window.document.form Objekt des Internet Explorer

Dieses Objekt dient der formular-orientierte Datenbeschaffung zum Zweck der Datenübersendung an den Server. Es ist möglich, mit einem Formular versteckt Daten zu transferieren, die dem User nicht angezeigt bzw. bewusst gemacht werden. Der Programmierer sollte bei der Nutzung von Scripten in Formularen die Sorgfaltspflicht bezüglich Userdaten umsetzen. Im Internet Explorer ist dem User das Abschalten der Autovervollständigung auch bezüglich Formulare anzuraten.
Das input Objekt ist der Container für alle Input-Varianten im Formular. Das Objekt img ist der Container für input image. Das Objekt button ist der Container für input button. Das Objekt textarea ist der Container für input textarea. Das Objekt select ist der Container für input option und input select.Container vererben ihre Eigenschaften an die Formularkomponenten.

Erzeugung unter HTML:

Beispiel:

```
<FORM   ID="ID_Formular"
        NAME="logischer_formular_name"      // muss für alle zu sendenden Formular-Elemente ebenfalls
                                            //      kodiert sein, wenn die Elemente Daten des
                                            //      Formulars darstellen
        TARGET="zielfenster"                // Ausgabefenster z.B. des CGI-Scripts für
        ACTION="auswertungs_url"            //   Auswertung und Antwort auf abgeschicktes Formular
        METHOD="get"  oder  "post"
        ENCTYPE="mime_typ"                  // z.B. text/* für E-Mail
        onreset="eventhandler1"             // erzeugt Event onreset
                                                    // Reaktion VOR dem Rücksetzen ist zu programmieren per
                                            Eventhandler:
                                            //      muß return-Anweisung haben:
                                            //          return true;  so wird Formular auch
                                            //                      zurückgesetzt
                                            //          return false; so wird Formular nicht
                                            //                      zurückgesetzt
                                            // Das Rücksetzen an sich macht der Browser per
                                            //          Methode .reset()

        onsubmit="eventhandler2"            // erzeugt Event onsubmit
                                            // Reaktion VOR dem Senden ist zu programmieren per Eventhandler:
                                            //      muß return-Anweisung haben:
                                            //          return true;  so wird Formular auch
                                            //                      abgesendet
                                            //          return false; so wird Formular nicht
                                            //                      abgesendet
                                            // Das Absenden an sich macht der Browser per
                                            //          Methode .submit()
>
        formularinhalt
</FORM>
```

Formularinhalte können diverse HTML-Elemente sein., vorallem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann.

Beispiele:

Beispiel 1:

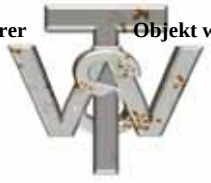
```
<HTML>
    <FORM ACTION="http://www.test.de/sample.asp"  METHOD="POST">
        <SELECT NAME="Flavor">
            <OPTION VALUE="Test1"> Test1
            <OPTION VALUE="Test2"> Test2
            <OPTION VALUE=" Test3" SELECTED> Test3
        </SELECT>
        <INPUT TYPE=SUBMIT>
    </FORM>
</HTML>
```

Beispiel 2:

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

Beispiel 3:

```
<HTML>
<HEAD>
<STYLE>
        .user_data_speicher_klasse {behavior:url(#default#userData);}
</STYLE>
<SCRIPT>
        // Cachname festlegen
        //      es können diverse Cachenamen definiert und somit Versionen von Cache
        //              verwaltet werden
        var FreierCacheName = "InputCache";

        // Cache-Attribut festlegen
        //      es können diverse Attribute definiert und somit Versionen von Input-Daten
        //              verwaltet werden
        var FreiesCacheAttribut = "InputCacheAttribut";

        // zu cachende Daten referenzieren
        var InputDatenObjekt = ID_Formular.ID_Input;

        function InputSichern()
        {
                // +++++++++ Zeitstempel ++++++++++++++++++++++
                var ZeitpunktJetzt = new Date();
                var ZeitpunktJetztInMinuten = ZeitpunktJetzt.getMinutes();

                // Zeitstempel festlegen: Ab jetzt + 20 Minuten
                var Zeitstempel = ZeitpunktJetztInMinuten + 20;

                // und als UTC-Format erzeugen
                var ZeitstempelUTC = Zeitstempel.toUTCString();

                // und Zeitstempel dem Input-Objekt verpassen
                ID_Input.expires = ZeitstempelUTC;

                // ++++++++ Daten chachen +++++++++++++++++++++
                // aktuelle Daten holen laut Input-Objekt
                var InputDaten = InputDatenObjekt.value;

                // Attrbibut instanzieren und mit Daten füllen
                InputDatenObjekt.setAttribute(FreiesCacheAttribut, InputDaten);

                // und Cache saven
                InputDatenObjekt.save(FreierCacheName);
        }

        function InputLaden()
        {
                // Cache laden
                InputDatenObjekt.load(FreierCacheName);

                // und Daten zum Attribut laut Sicherung lesen
                var InputDaten = InputDatenObjekt.getAttribute(FreiesCacheAttribut);
        }

</SCRIPT>
</HEAD>
<BODY>
        <FORM ID="ID_Formular">
                <INPUT   ID="ID_Input"
                         CLASS="user_data_speicher_klasse"
                         TYPE="text"
```

```
                        >
                        <INPUT TYPE="button" VALUE="sichern der Input-Daten" onclick="InputSichern()">
                        <INPUT TYPE="button" VALUE="laden der Input-Daten"    onclick="InputLaden()">
                </FORM>
        </BODY>
        </HTML>
```

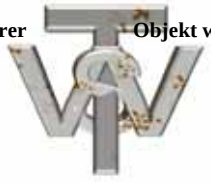
Beispiel 4:

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

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

Beispiel 5:

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

                        // wirklich senden ???
                        return( confirm("Wirklich senden ?"));
                                // true  so Submit durch Browser ausführen lassen
                                // false so kein Submit durch Browser
                }
        </SCRIPT>
        </HEAD>
        <BODY>
                <DIV>
                        <FORM NAME="Formular" ACTION=" ...."  METHOD="   "
                                onsubmit="EventHandler_AktionVorSubmit();">
                                <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
                                <BR>
                                <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
                                <BR><BR>
                                <INPUT TYPE="submit" VALUE="oder hiermit senden"
                                        onclick="form.submit()"
                                >
                        </FORM>
                </DIV>
                <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
        </BODY>
        </HTML>
```

Beispiel 6:



```
<FORM ACTION="test.htm">
    <INPUT TYPE="submit" VALUE="Gehe zu test.htm">
</FORM>
```

Zugriff:

Hinweise: Anstelle des logischen Namens laut NAME-Attribut kann auch der Wert des ID-Attributes verwendet werden. Das NAME-Atrribut muss kodiert werden, wenn Daten des Formularelementes zu senden sind per .submit() (zu kodieren im Formular selbst und für jedes Formular-Element, das Daten senden soll).

auf das Formular:

beim NS:

logischer_formular_name.eigenschaft
logischer_formular_name.methode()

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

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

beim IE:

document.all.**logischer_formular_name**.eigenschaft
document.all.**logischer_formular_name**.methode()

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

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

auf ein Formular-Element:

Formularinhalte können diverse HTML-Elemente sein., vorallem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann. **Jedes Element** innerhalb von <FORM> .. </FORM> wird zum Kind des Objektes form. Daher ist Punktnotation nötig, um den Formularinhalt, also die Kinders des Objektes form, referenzieren zu können.

beim NS:

logischer_formular_name.logischer_formular_element_name.eigenschaft
logischer_formular_name.logischer_formular_element_name.methode()

var ZeigerAufFormular = document.forms**[**index**]**;
ZeigerAufFormular.**logischer_formular_element_name**.eigenschaft
ZeigerAufFormular.**logischer_formular_element_name**.methode()

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

var ZeigerAufFormular = document.forms**[**index1**]**;
ZeigerAufFormular.elements**[**inde2**]**.eigenschaft
ZeigerAufFormular.elements**[**inde2**]**.methode()

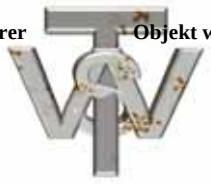
index1: ab 0
Nummer des Formulars im Dokument
muss in **[ ]** kodiert sein

index2: ab 0
Nummer des Formular-Elementes im Formular
muss in **[ ]** kodiert sein

var ZeigerAufFormular = document.forms**[**index1**]**;
var ZeigerAufFormularElement = ZeigerAufFormular.elements**[**index2**]**;
ZeigerAufFormularElement.eigenschaft
ZeigerAufFormularElement.methode()

index1: ab 0
Nummer des Formulars im Dokument
muss in **[ ]** kodiert sein

index2: ab 0



Nummer des Formular-Elementes im Formular
muss in **[ ]** kodiert sein

beim IE:

document.all.**logischer_formular_name.logischer_formular_element_name.**eigenschaft
document.all.**logischer_formular_name.logischer_formular_element_name.**methode()

var ZeigerAufFormular = document.forms**[**index**]**;
ZeigerAufFormular.**logischer_formular_element_name**.eigenschaft
ZeigerAufFormular.**logischer_formular_element_name**.methode()

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

var ZeigerAufFormular = document.forms**[**index1**]**;
ZeigerAufFormular.elements**[**inde2**]**.eigenschaft
ZeigerAufFormular.elements**[**inde2**]**.methode()

index1: ab 0
Nummer des Formulars im Dokument
muss in **[ ]** kodiert sein

index2: ab 0
Nummer des Formular-Elementes im Dokument
muss in **[ ]** kodiert sein

var ZeigerAufFormular = document.forms**[**index1**]**;
var ZeigerAufFormularElement = ZeigerAufFormular.elements**[**index2**]**;
ZeigerAufFormularElement.eigenschaft
ZeigerAufFormularElement.methode()

index1: ab 0
Nummer des Formulars im Dokument
muss in **[ ]** kodiert sein

index2: ab 0
Nummer des Formular-Elementes im Dokument
muss in **[ ]** kodiert sein

Ereignisse (ausgewählte):

onreset — ausgeführt beim Rücksetzen des Formulars
onsubmit — ausgeführt vor dem Senden des Formulars, also verwendbar für Formularprüfung oder Unterbinden des Sendens wegen fehlerhaften Formulardaten

Eigenschaften zum Internet Explorer:

.acceptCharset — Liste von UTF-8 Zeichen für Encoden der Eingabedaten durch den Server
Liste deklariert alle Zeichen, die nicht im Charset des Dokumentes erfasst werden
Liste wird vom Server benötigt
wenn nicht kodiert, so nur der Charset des Dokumentes verwendet
UTF-8 Zeichen: Teil des Unicode (0 bis 255)

.action — Url des Servers und des dortigen Dokumentes bei Formular

Beispiele für mailto-Formen:

Standard	mailto:aaa@bbb
carbon copy	mailto:aaa@bbb**?cc=**mailto:ccc@ddd (mit Kopie an anderen Empfänger)
blind copy	mailto:aaa@bbb**?bcc=**mailto:ccc@ddd (mit Blind-Kopie an anderen Empfänger)
carbon copy und Betreff	mailto:aaa@bbb**?cc=**mailto:ccc@ddd**&subject=**betreff_text
carbon copy und Betreff und Mailtext	mailto:aaa@bbb**?cc=**mailto:ccc@ddd**&subject=**betreff_text**&body=**mail_text

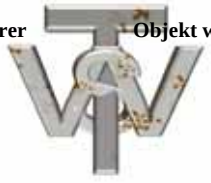
Betreff/Mailtext analog für Standard und blind copy

Beispiel für Spam-Schutz einer Email-Adresse (Schutz vor Missbrauch nach dem Scannen der Email-Adresse durch Suchmaschine):

```
// test@test.de        wird per Script und nicht als HTML im Quellcode kodiert

var user_name="test";
var host_name="test.de";

document.write(        '<A HREF="mailto:'
```

```
                    + user_name
                    + '@'
                    + host_name
                    +  '">'
                    + user_name
                    + '@'
                    + host_name
                    +'</A>'
                );
```

ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.autocomplete	Status des Autovervollständigung zum Formular
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.blockDirection	Umfluss um ein Objekt
.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 ohne Rahmen ohne Scrollbalken
.clientLeft	Abstand in Pixel zum linken Rand des Fensters
.clientTop	Abstand in Pixel zum oberen Rand des Fensters
.clientWidth	Objekt-Breite in Pixel ohne Abstand zum Umgebungsobjekt ohne Rahmen ohne Scrollbalken
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.enctype	Multipurpose Internet Mail Extensions (MIME) des Formulars für Encoding nach dem Senden der Daten ab IE 6.x
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen).
.innerHTML	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.method	Methode des Senden/Empfagen der Daten an/von den Server bei Formular
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes)



nur für Text- und Attribut-Elemente
nicht für Element-Knoten (Knotentyp 1)

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

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

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

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

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

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

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

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

.ownerDocument — Referenz auf das document Objekt zu dem der Koten gehört, also in dem der Knoten erzeugt wurde

.parentElement — Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM

.parentNode — Referenz auf Elternknoten innerhalb der DOM-Hierarchie

.previousSibling — Referenz auf das Vorgängerkind

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

.scopeName — Namensraum laut XMLNS-Attribut

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

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

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

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

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

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

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

.systemBitrate — wird hier nicht erklärt

.systemCaptions — wird hier nicht erklärt

.systemLanguage — Sprache festlegen für das Objekt

.systemOverdubOrSubtitle — wird hier nicht erklärt

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

Anspringen default nicht per TAB-Taste
für APPLET, DIV, FRAMESET, SPAN, TABLE, TD

.tagName — Tag-Bezeichner des Objektes

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

.target — Name des Ziel-Fenster bzw. Ziel-Frame

.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

Methoden zum Internet Explorer:

.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

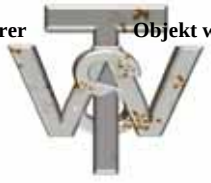
.getAttribute() Wert eines per HTML erzeugten Attributes liefern
DOM nicht geändert

.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

.getElementsByName() Referenz auf ein Feld (Collection) aller im Dokument befindlichen Objekte mit gemeinsamen NAME (analog zum Attribut NAME) liefern
Achtung: Objekte, die kein NAME besitzen, werden nicht erfasst !
Für Verwaltung per ID (analog zum ID-Attribut):
siehe Methode getElementsById()
Für Verwaltung per Tag-Name



siehe Methode .getElementsByTagName()
DOM nicht geändert

.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

.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 !!!

.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

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

.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

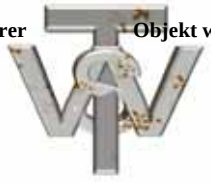
.removeChild() Kind-Objekt aus einem Objekt entfernen aus DOM und Referenz auf das entfernte Kind liefern
Sichtbarkeit erst, wenn Ende-Tag geparst wurde, also das Dokument neu geladen wurde
DOM wird geändert

.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

.reset() löst für Formular das Event onreset aus, dessen Eventhandler die Reset-Aktion beinhaltet, die gestartet wird VOR dem Reset der Elemente (Browser setzt zurück)
Eventhandler muss liefern: return true, für Ausführung des Reset durch Browser
return false; für Nicht-Ausführung des Reset durch Browser

Beispiel:

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

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

.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

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

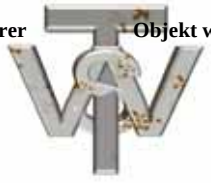
.setCapture() Maus-Überwachung einschalten für ein Objekt
Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick,
onmouseover und onmouseout.
ab IE 5.5
Hinweis: ausschalten per Methode .releaseCapture()

.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

.submit() löst für Formular das Event onsubmit aus, dessen Eventhandler die Submit-Aktion beinhaltet, die gestartet wird VOR dem Senden der Elemente (Browser sendet)
Eventhandler muss liefern: return true, für Ausführung des Submit durch Browser
return false; für Nicht-Ausführung des Submit durch Browser

Beispiel:

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

        // wirklich senden ???
```

```
                        return( confirm("Wirklich senden ?"));
                                // true  so Submit durch Browser ausführen lassen
                                // false so kein Submit durch Browser
                }
        </SCRIPT>
        </HEAD>
        <BODY>
                <DIV>
                        <FORM NAME="Formular" ACTION=" ...."  METHOD="   "
                                onsubmit="EventHandler_AktionVorSubmit();">
                                <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
                                <BR>
                                <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
                                <BR><BR>
                                <INPUT TYPE="submit" VALUE="oder hiermit senden"
                                        onclick="form.submit()"
                                >
                        </FORM>
                </DIV>
                <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
        </BODY>
        </HTML>
```

.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparst DOM wird geändert
.urns()	Referenz auf Feld aller Elemente mit gemeinsamer URN liefern

Objekt document.form.input und seine Varianten beim Internet Explorer

Formularinhalte können diverse HTML-Elemente sein., vorallem Elemente, die Daten speichern können wie z.B. TEXTAREA, SELECT, OPTION etc.. Ein sehr beliebtes HTML-Element ist das input Objekt, das in diversen Varianten kodiert werden kann. **Jedes Element** innerhalb von <FORM> .. </FORM> wird zum Kind des Objektes form. Daher ist Punktnotation nötig, um den Formularinhalt, also die Kinders des Objektes form, referenzieren zu können. Der Objekttyp input.image kann **nicht für eine Formular** referenziert werden, wenn er innerhalb <FORM> ... </FORM> kodiert wurde. Für input.image ist immer die children Collection zu verwenden.

Nachfolgend werden die Varianten des input Objektes kurz beschrieben, wobei nur eine Auswahl von Eigenschaften und Methoden genannt wird. Aus Übersichtsgründen sind weitere Eigenschaften und Methoden direkt bei der Beschreibung vom input Objekt **und nicht hier** zu finden. Ereignisse werden alle beim Objekt event beschrieben. Die Variante **input.image** wird nicht beschrieben.

Die Varianten des input Objekt basieren z.T. auch auf anderen Objekten.

Im Objekt input muss das Attribut TYPE mit dem Objekttyp belegt werden, von dem die Variante des Objektes input abstammt. Der Wert vom Attrivbut TYPE kann wahlweise mit " " bzw. ' ' oder ohne " " bzw. ' ' kodiert werden, wobei Gross-Kleinschreibung egal ist. Der Objekttyp image kann **nicht für eine Formular** referenziert werden, wenn er innerhalb <FORM> ... </FORM> kodiert wurde.

Die Varianten von input haben nur Sinn, wenn **zugleich** Eventhandler kodiert werden, da ein Formular die User-Interaktion behandeln sollte.

Objekt document.form.input.button des Internet Explorer

Diese Objekt basiert zusätzlich auf dem button Objekt (siehe dort).

Erzeugung durch HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=button
         VALUE="button_beschriftung"
         NAME="logischer_formular_element_name"
         onblur="eventhandler1"
         onfocus="eventhandler2"
         onclick="eventhandler3"
         onmousedown="eventhandler4"
         onmouseup="eventhandler5"
>
```

Beispiel:

```
<HEAD>
<SCRIPT>
        function ValueAendern()
        { ID_Input1.value = "Neuer Wert von VALUE";}

        function FarbeAendern()
        { ID_Input2.style.backgroundColor = "aqua";}

        function Anzeige()
        {alert(event.propertyName + " wurde verändert");}
```

```
</SCRIPT>
</HEAD>
<BODY>
        <INPUT  TYPE=button ID="ID_Input1"
                VALUE="Click um VALUE zu ändern"
                onclick="ValueAendern()"
                onpropertychange="Anzeige()"
        >
        <INPUT  TYPE=button ID="ID_Input2"
                VALUE="Click um Farbe zu ändern"
                onclick=" FarbeAendern()"
                onpropertychange="Anzeige()"
        >
</BODY>
```

Eigenschaften (ausgewählte) :

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	liefert immer den Typ button
.value	entspricht VALUE

Methoden (ausgewählte):

.blur()	entfernt den Cursor vom Button
.click()	bewirkt einen Klick auf die Schaltfläche belegt CHECKED und .checked und .defaultChecked
.focus()	setzt den den Cursor auf das Button

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User der Cursor vom Button entfernt
onclick	ausgelöst, wenn User auf das Button klickt
onfocus	ausgelöst, wenn User den Cursor auf das Button bewegt
onmousedown	ausgelöst, wenn User auf dem Button die Maustaste niederdrückt
onmouseup	ausgelöst, wenn User auf dem Button die gedrückte Maustaste loslässt

Objekt document.form.input.checkbox (Abhak-Kästchen) des Internet Explorer

Erzeugung durch HTML:

Beispiel:

```
<INPUT  ID="ID_Input"
        TYPE=checkbox
        VALUE="wert_der_gesendet_wird"          // Standardwert ist "on"
        NAME="logischer_formular_element_name"
        CHECKED                                 // Checkbox ist mit Erstellung bereits markiert
        onblur="eventhandler1"
        onfocus="eventhandler2"
        onclick="eventhandler3"
>
```

Beispiele:

Beispiel 1:

```
<HEAD>
<SCRIPT>
        function Anzeige()
        {alert("Checkbox-Status = " + ID_Checkbox.checked);}
</SCRIPT>
</HEAD>
<BODY>
        selektiere !
        <INPUT  TYPE=checkbox
                ID="ID_Checkbox"
                NAME="checkbox1"
                 onclick=Anzeige()
        >
</BODY>
```

Beispiel 2:

```
<INPUT  TYPE=checkbox
        ID="ID_InputCheckbox"
        CHECKED
        DISABLED
>
<SPAN   STYLE="font-weight:bold"
        onclick="ID_InputCheckbox.status=false"
>
        ablehnen
```

```
                </SPAN>.
                <SPAN   STYLE="font-weight:bold"
                        onclick="ID_InputCheckbox.status=true"
                 >
                        annehmen
                </SPAN>
```

Beispiel 3:

```
<HTML>
<HEAD>
<SCRIPT>
        function ClickMitFocus()
        {
                ID_CheckBox.focus();
                ClickOhneFocus();
        }

        function ClickOhneFocus ()
        { ID_CheckBox.click();}
</SCRIPT>
<SCRIPT FOR= ID_CheckBox EVENT=onfocus>
        alert("Check Box hat Focus");
</SCRIPT>
</HEAD>
<BODY>
        <INPUT Type="CHECKBOX" ID="ID_CheckBox"></INPUT>
        <BR>
        <BUTTON onclick="ClickMitFocus()">Klick mit Fokus</BUTTON>
        <BR>
        <BUTTON onclick="ClickOhneFocus()">Klick ohne Fokus</BUTTON>
</BODY>
</HTML>
```

Eigenschaften (ausgewählte):

.checked	nur lesen ist true, wenn Checkbox abgehakt
.defaultChecked	ist true, wenn CHECKED in <INPUT> kodiert kann belegt werden mit true oder false false bewirkt CHECKED in <INPUT> wird unwirksam
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	liefert immer Typ checkbox
.value	entspricht VALUE

Methoden (ausgewählte):

.blur()	entfernt Cursor von der Checkbox
.click()	bewirkt markieren oder nicht markieren belegt CHECKED und .checked und .defaultChecked
.focus()	setzt Cursor auf die Checkbox

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User der Cursor vom Checkbox entfernt
onclick	ausgelöst, wenn User auf die Checkbox klickt
onfocus	ausgelöst, wenn User den Cursor auf die Checkbox bewegt

Hinweise: onclick für weitere Reaktionen aufgrund markieren bzw. nicht markieren

Objekt document.form.input.fileupload des Internet Explorer

Erzeugung unter HTML:

Beispiel:

```
<INPUT  ID="ID_Input"
        TYPE=file
        NAME="logischer_formular_element_name"
        onblur="eventhandler1"
        onchange="eventhandler2"
        onfocus="eventhandler3"
>
```

Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	enthält den Dateinamen, nur lesen

Methoden (ausgewählte):

.blur()	entfernt Cursor vom Eingabefeld
.focus()	setzt den Cursor auf das Eingabefeld
.select()	markiert den Inhalt des Eingabefeldes, auf dem per .focus() der Cursor gesetzt sein muss

Events (ausgewählte):

onblur	ausgelöst, wenn User der Cursor aus der Eingabezeile bewegt
onchange	ausgelöst, wenn User auf den Inhalt der Eingabezeile ändert
onfocus	ausgelöst, wenn User den Cursor auf die Eingabezeile bewegt

Objekt document.form.input.hidden des Internet Explorer

Erzeugung unter HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=hidden
         NAME="logischer_formular_element_name"
         VALUE="wert"
>
```

Beispiel:

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

Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE

Objekt document.form.input.password des Internet Explorer

Erzeugung unter HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=password
         NAME="logischer_formular_element_name"
         VALUE="vorbelegung"
         SIZE="breite_eingabe_feld_in_zeichen"
         onblur="eventhandler1"
         onchange="eventhandler2"
         onfocus="eventhandler3"
         onselect="eventhandler4"
>
```

Beispiel:

```
<INPUT TYPE="password" AUTOCOMPLETE="off">
```

Eigenschaften (ausgewählte):

.defaultValue	ist standardgemäß eine Leerkette kann verändert werden
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE enthält die Benutzereingabe

Methoden (ausgewählte):

.blur()	entfernt Cursor vom Eingabefeld
.focus()	setzt Cursor auf Eingabefeld
.select()	markiert den Inhalt des Eingabefeldes, da zuvor mit focus() den Cursor bekommt

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User der Cursor aus dem Eingabefeld bewegt
onfocus	ausgelöst, wenn User den Cursor auf das Eingabefeld bewegt

Objekt document.form.input.radio des Internet Explorer

Erzeugung unter HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=radio
```

```
        VALUE="radio_wert_der_gesendet_wird_wenn_button_markiert_ist"
        NAME="logischer_radio_name" bzw. "logischer_radio_gruppen_name"
                                    // Radiobutton mit identischem Namen gehören einer Gruppe an
                                    // in der Gruppe kann pro Zeitpunkt maximal nur 1
                                    //                 Button markiert sein
        CHECKED
        onblur="eventhandler1"
        onfocus="eventhandler2"
        onclick="eventhandler3"
    >
```

Beispiele:

Beispiel 1:

```
<HEAD>
<SCRIPT>
    function KanalVerteilung(Wert)
    { ID_bgsound.balance = Wert;}

    function GenauEinmal()
    {
        ID_bgsound.loop = 1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }

    function Endlos()
    {
        ID_bgsound.loop = -1;
        ID_bgsound.src = ID_bgsound.src; // restart
    }
</SCRIPT>
<BGSOUND ID="ID_bgsound" SRC="sound.wav">
</HEAD>
<BODY>
    <BUTTON onclick="GenauEinmal()"></BUTTON>
    <BUTTON onclick="Endlos()"></BUTTON>
    <BUTTON onclick="KanalVerteilung(-10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung(10)"></BUTTON>
    <BUTTON onclick="KanalVerteilung (0)"></BUTTON>
    <B>Volume control:</B> 
    <INPUT  TYPE ="radio" NAME="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-10;"
    >Mute

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-7;"
    >25% Volume

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe" CHECKED
        onpropertychange="ID_bgsound.volume=-5;"
    >50% Volume

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=-2;"
    >75% Volume

    < INPUT  TYPE ="radio" NAME ="ID_RadioButtonGruppe"
        onpropertychange="ID_bgsound.volume=0;"
    >100% Volume
</BODY>
```

Beispiel 2:

```
<HTML>
<HEAD>
<SCRIPT>
    function Erzeuge()
    {
        var Zeiger = document.createElement(
                "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Eins'>"
                                    );
        document.body.insertBefore(Zeiger);

        Zeiger = document.createElement(
                "<INPUT TYPE='RADIO' NAME='RADIOTEST' VALUE='Zwei'>"
                                    );
```


```
                        document.body.insertBefore(Zeiger);
                }
        </SCRIPT>
        </HEAD>
        <BODY>
                <INPUT  TYPE="BUTTON"
                        ONCLICK=" Erzeuge()"
                        VALUE="Zwei Radio Buttons erzeugen">
                <BR>
                <INPUT  TYPE="BUTTON"
                        ONCLICK="alert(document.body.outerHTML)"
                        VALUE="Click um HTML zu sehen">
                <BODY>
        </HTML>
```

Eigenschaften (ausgewählte):

.checked	ist true wenn Radiobutton markiert ist (sonst false)
.defaultChecked	ist true wenn CHECKED in INPUT kodiert ist kann neu gesetzt werden --> Wirkung von CHECKED aus INPUT aufgehoben
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.index	Index des ausgeählten Elementes
.length	Anzahl Radiobutton innerhalb der Gruppe
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE

Methoden (ausgewählte):

.blur()	entfernt Cursor vom Radiobutton
.click()	bewirkt markieren bzw. entmarkieren des Buttons (Click auf das Radiobutton) sowie setzen der Eigenschaft checked auf true für das angeklickte Button Eigenschaft checked auf false für alle anderen Button der Gruppe
.focus()	setzt den Cursor auf das Radiobutton

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User den Cursor vom Radiobutton entfernt
onclick:	ausgelöst, wenn User auf das Radiobutton klickt Eventhandler für weitere Reaktion nach Anklicken: muss return true; oder return false; besitzen: true, so erfolgt das Senden des radio_wertes auch wirklich false, so erfolgt kein Senden trotz angeklicktem Button
onfocus	ausgelöst, wenn User den Cursor auf das Radiobutton bewegt

Objekt document.form.input.reset des Internet Explorer

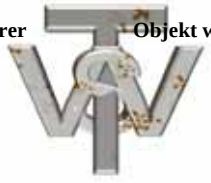
Erzeugung unter HTML:

Beispiel:

```
<INPUT  ID="ID_Input"
        TYPE=reset
        VALUE="reset_button_beschriftung"
        NAME="logischer_formular_element_name"
        onblur="eventhandler1"
        onfocus="eventhandler2"
        onclick="eventhandler3"
        onreset="eventhandler4"
>
```

Beispiel:

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

                // wirklich rücksetzen ???
                return( confirm("Wirklich rücksetzen ?"));
                // true  so Reset durch Browser ausführen lassen
                // false so kein Reset durch Browser
        }
</SCRIPT>
</HEAD>
<BODY>
        <DIV>
                <FORM NAME="Formular" onreset="EventHandler_AktionVorReset();">
```

```
                                    <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
                                    <BR>
                                    <BUTTON onclick="form.reset();">OnReset ausloesen</BUTTON>
                                    <BR><BR>
                                    <INPUT TYPE="reset" VALUE="oder hiermit zuruecksetzen" onclick="form.reset()">
                            </FORM>
                    </DIV>
                    <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
            </BODY>
            </HTML>
```

Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE wenn VALUE nicht kodiert, so "Reset" als Standard nur lesbar

Methoden (ausgewählte):

.blur()	entfernt den Cursor vom Resetbutton
.click()	bewirkt Anklicken des Resetbuttons und löschen des Formulars
.focus()	setzt den Cursor auf das Resetbutton

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn der User denCursor vom Resetbutton entfernt
onclick	ausgelöst, wenn der User auf das Resetbutton clickt
onfocus	ausgelöst, wenn User den Cursor auf das Resetbutton bewegt
onreset	ausgelöst direkt vor dem Formular-Reset durch den Browser Eventhandler für weitere Reaktionen aufgrund des Anklicken auslösen: muss return true; oder return false; liefern true, so Reset auch wirklich ausgeführt

Objekt document.form.input.submit des Internet Explorer

Erzeugung unter HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=submit
         NAME="logischer_formular_element_name"
         VALUE="beschriftung_des_button"
         onblur="eventhandler1"
         onfocus="eventhandler2"
         onclick="eventhandler3"
         onsubmit="eventhandler4"
>
```

Beispiel:

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

                  // wirklich senden???
                  return( confirm("Wirklich senden ?"));
                                    // true  so Submit durch Browser ausführen lassen
                                    // false so kein Submit durch Browser
         }
</SCRIPT>
</HEAD>
<BODY>
         <DIV>
                  <FORM NAME="Formular" ACTION=" ...."  METHOD="   "
                                    onsubmit="EventHandler_AktionVorSubmit();">
                           <INPUT TYPE="text" NAME="InputText" VALUE="Text eingeben">
                           <BR>
                           <BUTTON onclick="form.submit();">OnSubmit ausloesen</BUTTON>
                           <BR><BR>
                           <INPUT TYPE="submit" VALUE="oder hiermit senden" onclick="form.submit()">
                  </FORM>
         </DIV>
         <TEXTAREA ID="ID_Textarea" VALUE="" ROWS="5" COLS="75"></TEXTAREA>
</BODY>
```

```
</HTML>
```

Eigenschaften (ausgewählte):

.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht VALUE wenn VALUE in <INPUT> nicht kodiert, so "Submit Query" als Wert nur lesen

Methoden (ausgewählte):

.blur()	entfernt den Cursor vom Submitbutton
.click()	bewirkt standardgemäß **sofortiges** Abschicken des Formulares (aufgrund Anklicken des Buttons), wenn es sich NICHT um E-Mail handelt --> bei E-Mail erfolgt erst Nachfrage
.focus()	setzt den Cursor auf das Submitbutton

Ereignisse (ausgewählte):

onblur	ausgelöst, wenn User den Cursor vom Submittbutton entfernt
onclick	ausgelöst, wenn User das Submitbutton clickt
onfocus	ausgelöst, wenn User den Cursor auf das Submitbutton bewegt
onsubmit	ausgelöst direkt vor dem Formular-Senden durch den Browser Eventhandler für weitere Reaktionen aufgrund des Anklicken auslösen: muss return true; oder return false; liefern true, so Senden auch wirklich ausgeführt false, so Senden NICHT ausgeführt trotz Anklicken

Objekt document.form.input.text des Internet Explorer

Dieses Objekt basiert zusätzlich auf dem Objekt text (siehe dort).

Erzeugung unter HTML:

Beispiel:

```
<INPUT   ID="ID_Input"
         TYPE=text
         NAME="logischer_formular_element_name"
         VALUE="zeichenkette_als_vorgabe_wert"
         SIZE="breite_eingabefeld_in_zeichen"
         onblur="eventhandler1"
         onchange="eventhandler2"
         onfocus="eventhandler3"
         onkeydown="eventhandler4"
         onkeypress="eventhandler5"
         onkeyup="eventhandler6"
         onselect="eventhandler7"
>
```

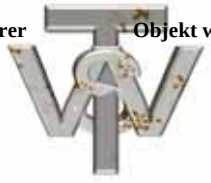
Beispiele:

Beispiel 1:

```
<LABEL FOR="ID_Input" ACCESSKEY="1">
        #<SPAN>1</SPAN>:
        Alt+1 fuer Sprung zur Textbox
</LABEL>
<INPUT   TYPE="text"
         ID="ID_Input "
         NAME="T1"
         VALUE=text1
         SIZE="20"
         TABINDEX="1"
>
```

Beispiel 2:

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

```
		<P>
			TEXTAREA:
			<TEXTAREA ID="ID_Textarea">
				Test
			</TEXTAREA>
			<BUTTON onclick="Antworten(ID_Textarea.isMultiLine);">
				Mehrzeiligkeit eines Textbereiches
			</BUTTON>
		</P>
	</BODY>
	</HTML>
```

Beispiel 3:

```
	<INPUT TYPE=text SIZE=33>
```

Beispiel 4 für Auslesen eines Eingabefeldes:

```
	<HTML>
	<HEAD>
	 <SCRIPT  LANGUAGE="JavaScript">
	 <!
		function gruss()
		{alert("Hallo " + document.meinFormular.Eingabe.value + "!");}
	// -->
	 </SCRIPT>
	</HEAD>
	<BODY>
	 Bitte Namen eingeben und danach den Knopf dr&uuml;cken
	 <FORM NAME="meinFormular">
		<INPUT	TYPE="text"
			NAME="Eingabe"
			VALUE=""
		>
		<BR>
		<INPUT	TYPE="button"
			NAME="Knopf"
			VALUE="Bitte dr&uuml;cken"
			onClick="gruss()"
		>
	 </FORM>
	</BODY>
	</HTML>
```

Beispiel 5 für Wert einem Eingabefeld zuweisen:

```
	<HTML>
	<HEAD>
	 <SCRIPT  LANGUAGE="JavaScript">
	 <!--
		function umrechnung(celsius)
		{
			var fahrenheit;
			fahrenheit = 9 / 5 * celsius + 32;
			document.Formular.Ausgabe.value = fahrenheit;
		}
	// -->
	 </SCRIPT>
	</HEAD>

	<BODY>
	 Bitte geben Sie einen Temperaturwert in Celsius ein:
	 <FORM NAME="Formular">
		<INPUT	TYPE="text"
			NAME="Eingabe"
			VALUE=""
		>
		<BR>
		<INPUT	TYPE="button"
			NAME="Knopf"
			VALUE="Berechnung"
			onClick="umrechnung(this.form.Eingabe.value)"
		>
		<BR>
```

```
                    Entpricht
                    <INPUT   TYPE="text"
                             NAME="Ausgabe"
                             VALUE=""
                    >
                    Fahrenheit
          </FORM>
          </BODY>
          </HTML>
```

Beispiel 6 für Laufschrift in einem Formular

```
          <HTML>
          <HEAD>
           <SCRIPT LANGUAGE="JavaScript">
           <!--
                    text="Laufschrift";

                    function anzeigen()
                    {
                             teilkette=text.substring(0, 1);
                             text=text.substring(1, text.length);
                             text= text + teilkette;

                             window.document.formular.schrift.value=text;
                             setTimeout('anzeigen()',200);
                    }
          //-->
           </SCRIPT>
          </HEAD>

          <BODY>
           <FORM NAME="formular">
                    <INPUT   TYPE=button
                             VALUE="Laufschrift im Formularfeld"
                             onClick="anzeigen()"
                    >
                    <INPUT   TYPE="text"
                             NAME="schrift"
                             SIZE="20"
                             READONLY
                    >
           </FORM>
          </BODY>
          </HTML>
```

Eigenschaften (ausgewählte):

.defaultValue	entspricht VALUE
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.name	entspricht NAME
.type	entspricht TYPE
.value	entspricht **nicht** VALUE enthält den eigegebenen Text

Methoden (ausgewählte):

.blur()	entfernt den Cursor aus dem Eingabefeld (deaktiviert Textfeld für Eingabe)
.focus()	setzt den Cursor auf das Eingabefeld
.select()	markiert den Text im Eingabefeld Cursor muss zuvor mit focus() auf das Eingabefeld bewegt worden sein

Eereignisse (ausgewählte):

onblur	ausgelöst, wenn User den Cursor vom Eingabefeld entfernt
onchange	ausgelöst, wenn User das Eingabefeld inhaltlich ändert
onfocus	ausgelöst, wenn User den Cursor auf das Eingabefeld bewegt
onselect	ausgelöst, wenn Teil des Inhaltes vom Eingabefeld markiert wird