textarea Objekt des Internet Explorer

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

Beispiel 1:

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

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

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

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

Beispiel 2:

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

Beispiel 3:

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

Beispiel 4:

```
<SPAN UNSELECTABLE="on" >
```

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

Beispiel 5:

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

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

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

Beispiel 6:

```
	<BODY	onload="ID_Div.setCapture();"
		onclick="document.releaseCapture();"
	>
		<DIV ID="ID_Div"
			onmousemove="ID_Textarea.value = event.clientX + event.clientY;"
			onlosecapture="alert(event.srcElement.id + ' hat keine Mausueberwachung mehr');"
		>
			<TEXTAREA ID="ID_Textarea" COLS=2>Test</TEXTAREA>
		</DIV>
	</BODY>

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

			// wirklich rücksetzen ???
			return( confirm("Wirklich rücksetzen ?"));
				// true  so Reset durch Browser ausführen lassen
				// false so kein Reset durch Browser
		}
	</SCRIPT>
```


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

Eigenschaften:

.accessKey	Tastaturzugriff auf ein Objekt per Alt + Taste bei Ausführung der Tastenkombination wird das Sprungziel wird focussiert die Sprungquelle defocussiert das Focus-Ereignis ausgelöst vor IE 5.x funktioniert Tastenkombination nicht mit jedem Objekt
ATOMICSELECTION	Selektierbarkeit des Objektes einstellen
.begin	Wartezeit einstellen und dann Objekt aktivieren laut Eigenschaft .timeAction siehe Objekt currTimeState und Behavior .style.time2
.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.className	Klassenreferenz, Klassenname
.clientHeight	Objekthöhe in Pixel ohne Abstand zum Umgebungsobjekt 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
.cols	Anzahl der sichtbaren Zeichen in einer Zeile der TEXTAREA Hinweis: Es können mehr Zeichen in der TEXTAREA enthalten sein als sichtbar siehe .rows für Mehrzeiligkeit .wrap für Wortumbruch Scrollleisten sind erzeugbar
.contentEditable	Objekt-Content-Editierbarkeit (auch wenn Objekt kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.dataFld	Datenquelle-Name vergeben (ID)
.dataSrc	Datenquelle als Anker festlegen
.defaultValue	Vorbelegung der TEXTAREA sichtbare Auswirkungen nur bei Instanzierung des Objektes Formular-Reset
.dir	Umflussrichtung
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.end	Objektaktivitäten laut Eigenschaft .timeAction beenden ab IE 6.x alternativ: Eigenschaft .dur siehe Objekt currTimeState und Behavior .style.time2
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.form	Zeiger auf das Formular (Formular als Container) ab IE 6.x für Elemente fieldSet, label, legend
.hasMedia	Objekt ist HTML-Media-Objekt siehe Objekt currTimeState und Behavior .style.time2
.hideFocus	Focussierbarkeit
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.innerText	Referenz auf den Bereich zwischen HTML-Start und -Ende-Tag ein Tag ohne Ende-Tag kann kein innerHTML haben, z.B. <B> dient zur Veränderung dieses Bereiches zur Laufzeit des Dokumentes, also nach dem Laden des Objektes aber wirksam erst mit parsen des HTML-Endetag

.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lang	Sprache für Anzeige von Sonderzeichen etc.
.language	Sprache für Script festlegen
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.offsetHeight	Y-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetLeft	X-Koordinate der linken oberen Ecke Objektes bezüglich Koordinatensystem des Elternobjektes (.offsetParent)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeigt, .offserLeft, .offsetTop und .offsetWidth
.offsetTop	Y-Koordinate der linken oberen Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)
.offsetWidth	X-Koordinate der rechten unteren Ecke des Objektes bezüglich Koordinatensystem des Elternobjektes (.offestParent)
.onOffBehavior	deprecated ab IE 5.x Unterstützung von DirectAnimation z.B. für 2D, 3D, Sound
.outerHTML	Referenz auf Bereich ab inklusive HTML-Start bis hinter -Ende-Tag wirksam mit parsen des Ende-Tag nur nach kompletten Einlesen des Dokumentes nutzbar
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Koten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readOnly	Editierbarkeit eines Text-Controls Editierung bedeutet Focuserhalt ! false Default Objekt ist **nicht** read-only also ist es editierbar kann also Focus erhalten true Objekt ist read-only also ist es nicht editierbar kann kein also Focus erhalten
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.recordNumber	Datenquelle-Satznummer eines Datenfeldes
.rows	Anzahl der sichtbaren Zeilen der TEXTAREA
.scopeName	Namensraum laut XMLNS-Attribut
.scrollHeight	Höhe des vertikalen Scrollbereiches, also Abstand von oberen und unteren sichtbaren Rand des Umgebungsobjektes
.scrollLeft	Abstand von linken sichtbaren Randes des Umgebungsobjektes zum linken Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollTop	Abstand von oberen sichtbaren Randes des Umgebungsobjektes zum oberen Rand des Objektes nur nutzbar nach dem kompletten Laden des Dokumentes
.scrollWidth	Breite des horizontalen Scrollbereiches, also Abstand von linken und rechten sichtbaren Rand des Umgebungsobjektes
.sourceIndex	Index des Objektes in der Collection document.all
.status	Selektionsstatus eines Control-Elementes Control-Element: kann durch User interaktiv verändert werden wird bei Ceckbox-Control auch verändert durch Eigenschaft .indeterminate bei Statusveränderung wird Event onpropertychange erzeugt false Default außer bei textArea Objekt Control ist nicht selektiert true Control ist selektiert null Control ist nicht initialisiert Default bei textArea Objekt
STYLE	direkt im HTML-Element kodierter Style (Inline-Style)

Hinweis: für Scripting ist das Style-Objekt zu nutzen

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

.systemBitrate — wird hier nicht erklärt

.systemCaptions — wird hier nicht erklärt

.systemLanguage — Sprache festlegen für das Objekt

.systemOverdubOrSubtitle — wird hier nicht erklärt

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

.tagName — Tag-Bezeichner des Objektes

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

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

.title — Tooltip-Text bei Mouse over über Objekt

.type — Type des TEXTAREA-Control

.uniqueID — durch den Browser automatisch-genriertes ID des Objektes
Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde
kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

UNSELECTABLE — Selektionsfähigkeit eines Objektes

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

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

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

"off" Wortumbruch nicht aktiv

Methoden:

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

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

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

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

.blur() — Element den Focus wegnehmen und Event onblur auslösen
Der Focus wird nicht automatisch auf irgend ein anderes Element gesetzt !

vor IE 5.0 TABINDEX-Attribut muss kodiert sein
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein

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

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

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

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

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

.createTextRange() Textbereich erzeugen

Beispiel 1:

```
<SCRIPT LANGUAGE="JScript">
    var FeldAllerButtonElemente coll = document.all.tags("BUTTON");

    if (    (FeldAllerButtonElemente !=null )
         && (FeldAllerButtonElemente.length>0)
       )
    {
        var ZeigerAufRange = FeldAllerButtonElemente[0].createTextRange();
        ZeigerAufRange.text = "Clicked";
    }
</SCRIPT>
```

Beispiel 2:

```
function ErzeugeUndSelektiereTextRange()
{
    var TextBereich = document.body.createTextRange();
    TextBereich.findText("Testtext");
    TextBereich.select();
}
```

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

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

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

Beispiel:

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

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

    function ScrolleSeiteDown()
    {ID_Textaerea.doScroll("scrollbarPageDown");}
</SCRIPT>
</HEAD>
```


```
<BODY>
    <BUTTON onclick=" ScrolleSeiteRechts()">
        scrolle Seite rechts
    </BUTTON>
    <BR>
    <BUTTON
        onclick=" ScrolleDown()"
        ondblclick=" ScrolleSeiteDown()"
    >
        Texarea: Click = scrolle down Doppelklick = scrolle Seite down
    </BUTTON>
    <BR>
    <BR>
    <TEXTAREA ID="ID_Textaerea" >
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
        ABCDEFGHIJKLMNOPQRSTUVWXYZ
        1234567890123456789012345678901234567890
    </TEXTAREA>
</BODY>
```

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

.fireEvent() ein Event auslösen

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

.removeNode() Knoten entfernen aus DOM und Referenz auf den entfernten Knoten liefern
Sichtbarkeit erst wenn Ende-Tag geparst wurde
DOM wird geändert

.replaceAdjacentText() Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern
DOM wird nicht geändert

.replaceChild() Kind-Objekt ersetzen durch ein Objekt
ersetzende Objekt muss per Methode .createElement() erzeugt worden sein
Sichtbarkeit erst wenn Ende-Tag geparst wurde
DOM wird geändert

.replaceNode() Objekt durch anderes Objekt komplett ersetzen und Referenz auf das komplett ersetzte Objekt liefern
sichtbar erst mit parsen des Endetags
DOM wird geändert

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

.select() Objekt selektieren
z.B. Textbereich: wird hervorgehoben
ControlRange: es wird Rechteck-Rahmen erzeugt
nur unter Windows 32-Bit

Beispiel:

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

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

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

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

.setCapture()	Maus-Überwachung einschalten für ein Objekt Maus-Events sind : onmousedown, onmouseup, onmousemove, onclick, ondblclick, onmouseover und onmouseout. ab IE 5.5 Hinweis: ausschalten per Methode .releaseCapture()
.setExpression()	Wert definieren, der als Ausdruck für die Methode .getExpression() zur Berechnung einer Style-Eigenschaft als Objektreferenz der Form objekt.style.eigenschaft. dient Ausdruck nur als Script kodierbar DOM wird nicht geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparst DOM wird geändert