 58, 64, 71, 78, 85
.cols....20
.colSpan....75, 82
.componentFromPoint()....25, 36, 41, 45, 51, 58, 64, 71, 78, 85
.contains()....25, 36, 41, 45, 51, 58, 64, 72, 78, 85
.createCaption()....25
.createElement()....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
.createTFoot()....25
.createTHead()....25
.dataPageSize....20
.dataSrc....22
.deleteCaption()....25
.deleteCell()....72
.deleteRow()....25, 51, 58, 64
.deleteTFoot()....25, 51
.deleteTHead()....25, 51
.detachEvent()....25, 36, 41, 45, 51, 58, 64, 71, 72, 78, 84, 85
.dir....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
.disabled....22, 34, 39, 69, 75
.dragDrop()....26, 36, 41, 72, 78
.end....22, 34, 49, 56, 62, 69, 75, 82
.fireEvent()....26, 36, 41, 45, 52, 58, 64, 72, 78, 85
.firstChild....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
.firstPage()....26
.focus()....27, 36, 52, 58, 64, 72, 78, 85
.frame....22
.getAdjacentText()....27, 36, 41, 45, 52, 58, 64, 72, 78, 85
.getAttribute()....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
.getAttributeNode()....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
.getBoundingClientRect()....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
.getClientRects()....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
.getElementsByName()....27, 37, 41, 45, 52, 59, 65, 72, 78, 85
.getElementsByTagName()....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
.getExpression()....27, 37, 41, 45, 52, 59, 65, 72, 78, 85
.hasChildNodes()....27, 37, 41, 46, 52, 59, 65, 72, 78, 85
.hasMedia....22, 34, 49, 56, 62, 69, 75, 82
.height....22, 69, 75, 82
.hideFocus....22, 34, 49, 56, 62, 69, 75, 82
.id....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
.innerHTML....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
.innerText....22, 34, 49, 56, 62, 69, 76, 83
.insertAdjacentElement()....28, 37, 41, 46, 52, 59, 65, 72, 79, 85
.insertAdjacentHTML()....37, 79, 85
.insertAdjacentText()....37, 79, 86
.insertBefore()....28, 37, 42, 46, 52, 59, 65, 72, 79, 86
.insertCell()....72
.insertRow()....28, 52, 59, 65
.isContentEditable....22, 34, 39, 43, 49, 56, 62, 70, 76, 83
.isDisabled....22, 34, 39, 43, 49, 56, 62, 70, 76, 83
.isMultiLine....22, 34, 39, 43, 50, 56, 62, 70, 76, 83
.isTextEdit....22, 34, 39, 43, 50, 57, 63, 70, 76, 83
.item()....47, 48, 55
.lang....22, 34, 39, 43, 50, 57, 63, 70, 76, 83
.language....22, 34, 50, 57, 63, 70, 76, 83
.lastChild....22, 34, 39, 43, 50, 57, 63, 70, 76, 83
.lastPage()....28
.length....47, 48, 55
.mergeAttributes()....29, 37, 42, 46, 52, 59, 65, 72, 79, 86
.moveRow()....29, 52, 59, 65
.namedItem()....47, 48, 55
.nextPage()....30
.nextSibling....22, 34, 39, 43, 50, 57, 63, 70, 76, 83
.nodeName....22, 35, 39, 43, 50, 57, 63, 70, 76, 83
.nodeType....23, 35, 39, 43, 50, 57, 63, 70, 76, 83
.nodeValue....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
.normalize()....31, 37, 42, 46, 53, 60, 66, 73, 79, 86
.noWrap....76, 83
.offestParent....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.offserLeft....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.offsetHeight....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
.offsetHeigt....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.offsetLeft....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
.offsetParent....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
.offsetTop....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.offsetWidth....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.onOffBehavior....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.outerHTML....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.outerText....23, 50, 57, 63, 70, 76, 83
.ownerDocument....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.parentElement....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.parentNode....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.parentTextEdit....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.previousPage()....31
.previousSibling....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.readyState....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
.refresh()....33, 35
.releaseCapture()....33, 37, 53, 60, 66, 73, 79, 86
.removeAttribute()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.removeAttributeNode()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.removeBehavior()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.removeChild()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.removeExpression()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.removeNode()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.replaceAdjacentText()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.replaceChild()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
.replaceNode()....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
.rowIndex....70
.rowSpan....77, 84
.rules....23
.scope....77, 84
.scopeName....24, 35, 40, 44, 50, 57, 63, 70, 77, 84
.scrollHeight....24, 35, 40, 44, 50, 57, 63, 70, 77, 84
.scrollIntoView()....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
.scrollLeft....24, 35, 40, 44, 50, 57, 63, 70, 77, 84
.scrollTop....24, 35, 40, 44, 50, 57, 63, 70, 77, 84
.scrollWidth....24, 35, 40, 44, 50, 57, 63, 70, 77, 84
.sectionRowIndex....71, 77
.setActive()....33, 38, 53, 60, 66, 73, 79, 86
.setAttribute()....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
.setAttributeNode()....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
.setCapture()....33, 38, 53, 60, 66, 73, 80, 86
.setExpression()....33, 38, 42, 46, 53, 60, 66, 73, 80, 86
.sourceIndex....24, 35, 40, 44, 50, 57, 63, 71, 77, 84
.span....40, 44
.summary....24
.swapNode()....33, 38, 42, 46, 53, 60, 66, 73, 80, 87

.syncMaster....24, 35, 50, 57, 63, 71, 77, 84
.systemBitrate....24, 35, 50, 57, 63, 71, 77, 84
.systemCaptions....24, 35, 50, 57, 63, 71, 77, 84
.systemLanguage....24, 35, 50, 57, 63, 71, 77, 84
.systemOverdubOrSubtitle....24, 35, 50, 57, 63, 71, 77, 84
.tabIndex....24, 35, 50, 57, 63, 71, 77, 84
.tagName....24, 35, 40, 44, 50, 57, 63, 71, 77, 84
.tags()....47, 48, 55
.tagUrn....24, 35, 40, 44, 50, 57, 63, 71, 77, 84
.tFoot....24
.tHead....24
.timeContainer....24, 35, 51, 57, 63, 71, 77, 84
.title....24, 35, 51, 58, 64, 71, 77, 84
.uniqueID....22, 24, 34, 35, 39, 40, 43, 44, 49, 51, 56, 58, 62, 64, 69, 71, 75, 77, 82, 84
.urns()....47, 48, 55
.vAlign....35, 40, 44, 51, 58, 64, 71, 77, 84
.width....24, 40, 44, 71, 77, 84
3D-Rahmen Farbe....20, 69, 75, 82
above....22
Abstand Objekt zum linken Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Abstand Objekt zum oberen Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
ActiveX-Control TDC....3
aktivieren Eventdurchreichung....33, 38, 53, 60, 66, 73, 79, 86
all23
Alt + Taste....20, 34, 39, 49, 56, 62, 69, 75, 82
anhängen Kind....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
anhängen Knoten....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Anker....22, 35, 39, 43, 50, 57, 63, 70, 76, 83
Anzahl der Feldelemente also Feldlänge....47, 48, 55
Anzeigesprache....22, 34, 39, 43, 50, 57, 63, 70, 76, 83
ATOMICSELECTION....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Attribut automatisch erzeugen und mit Wert belegen ..33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Attribut eines Knoten....33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Attribut erzeugen und mit Wert belegen33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Attribut Wert27, 33, 36, 38, 41, 42, 45, 46, 52, 53, 59, 60, 65, 66, 72, 73, 78, 79, 85, 86
Attribute eines Elementes....29, 37, 42, 46, 52, 59, 65, 72, 79, 86
Attribute HTML....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
attribute Objekt....27, 36, 41, 45, 52, 59, 65, 72, 78, 85
attributes Collection....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Attribut-Name....22, 35, 39, 43, 50, 57, 63, 70, 76, 83
auslösen Event....26, 36, 41, 45, 52, 58, 64, 72, 78, 85
Ausrichtung....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Ausrichtung Objektinhalt....40, 44, 51, 58, 64, 71, 77, 84
automatisch-genriertes ID des Objektes 24, 35, 40, 44, 51, 58, 64, 71, 77, 84
baseline....40, 44, 51, 58, 64, 71, 77, 84
below....22
Bereich ab inklusive HTML-Start bis hinter -Ende-Tag 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Bereich zwischen HTML-Start und -Ende-Tag 22, 34, 39, 43, 49, 56, 62, 69, 75, 76, 82, 83
Bezeichner des Objektes....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Bezeichner des Tag eines Objektes. 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Body Eltern....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Body Referenz auf das Elternobjekt 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
border....22
Borderdicke....20
Borderfarbe....20, 69, 75, 82
bottom....36, 40, 44, 51, 58, 64, 71, 77, 84
box....22
Breite des horizontalen Scrollbereiches 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Breite Objekt....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
canHaveHTML....20, 34, 39, 43, 49, 56, 62, 69, 75, 82
CAPTION....25, 34
childNodes Collection....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
clsid:333C7BC4-460F-11D0-BC04-0080C7055A83....3
col....77, 84
COL....39
colgroup....77, 84
COLGROUP....43
Collection Anzahl Elemente....47, 48, 55
Collection attributes....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Collection childNodes....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Collection document.all....24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Collection table.rows....47, 70
Collection table.rows.cells....48, 75, 82
Collection table.tBodies....55
Collection table.tBody.rows....53, 71, 77
Collection table.tBody.rows.cells....54
Collection table.tFoot.rows....60, 71, 77
Collection table.tFoot.rows.cells....61
Collection table.tHead.rows....66, 71, 77
Collection table.tHead.rows.cells....67
cols....23
Container....25, 36, 41, 45, 51, 58, 64, 72, 78, 85
createAttribute()....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
currTimeState Objekt....20, 24, 34, 35, 49, 51, 56, 58, 62, 64, 69, 71, 75, 77, 82, 84
Cursorgeschwindigkeit....25, 36, 41, 45, 51, 58, 64, 72, 78, 85
Datenquelle als Anker festlegen....22
DEFER....22, 34, 37, 39, 43, 49, 56, 62, 69, 76, 79, 83, 86
DHTML-Eigenschaft hinzufügen ...25, 36, 40, 45, 51, 58, 64, 71, 77, 84
document Objekt des Koten....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
document.all Collection....24, 35, 40, 44, 50, 57, 63, 71, 77, 84
DOM....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
DOM Elementeigenschaft....25, 36, 41, 45, 51, 58, 64, 71, 77, 84
DOM Normalisierung....31, 37, 42, 46, 53, 60, 66, 73, 79, 86
Drag-Manipulation Status....26, 36, 41, 72, 78
durchreichen Event....33, 38, 53, 60, 66, 73, 79, 86
Editierbarkeit des Objekt-Content ..22, 34, 39, 43, 49, 56, 62, 70, 76, 83
Eigenschaft des attribute-Objektes . 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Eigenschaft Element....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Eigenschaft hinzufügen....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Eigenschaften IE Standard....25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Element Attribute....29, 37, 42, 46, 52, 59, 65, 72, 79, 86
Element den Focus wegnehmen....25, 36, 51, 58, 64, 71, 78, 84
Element Eigenschaft....33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Element innerhalb eines Elementes 25, 36, 41, 45, 51, 58, 64, 72, 78, 85
Elementeigenschaft im DOM....25, 36, 41, 45, 51, 58, 64, 71, 77, 84
Elementes Tab-Tasten-Folge....24, 35, 50, 57, 63, 71, 77, 84
Element-Knoten....23, 35, 39, 43, 50, 57, 63, 70, 76, 83
Eltern....25, 36, 41, 45, 51, 58, 64, 71, 77, 84
Eltern Body....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Eltern Referenz....23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Elternknoten....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Elternobjekt Textbereich....23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Ereignis Standardbehandlung 25, 36, 41, 45, 52, 58, 64, 72, 78, 85
erstes Kind Referenz....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
erstes Kind Zeiger....22, 34, 39, 43, 49, 56, 62, 69, 75, 82
erzeugen Attribut automatisch und mit Wert belegen ...33, 38, 42, 46, 53, 60, 66, 73, 79, 86
erzeugen Attribut und mit Wert belegen33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Event auslösen....26, 36, 41, 45, 52, 58, 64, 72, 78, 85
Event durchreichen....33, 38, 53, 60, 66, 73, 79, 86
Event onblur....25, 36, 51, 58, 64, 71, 78, 84
Event onfocus....27, 36, 52, 58, 64, 72, 78, 85
Event onmouseover Pixelgenauigkeit...25, 36, 41, 45, 51, 58, 64, 71, 78, 85
Event registrieren....25, 36, 41, 45, 51, 58, 64, 71, 78, 84
Eventdurchreichung aktivieren....33, 38, 53, 60, 66, 73, 79, 86
Events regisitrieren....25, 36, 41, 45, 51, 58, 64, 72, 78, 85
Existenz Kind möglich....20, 34, 39, 43, 49, 56, 62, 69, 75, 82

Existenz von Kinder 27, 37, 41, 46, 52, 59, 65, 72, 78, 85
Farbe Border 20, 69, 75, 82
Farbe des 3D-Rahmens 20, 69, 75, 82
Farbe Rahmen 20, 69, 75, 82
Feld aller Elemente mit gemeinsamer URN 47, 48, 55
Feld aller HTML-Elemente mit gemeinsamen Tag-Namen 48
Feld der Zeiger auf TextRectangle-Objekte 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Feldelement Zeiger 47, 48, 55
Feldelemente Anzahl 47, 48, 55
Feldlänge 47, 48, 55
Fenster linker Rand 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Fenster oberer Rand 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Focus 25, 36, 51, 58, 64, 71, 78, 85
Focus setzen 27, 36, 52, 58, 64, 72, 78, 85
Focus wegnehmen 25, 36, 51, 58, 64, 71, 78, 84
Focus-Event 27, 36, 52, 58, 64, 72, 78, 85
Focussierbarkeit Objekt 22, 34, 49, 56, 62, 69, 75, 82
gemeinsamen Tag-Namen 48
groups 23
Hintergrundbild eines Objektes 20, 34, 49, 75, 82
hinzufügen DHTML-Eigenschaft ... 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
hinzufügen Eigenschaft 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Höhe des Objektes 22, 69, 75, 82
Höhe des vertikalen Scrollbereiches 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Höhe Objekt 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
horizontaler Scrollbereich 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
hsides 22
HTML-Attribut Wert 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
HTML-Attribute 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
HTML-Attribute eines Objektes 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
HTML-Code und/oder Script-Code einfügen 37, 79, 85
HTML-Elemente mit gemeinsamen Tag-Namen 48
HTML-Tags im Objekt 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
ID 27, 37, 41, 45, 52, 59, 65, 72, 78, 85
ID des Objektes 24, 35, 40, 44, 51, 58, 64, 71, 77, 84
ID internes 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
ID Objekt 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
IE Standard-Eigenschaften 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Index des Elementes in der Tab-Tasten-Folge .. 24, 35, 50, 57, 63, 71, 77, 84
Index des Objektes in der Collection document.all 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Inhalt eines Objektes ausrichten 40, 44, 51, 58, 64, 71, 77, 84
Inkonsistenz 31, 37, 42, 46, 53, 60, 66, 73, 79, 86
Inline-Style 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
interactive 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Interaktionsfähigkeit Objekt 22, 34, 39, 43, 49, 56, 62, 69, 70, 75, 76, 83
internes ID 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Kind 25, 36, 41, 45, 51, 58, 64, 71, 77, 84
Kind anhängen 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Kind erstes 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Kind letztes 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Kind nachfolgenes 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Kind Name 22, 35, 39, 43, 50, 57, 63, 70, 76, 83
Kind Objekt 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Kind Vorgänger 23, 35, 40, 44, 50, 57, 63, 70
Kinder Existenz 27, 37, 41, 46, 52, 59, 65, 72, 78, 85
Kind-Existenz möglich 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Kindknoten 28, 37, 42, 46, 52, 59, 65, 72, 79, 86
Klassenname 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Klassenreferenz 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Klick auf das Element 25, 36, 51, 58, 64, 71, 78, 85
klonen Objekt 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
Knoten als Kind anhängen 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Knoten Attribut 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Knoten Eltern 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Knoten Elternobjekt 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Knoten entfernen 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Knoten Existenz 27, 37, 41, 46, 52, 59, 65, 72, 78, 85
Knoten im DOM tauschen 33, 38, 42, 46, 53, 60, 66, 73, 80, 87
Knoten Kind 28, 37, 42, 46, 52, 59, 65, 72, 79, 86
Knotentyp 23, 35, 39, 43, 50, 57, 63, 70, 76, 83
Knotenwert 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Kommentar in einer Tabelle 24
konsistenten Struktur 31, 37, 42, 46, 53, 60, 66, 73, 79, 86
-Koordinate der linken oberen Ecke Objektes .. 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Layout-Komponente eines Objektes 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
letztes Kind Zeiger 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
lhs 22
linke obere Ecke des Objektes ... 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
linke oberen Ecke Objektes 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
linker Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
linker Rand des Objektes 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
loaded 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
loading 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
logischer Objektname 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Maus-Überwachung ausschalten für ein Objekt 33, 37, 53, 60, 66, 73, 79, 86
Maus-Überwachung einschalten für ein Objekt 33, 38, 53, 60, 66, 73, 80, 86
middle 40, 44, 51, 58, 64, 71, 77, 84
nachfolgenes Kind Zeiger 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Name des Kindes 22, 35, 39, 43, 50, 57, 63, 70, 76, 83
Namensraum laut XMLNS-Attribut 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
none 23
Normalisierung des DOM 31, 37, 42, 46, 53, 60, 66, 73, 79, 86
oberer Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
oberer Rand des Objektes 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Objekt Abstand zum linken Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt Abstand zum oberen Rand des Fensters 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt aktivieren 20, 34, 49, 56, 62, 69, 75, 82
Objekt attribute 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Objekt aus einem Objekt entfernen. 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Objekt Ausrichtung 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt Ausrichtung des Inhaltes 40, 44, 51, 58, 64, 71, 77, 84
Objekt Bezeichner 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt currTimeState 20, 24, 34, 35, 49, 51, 56, 58, 62, 64, 69, 71, 75, 77, 82, 84
Objekt durch anderes Objekt komplett ersetzen 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Objekt ersetzen durch ein Objekt 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Objekt Farbe des 3D-Rahmens 20, 69, 75, 82
Objekt Focussierbarkeit 22, 34, 49, 56, 62, 69, 75, 82
Objekt füllen mit Daten 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Objekt Hintergrundbild 20, 34, 49, 75, 82
Objekt Höhe 22, 69, 75, 82
Objekt HTML-Attribute 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
Objekt ID 22, 24, 34, 35, 39, 40, 43, 44, 49, 51, 56, 58, 62, 64, 69, 71, 75, 77, 82, 84
Objekt in eine Objekt einfügen .. 28, 37, 41, 46, 52, 59, 65, 72, 79, 85
Objekt Index in der Collection document.all 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Objekt Inhalt Ausrichtung 40, 44, 51, 58, 64, 71, 77, 84
Objekt Interaktionsfähigkeit 22, 34, 39, 43, 49, 56, 62, 69, 70, 75, 76, 83
Objekt internes ID 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt Kind 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Objekt klonen 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
Objekt Layout-Komponente 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
Objekt linke obere Ecke 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Objekt linke oberen Ecke 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Objekt linker Rand 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Objekt Maus-Überwachung ausschalten 33, 37, 53, 60, 66, 73, 79, 86
Objekt Maus-Überwachung einschalten 33, 38, 53, 60, 66, 73, 80, 86
Objekt mit HTML-Tags 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt oberer Rand 24, 35, 40, 44, 50, 57, 63, 70, 77, 84

Objekt Plain-Text 23, 50, 57, 63, 70, 76, 83
Objekt Rahmenfarbe 20, 69, 75, 82
Objekt rechte untere Ecke 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Objekt rechte unteren Ecke 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Objekt Referenz auf TextRectangle 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Objekt scrollen 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Objekt Selektierbarkeit 20, 34, 39, 43, 49, 56, 62, 69, 75, 77, 82
Objekt sichtbar 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Objekt Sprache 24, 35, 50, 57, 63, 71, 77, 84
Objekt Status 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Objekt table 1, 25, 28, 29, 33, 51, 52, 58, 59, 64, 65
Objekt table.caption 20, 34
Objekt table.col 39
Objekt table.colGroup 43
Objekt table.tBody 25, 28, 29, 49, 51, 52, 58, 59, 64, 65
Objekt table.tFoot 24, 25, 28, 29, 51, 52, 56, 58, 59, 64, 65
Objekt table.tHead 24, 25, 28, 29, 51, 52, 58, 59, 62, 64, 65
Objekt table.tr 68
Objekt table.tr.td 74, 75, 77, 82, 84
Objekt table.tr.th 75, 77, 81, 82, 84
Objekt Tag-Bezeichner 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Objekt Text 27, 36, 41, 45, 52, 58, 64, 72, 78, 85
Objekt Textbereich 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Objekt Umflussrichtung 22, 34, 39, 43, 49, 56, 62
Objekt Umgebung 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Objekt Viewbereich 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Objekt Zeiger auf TextRectangle 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Objekt Zugriff per Alt + Taste 20, 34, 39, 49, 56, 62, 69, 75, 82
Objektaktivität 22, 34, 49, 56, 62, 69, 75, 82
Objekt-Breite 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objekt-Content Editierbarkeit 22, 34, 39, 43, 49, 56, 62, 70, 76, 83
Objekthöhe 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Objektinhalt Mehrzeiligkeit 22, 34, 39, 43, 50, 56, 62, 70, 76, 83
Objektname logischer 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
onblur 25, 36, 51, 58, 64, 71, 78, 84
onclick 25, 33, 36, 38, 51, 53, 58, 60, 64, 66, 71, 73, 78, 80, 85, 86
ondblclick 33, 38, 53, 60, 66, 73, 80, 86
onmousedown 33, 37, 38, 53, 60, 66, 73, 79, 80, 86
onmousemove 33, 37, 38, 53, 60, 66, 73, 79, 80, 86
onmouseout 33, 37, 38, 53, 60, 66, 73, 79, 80, 86
onmouseover 33, 37, 38, 53, 60, 66, 73, 79, 80, 86
onmouseover Pixelgenauigkeit 25, 36, 41, 45, 51, 58, 64, 71, 78, 85
onmouseup 33, 37, 38, 53, 60, 66, 73, 79, 80, 86
Plain-Text 22, 23, 33, 34, 35, 37, 39, 40, 42, 43, 44, 46, 49, 50, 53, 56, 57, 60, 62, 63, 66, 69, 70, 73, 76, 79, 82, 83, 86
Plain-Text im Objekt 23, 50, 57, 63, 70, 76, 83
Positionen von 2 Knoten im DOM tauschen 33, 38, 42, 46, 53, 60, 66, 73, 80, 87
Rahmen um eine Tabelle 22
Rahmendicke 20
Rahmenfarbe 20, 69, 75, 82
rechte untere Ecke des Objektes 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
rechte unteren Ecke des Objektes 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Rectangle 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Referenz auf Feld der Zeiger auf TextRectangle-Objekte 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Referenz auf das document Objekt des Koten 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Referenz auf das Elternobjekt Body 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Referenz auf das ERSTE Kind 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Referenz auf das LETZTE Kind 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Referenz auf das NACHFOLGENDE Kind 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Referenz auf das Vorgängerkind 23, 35, 40, 44, 50, 57, 63, 70
Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag 22, 34, 39, 43, 49, 56, 62, 69, 75, 76, 82, 83
Referenz auf den gesamten Plain-Text im Objekt 23, 50, 57, 63, 70, 76, 83
Referenz auf Elternknoten 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Referenz auf Feld aller Elemente mit gemeinsamer URN 47, 48, 55
Referenz auf Feldelement 47, 48, 55
Referenz auf TextRectangle-Objekt 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Referenz der Eltern 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
regisitrieren eines Events 25, 36, 41, 45, 51, 58, 64, 72, 78, 85
registrieren eines Events 25, 36, 41, 45, 51, 58, 64, 71, 78, 84
rhs 22
row 77, 84
rowgroup 77, 84
rows 23
Script-Code einfügen 37, 79, 85
Scriptsprache 22, 34, 50, 57, 63, 70, 76, 83
Scrollbereich horizontal 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Scrollbereich vertikal 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Selektierbarkeit des Objektes 20, 34, 39, 43, 49, 56, 62, 69, 75, 82
Selektierbarkeit eines Objektes 77
Selektionsfähigkeit eines Objektes 77
setCapture() 33, 37, 53, 60, 66, 73, 79, 86
setExpression() 33, 37, 42, 46, 53, 60, 66, 73, 79, 86
Sonderzeichen 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
SPAN 43
Sprache für Anzeige von Sonderzeichen 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Sprache für Script 22, 34, 50, 57, 63, 70, 76, 83
Sprache zum Objekt 24, 35, 50, 57, 63, 71, 77, 84
Standardbehandlung Ereignis 25, 36, 41, 45, 52, 58, 64, 72, 78, 85
Standard-IE-Eigenschaften 25, 36, 40, 45, 51, 58, 64, 71, 77, 84
Status des Objektes 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Status Drag-Manipulation 26, 36, 41, 72, 78
STYLE 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Style-Eigenschaft Wert 27, 33, 37, 38, 41, 42, 45, 46, 52, 53, 59, 60, 65, 66, 72, 73, 78, 79, 80, 85, 86
Tabelle Abstand zwischen den Zellen 20
Tabelle Abstand zwischen Zellrahmen und dem Inhalt der Zelle 20
Tabelle Änderung der Anzahl Zeilen bzw. Spalten 33
Tabelle Anzahl der Sätze pro Dataset-Anzeige 20
Tabelle Anzahl der sichtbaren Datensätze 20
Tabelle Anzahl der Spalten einer Zelle 75, 82
Tabelle Anzahl der Spalten in der Tabelle 20
Tabelle Anzahl der Zeilen einer Zelle 77, 84
Tabelle Anzeige der Tabelle neu erzeugen 33
Tabelle äußerer Rahmen 23
Tabelle Datenbereitsstellung 3
Tabelle Datenbereitstellung in HTML 3
Tabelle Datenbereitstellung in JScript 3
Tabelle Datenbereitstellung per Active-X-Control 3
Tabelle Dateneinbindung 3
Tabelle Dateneinbindung in HTML 3
Tabelle Dateneinbindung in JScript 3
Tabelle Dateneinbindung per Active-X-Control 3
Tabelle Datenerzeugung zur Laufzeit 10
Tabelle Datensätze 20, 26, 28, 30, 31
Tabelle dynamische Daten 6
Tabelle dynamische Struktur 6
Tabelle erste Seite des Datasets 26
Tabelle erzeugen 1
Tabelle erzeugen in HTML 1
Tabelle erzeugen in JScript 2
Tabelle Fuss erzeugen 25
Tabelle Fuss löschen 25, 51
Tabelle Gruppierung von Spalten 77, 84
Tabelle Gruppierung einer Zelle 77, 84
Tabelle Gruppierung von Zeilen 77, 84
Tabelle Gruppierung, in der eine Zelle liegt 77, 84
Tabelle innere Rahmen 23
Tabelle Kommentar 24
Tabelle Kopf erzeugen 25
Tabelle Kopf löschen 25, 51
Tabelle Lage der Überschrift 35
Tabelle Lage der Zelle 77, 84
Tabelle Layoutveränderung per Style 19

Tabelle letzte Seite des Datasets ... 28
Tabelle nächste Seite des Datasets ... 30
Tabelle Rahmen ... 22
Tabelle Rahmen auf allen Seiten ... 22
Tabelle Rahmen außen ... 23
Tabelle Rahmen innen ... 23
Tabelle Rahmen links ... 22
Tabelle Rahmen links und rechts ... 22
Tabelle Rahmen oben und unten ... 22
Tabelle Rahmen oberhalb ... 22
Tabelle Rahmen rechts ... 22
Tabelle Rahmen unterhalb ... 22
Tabelle Spalten gruppieren ... 43
Tabelle Spaltengruppe Anzahl der Spalten ... 40, 44
Tabelle Spalten-Gruppierung ... 77, 84
Tabelle Spaltenüberschrift ... 81
Tabelle Strukturveränderung zur Laufzeit ... 10
Tabelle Trennlinie ... 23
Tabelle Überschrift erzeugen ... 25
Tabelle Überschrift Lage ... 35
Tabelle Überschrift löschen ... 25
Tabelle und Daten ... 3
Tabelle vorhergehende Seite des Datasets ... 31
Tabelle Zeile einfügen ... 28, 52, 59, 65
Tabelle Zeile löschen ... 58, 64
Tabelle Zeilen tauschen ... 29, 52, 59, 65
Tabelle Zeilen-Gruppierung ... 77, 84
Tabelle Zelle in Zeile einfügen ... 72
Tabelle Zelle in Zeile löschen ... 72
Tabelle Zelle und Gruppierung ... 77, 84
Tabellenfuss ... 56
Tabellenkopf ... 62
Tabellenkörper ... 49
Tabellen-Objektmodell ... 4
Tabellenspalte ... 39
Tabellenüberschrift ... 34
Tabellenzeile ... 68
Tabellenzelle ... 74, 81
TABINDEX ... 25, 27, 36, 51, 52, 58, 64, 71, 72, 78, 85
table Objekt ... 1, 25, 28, 29, 33, 51, 52, 58, 59, 64, 65
table.caption Objekt ... 20, 34
table.col Objekt ... 39
table.colGroup Objekt ... 43
table.rows Collection ... 47, 70
table.rows.cells Collection ... 48, 75, 82
table.tBodies Collection ... 55
table.tBody Objekt ... 25, 28, 29, 49, 51, 52, 58, 59, 64, 65
table.tBody.rows Collection ... 53, 71, 77
table.tBody.rows.cells Collection ... 54
table.tFoot Objekt ... 24, 25, 28, 29, 51, 52, 56, 58, 59, 64, 65
table.tFoot.rows Collection ... 60, 71, 77
table.tFoot.rows.cells Collection ... 61
table.tHead Objekt ... 24, 25, 28, 29, 51, 52, 58, 59, 62, 64, 65
table.tHead.rows Collection ... 66, 71, 77
table.tHead.rows.cells Collection ... 67
table.tr Objekt ... 68
table.tr.td Objekt ... 74, 75, 77, 82, 84
table.tr.th Objekt ... 75, 77, 81, 82, 84
Tab-Tasten-Folge ... 24, 35, 50, 57, 63, 71, 77, 84
Tabular Data Container ... 3
TAG-Bezeichner ... 22, 35, 39, 43, 50, 57, 63, 70, 76, 83
Tag-Bezeichner des Objektes ... 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
Tag-Bezeichner gemeinsam ... 48
Tag-Namen gemeinsam ... 48
Tastaturzugriff auf ein Objekt per Alt + Taste . 20, 34, 39, 49, 56, 62, 69, 75, 82
TBODY ... 49
TD ... 74
TDC ... 3
Text eines Objektes ... 27, 36, 41, 45, 52, 58, 64, 72, 78, 85
Textbereich des Elternobjektes ... 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Textknoten ... 23, 35, 39, 43, 50, 57, 63, 70, 76, 83
TextRectangle-Objekt ... 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
TFOOT ... 25, 51, 56, 71, 77
TH ... 81
THEAD ... 25, 51, 62, 71, 77
TIMEACTION ... 20, 22, 34, 49, 56, 62, 69, 75, 82
Timeline ... 20, 34, 49, 56, 62, 69, 75, 82
Timeline Synchronisierung der Animation des im Container liegenden Elementes auf Timeline ... 24, 35, 50, 57, 63, 71, 77, 84
TOBDY ... 71, 77
TOM ... 4
Tooltip-Text ... 24, 35, 51, 58, 64, 71, 77, 84
top ... 35, 40, 44, 51, 58, 64, 71, 77, 84
TR ... 68
Umflussrichtung ... 22, 34, 39, 43, 49, 56, 62
Umgebungsobjekt ... 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Uniform Resource Name ... 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
uninitialized ... 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
UNSELECTABLE ... 24, 35, 51, 58, 64, 77
URN ... 24, 35, 40, 44, 50, 57, 63, 71, 77, 84
vertikaler Scrollbereich ... 24, 35, 40, 44, 50, 57, 63, 70, 77, 84
Viewbereich Objekt ... 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
void ... 22
Vorgängerkind ... 23, 35, 40, 44, 50, 57, 63, 70
vsides ... 22
Wert einer Style-Eigenschaft ... 27, 33, 37, 38, 41, 42, 45, 46, 52, 53, 59, 60, 65, 66, 72, 73, 78, 79, 80, 85, 86
Wert eines Attributes belegen ... 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Wert eines per HTML-Attributes 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Wert vom Attribut ... 33, 38, 42, 46, 53, 60, 66, 73, 79, 86
Wortumbruch ... 76, 83
X-Koordinate der rechten unteren Ecke des Objektes ... 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
XMLNS ... 24, 35, 40, 44, 50, 57, 63, 70, 71, 77, 84
Y-Koordinate der linken oberen Ecke des Objektes 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Y-Koordinate der rechten unteren Ecke des Objektes ... 23, 35, 39, 44, 50, 57, 63, 70, 76, 83
Zeiger auf das ERSTE Kind ... 22, 34, 39, 43, 49, 56, 62, 69, 75, 82
Zeiger auf das LETZTE Kind ... 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Zeiger auf das NACHFOLGENDE Kind ... 22, 34, 39, 43, 50, 57, 63, 70, 76, 83
Zeiger auf das Vorgängerkind ... 23, 35, 40, 44, 50, 57, 63, 70
Zeiger auf Elternknoten ... 23, 35, 40, 44, 50, 57, 63, 70, 76, 83
Zeiger auf TextRectangle-Objekt ... 27, 36, 41, 45, 52, 59, 65, 72, 78, 85
Zeigertausch ... 33, 38, 42, 46, 53, 60, 66, 73, 80, 87
Zugriff per Alt + Taste ... 20, 34, 39, 49, 56, 62, 69, 75, 82