

window.document.frameset Objekt des Internet Explorer

window.document.frameset Objekt des Internet Explorer1
window.document.frameset.frame Objekt des Internet Explorer 10

Entweder BODY oder FRAMESET
FRAME ist Teil des FRAMESET

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

Erzeugung unter HTML:

```

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

```

Rahmen zwischen den einzelnen Fenstern des Framesets entfernen:

```

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

```

Hinweise:

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

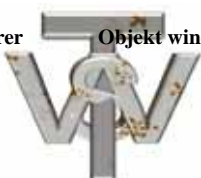
Zugriff:

```

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

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

```



index: ab 0
 Nummer des Frames im Dokument
 muss in [] kodiert sein

Möglichkeiten des Laden eines Dokumentes:

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

Laden eines fremden Dokumentes ohne Framedarstellung:

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

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

Eigenes Dokument wird durch fremde Webseite geladen:

CopyRight-Meldung auf fremden Host erzeugen:

Beispiel 1:

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

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

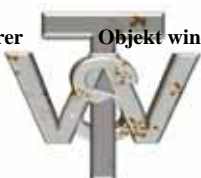
    if(host != HostUrl)
    {
        document.write(
            "Diese Seite wurde ausgeliehen bei "
            + "<A HREF=" + location.href
            + "' TARGET='_parent'"
            + ">"
            + HostUrl
            + "</A>"
        );
    }
}
```

Beispiel 2:

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

    var fremder_host_name = parent.location.hostname;

    if ( fremder_host_name != mein_host_name)
    {
```



```

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

```

Framedarstellung der eigenen Webseite sofort und ohne Bildschirmmeldung aktivieren:

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

```

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

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

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

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

```

Reload eines Dokumentes mit allen seinen FRAMES:

```

function frame_neu_laden()
{

```



```

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

    <FRAMESET onResize="frame_neu_laden()">

```

Datei ohne FRAMESET in eine Datei mit FRAMESET laden

```

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

```

Mehrere Frameinhalte gleichzeitig ändern:**Inhalte zweier Frames tauschen:**

Kodierung im Dokument, dass die beiden Frames definiert !

```

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

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

```

Inhalte beliebig vieler Frames als Ring tauschen:

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

Tausch:

erster Frame retten und dann mit zweiten Frame überschreiben
 zweiten Frame mit drittem Frame überschreiben

 vorletzten Frame mit letztem Frame überschreiben
 letzten Frame mit geretteten ersten Frame überschreiben

```

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

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

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

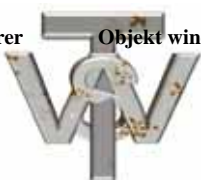
                    if(pos_doppelpunkt != -1) // nächstes Argument ist vorhanden
                    {
                        // ersten zu ersetzenden Framenamen retten

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

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

                        logischer_framename_ersetzender_frame =

```



```

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

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

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

```

Frame und Datenaustausch:

Zwischen Frames können Daten ausgetauscht werden.

Beispiel Adressierung eines Frame über das FRAMESET

```

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

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

```

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

```

im FRAMESET-Dokument wird kodiert          var Kette="Hallo !";
im FRAME-Dokument wird auf Kette zugegriffen: alert(parent.Kette);

```

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

```

im FRAME-Dokument wird kodiert          var Kette="Hallo !";
                                          <FRAME ...NAME="test2" ...>
im FRAMESET-Dokument wird auf Kette zugegriffen: alert(parent.test2.Kette);

```

Beispiel: Textdaten-Übergabe durch an Url angehängte Textdaten

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

```

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

<BODY>
An ziel.htm zu &uuml;bergabenden Text eingeben:
<FORM>
  <TEXTAREA       NAME=eingabe
                  ROWS=5
                  COLS=40
  >
</TEXTAREA>
<BR>
<INPUT   TYPE=button
          VALUE="Zielseite laden"

```



```

onClick=" ziel_seite_laden_mit_dateneubergabe(this.form.eingabe.value)">
</FORM>
</BODY>
</HTML>

```

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

```

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

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

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

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

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

```

Beispiel: Textdaten-Übergabe durch Fenster-Handle

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

HTML-Dokument, das die Textdaten enthält:

```

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

```

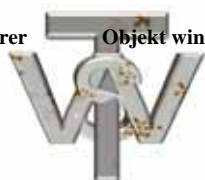
Dokument, das die Textdaten verarbeitet:

```

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

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

```



```

        handle.close();

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

```

frameset Objekt des Internet Explorer:

Container aller Frames

Die Collection frames wird unter document.frames beschrieben !

Das Laden einer neuen Webseite bei vorhandenem Frameset führt nur dann zur Darstellung außerhalb des Framesets, wenn ein Link z.B. <A> benutzt und dort ein **neues** Fenster geöffnet wird (TARGET-Attribut). Mit anderen Worten: Ein instanzierter Frameset ist z.B. per window.location.href oder window.location.replace() **nicht** aufhebbar, auch wenn das neue Dokument selbst einen Frameset instanziiert. Im letzteren Fall passiert optisch gesehen die Überlagerung des alten mit dem neuen Frameset. Allerdings hat der Programmierer mit Javascript ein riesen Problem: Der Zeiger parent als Zeiger auf dasjenige Dokument, das den Frameset deklariert, zeigt **nicht** auf das neu geladene Dokument mit neuem Frameset ! Daher werden alle Versuche, per Zeiger parent sich auf den neuen Eltern-Frameset zu beziehen, stets beim **vorherigen** Frameset landen und **nicht** beim neuen Dokument, das damit seine eigenen Frames nicht kennt. Somit werden nur Instanzen des alten Framesets vererbt, der auch die Kinder des neuen Frameset kennt, jedoch diese **ihn nicht**. Denn woher sollten die Kinder des neuen Framesets wissen, dass vorher bereits ein Frameset existiert und selbst wenn, sie können mit den Zwangs-Eltern in Form des alten Frameset nichts anfangen. Das Setzen des Zeigers parent auf den null-Wert wird vom Browser bemängelt. Die Verwendung des ID-Attributes im FRAMESET und das Null-Setzen per ID wird vom Browser akzeptiert, aber nicht ausgeführt. Das Schliessen des Eltern-Fensters per self.close() vor dem Umleiten per windows.location.href etc. wird ignoriert. Der Zeiger parent ist also nicht umzubiegen. Die Anwendung eines Zeigerbezuges vom alten Frameset über den neuen Frameset auf die Kinder des neuen Frameset, also die Verkettung von parent-Zeigern, ist ziemlich unsinnig, da parent ansich der Zeiger auf das aktuelle Frameset sein sollte, auch wenn in der Objekthierarchie der neue Frameset ein Kind vom alten Frameset sein müsste. Alternativ zu dieser unangenehmen Framesetüberlagerung sind Konstruktionen von DIV-Objekten innerhalb eines **gemeinsamen** Dokumentes möglich, wobei in die DIV's dann je ein Dokument geladen wird (document.open()). Es können dann alle Dokumente direkt über das ID des jeweiligen DIV's adressiert werden, allerdings haben diese DIV's eben **keine** Fenstereigenschaften wie Frames im gemeinsamen Frameset-Fenster. (DIV- und FRAME/IFRAME-Objekte unterscheiden sich in Eigenschaften und Methoden). So gesehen stellt document.open() eine eingeschränkte Lösung dar. HTML-orientiert ist die Nutzung eines Links, z.B. <A> mit Targetwert _top, also dem Zeiger auf das oberste Fenster, in dem der alte Frameset sitzt. Nur ist ein Link in der Regel visuell vorhanden und muss durch den User aktiviert werden. Letzte Alternative ist die Umlenkung per META-Tag anhand eines Timers.

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

Zugriff auf Frameset-Eigenschaft:

Beispiel

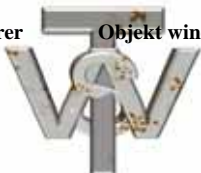
```
var Wert = document.all.ZeigerAufFrameset.eigenschaft;
```

mit ZeigerAufFrameset laut ID-Attribut

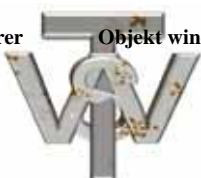
```
<FRAMESET ID="ZeigerAufFrameset" ..... >
```

Eigenschaften:

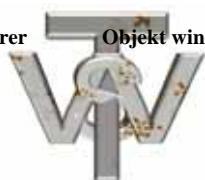
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.border	Rahmendicke in Pixel
.borderColor	Borderfarbe (Rahmenfarbe)
	wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no"
	Eigenschaft .border mit Wert 0
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassename
.cols	Breite aller Frames eines Frameset (Breite des Frameset als Container)
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.frameBorder	Borderanzeige ein/aus beim Frame
.frameSpacing	Zwischenraum in Pixel zwischen 2 benachbarten Frames
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname)
	Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und



	betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id); Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetags
.innerHTML	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isContentEditable	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isDisabled	Mehrzeiligkeit des Objekthinhaltes
.isMultiLine	Erzeugbarkeit eines Textbereiches
.isTextEdit	Sprache für Anzeige von Sonderzeichen etc.
.lang	Sprache für Script festlegen
.language	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.lastChild	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !!
.name	Element darf nicht per Methode .createElement() erzeugt worden sein Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nextSibling	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeName	Knotentyp laut attributes Collection Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.nodeType	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.nodeValue	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.ownerDocument	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentElement	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentNode	Textbereich des Elternobjektes referenzieren
.parentTextEdit	Referenz auf das Vorgängerkind
.previousSibling	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.readyState	Höhe aller Frames eines Frameset (Höhe des Frameset als Container)
.rows	Namensraum laut XMLNS-Attribut
.scopeName	Index des Objektes in der Collection document.all
.sourceIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup
.tabIndex	Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.width	Breite des Objektes in Pixel
Methoden:	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert



	Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt 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
.cloneNode()	DOM wird geändert 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
.getElementsByTagName()	Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByName()
.hasChildNodes()	DOM nicht geändert prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert
.insertAdjacentElement()	Objekt in eine Objekt einfügen und Referenz liefern, wobei die Lage definiert werden kann wenn Element bereits eingefügt vorhanden, so wird dieses nur verschoben laut Lage des Objektes im DOM nur nach dem kompletten Laden des Dokumentes möglich DOM wird geändert
.insertAdjacentHTML()	HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen nicht ausgeführt eingefügter Code wird nur dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER> muss kodiert werden DOM wird geändert



.insertAdjacentText()	Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert
.insertBefore()	Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag 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
.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
.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
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparst DOM wird geändert

window.document.frameset.frame Objekt des Internet Explorer

Frame muss im FRAMESET liegen.

Die Collection frames wird unter document.frames beschrieben !

Zugriff auf Frame-Eigenschaft:

Beispiel

```
var Wert = document.all.ZeigerAufFrame.eigenschaft;
```

mit ZeigerAufFrame laut ID-Attribut

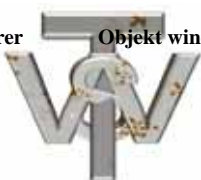
```
<FRAME ID="ZeigerAufFrame" ..... >
```

Transparenter Content des Frame:

ab IE 5.5

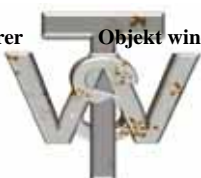
für den Frame muss das Attribut .ALLOWTRANSPARENCY auf true gesetzt sein

Im Dokument, das in den Frame geladen wird, muss im BODY die Hintergrundfarbe (background-color oder BGCOLOR-Attribut) auf transparent gesetzt sein.

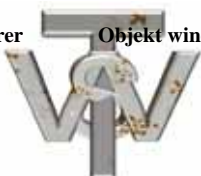


Eigenschaften:

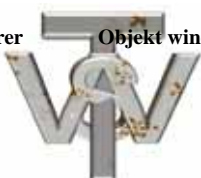
.allowTransparency	Transparenz ein/aus Transparenz: z.B. Hintergrundfarbe des Elternobjektes wird zur Hintergrundfarbe des Frames Achtung: Wenn Frame-eigenes STYLE-Attribut mit Wert für background-color also STYLE="background-color: farb_bezeichner" kodierte wurde, so wird die Transparenz ignoriert und die Hintergrundfarbe laut STYLE verwendet
APPLICATION	Umgebung des Frames mit Vertrauen-Status im Rahmen des Browser-Sicherheitsmodells
ATOMICSELECTION	Attribut wird nicht vererbt an Kinder-Frame
.borderColor	Selektierbarkeit des Objektes einstellen Borderfarbe (Rahmenfarbe) wird ignoriert wenn Eigenschaft .frameBorder mit Wert 0 oder "no" Eigenschaft .border mit Wert 0
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.contentWindow	Referenz auf das zugehörige Fenster-Objekt laut Collection document.all
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.frameBorder	Borderanzeige ein/aus beim Frame
.height	Höhe des Objektes in Pixel
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.longDesc	Uniform Resource Identifier (URI) zur einer "long description" umwandeln
.marginHeight	Abstand in Pixel vom unteren zum oberen Rand des Objektes
.marginWidth	Abstand in Pixel vom linken zum rechten Rand des Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker Knotentyp laut attributes Collection
.nextSibling	
.nodeName	
.nodeType	
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.noResize	Frame-Größenänderung ein/aus Größenänderung z.B. durch Maus etc
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.scopeName	Namensraum laut XMLNS-Attribut
.scrolling	Scrollenbar erzeugen
SECURITY	temporäre Sicherheit des IFRAME



	wird an Kinder weitervererbt ab IE 6.0
.self	Referenz auf das aktuelle Fenster oder den aktuellen Frame
.sourceIndex	Index des Objektes in der Collection document.all
.src	Url der Daten z.B. vom Image in normaler Auflösung
.tabIndex	Index des Elementes in der Tab-Tasten-Folge für Anspringen des Dokumentes Anspringen verbunden mit Focus erhalten --> Ereignisse werden ausgelöst !! unter IE 5.x onblur, onfocus ab IE 5.x onblur, onfocus, onkeydown, onkeypress, onkeyup Anspringen default per TAB-Taste für A, BODY, BUTTON, FRAME, IFRAME, IMG, INPUT, SELECT, TEXTAREA Anspringen default nicht per TAB-Taste für APPLET, DIV, FRAMESET, SPAN, TABLE, TD
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.title	Tooltip-Text bei Mouse over über Objekt
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden
UNSELECTABLE	Selektionsfähigkeit eines Objektes
.width	Breite des Objektes in Pixel
Methoden:	
.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.attachEvent()	Einschalten des Registrieren eines Events durch Eventhandler Hinweis: Abschalten mit Methode .detachEvent() Achtung: Wenn mehrere Eventhandler zum Event, so Aufruf der Handler leider NICHT verkettet sondern in Zufallsfolge , es sei denn die Handler prüfen ihre Aufruffolge (muss programmiert werden)
.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
.cloneNode()	DOM wird geändert 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

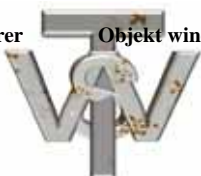


	nur nach dem kompletten Laden des Dokumentes vor IE 5.x: Objekt muss TABINDEX-Attribut besitzen
.getAdjacentText()	Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht
.getAttribute()	DOM nicht geändert Wert eines per HTML erzeugten Attributes liefern
.getAttributeNode()	DOM nicht geändert 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
.getElementsByTagName()	DOM nicht geändert Referenz auf ein Feld (Collection) aller im Objekt befindlichen Kinder-Objekte mit gemeinsamen Tagnamen liefern, inklusive aller Kinder und Unterkinder etc. Hinweis: Natürlich kann auch das document-Objekt so verarbeitet werden (beachte dabei document.all Collection) Achtung: Kinder-Objekte, die keinen Tag-Name besitzen, werden nicht erfasst ! Für Verwaltung per ID (analog zum ID-Attribut): siehe Methode getElementById() Für Verwaltung per NAME (analog zum NAME-Attribut): siehe Methode .getElementsByTagName()
.hasChildNodes()	DOM nicht geändert prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt
.insertAdjacentElement()	DOM nicht geändert 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
.mergeAttributes()	DOM wird geändert alle Attribute eines Elementes in ein anderes Element kopieren und eventuell die Attribute im Ziel mischen Attribute sind: HTML Events Styles ab IE 5.01 auch ID, NAME Achtung: Diese Methode ist mir Vorsicht zu geniessen !!
.normalize()	DOM wird geändert Normalisierung des DOM zur Erreichung einer konsistenten Struktur 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
.removeAttributeNode()	DOM wird geändert entfernen von Attribut, egal ob es mit oder ohne HTML-Anweisung erzeugt wurde, und Referenz auf das entfernte Attribut liefern
.removeBehavior()	DOM wird geändert per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie)
.replaceAdjacentText()	DOM wird geändert Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
.setActive()	DOM wird nicht geändert 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
.setAttributeNode()	DOM wird nur bei Erzeugung geändert Attribut einem Knoten zuweisen und Referenz liefern
.swapNode()	DOM wird geändert Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparkt



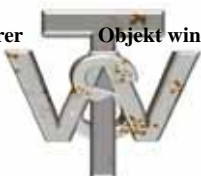
"#default#behaviorName 8, 12
#text 8, 11
.addBehavior() 8, 12
.allowTransparency 11
.appendChild() 8
.applyElement() 8, 12
.attachEvent() 9, 12
.blur() 9, 12
.border 7
.borderColor 7, 11
.canHaveChildren 7
.className 7, 11
.clearAttributes() 9, 12
.cloneNode() 9, 12
.cols 7
.componentFromPoint() 9, 12
.contains() 9, 12
.contentWindow 11
.createElement() 10
.dataFld 11
.dataSrc 11
.detachEvent() 9, 12
.disabled 11
.dragDrop() 12
.fireEvent() 9, 12
.firstChild 7, 11
.focus() 9, 12
.frameBorder 7, 11
.frameSpacing 7
.getAdjacentText() 9, 13
.getAttribute() 9, 13
.getAttributeNode() 9, 13
.getElementsByName() 9, 13
.getElementsByTagName() 9, 13
.hasChildNodes() 9, 13
.height 11
.hideFocus 7, 11
.id 7, 11
.innerHTML 8
.insertAdjacentElement() 9, 13
.insertAdjacentHTML() 9
.insertAdjacentText() 10
.insertBefore() 10
.isContentEditable 8, 11
.isDisabled 8, 11
.isMultiLine 8, 11
.isTextEdit 8, 11
.lang 8, 11
.language 8, 11
.lastChild 8, 11
.longDesc 11
.marginHeight 11
.marginWidth 11
.mergeAttributes() 10, 13
.name 8, 11
.nextSibling 8, 11
.nodeName 8, 11
.nodeType 8, 11
.nodeValue 8, 11
.noResize 11
.normalize() 10, 13
.offestParent 11
.offserLeft 11
.offsetHeight 11
.offsetHeigt 11
.offsetLeft 11
.offsetParent 11
.offsetTop 11
.offsetWidth 11
.outerHTML 8
.ownerDocument 8, 11
.parentElement 8, 11

.parentNode 8, 11
.parentTextEdit 8, 11
.previousSibling 8, 11
.readyState 8, 11
.recordNumber 11
.removeAttribute() 10, 13
.removeAttributeNode() 10, 13
.removeBehavior() 10, 13
.removeChild() 10
.removeNode() 10
.replaceAdjacentText() 10, 13
.replaceChild() 10
.replaceNode() 10
.rows 8
.scopeName 8, 11
.scrolling 11
.self 12
.setActive() 10, 13
.setAttribute() 10, 13
.setAttributeNode() 10, 13
.sourceIndex 8, 12
.src 12
.swapNode() 10, 13
.tabIndex 8, 12
.tagName 8, 12
.tagUrn 8, 12
.title 8, 12
.uniqueID 7, 8, 11, 12
.width 8, 12
Abmahnungsgefahr 2
aktivieren Eventdurchreichung 10, 13
aktueller Frame Referenz 12
aktuelles Fenster Referenz 12
an Url angehängte Textdaten 5
anhängen Kind 8
anhängen Knoten 8
Anker 8, 11
Anzeigesprache 8, 11
APPLICATION 11
ATOMICSELECTION 7, 11
Attribut automatisch erzeugen und mit Wert belegen 10, 13
Attribut eines Knoten 10, 13
Attribut erzeugen und mit Wert belegen 10, 13
Attribut Wert 9, 10, 13
Attribute eines Elementes 10, 13
Attribute HTML 10, 13
attribute Objekt 9, 13
attributes Collection 8, 11
Attribut-Name 8, 11
auslösen Event 9, 12
automatisch-generiertes ID des Objektes 8, 12
Bereich ab inklusive HTML-Start bis hinter -Ende-Tag 8
Bereich zwischen HTML-Start und -Ende-Tag 8
Bezeichner des Objektes 7, 11
Bezeichner des Tag eines Objektes 8, 12
Body Eltern 8, 11
Body Referenz auf das Elternobjekt 8, 11
Borderdicke 7
Borderfarbe 7, 11
Breite Frame 7
canHaveHTML 7, 11
childNodes Collection 7, 11
Collection attributes 8, 11
Collection childNodes 7, 11
Collection document.all 8, 11, 12
Collection document.frames 7, 10
Container 9, 12
Copyright-Meldung 2
createAttribute() 10, 13
Cursorgeschwindigkeit 9, 12
Daten Url 12
Datenaustausch zwischen Frames 5



Datenfeld..... 11
 Datenquelle als Anker festlegen..... 11
 Datenquelle-Name vergeben..... 11
 Datenquelle-Satznummer..... 11
 DEFER..... 9
 DHTML-Eigenschaft hinzufügen..... 8, 12
 document Objekt des Knoten..... 8, 11
 document.all Collection..... 8, 11, 12
 document.frames Collection..... 7, 10
 Dokument Eigenes Dokument wird durch fremde Webseite geladen..... 2
 Dokument fremdes Dokument laden ohne Framedarstellung..... 2
 Dokument Laden eines fremden Dokumentes ohne Framedarstellung..... 2
 Dokument Reload mit allen seinen FRAMES..... 3
 DOM..... 10
 DOM Elementeigenschaft..... 8, 12
 DOM Normalisierung..... 10, 13
 Drag-Manipulation Status..... 12
 durchreichen Event..... 10, 13
 Editierbarkeit des Objekt-Content..... 8, 11
 Eigenes Dokument wird durch fremde Webseite geladen..... 2
 Eigenschaft des attribute-Objektes..... 9, 13
 Eigenschaft Element..... 10, 13
 Eigenschaft hinzufügen..... 8, 12
 Eigenschaften IE Standard..... 8, 12
 Element Attribute..... 10, 13
 Element den Focus wegnehmen..... 9, 12
 Element Eigenschaft..... 10, 13
 Element innerhalb eines Elementes..... 9, 12
 Elementeigenschaft im DOM..... 8, 12
 Elementes Tab-Tasten-Folge..... 8, 12
 Eltern..... 8, 12
 Eltern Body..... 8, 11
 Eltern Referenz..... 11
 Elternknoten..... 8, 11
 Elternobjekt Textbereich..... 8, 11
 Ereignis Standardbehandlung..... 9, 12
 erstes Kind Referenz..... 7, 11
 erstes Kind Zeiger..... 7, 11
 erzeugen Attribut automatisch und mit Wert belegen..... 10, 13
 erzeugen Attribut und mit Wert belegen..... 10, 13
 escape()..... 5
 Event auslösen..... 9, 12
 Event durchreichen..... 10, 13
 Event onblur..... 9, 12
 Event onfocus..... 9, 12
 Event onmouseover Pixelgenauigkeit..... 9, 12
 Event registrieren..... 9, 12
 Eventdurchreichung aktivieren..... 10, 13
 Events registrieren..... 9, 12
 Existenz Kind möglich..... 7
 Existenz von Kinder..... 9, 13
 Farbe Border..... 7, 11
 Farbe Rahmen..... 7, 11
 Fenster aktuell Referenz..... 12
 Fenster-Handle..... 6
 Fenster-Objekt eines Objektes..... 11
 Focus setzen..... 9, 12
 Focus wegnehmen..... 9, 12
 Focus-Event..... 9, 12
 FRAME..... 1
 Frame aktuell Referenz..... 12
 Frame Breite..... 7
 Frame Größenänderung..... 11
 Frame Höhe..... 8
 frame Objekt..... 1
 FRAME Reload eines Dokumentes mit allen seinen FRAMES..... 3
Frame und Datenaustausch..... 5
 Frame und Status im Rahmen des Browser-Sicherheitsmodells..... 11
 Framedarstellung und Abmahnungsgefahr..... 2
 Framedarstellung und rechtliche Zulässigkeit..... 2
Frameinhalte..... 4
 Frames Zwischenraum..... 7

frames[]..... 4
 FRAMESET..... 1, 10
 fremde Webseite..... 2
 Fremdes Dokument laden ohne Framedarstellung..... 2
 Größenänderung Frame..... 11
 Handle..... 6
 hinzufügen DHTML-Eigenschaft..... 8, 12
 hinzufügen Eigenschaft..... 8, 12
 Höhe des Objektes..... 11
 Höhe Frame..... 8
 HTML-Attribut Wert..... 9, 13
 HTML-Attribute..... 10, 13
 HTML-Attribute eines Objektes..... 9, 12
 HTML-Code und/oder Script-Code einfügen..... 9
 HTML-Tags im Objekt..... 7, 11
 ID..... 8, 9, 11, 13
 ID des Objektes..... 8, 12
 ID internes..... 7, 11
 ID Objekt..... 7, 11
 IE Standard-Eigenschaften..... 8, 12
 Index des Elementes in der Tab-Tasten-Folge..... 8, 12
 Index des Objektes in der Collection document.all..... 8, 12
 Inkonsistenz..... 10, 13
 Interaktionsfähigkeit Objekt..... 8, 11
 internes ID..... 7, 11
 Kind..... 8, 12
 Kind anhängen..... 8
 Kind erstes..... 7, 11
 Kind letztes..... 8, 11
 Kind nachfolgenes..... 8, 11
 Kind Name..... 8, 11
 Kind Objekt..... 10
 Kind Vorgänger..... 8, 11
 Kinder Existenz..... 9, 13
 Kind-Existenz möglich..... 7
 Kindknoten..... 10
 Klassenname..... 7, 11
 Klassenreferenz..... 7, 11
 klonen Objekt..... 9, 12
 Knoten als Kind anhängen..... 8
 Knoten Attribut..... 10, 13
 Knoten Eltern..... 8, 11
 Knoten Elternobjekt..... 8, 11
 Knoten entfernen..... 10
 Knoten Existenz..... 9, 13
 Knoten im DOM tauschen..... 10, 13
 Knoten Kind..... 10
 Knotentyp..... 8, 11
 Knotenwert..... 8, 11
 konsistenten Struktur..... 10, 13
 -Koordinate der linken oberen Ecke Objektes..... 11
 Laden eines fremden Dokumentes ohne Framedarstellung..... 2
 Layout..... 8, 11
 Layout-Komponente eines Objektes..... 9, 12
 letztes Kind Zeiger..... 8, 11
 linke obere Ecke des Objektes..... 11
 linke obere Ecke Objektes..... 11
 logischer Objektname..... 7, 11
 Mehrere Frameinhalte gleichzeitig ändern..... 4
 Mehrzeiligkeit des Objektinhaltes..... 8, 11
 nachfolgenes Kind Zeiger..... 8, 11
 Name des Kindes..... 8, 11
 Name des Objektes..... 8, 11
 Normalisierung des DOM..... 10, 13
 Objekt attribute..... 9, 13
 Objekt aus einem Objekt entfernen..... 10
 Objekt Bezeichner..... 7, 11
 Objekt durch anderes Objekt komplett ersetzen..... 10
 Objekt ersetzen durch ein Objekt..... 10
 Objekt frame..... 1
 Objekt füllen mit Daten..... 8, 11
 Objekt Höhe..... 11
 Objekt HTML-Attribute..... 9, 12



Objekt ID 7, 8, 11, 12

Objekt in eine Objekt einfügen 9, 13

Objekt Index in der Collection document.all 8, 12

Objekt Interaktionsfähigkeit 8, 11

Objekt internes ID 7, 11

Objekt Kind 10

Objekt klonen 9, 12

Objekt Layout-Komponente 9, 12

Objekt linke obere Ecke 11

Objekt linke oberen Ecke 11

Objekt mit HTML-Tags 7, 11

Objekt Name 8, 11

Objekt Rahmenfarbe 7, 11

Objekt rechte untere Ecke 11

Objekt rechte unteren Ecke 11

Objekt Selektierbarkeit 7, 8, 11, 12

Objekt Status 8, 11

Objekt Tag-Bezeichner 8, 12

Objekt Text 9, 13

Objekt Textbereich 8, 11

Objekt Transparenz 11

Objekt-Content Editierbarkeit 8, 11

Objektes Fenster-Objekt 11

Objektinhalt Mehrzeiligkeit 8, 11

Objektname logischer 7, 11

onblur 9, 12

onmouseover Pixelgenauigkeit 9, 12

Plain-Text 10, 13

Positionen von 2 Knoten im DOM tauschen 10, 13

Rahmendicke 7

Rahmenfarbe 7, 11

rechte untere Ecke des Objektes 11

rechte unteren Ecke des Objektes 11

Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag 8

Referenz auf das document Objekt des Knoten 8, 11

Referenz auf das Elternobjekt Body 8, 11

Referenz auf das ERSTE Kind 7, 11

Referenz auf das LETZTE Kind 8, 11

Referenz auf das NACHFOLGENDE Kind 8, 11

Referenz auf das Vorgängerkind 8, 11

Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag 8

Referenz auf Elternknoten 8, 11

Referenz der Eltern 11

registrieren eines Events 9, 12

registrieren eines Events 9, 12

Reload eines Dokumentes mit allen seinen FRAMES 3

Reload mit allen FRAMES 3

reload()4

Script-Code einfügen9

Scriptsprache8, 11

SECURITY 11

Selektierbarkeit des Objektes7, 11

Selektierbarkeit eines Objektes8, 12

Selektionsfähigkeit eines Objektes8, 12

Sonderzeichen8, 11

Sprache für Anzeige von Sonderzeichen8, 11

Sprache für Script8, 11

Standardbehandlung Ereignis9, 12

Standard-IE-Eigenschaften8, 12

Status des Objektes8, 11

Status Drag-Manipulation12

TABINDEX9, 12, 13

Tab-Tasten-Folge8, 12

TAG-Bezeichner8, 11

Tag-Bezeichner des Objektes8, 12

Text eines Objektes9, 13

Textbereich des Elternobjektes8, 11

Textdaten-Übergabe durch an Url angehängte Textdaten5

Textdaten-Übergabe durch Fenster-Handle6

Tooltip-Text8, 12

Transparenz11

unescape()6

Uniform Resource Identifier11

Uniform Resource Name8, 12

UNSELECTABLE8, 12

URI11

Url der Daten12

Url mit angehängten Textdaten5

URN8, 12

Vorgängerkind8, 11

Wert eines Attributes belegen10, 13

Wert eines per HTML-Attributes9, 13

Wert vom Attribut10, 13

X-Koordinate der rechten unteren Ecke des Objektes11

XMLNS8, 11, 12

Y-Koordinate der linken oberen Ecke des Objektes11

Y-Koordinate der rechten unteren Ecke des Objektes11

Zeiger auf das ERSTE Kind7, 11

Zeiger auf das LETZTE Kind8, 11

Zeiger auf das NACHFOLGENDE Kind8, 11

Zeiger auf das Vorgängerkind8, 11

Zeiger auf Elternknoten8, 11

Zeigertausch10, 13

Zwischenraum zwischen 2 benachbarten Frames7

