window.event Objekt des Internet Explorer

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

Beispiel: Wird per

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

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

Beispiel: Wird per

```
function OnContextMenuHandler()
// Unterbindung des Kontextmenüs
// Achtung: return false; unterdrückt nicht das Kontextmenü
{event.returnValue=false;}

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

ein Eventhandler zum Ereignis oncontextmenu implementiert, so bleibt der rechte Maustastendruck im gesamten Dokument und für alle Kinder wirkungslos, es sei denn, das Kind implementiert einen eigenen Handler für das Ereignis des Drückens der rechten Maustaste (z.B. per Event onmousedown).

Zugriff:

event.eigenschaft

event.methode()

window.event.eigenschaft
window.event.methode()

window kann entfallen wenn aktuelles Fenster gemeint ist

zeiger_auf_fenster.event.eigenschaft
zeiger_auf_fenster.event.methode()

1. Eigenschaften

.altKey	ALT-Tasten-Status
.altLeft	linke ALT-Taste-Status nicht für Win9x nur für Dokument, das den Fokus hat
.button	Maustaste die das Event auslöst NUR per onmousedown, onmouseup, und onmousemove events
.cancelBubble	Event durchreichen über Eventhandlerkette
.clientX	X-Koordinate Maus relativ zum Fenster aber ohne Rahmen, Scrollbar etc.
.clientY	Y-Koordinate Maus relativ zum Fenster aber ohne Rahmen, Scrollbar etc.
.ctrlKey	CTRL-Tasten-Status
.ctrlLeft	linken CTRL-Taste Status nicht für Win9x nur für Dokument, das den Fokus hat
.fromElement	Referenz auf Maus-Eventauslösendes Quell-Element bei Mausbewegung nicht für Event ondragleave
.keyCode	Unicode der Taste, die das Event auslöst per onkeydown, onkeyup und onkeypress
.offsetX	X-Koordinate Maus relativ zum Objekt, das das Event auslöst
.offsetY	Y-Koordinate Maus relativ zum Objekt, das das Event auslöst
.propertyName	Name der Eigenschaft, die wertmässig verändert wurde durch onpropertychange Event auch Style-Eigenschaft etc.
.returnValue	Returnwert für den Eventhandler festlegen anstelle von return-Anweisung Achtung: Wenn kodiert, so wird eine ebenfalls im Eventhandler kodierte die Anweisung return ....; ignoriert
.screenX	X-Koordinate Maus relativ zum Bildschirm
.screenY	Y-Koordinate Maus relativ zum Bildschirm
.shiftKey	SHIFT-Tasten-Status
.shiftLeft	SHIFT-Taste links Status nur für Dokument mit Focus
.srcElement	Referenz auf Objekt das das Event auslöst
.toElement	Referenz auf Maus-Eventauslösendes Ziel-Element bei Mausbewegung nicht für Event ondragleave
.type	Bezeichner des Events, also Eventart
.URL	Url laut einem Kommando aus einer Advanced Streaming Format (ASF)-Datei von Element auf der Timeline es muss Ereignis onURLFlip aufgetreten sein siehe Objekt currTimeState und Behavior .style.time2
.wheelDelta	liefert Umdrehung und Richtung in der das Mausrad gedreht wurde Verwendung bei Event onmousewheel ab IE 6.x
.x	X-Koordinate Maus relativ zum Eltern-Objekt ab IE 5.x zum Fenster unter IE 5.x X-Koordinate ist relativ zum BODY-Element wenn Objekt, das das Event auslöst, absolut positioniert ist (z.B. per Style) oder Eltern nicht absolut positioniert sind
.y	Y-Koordinate Maus relativ zum Eltern-Objekt ab IE 5.x zum Fenster unter IE 5.x Y-Koordinate ist relativ zum BODY-Element wenn Objekt, das das Event auslöst, absolut positioniert ist (z.B. per Style) oder Eltern nicht absolut positioniert sind

2. Methoden

keine

Methoden anderer Objekte zur Verwaltung von Events:

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

3. event.bookmarks Collection

Feld der Zeiger auf ActiveX Data Objects (ADO)-Bookmarks

Syntax:

[var ZeigerAufFeld =] object.event.bookmarks
[var ZeigerAufFeldElement =] object.event.bookmarks[Index]

object kann entfallen wenn aktuelles Fenster gemeint ist, also window
zeiger_auf_fenster

Index Integer ab 0
oder String Name oder ID des Elementes
muss in [ ] kodiert sein

Eigenschaften:

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

Methoden:

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

4. event.dataTransfer Objekt des Internet Explorer

Objekt für Drag und Drop zwischen Quelle und Ziel über Clipboard (Zwischenablage)
ab IE 5.x
siehe auch Objekt clipboardData des IE

Drag = aus Quelle verschieben: Objekt mit Maus anklicken/selektieren, dann ziehen bei gedrückter Maustaste
oder Objekt mit Maus markieren und per Tastatur CTRL-X

kopieren bzw. ausschneiden:
Objekt mit Maus selektieren, dann per Menü der rechten Maustaste kopieren bzw. ausschneiden
oder Objekt mit Maus markieren und per Tastatur CTRL-C bzw. CTRL-X

Drop = im Ziel ablegen
nach verschieben aus Quelle:
gezogenes Objekt über Objekt ablegen durch loslassen der Maustaste
keine Mehrfachablage

nach kopieren bzw. ausschneiden aus Quelle:
auf dem Zielobjekt per Menü der rechten Maustaste in das Ziel einfügen
Mehrfacheinfügung möglich

Tastatur: über dem Ziel CTRL-V
Mehrfacheinfügung möglich

Für die Verwaltung der Zwischenablage wird das Event-Objekt mit den Ereignissen ondragXXX benutzt.

Standard-Drag und Drop (Standard-Eventhandler) bei folgenden HTML-Objekten
A, IMG, TEXTAREA, INPUT TYPE=text

Für Elemente ohne Standard-Drag und Drop müssen Eventhandlern für Drag und Drop **programmiert** werden.

Das Clipboard funktioniert nur innerhalb ein und derselben Domain,
also nicht per HTTP
HTTPS
FTP etc.
nur innerhalb ein und derselben Browserinstanz
nicht zwischen Browser und externe Anwendungen z.B. MS Word

verwaltet folgende Datenformate: Text, URL, File, HTML, Image

DIV-Objekt und IFRAME-Objekt unterstützen den Datentransfer mit folgenden Attributen:

HTML-Attribut	Eigenschaft
DATAFLD	.dataFld
DATAFORMATAS	.dataFormatAs
DATASRC	.dataSrc
---------	.recordNumber

Syntax:
object.event.dataTransfer.eigenschaft
object.event.dataTransfer.methode()

object kann entfallen wenn aktuelles Fenster gemeint ist, also window
zeiger_auf_fenster

Beispiel 1:

```
<HEAD>
<SCRIPT>
    function DragStart() // Url eines Ankers als Datenformat festlegen und Daten holen
    {event.dataTransfer.setData("URL", ID_A.href);}

    function DragEnde() // Daten im URL-Format als Textdaten in den Span ablegen
    {ID_Span.innerText = event.dataTransfer.getData("URL");}
</SCRIPT>
</HEAD>
<BODY>
    <A ID="ID_A" HREF="anker"
        onclick="return(false);"
        ondragstart=" DragStart()"
    >
```


```
                Testanker
        </A>
        <SPAN ID="ID_Span" ondragenter=" DragEnde()">
                obigen Link auf diese Stelle hierher droppen
        </SPAN>
</BODY>
```

Beispiel 2:

```
<HEAD>
<SCRIPT>
        function InitiateDrag()
        {event.dataTransfer.setData("URL", ID_Image.src);}

        function FinishDrag()
        {ID_Span.innerText = event.dataTransfer.getData("URL");}
</SCRIPT>
</HEAD>
<BODY>
        <IMAGE ID="ID_Image" SRC="/test/graphics/test.gif"
                                        ondragstart="InitiateDrag()">
        <SPAN ID="ID_Span" ondragenter="FinishDrag()"
        >
                Image hierher droppen
        </SPAN>
</BODY>
```

Beispiel 3:

```
<HTML>
<HEAD>
<SCRIPT>
        var Wert;

        function TextBereichErzeugen()
        {
                // Text im gesamten Body adressieren
                var TextBereich = document.body.createTextRange();

                // davon den Textteil des Source-Div adressieren
                TextBereich.findText(ID_DivSource.innerText);

                // und diesen selektieren
                TextBereich.select();
        }

        // Ausschneiden aktivieren, da standardgemäß für DIV dekativ ist
        // Ausschneiden bedeutet: Browser verschiebt automatisch in das Clipboard
        function AusschneidenAktivieren()  // ist Eventhandler für onbeforecut
        {event.returnValue = false;}         // Event-Handler-Rückkehrcode

        function Ausschneiden()            // ist Eventhandler für  oncut
        {
                // Clipboard füllen mit Daten vom Texttyp
                //                                      als Text des Source-Div
                Wert = window.clipboardData.setData("Text", ID_DivSource.innerText);

                // Source-Div Textbereich löschen
                ID_DivSource.innerText = "";

                // Rückgabewert vom Clipboardfüllen als Text (true oder false)
                //              im Text des Input-Button anzeigen
                ID_Input.innerText += Wert; // Wert mit als Text

                // Event-Handler-Rückkehrcode
                event.returnValue = false;
        }
        // Einfügen aktivieren, da standardgemäß für DIV dekativ ist
        // Einfügen bedeutet: Browser fügt aus Clipboard in das Zielobjekt ein
        function EinfuegenAktivieren()   // ist Eventhandler für onbeforepaste
        {event.returnValue = false;}       // Event-Handler-Rückkehrcode

        function Einfuegen()  // ist Eventhandler für onpaste
```


```
		{
			// aus Clipboard Daten vom Texttyp holen und in den Textbereich
			// des Target-Div ablegen
			ID_DivTarget.innerText = window.clipboardData.getData("Text");

			// Event-Handler-Rückkehrcode
			event.returnValue = false;
		}
	</SCRIPT>
	</HEAD>
	<BODY	onload="TextBereichErzeugen()">
		<DIV ID="ID_DivSource" onbeforecut="AusschneidenAktivieren()"
					oncut="Ausschneiden()"
		>
			Diesen Text mit Maus markieren und ausschneiden
		</DIV>
		<DIV ID="ID_DivTarget" onbeforepaste="EinfuegenAktivieren()"
					onpaste="Einfuegen()"
		>
			An diese Stelle den ausgeschnittenen Text einfügen
		</DIV>
		<BR>
		<INPUT ID="ID_Input" TYPE="text" VALUE="Rückkehrcode = ">
	</BODY>
	</HTML>
```

Beispiel 4:

```
	<HTML>
	<HEAD>
	<SCRIPT>
		// ###############  Quellobjekt ############################
		function EventHandlerFuerOnDragStart()
			// Quellobjekt löst Event aus:
			//		in Quelle wird markiert und selektiert
		{
			// selektierte Quell-Daten in die Zwischenablage puffern
			//			(Puffer wird per window.event-Objekt verwaltet)
			//		Datentyp ist hier uninteressant, da für die Zwischenablage
			//			der Typ automatisch erkannt wird, also
			//			Speicherung der Daten in der Zwischenablage
			//			immer typgerecht ist
			var ZwischenAblage = window.event.dataTransfer;

			// und diese Daten als verschiebbar erklären:
			//			aus der Quelle in die Zwischenablage
			ZwischenAblage.effectAllowed = "move";
		}

		// ############## Zielobjekt ############################
		function EventHandlerFuerOnDragEnter
			// Zielobjekt löst Event aus
			// Maus über Ziel nach dem Draggen der Daten,
			//		UND Maustaste ist noch nicht losgelassen
			//   also Zielobjekt wird mit den Daten betreten
		{
			// Daten adressieren
			var ZwischenAblage = window.event.dataTransfer;

			// und diese Daten als verschiebbar erklären:
			//		aus der Zwischenablage in das Zielobjekt
			ZwischenAblage.dropEffect = "move";

			// Event-Handler-Rückkehrcode
			var EventObjekt = window.event;
			EventObjekt.returnValue = false;
		}

		function EventHandlerFuerOnDrop
			// Zielobjekt löst Event aus
			// Maus über Ziel nach dem Droppen der Daten,
			//		UND Maustaste wird losgelassen
		{
```


```
                        // Ziel adressieren, das mit ondrop-Ereignis behandelt werden soll
                        var Ziel = window.event.srcElement;

                        // Daten in der Zwischenablage adressieren und holen
                        //                      (Puffer wird per window.event-Objekt verwaltet)
                        var ZwischenAblage = window.event.dataTransfer;

                        // und Daten vom Texttyp aus der Zwischenablage im Ziel ablegen,
                        //          da Ziel nur Textdaten empfangen kann
                        Ziel.innerText += Daten.getData("text");

                        // Event-Handler-Rückkehrcode
                        var EventObjekt = window.event;
                        EventObjekt.returnValue = false;
                }

                function EventHandlerFuerOnDragOver
                        // Zielobjekt löst Event aus
                {
                        // nichts tun ausser Rückkehrcode des Handlers liefern
                        var EventObjekt = window.event;
                        EventObjekt.returnValue = false;
                }
        </SCRIPT>
        </HEAD>
        <BODY>
                <DIV ondragstart=" EventHandlerFuerOnDragStart ()"
                >
                        Diesen Text markieren und draggen
                </DIV>
                <BR>
                <DIV    ondragover="EventHandlerFuerOnDragOver()"
                        ondragenter="EventHandlerFuerOnDragEnter()"
                        ondrop="EventHandlerFuerOnDrop()"
                >
                        An diese Stelle den Text droppen
                </DIV>
        </BODY>
        </HTML>
```

Beispiel 5 für Clipboard-Nutzung ohne Drag & Drop:

```
        <HTML>
        <HEAD>
        <SCRIPT>
                function Init()
                {
                        var TextBereich  = document.body.createTextRange();
                        TextBereich.findText(ID_Div1.innerText);
                        TextBereich.select();  // selektieren
                }

                function VorDemAusschneiden()
                {event.returnValue = false;} // Cut per Menü aktivieren

                function Ausschneiden()
                {
                        ID_Div1.innerText = "";
                        ID_Input.innerText += window.clipboardData.setData("Text", ID_Div1.innerText);
                                                        // true oder false
                                                        // Clipboard-Daten sind Texttyp
                        event.returnValue = false;
                }

                function VorDemEinfuegen()
                {event.returnValue = false;} // Paste per Menü aktivieren

                function Einfuegen()
                {
                        ID_Div2.innerText = window.clipboardData.getData("Text");
                        event.returnValue = false;
                }
        </SCRIPT>
```


```
</HEAD>
<BODY onload="Init()">
        <DIV ID=" ID_Div1" onbeforecut="VorDemAusschneiden()" oncut="Ausschneiden()">
                Dieses Text selektieren und ausschneiden
        </DIV>
        <DIV ID="ID_Div2" onbeforepaste="VorDemEinfuegen()" onpaste="Einfuegen()">
                den ausgeschnittenen Text hier einfügen
        </DIV><BR>
        <INPUT ID="ID_Input" TYPE="text" READONLY VALUE="" SIZE="6">
</BODY>
</HTML>
```

Eigenschaften:

.dropEffect	Cursor-Layout bei Drop wird von Eigenschaft .effectAllowed beeinflusst
.effectAllowed	Art von Drag und Drop beeinflusst Eigenschaft .dropEffect

Methoden:

.clearData()	Clipboardinhalt löschen
.getData()	Clipboard auslesen
.setData()	Clipboard füllen, also Daten dort ablegen wenn Clipboard nicht leer so immer anhängen

5. Eventarten (Auswahl)

Hinweis zum Test von Mausereignissen: Bitte bei forlaufenden Ereignissen wie onmousemove etc. **kein** alert() verwenden, da das Betreten/Betätigen der Alert-Box selbst das Ereignis erzeugen kann. Anstelle dafür einen DIV verwenden, dessen innerHTML bzw. innerText per gleichnamige Eigenschaften belegt wird zum Mausereingis und so keine zusätzlichen Mausereignisse auftreten können. Alternativ ist auch die Fenster-Status-Zeile belegbar.

onabort — Abbruch des Download vom Image

onactivate — erzeugt wenn Objekt als aktives gesetzt wird (event.fromElement to the event.srcElement)
aktiv setzen bedingt nicht den Erhalt des Focus:
aber Aktivierung per Fokus-Methode möglich
wird immer direkt vor onload erzeugt
ab IE 5.5

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    function EventOnActivate()
    {ID_Div.innerHTML +=    "onactivate erkannt";}

    function EventOnLoad()
    {ID_Div.innerHTML += "onload erkannt";}
</SCRIPT>
</HEAD>
<BODY onactivate="EventOnActivate();" onload="EventOnLoad();">
    <DIV ID="ID_Div"></DIV>
</BODY>
</HTML>
```

onafterprint — erzeugt mit Ende des Druck bzw. Druckvorschau

onafterupdate — erzeugt wenn Daten in einem Datasource-Objekt erfolgreich geupdatet wurden

onbeforeactivate — erzeugt direkt vor der Aktivierung eines Objektes

onbeforecopy — erzeugt direkt vor dem Kopieren einer Selektion im Objekt in das Clipboard

Beispiel

```
<HEAD>
<SCRIPT>
    var Daten= "Das sind Text-Daten";

    function Quelle_Beforecopy()
    {event.returnValue = false; }

    function Quelle_Copy()
    {window.clipboardData.setData("Text", Daten); }

    function Ziel_BeforePaste()
    {event.returnValue = false; }

    function Ziel_Paste()
    {
        ID_Input.value = window.clipboardData.getData("Text");
        event.returnValue = false;
    }
</SCRIPT>
</HEAD
<BODY>
    <SPAN ID="ID_Span" onbeforecopy="Quelle_Beforecopy()"
                       oncopy="Quelle_Copy()"
    >
        diesen Text kopieren
    </SPAN>
    <INPUT ID="ID_Input" onbeforepaste="Ziel_BeforePaste()"
                         onpaste="Ziel_Paste()"
    >
</BODY>
```

onbeforecut — erzeugt dirket vor dem Ausschneiden einer Selektion im Objekt

Beispiel 1

```
<HEAD>
<SCRIPT>
    var Kette = "";

    function EventBeforeCut()
    {event.returnValue = false; }
```


```
                function EventCut()
                {
                        Kette= ID_Span.innerText;
                        ID_Span.innerText = "";
                        event.returnValue = false;
                }

                function EventBeforePaste()
                {event.returnValue = false; }

                function EventPaste()
                {
                        ID_Div.innerText = Kette;
                        event.returnValue = false;
                }
        </SCRIPT>
        </HEAD>
        <BODY>
                <SPAN ID="ID_Span" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                        diesen Text selektieren aund ausschneiden
                </SPAN>
                <BR>
                <DIV ID="ID_Div" onbeforepaste="EventBeforePaste()"onpaste="EventPaste()">
                        hierher den ausgeschnittenen Text einfügen
                </DIV>
        </BODY>
```

Beispiel 2

```
        <HEAD>
        <SCRIPT>
                var TextDaten = "new content associated with this object";
                var Kette = "";

                function Init()
                {
                        var TextBereich = document.body.createTextRange();  // Bereich aller Texte
                        TextBereich.findText(ID_Span1.innerText);           // aber den Text im ID_Span1 suchen
                        TextBereich.select();                               // und dann selektieren
                }

                function EventBeforeCut()
                {
                        Kette = ID_Span1.innerText;
                        event.returnValue = false;
                }

                function EventCut()
                {window.clipboardData.setData("Text", TextDaten); }         // setData liefert return-Wert
                                                                                    //        also
                event.returnValue = ...;
                                                                                    //        noch nötig zu
                kodieren

                function EventBeforePaste()
                {event.returnValue = false; }

                function EventPaste()
                {
                        ID_Span2.value = window.clipboardData.getData("Text", TextDaten);
                        event.returnValue = false;
                }
        </SCRIPT>
        </HEAD>
        <BODY onload="Init()">
                <SPAN ID="ID_Span1" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                        Diesen Text ausschneiden
                </SPAN>
                <INPUT  ID=" ID_Span2" TYPE="text" VALUE="Den ausgeschnittenen Text hier einfuegen"
                        onbeforepaste="EventBeforePaste()"
                        onpaste="EventPaste()"
                >
        </BODY>
```

onbeforedeactivate erzeugt direkt bevor ein Objekt als deaktiv gesetzt wird

onbeforeeditfocus erzeugt direkt vor der User-Ineraktivität auf ein editierbares Objekt

immer erzeugt wenn INPUT-Objekt oder TEXTAREA-Objekt den focus erhält

onbeforepaste erzeugt direkt vor dem Eifügen aus dem Clipboard in ein Objekt

onbeforeprint erzeugt direkt vor dem Druck bzw. Druckvorschau

onbeforeunload erzeugt direkt vor dem Unload eines Dokumentes
unload = verlassen des Dokumentes durch Laden eines neuen Dokumentes in das selbe Fenster

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    function DokumentSchliessen()
    {event.returnValue = "Irgendeinen Stringwert liefern. Vor dem Schliessen wird Dialogbox angezeigt."}
</SCRIPT>
</HEAD>
<BODY onbeforeunload="DokumentSchliessen ()">
    <A HREF="http://www.test.de">zu www.test.de</A>
</BODY>
</HTML>
```

onbeforeupdate erzeugt mit dem Beginn des Update von Daten eines Datasource-Objektes

onbegin erzeugt wenn ein Element auf der Timeline aktiviert und damit die Timeline aktiv wird
nicht erzeugt für eine Wiederholung der Animation des Elementes, also wenn Eigenschaft .repeatCount > 1 ist für das Element
erzeugt für eine Wiederholung der Animation des Eltern-Elementes, also wenn Eigenschaft .repeatCount > 1 ist für das Eltern-Element
siehe onend und onrepeat
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:EXCL  ID="ID_Excl"
        BEGIN="indefinite"
        DUR="5"
        REPEATCOUNT="5"
        onbegin="alert('Start der Animation');"
    >
        <t:ANIMATEMOTION        ID="ID_Animatemotion"
                                TARGETELEMENT="ID_Div"
                                TO="200,0"
                                BEGIN="0"
                                DUR="2"
                                AUTOREVERSE="true"
        >
        </t:ANIMATEMOTION>
    </t:EXCL>
    <DIV    ID="ID_Div"
        CLASS="time_line_klasse"
        STYLE="position:relative;top:15px;left:25px;height:100px;width:100px;
            border:solid black 1px;
            "
    >
        sich bewegender DIV
    </DIV>
    <BUTTON onclick=" ID_Animatemotion.beginElement();">Start</BUTTON>
    <BUTTON onclick=" ID_Animatemotion.endElement();">Stop</BUTTON>
</BODY>
</HTML>
```

onblur erzeugt wenn Objekt den Focus verliert

Beispiel

```
<HTML>
<BODY>
    <INPUT TYPE=text NAME="Test" VALUE="onblur Test" onblur="alert(event.srcElement.name)">
</BODY>
</HTML>
```

onbounce erzeugt wenn Behavior-Eigenschaft des Marquee-Objektes auf "alternate" gesetzt ist **und** der Marquee-Text eine der beiden Seiten des Elternfensters erreicht hat

Beispiel

```
<BODY>
```


```
                    <MARQUEE BEHAVIOR="alternate" WIDTH=200 LOOP=3
                                onbounce="alert('onbounce-Ereignis erkannt')"
                    >
                            Marquee Text
                    </MARQUEE>
            </BODY>
```

oncellchange erzeugt wenn Daten verändert wurden in der Datenquelle z.B. Objekt
onclick erzeugt mit Druck der linken Maustaste auf ein Objekt
benötigt vorher die Eventfolge erzeugt onmousedown und onmouseup
wobei die Maus auch dabei über dem Objekt sein muss
bei Radio-Buttons-Gruppe: onclick event immer nach
onbeforeupdate und onafterupdate events for the control group.
bei focusfähigem Objekt: onfocus event erzeugt vor dem onclick event
bei Doppelkick per linker Maustaste:
zuerst onclick ausgelöst, dann ondblclick
oncontextmenu erzeugt wenn rechte Maustaste gedrückt wurde
Unterdrückung, wenn der Eventhandler event.returnValue=false; enthält
(mit return false; erfolgt keine Unterdrückung)

Beispiel

```
<SPAN STYLE="width:300; background-color:blue; color:white;" oncontextmenu="return false">
        <P>Das Kontextmenue ist innerhalb dieses SPAN abgeschaltet</P>
</SPAN>
```

oncontrolselect erzeugt wenn User eine Control-Selektion im Objekt tätigt (z.B. CTRL+ Maus)
oncopy erzeugt wenn Quellelement oder Selektion dem Clipboard hinzugefügt wird
per rechte Maus-Click und Anwahl des Menüpunktes Copy
Hinweis: Festlegung der Datenart per Methode .setData()
oder CTRL-C
oncut erzeugt durch Quell-Objekt wenn Daten aus Quellobjekt in das Clipboard verschoben werden

Beispiel 1

```
<HEAD>
<SCRIPT>
        var Kette = "";

        function EventBeforeCut()
        {event.returnValue = false; }

        function EventCut()
        {
                Kette= ID_Span.innerText;
                ID_Span.innerText = "";
                event.returnValue = false;
        }

        function EventBeforePaste()
        {event.returnValue = false; }

        function EventPaste()
        {
                ID_Div.innerText = Kette;
                event.returnValue = false;
        }
</SCRIPT>
</HEAD>
<BODY>
        <SPAN ID="ID_Span" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                diesen Text selektieren aund ausschneiden
        </SPAN>
        <BR>
        <DIV ID="ID_Div" onbeforepaste="EventBeforePaste()"onpaste="EventPaste()">
                hierher den ausgeschnittenen Text einfügen
        </DIV>
</BODY>
```

Beispiel 2

```
<HEAD>
<SCRIPT>
        var TextDaten = "new content associated with this object";
        var Kette = "";

        function Init()
        {
                var TextBereich = document.body.createTextRange();  // Bereich aller Texte
```


```
                    TextBereich.findText(ID_Span1.innerText);          // aber den Text im ID_Span1 suchen
                    TextBereich.select();                              // und dann selektieren
                }

                function EventBeforeCut()
                {
                    Kette = ID_Span1.innerText;
                    event.returnValue = false;
                }

                function EventCut()
                {window.clipboardData.setData("Text", TextDaten); }    // setData liefert return-Wert
                                                                       //          also
                event.returnValue = ...;
                                                                       //          noch nötig zu
                kodieren

                function EventBeforePaste()
                {event.returnValue = false; }

                function EventPaste()
                {
                    ID_Span2.value = window.clipboardData.getData("Text", TextDaten);
                    event.returnValue = false;
                }
        </SCRIPT>
        </HEAD>
        <BODY onload="Init()">
            <SPAN ID="ID_Span1" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                Diesen Text ausschneiden
            </SPAN>
            <INPUT  ID=" ID_Span2" TYPE="text" VALUE="Den ausgeschnittenen Text hier einfuegen"
                onbeforepaste="EventBeforePaste()"
                onpaste="EventPaste()"
            >
        </BODY>
```

ondataavailable — erzeugt von der Datenquelle bei jedem Senden von Daten (asynchrones Senden von der Datenquelle (Objekt))

ondatasetchanged — erzeugt von der Datenquelle wenn Daten geändert wurden in der Datenquelle
verwendet bei Filter-Operationen

ondatasetcomplete — erzeugt von Datenquelle wenn alle Daten der Quelle verfügbar also sendbar sind

ondblclick — erzeugt wenn Doppelklick mit Maus mit folgender Eventfolge
onmousedown, onmouseup, onclick, onmouseup, und then ondblclick.

ondeactivate — erzeugt mit Deaktivierung eines Objektes

ondrag — erzeugt kontinuierlich während allen Drag-Operationen, aber erst nach ondragstart Event
Drag = Maus gedrückt halten

ondragend — erzeugt am Ende der Dragoperation also wenn Maus losgelassen wird
erzeugt immer vom Quellobjekt
Hinweis: Zielobjekt erzeugt anschliessend ondragleave Event

ondragenter — erzeugt wenn Maus über Objekt gedrückt wurde UND der User das Quell-Objekt bei gedrückter Maustaste zieht
erzeugt immer vom Ziel-Objekt wenn Ziel erreicht wurde
ist das ERSTE Event des Ziels, also das als Erstes augelöste Event
erzeugt immer vom Quell-Objekt solange Ziel nicht erreicht wurde
wird immer direkt vor dem ondragover Event erzeugt

ondragleave — erzeugt vom Ziel-Objekt wenn User die Maus aus dem Ziel bewegt während Drag-Operation

ondragover — erzeugt vom Ziel-Objekt wenn User die Maus über das Ziel bewegt während Drag-Operation
UND immer direkt nach dem ondragenter Event

ondragstart — erzeugt vom Quell-Objekt wenn User die Selektion für Drag-Operationen ausführt
auch bei Textelement
ist ERSTES Event für Drag-und Drop
Hinweis: Quell-Objekt nutzt Methode .setData() für Übergabe der Daten

ondrop — erzeugt vom Ziel-Objekt wenn Maustaste losgelassen wird über dem Ziel
Drop = Ablegen
direkt erzeugt vor ondragleave und ondragend Events.

onend — erzeugt wenn das aktive Element das Ende der aktiven Timeline erreicht inklusive aller Wiederholungen laut .repeatCount
bzw. wenn das aktive Element gestoppt wird
erzeugt wenn das Elternelement das Ende seiner Timeline erreicht hat
und somit des Kindelement ebenfalls endet
nicht erzeugt wenn .fill mit Wert "freeze" oder "hold" für das Element kodiert wurde,
es sei denn, das Elternelement hat das Ende seiner Timeline erreicht
siehe Methode .endElement() und Eigenschaft .end

siehe onbegin und onrepeat
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:EXCL  ID="ID_Excl"
        BEGIN="indefinite"
        DUR="5"
        REPEATCOUNT="5"
        onend="alert('Ende der Animation');"
    >
        <t:ANIMATEMOTION        ID="ID_Animatemotion"
                                TARGETELEMENT="ID_Div"
                                TO="200,0"
                                BEGIN="0"
                                DUR="2"
                                AUTOREVERSE="true"
        >
        </t:ANIMATEMOTION>
    </t:EXCL>
    <DIV    ID="ID_Div"
        CLASS="time_line_klasse"
        STYLE="position:relative;top:15px;left:25px;height:100px;width:100px;
            border:solid black 1px;
            "
    >
        sich bewegender DIV
    </DIV>
    <BUTTON onclick=" ID_Animatemotion.beginElement();">Start</BUTTON>
    <BUTTON onclick=" ID_Animatemotion.endElement();">Stop</BUTTON>
</BODY>
</HTML>
```

onerror erzeugt wenn Laufzeit-Fehler beim Laden eines Objektes

Beispiel 1:

```
<SCRIPT ...>
<!--
    function eigenes_fehler_fenster(meldungs_text,url,zeilen_nr)
    {
        // Fehlermeldung bilden
        var url_als_kette=url.toString();
        var zeilen_nr_als_kette=zeilen_nr.toString();
        var meldung_komplett=medlungs_text + url_als_kette + zeilen_nr_als_kette;

        // Meldungsfenster
        var fenster=window.open();
        with (fenster.document)
        {
            open("text/html");   // HTML-Dokument im Fenster erzeugen

            writeln("<HTML><HEAD><TITLE>Private Errormeldung</TITLE></HEAD>");
            writeln("<BODY><H1>Fehlermeldung</H1>");

            writeln(meldung_komplett);

                writeln("<FORM><INPUT TYPE=BUTTON VALUE='OK'
              onClick='self.close()'>"
            );          // ist EINE Zeile

            // Button anklicken, damit das Meldungs-Fenster geschlossen wird
            close();  //HTML-Dokument schliessen
        }
        return true;        // muss true liefern für window.onerrer
    }

    var RetteOnError=window.onerror;

    window.onerror= eigenes_fehler_fenster;     // NICHT onError kodieren !
```


```
                                                        // keine () kodieren, da sonst die Funktion
                                                        //          sofort ausgeführt wird !!
                                                        // window.onerror verlangt die Zuweisung
                                                        //          von true für Abschaltung des
                                                        //       Standard-onerror
        // -->
        </SCRIPT>
        ...
        <BODY>
        ...
        </BODY>
```

Beispiel 2:

```
        <SCRIPT>
                function FehlerAufspueren (MeldungString, UrlString, ZielenNummerString)
                {
                        ID_Div.innerHTML  ="<B>Fehler erkannt</B>";
                        ID_Div.innerHTML+="Error: "  + MeldungString         + "<BR>";
                        ID_Div.innerHTML+="Line: "   + ZielenNummerString + "<BR>";
                        ID_Div.innerHTML+="URL: "    + UrlString                + "<BR>";

                        return false;
                }

                window.onerror=FehlerAufspueren;          // ohne () kodieren
        </SCRIPT>
        <BODY>
        <DIV ID="ID_Div"></DIV>
        <BODY>
```

Beispiel 3 für Bild:

```
        <SCRIPT>
                var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild ">';

                function Laden()
                {
                        ID_Div.innerHTML=Kette;                 // wird sofort geparst, also IMG-Tag ausgeführt
                                                                //          ( anstelle von eval()
                                                                //            bzw. document.write() )
                        ID_IMG.src="";
                        ID_IMG.style.display="block";

                        ID_IMG.onerror= IMGAltTextAufFehlerMeldung;     // ohne  ()

                }

                function IMGAltTextAufFehlerMeldung ()
                {
                        ID_IMG.alt="Das Bild konnte nicht geladen werden.";
                        return true;
                }
        </SCRIPT>
        <INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
        <DIV ID="ID_Div"></DIV>
```

onerrorupdate — erzeugt wenn Fehler während Datenupdate für ein Datasource-Objekt auftrat

onfilterchange — erzeugt wenn ein visueller Filter komplett abgelaufen ist

onfinish — erzeugt wenn alle Durchläufe des Marquee-Objektes komplett beendet sind
Anzahl der Durchläufe laut LOOP-Attribut
LOOP-Attribut muss Wert > 1 haben, wenn onfinish erzeugt werden soll

Beispiel

```
        <BODY>
                <MARQUEE LOOP=2  onfinish="alert(event.srcElement.id + ' onfinish-Ereignis erkannt')">
                        Marquee Text
                </MARQUEE>
        </BODY>
```

onfocus — erzeugt, wenn Objekt den Focus erhält
Fokus nur erhaltbar nach dem kompletten Laden des Dokumentes

Beispiel für Label

```
        <STYLE>
             .normal {background-color:"#FFFFFF"; color:"#000000"; font-weight:normal; font-size:8pt; font-family:Arial;}
             .accessible { background-color:"beige"; font-weight:bold; font-size:10pt;}
```

```
		</STYLE>
		<SCRIPT>
			function StyleWechsel()
			{
				// Input-Objekt
				event.srcElement.className="accessible";  // Input-Objekt

				// Label-Objekt
				var ZeigerAuf Label =eval(event.srcElement.id + "_Label ");
							// ID ist "ID_Input"
							//	also "ID_Input" + "_Label" = "ID_Input_Label"
							//  Zeichenkettenoperation  leider nötig, wenn mehrere
							//	Objekte mit Eventerzeugung vorhanden wären
							//  und ebenfalls nötig wegen Bezug im Label auf das
							//	ID des Objektes, das Label haben soll
				ZeigerAuf Label.className="accessible";
			}
		</SCRIPT>
		<LABEL FOR="ID_Input" oInput" CLASS="normal" ID="ID_Input_Label">Text eingeben</LABEL>
		<INPUT TYPE="text" CLASS="normal" onfocus="StyleWechsel()" ID="ID_Input">
onfocusin		erzeugt bevor das Element den Focus erhält
onfocusout		erzeugt nachdem das Element den Focus verloren hat
onhelp		erzeugt wenn F1-Taste gedrückt im aktuellen Fenster
onhide		erzeugt wenn der Media Bar Player gerade unsichtbar gemacht wird (versteckt wird)
			siehe Eigenschaft .enabled
		entspricht dem Schliessen des Media Bar Player durch den User
		Media Bar Player ist der Windows Media Player
		siehe Behavior .style.mediaBar
onkeydown		erzeugt wenn irgend eine Tastatur-Taste gedrückt wurde
	Beispiel
		<SCRIPT>
			function TastenCodeHolen()
			{
				if(ID_Input.checked)
				{
					ID_Textarea.innerText+="[Keycode = " + event.keyCode + "]";
					event.returnValue=false;
				}
				else
				{ ID_Textarea.innerText+=String.fromCharCode(event.keyCode); }
			}
		</SCRIPT>
		<INPUT TYPE="checkbox" ID="ID_Input">
		<INPUT TYPE="text" onkeydown=" TastenCodeHolen()">
		<TEXTAREA ID="ID_Textarea" ROWS="10" COLS="50"></TEXTAREA>
onkeypress		erzeugt bei Druck einer alphanumerischen Tastatur-Taste
	Beispiel
		<HEAD>
		<SCRIPT>
			function ShiftPruefen()
			{
				if (window.event.shiftKey)
				{ID_Input.value = "Shift erkannt";}
			}
		</SCRIPT>
		</HEAD>
		<BODY>
			drücke SHIFT mit anderer Taste
			<INPUT TYPE=text onkeypress="ShiftPruefen()">
			<INPUT TYPE=text ID="ID_Input">
		</BODY>
onkeyup		erzeugt wenn gedrückte Tastatur-Taste losgelassen wird
onlayoutcomplete		erzeugt wenn das Druckobjekt bzw. Druckvorschau-Objekt komplett gefüllt ist
			setzt Eigenschaft .contentOverflow auf true
onload		erzeugt mit dem Laden eines Objektes
	Beispiel 1
		<BODY>
			<SCRIPT FOR=window EVENT=onload LANGUAGE="JScript">
				window.status = "Seite ist geladen!";
			</SCRIPT>
		</BODY>

	Beispiel 2
```


```
<SCRIPT>
        function Meldung()
        {window.status = "Image " + event.srcElement.src + " ist geladen";}
</SCRIPT>
<BODY>
        <IMG SRC="test.gif" onload="Meldung()">
</BODY>
```

onlosecapture — erzeugt, wenn Objekt die Mausüberwachung verliert

Beispiel

```
<BODY onload="ID_Div.setCapture()"
      onclick="ID_Div.releaseCapture();"
>
        <DIV ID="ID_Div"divOwnCapture
            onmousemove="ID_Textarea.value=event.clientX + event.clientY"; // kein alert() !!!!!!
            onlosecapture="alert(event.srcElement.id + ' ohne Maus-Ueberwachung')";
        >
                Ueber diesen Text mit der Maus fahren
                <BR>
                <TEXTAREA ID="ID_Textarea" COLS=2></TEXTAREA>
        </DIV>
        <DIV>
                Klick hier fuer Event onlosecapture per Methode .releaseCapture()
        </DIV>
</BODY>
```

onmediacomplete — erzeugt wenn das Medium zum Element (Media-Element) komplett geladen wurde
für Animation per Timeline
Media-Element benötigt vor seiner Animation dern Datenfluss des Mediums aus einer Datei (streaming media file) z.B. Gif-Bild, Video etc.
sinnvoll für Start des Elementes auf der Timeline
siehe onmediaerror
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
        <t:VIDEO        ID="ID_Video"
                        SRC="test.wmv"
                        onmediacomplete="ID_Span.innerText +=
                                        'Datei test.wmv wurde komplett geladen !'
                                "
        >
        </t:VIDEO>
        <BR>
        <SPAN ID="ID_Span"></SPAN>
</BODY>
</HTML>
```

onmediaerror — erzeugt wenn das Medium zum Element (Media-Element) fehlerhaft geladen wurde
oder das Medium einen Fehler enthält
für Animation per Timeline
Media-Element benötigt vor seiner Animation dern Datenfluss des Mediums aus einer Datei (streaming media file) z.B. Gif-Bild, Video etc.
sinnvoll für Start des Elementes auf der Timeline
ersetzt das Ereignis onmedialoadfailed, das deprecated ist und nicht mehr verwendet werden darf !
siehe onmediacomplete
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
        <t:IMG  ID="ID_Img"
                SRC="test.gif"
```


```
                        STYLE="position:relative;top:25px;left:50px;height:100px;width:100px;"
                        onmediaerror="alert(' Datei test.gif  nicht ladbar !')"
                >
                </t:IMG>
                <t:ANIMATION    ID="ID_Animation"
                                TARGETELEMENT="ID_Img"
                                TO="0,400"
                                DUR="2"
                                BEGIN="0"
                                ACCELERATE="1"
                                AUTOREVERSE="true"
                                REPEATCOUNT="5"
                >
                </t:ANIMATION>
        </BODY>
        </HTML>
```

onmousedown — erzeugt wenn irgendeine Maustaste gedrückt wird

onmouseenter — erzeugt wenn Maus ein Obhjektbereich betritt

onmouseleave — erzeugt wenn Maus gerade den Objektbereich verlässt

onmousemove — erzeugt wenn Maus im Bereich eines Objekt bewegt wird

Beispiel

```
<SCRIPT>
        function Anzeige()
        {ID_Span.innerText="Coords: (" + event.clientX + ", " + event.clientY + ")";}          // kein alert() !!!!
</SCRIPT>
<DIV onmousemove="Anzeige()">
        <SPAN ID="ID_Span"></SPAN>
</DIV>
```

onmouseout — erzeugt wenn Maus den Bereich eines Objektes gerade verlässt
genau 1x erzeugt pro Verlassen

onmouseover — erzeugt wenn Maus den Bereich eines Objektes gerade betritt
genau 1x erzeugt pro Betreten

onmouseup — erzeugt wenn Maustaste losgelassen wird
Hinweis: Eigenschaft .button liefert die losgelassene Taste

onmousewheel — erzeugt wenn Mausrad gedreht wird
mit dem Drehen wird Eigenschaft .wheelDelta gefüllt
mit Integer-Vielfachen von 120 Grad Umdrehung
wenn > 0 so Drehung vom User weg
wenn < 0 so Drehung zum User hin
ab IE 6.0

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
        var ZoomFaktorStartWert = 10;                   // > 0, ganzzahlig, in Prozent
        var ZoomSchrittWeite = 1;                       // > 0, ganzzahlig
        var DrehSchrittWeite = 1;                       // 1 oder 2 oder 3 da Faktor von 120 Grad

        var ZoomFaktor = ZoomFaktorStartWert;           // > 0, ganzzahlig, in Prozent
        function VergroessernVerkleinern()
        {
                if (event.wheelDelta >= (DrehSchrittWeite  * 120))
                { ZoomFaktor += ZoomSchrittWeite;}
                else
                {
                        if (event.wheelDelta <= (-1 * (DrehSchrittWeite  * 120)) )
                        { ZoomFaktor -= ZoomSchrittWeite; }
                }

                if (ZoomFaktor <= 0)
                {ZooFaktor = ZoomFaktorStartWert;}

                ID_Image.style.zoom = ZoomFaktor + '0%';
        }

</SCRIPT>
</HEAD>
<BODY>
        <IMG ID="ID_Image" SCR="test.jpg" onmousewheel="VergroessernVerkleinern()">
</BODY>
</HTML>
```

onmove — erzeugt, wenn Objekt bewegt wird:
Objekt muss **absolute positioniert** sein (im STYLE per "**position:absolute;**")

nicht erzeugt, wenn ein Unterobjekt bewegt wird:
Unterobjekt muss **eigenen** Handler haben

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    function HandleFuerOnMoveStart()
    {
        var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                            //        falls mehrere bewegbare
                                                            //        Objekte vorhanden sind
        ZeigerAufObjektMitEvent.style.backgroundColor = "green";
        ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
    }

    function HandlerFuerOnMove ()
    {
        ID_Span1.innerHTML = event.srcElement.offsetLeft;
        ID_Span2.innerHTML = event.srcElement.offsetTop;
    }

    function HandleFuerOnMoveEnd()
    {
        var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                            //        falls mehrere bewegbare
                                                            //        Objekte vorhanden sind
        ZeigerAufObjektMitEvent.style.backgroundColor = "red";
        ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
    }

    document.execCommand("2D-position",false,true);     // 2-D Positionierung einschalten
</SCRIPT>
</HEAD>
<BODY   onmovestart="HandleFuerOnMoveStart();"
        onmove="HandlerFuerOnMove();"
        onmoveend="HandleFuerOnMoveEnd();"
>
    offsetLeft = <SPAN ID="ID_Span1"></SPAN>
    <BR>
    offserTop = <SPAN ID="ID_Span2"></SPAN>
    <BR>
    <DIV CONTENTEDITABLE="true">
        <DIV STYLE="position:absolute;width:300px;height:100px; background-color:red;">
            bewegbarer DIV
        </DIV>
    </DIV>
</BODY>
</HTML>
```

onmoveend

erzeugt, wenn das Bewegen eines Objektes endet:
Objekt muss **absolute positioniert** sein (im STYLE per "**position:absolute;**")
nicht erzeugt, wenn ein Unterobjektbewegung endet:
Unterobjekt muss **eigenen** Handler haben

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    function HandleFuerOnMoveStart()
    {
        var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                            //        falls mehrere bewegbare
                                                            //        Objekte vorhanden sind
        ZeigerAufObjektMitEvent.style.backgroundColor = "green";
        ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
    }

    function HandlerFuerOnMove ()
    {
        ID_Span1.innerHTML = event.srcElement.offsetLeft;
        ID_Span2.innerHTML = event.srcElement.offsetTop;
    }

    function HandleFuerOnMoveEnd()
    {
```


```
                    var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                                        //      falls mehrere bewegbare
                                                                        //      Objekte vorhanden sind
                    ZeigerAufObjektMitEvent.style.backgroundColor = "red";
                    ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
                }

                document.execCommand("2D-position",false,true);     // 2-D Positionierung einschalten
        </SCRIPT>
        </HEAD>
        <BODY   onmovestart="HandleFuerOnMoveStart();"
                onmove="HandlerFuerOnMove();"
                onmoveend="HandleFuerOnMoveEnd();"
        >
                offsetLeft = <SPAN ID="ID_Span1"></SPAN>
                <BR>
                offserTop = <SPAN ID="ID_Span2"></SPAN>
                <BR>
                <DIV CONTENTEDITABLE="true">
                    <DIV STYLE="position:absolute;width:300px;height:100px; background-color:red;">
                        bewegbarer DIV
                    </DIV>
                </DIV>
        </BODY>
        </HTML>
```

onmovestart

erzeugt, wenn das Bewegen eines Objektes beginnt:
Objekt muss **absolute positioniert** sein (im STYLE per "**position:absolute;**")
nicht erzeugt, wenn ein Unterobjektbewegung beginnt:
Unterobjekt muss **eigenen** Handler haben

Beispiel

```
        <HTML>
        <HEAD>
        <SCRIPT>
                function HandleFuerOnMoveStart()
                {
                    var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                                        //      falls mehrere bewegbare
                                                                        //      Objekte vorhanden sind
                    ZeigerAufObjektMitEvent.style.backgroundColor = "green";
                    ZeigerAufObjektMitEvent.innerText = "DIV wird bewegt ";
                }

                function HandlerFuerOnMove ()
                {
                    ID_Span1.innerHTML = event.srcElement.offsetLeft;
                    ID_Span2.innerHTML = event.srcElement.offsetTop;
                }

                function HandleFuerOnMoveEnd()
                {
                    var ZeigerAufObjektMitEvent = event.srcElement;     // anstelle des ID vom DIV
                                                                        //      falls mehrere bewegbare
                                                                        //      Objekte vorhanden sind
                    ZeigerAufObjektMitEvent.style.backgroundColor = "red";
                    ZeigerAufObjektMitEvent.innerText = "DIV wird nicht mehr bewegt";
                }

                document.execCommand("2D-position",false,true);     // 2-D Positionierung einschalten
        </SCRIPT>
        </HEAD>
        <BODY   onmovestart="HandleFuerOnMoveStart();"
                onmove="HandlerFuerOnMove();"
                onmoveend="HandleFuerOnMoveEnd();"
        >
                offsetLeft = <SPAN ID="ID_Span1"></SPAN>
                <BR>
                offserTop = <SPAN ID="ID_Span2"></SPAN>
                <BR>
                <DIV CONTENTEDITABLE="true">
                    <DIV STYLE="position:absolute;width:300px;height:100px; background-color:red;">
                        bewegbarer DIV
                    </DIV>
                </DIV>
```


```
</BODY>
</HTML>
```

onopenstatechange erzeugt, wenn Media Bar Player den Status bezüglich Playliste, Codec, Lizenz und Individualisierung ändert
siehe Eigenschaft .openState
Media Bar Player ist der Windows Media Player
siehe Behavior .style.mediaBar

Beispiel:

```
<DIV    ID="ID_Div"
        STYLE="behavior:url(#default#mediaBar)"
        onopenstatechange="alert(ID_Div.openState);"
>
</DIV>
<INPUT  TYPE=button
        VALUE='abspielen von test.asx'
        onclick="  ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
                   ID_Div.disabledUI = true;
                "
>
```

onoutofsync erzeugt wenn das Element seine Timeline verliert, also nicht mehr auf seiner Timleline animiert wird (nicht synchron zur Timeline des Elementes)
sinnvoll nur bei Zwangssynchronisierung per .syncBehavior auf "locked":
Eventhandler sollte neu synchronisieren z.B. Reset des Elementes durch .resetElement()
siehe onsyncrestored
siehe Objekt currTimeState und Behavior .style.time2
Syntax:

HTML	<ELEMENT onoutofsync = "handler" ... >
Script	object.onoutofsync = handler <SCRIPT FOR = object EVENT = onoutofsync>

onpaste erzeugt vom Zielobjekt wenn Daten aus Clippboard in das Objekt eingefügt werden

Beispiel 1

```
<HEAD>
<SCRIPT>
        var Kette = "";

        function EventBeforeCut()
        {event.returnValue = false; }

        function EventCut()
        {
                Kette= ID_Span.innerText;
                ID_Span.innerText = "";
                event.returnValue = false;
        }

        function EventBeforePaste()
        {event.returnValue = false; }

        function EventPaste()
        {
                ID_Div.innerText = Kette;
                event.returnValue = false;
        }
</SCRIPT>
</HEAD>
<BODY>
        <SPAN ID="ID_Span" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                diesen Text selektieren aund ausschneiden
        </SPAN>
        <BR>
        <DIV ID="ID_Div" onbeforepaste="EventBeforePaste()"onpaste="EventPaste()">
                hierher den ausgeschnittenen Text einfügen
        </DIV>
</BODY>
```

Beispiel 2

```
<HEAD>
<SCRIPT>
        var TextDaten = "new content associated with this object";
        var Kette = "";
```


```
                        function Init()
                        {
                                var TextBereich = document.body.createTextRange();  // Bereich aller Texte
                                TextBereich.findText(ID_Span1.innerText);            // aber den Text im ID_Span1 suchen
                                TextBereich.select();                                // und dann selektieren
                        }

                        function EventBeforeCut()
                        {
                                Kette = ID_Span1.innerText;
                                event.returnValue = false;
                        }

                        function EventCut()
                        {window.clipboardData.setData("Text", TextDaten); }          // setData liefert return-Wert
                                                                                     //          also
                        event.returnValue = ...;
                                                                                     //          noch nötig zu
                        kodieren

                        function EventBeforePaste()
                        {event.returnValue = false; }

                        function EventPaste()
                        {
                                ID_Span2.value = window.clipboardData.getData("Text", TextDaten);
                                event.returnValue = false;
                        }
                </SCRIPT>
                </HEAD>
                <BODY onload="Init()">
                        <SPAN ID="ID_Span1" onbeforecut="EventBeforeCut()"oncut="EventCut()">
                                Diesen Text ausschneiden
                        </SPAN>
                        <INPUT  ID=" ID_Span2" TYPE="text" VALUE="Den ausgeschnittenen Text hier einfuegen"
                                onbeforepaste="EventBeforePaste()"
                                onpaste="EventPaste()"
                        >
                </BODY>
```

onpause

erzeugt wenn das aktive Element auf der aktiven Timline gerade beginnt zu pausieren
auch bei body Objekt
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
        <t:VIDEO        ID="ID_Video"
                        SRC="test.wmv"
                        onpause="ID_Span.innerText = "Pause !"
                        onmediacomplete="ID_Span.innerText =
                                                'Datei test.wmv wurde komplett geladen !'
                                        "
        >
        </t:VIDEO>
        <BR>
        <SPAN ID="ID_Span"></SPAN>
</BODY>
</HTML>
```

onplaystatechange

erzeugt, wenn Media Bar Player den Status bezüglich Wiedergabe ändert
siehe Eigenschaft .playState
Media Bar Player ist der Windows Media Player
siehe Behavior .style.mediaBar

Beispiel:

```
<DIV    ID="ID_Div"
        STYLE="behavior:url(#default#mediaBar)"
        onplaystatechange="alert(ID_Div.playState);"
```


```
>
</DIV>
<INPUT   TYPE=button
         VALUE='abspielen von test.asx'
         onclick="  ID_Div.playURL('http://www.test.de/test.asx','video/x-ms-asf');
                    ID_Div.disabledUI = true;
                  "
>
```

onpropertychange — erzeugt wenn eine Objekteigenschaft geändert wird
außer Änderung von .innerText und .innerHTML

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>
```

onreadystatechange — erzeugt wenn Status eines Objektes sich ändert
immer erzeugt von datenladenden folgender Objekte:
applet, document, frame, frameSet, iframe, img, link, object, script und xml elements.

Beispiel

```
function Preufen ()
{
         if (document.readyState=="complete")
         {alert('Dokument komplett geladen und mit den Daten initialisiert.);}
}

document.onreadystatechange=Preufen;      // ohne () kodieren
```

onrepeat — erzeugt mit Start **jeder** Wiederholung der Animation des aktiven Elementes
nicht erzeugt beim Start des aktiven Elementes (siehe onbegin), also des ersten Durchlaufes, der keine Wiederholung ist
nicht erzeugt durch Kind des Elementes, wenn Kind nicht selbst einen Handler für onrepeat kodiert hat (kein Heraufreichen des Ereignisses onrepeat zum Elternelement)
.repeatCount bzw. .repeat muss > 1 sein:
onrepeat wird also .repeatCount –1 mal erzeugt
Kind eines Elementes
siehe onend, onbegin, .repeatCount und .repeat
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
         .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
         <t:IMG   ID="ID_Img"
                  SRC="test.gif"
                  STYLE="position:relative;top:25px;left:50px;height:100px;width:100px;"
                  onmediaerror="alert(' Datei test.gif  nicht ladbar !')"
```


```
                        onrepeat="alert('Aktuelle Wiederholung: ' + ID_Animation.currTimeState.repeatCount);"
                    >
                    </t:IMG>
                    <t:ANIMATION    ID="ID_Animation"
                                    TARGETELEMENT="ID_Img"
                                    TO="0,400"
                                    DUR="2"
                                    BEGIN="0"
                                    ACCELERATE="1"
                                    AUTOREVERSE="true"
                                    REPEATCOUNT="5"
                    >
                    </t:ANIMATION>
                </BODY>
                </HTML>
```

onreset

erzeugt wenn Formular zurückgesetzt wird
Resetaktion erzeugbar per
INPUT TYPE="reset"
BUTTON TYPE="reset"

Beispiel

```
<HTML>
<HEAD>
<SCRIPT>
    function Anzeige()
    { ID_Span.innerText += "Anzeige zum Formular Reset";}
</SCRIPT>
</HEAD>
<BODY>
    <FORM NAME="Formular" ID="ID_Formular" onreset="Anzeige();"
        <INPUT TYPE="text" NAME="InputText" VALUE="">
        <BR>
        <INPUT TYPE="reset" VALUE="Formular Reset">
        <BUTTON onclick="ID_Formular.reset();">Formular Reset</BUTTON>
    </FORM>
    <SPAN ID="ID_Span"></SPAN>
    <BR>
    <BUTTON onclick="location.reload(true);">Refresh der Seite</BUTTON>
</BODY>
</HTML>
```

onreset

erzeugt wenn Element zurückgesetzt wurde per Methode .resetElement(),
also die Timeline des Elementes den aktuellen Wert laut .begin wieder erreicht
und damit zurückgesetzt wurde
also nicht beim Initialisieren des Elemtes und seiner Timeline
durch Instanzierung des Elementes
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
    <t:EXCL  ID="ID_Excl"
        BEGIN="indefinite"
        DUR="5"
        REPEATCOUNT="5"
        onreset="alert('Rücksetzen der Animation und Timeline');"
    >
        <t:ANIMATEMOTION        ID="ID_Animatemotion"
                                TARGETELEMENT="ID_Div"
                                TO="200,0"
                                BEGIN="0"
                                DUR="2"
                                AUTOREVERSE="true"
        >
        </t:ANIMATEMOTION>
    </t:EXCL>
    <DIV    ID="ID_Div"
        CLASS="time_line_klasse"
        STYLE="position:relative;top:15px;left:25px;height:100px;width:100px;
            border:solid black 1px;
```


```
                                        "
                        >
                                sich bewegender DIV
                        </DIV>
                        <BUTTON onclick=" ID_Animatemotion.beginElement();">Start</BUTTON>
                        <BUTTON onclick=" ID_Animatemotion.endElement();">Stop</BUTTON>
                        <BUTTON onclick=" ID_Animatemotion.resetElement();">Reset</BUTTON>

                </BODY>
                </HTML>
```

onresize — erzeugt, wenn Objektgröße verändert wird
nicht für embedded-Objekt

onresizeend — erzeugt, wenn Änderung der Objektgröße endet

onresizestart — erzeugt wenn Veränderung Objektgröße endet

onresume — erzeugt wenn die Pause eines aktiven Elementes, das auf der Timeline pausiert, beendet wird
siehe Objekt currTimeState und Behavior .style.time2
auch für body Objekt

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
        <t:VIDEO        ID="ID_Video"
                        SRC="test.wmv"
                        onpause="ID_Span.innerText = "Pause !"
                        onresume=" ID_Span.innerText = "Pause beendet ""
                        onmediacomplete="ID_Span.innerText =
                                                'Datei test.wmv wurde komplett geladen !'
                                        "
        >
        </t:VIDEO>
        <BR>
        <SPAN ID="ID_Span"></SPAN>
</BODY>
</HTML>
```

onreverse — erzeugt wenn das aktive Element rückwärts auf der Timeline animiert wird
(auch wenn es sich um eine Wiederholung handelt, also .repeatCount > 0 ist)
.autoReverse muss auf "true" stehen
siehe Objekt currTimeState und Behavior .style.time2

Beispiel:

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
        .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
</HEAD>
<BODY>
        <t:EXCL  AUTOREVERSE="true"
                DUR="6"
                REPEATCOUNT="indefinite"
                onreverse="alert('Rückwärts auf der Timeline);"
        >
                <DIV CLASS="time_line_klasse" BEGIN="0" DUR="2">Zeile 1</DIV>
                <DIV CLASS="time_line_klasse" BEGIN="2" DUR="2">Zeile 2</DIV>
                <DIV CLASS="time_line_klasse" BEGIN="4" DUR="2">Zeile 3</DIV>
        </t:EXCL>
</BODY>
</HTML>
```

onrowenter — erzeugt wenn neue Daten verfügbar sind in der akuellen Spalte
aufgrund Änderung der Daten in der Datenquelle zu Spalte
nur für databound-Objekte, die selbst Daten halten (Quelle und Ziel identisch)

onrowexit — erzeugt direkt Spaltenwechsel, also vor dem Verlassen der aktuellen Spalte
nur für databound-Objekte, die selbst Daten halten (Quelle und Ziel identisch)

onrowsdelete — erzeugt direkt vor dem Löschen von Spalten

onrowsinserted — erzeugt direkt nach dem Einfügen einer Spalte per Methode .AddNew()

onsave — erzeugt, wenn Webseite als Datei gespeichert wird
oder Webseite als Bookmark in die Favoritenliste eingetragen wird
oder Webseite verlassen wird

Beispiel

```
<HTML>
<HEAD>
<META NAME="save" CONTENT="favorite">
<STYLE>
    .saveFavorite {behavior:url(#default#savefavorite);}
</STYLE>
</HEAD>
<BODY>
    <INPUT   TYPE=text ID="ID_Input" CLASS=saveFavorite
             onsave="javascript:alert('Event onsave erkannt');"
>
</BODY>
</HTML>
```

onscroll — erzeugt wenn User die Scrollpfeile benutzt (nicht den Scrollbalken)

Syntax:

HTML	<ELEMENT onscroll = "handler" ... >
Script	object.onscroll = handler
	<SCRIPT FOR = object EVENT = onscroll>

onseek — erzeugt wenn eine Seek-Methode zum Element aktiviert wurde:

- .seekActiveTime()
- .seekSegmentTime()
- .seekTo()
- .seekToFrame()

siehe Objekt currTimeState und Behavior .style.time2

Beispiel :

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<HEAD>
<?IMPORT namespace="t" implementation="#default#time2">
<STYLE>
    .time_line_klasse { behavior: url(#default#time2) }
</STYLE>
<SCRIPT>
    var FrameAnzahl=600;

    function Suchen()
    {
        // prüfen ob Element nicht aktiv ist
        if  (!ID_Video.currTimeState.isActive)
        {
            // nicht aktiv, also starten
            ID_Video.beginElement();
        }
        else
        {
            // prüfen des Eingabewertes
            if(     (  isFinite(ID_Input.value) )
                && (ID_Input.value   < FrameAnzahl )
                && (ID_Input.value > 0)
              )
            {
                // ist okay, also ab Zeitpunkt animieren
                ID_Video.seekToFrame(ID_Input.value);
            }
            else
            {
                // fehlerhaft
                alert(   "Fehler ! Frame-Wert muss > 0 und < "
                    + FrameAnzahl
                    +   " sein"
                   );
                ID_Input.focus();
            }
        }
    }
</SCRIPT>
</HEAD>
<BODY>
    <SPAN    CLASS="time_line_klasse"
             DUR=".01"
             REPEATCOUNT="indefinite"
             FILL="hold"
             onrepeat="innerText=parseInt(ID_Video.currentFrame);"
```


```
                        >
                                0
                        </SPAN>
                        <BR>
                        <t:VIDEO        ID="ID_Video"
                                        SRC"test.avi"
                                        STYLE="width:175px; height:150px;"
                                        onmediacomplete="ID_Span.innerText= ID_Video.mediaDur;"
                                        onseek="alert('Der Frame ' + ID_Input.value + ' wird eingestellt !')
                        >
                        </t:VIDEO>
                        <BR>
                        Dauer des AVI:
                         <SPAN ID="ID_Span"></SPAN>
                         Sekunden
                        <BR>
                        setze seekToFrame:
                        <INPUT  TYPE="text"
                                VALUE=""
                                NAME="ID_Input"
                                SIZE="3"
                        > Sekunden
                        <BR>
                        <BUTTON onclick="Suchen();">
                                Klick fuer Seek
                        </BUTTON>
                        <BUTTON onclick=" ID_Video.beginElement()">
                                Restart
                        </BUTTON>
                </BODY>
                </HTML>
```

onselect — erzeugt wenn etwas selektiert wird

onselectionchange — erzeugt wenn Selektionsstatus im Dokument verändert wird

onselectstart — erzeugt wenn Objekt selektiert wird

onshow — erzeugt wenn der Media Bar Player gerade sichtbar gemacht wird (nicht versteckt wird)
siehe Eigenschaft .enabled
entspricht dem Öffnen des Media Bar Player durch den User
Media Bar Player ist der Windows Media Player
siehe Behavior .style.mediaBar

onstart — erzeugt mit Beginn eines jedem Loop-Durchlaufes des des Marquee-Objektes
Anzahl der Durchläufe laut LOOP-Attribut
LOOP-Attribut muss Wert > 0 haben, wenn onstart erzeugt werden soll

Beispiel

```
<BODY>
        <MARQUEE BEHAVIOR="alernate" LOOP=2 onstart="alert('onstart-Ereignis erkannt')">
                Marquee Text
        </MARQUEE>
</BODY>
```

onstop — erzeugt wenn STOP-Button im Browser-Menü gedrückt wurde
wird direkt nach den onbeforeunload Event erzeugt
wird vor dem onunload Event erzeugt, da die Webseite verlassen wird

Beispiel

```
var TimerID;

function Rekursion()
{
        // nach belieben etwas tun
}

function TimerLoeschen()
{window.clearInterval(TimerID); }

function Init()
{TimerID = window.setInterval("Rekursion()",1000); }

document.onstop= TimerLoeschen;         // ohne ()
window.onload=Init;                     // ohne ()
```

onsubmit — erzeugt wenn das Formular gesendet wird
Eventhandler **muss** einen Rückkehrcode liefern (true oder false) z.B. per return-Anweisung
Sendeaktion erzeugbar per
INPUT TYPE="submit"

BUTTON TYPE="submit"
Art der Formularaktion laut ACTION-Attribut des Formular-Tag
Senden unterbindbar, wenn Eventhandler false liefert als Rückkercode (muss programmiert werden)
z.B. wenn Daten im Formular fehlerhaft vom User eingegeben wurden

Beispiel

```
<BODY>
    <FORM NAME="Formular" onsubmit="return(SubmitEventHandler());"></FORM>
</BODY>
```

onsyncrestored erzeugt wenn Element gerade mit seiner Timeline synchronisiert wird
nach vorausgegangenem Abbruch der Synchronisation
siehe onoutofsync
sinnvoll nur bei Zwangssynchronisierung per .syncBehavior auf "locked"
siehe Objekt currTimeState und Behavior .style.time2

ontimeerror erzeugt bei Zeitfehler nur für BODY-Objekt bei benutzung der Timeline

Beispiel

```
<SCRIPT FOR="BODY" event="ontimeerror">
    alert("HTML+TIME error erkannt");
</SCRIPT>
```

ontrackchange erzeugt wenn der Track einer Advanced Stream Redirector (ASX) Datei (*.asx)
in der Playliste gewechselt wurde
Beispiel für Aufbau einer ASX-Datei:

```
<ASX Version="1.0" PreviewMode="No" >
<ENTRY>
    <TITLE>First title</TITLE>
    <AUTHOR>Test</ AUTHOR>
    <COPYRIGHT>2002</ COPYRIGHT>
    <ABSTRACT>WAV File</ABSTRACT>
    <REF href=""></REF>
    <banner href = "first_title.gif">
        <moreinfo href = "first_title.doc"><moreinfo>
        <abstract>Visit the first abstract Web site</abstract>
    </banner>
</ENTRY>
</ASX>
```

siehe Objekt currTimeState und Behavior .style.time2

onunload erzeugt mit dem direkt vor dem Unload eines Dokumentes
unload = verlassen des Dokumentes durch Laden eines neuen Dokumentes in das selbe Fenster

Beispiel

```
<HEAD>
<SCRIPT FOR=window EVENT=onunload>
    alert("Event onunload erkannt");
</SCRIPT>

<SCRIPT>
    function Umlenken()
    {location.href="test.htm";}
</SCRIPT>
</HEAD>
<BODY>
    <INPUT TYPE=button VALUE="Klicken ... weiter zu TEST.HTM" onclick="Umlenken()">
    <IMG SRC="test.gif">
</BODY>
```

onURLFlip erzeugt wenn ein Scriptkommando, das in einer Advanced Streaming Format (ASF)-Datei (*.asf)
liegen, ausgeführt wird
siehe Objekt currTimeState und Behavior .style.time2
Hinweis: window.event.URL Url laut einem Kommando aus einer Advanced Streaming Format (ASF)-Datei von Element auf der Timeline
es muss Ereignis onURLFlip aufgetreten sein
siehe Objekt currTimeState und Behavior .style.time2

Event-Behandlung beim Internet Explorer bzw. Netscape

Ansatz

Zwischen Internet Explorer und Netscape bestehen prinzipielle Differenzen der Ereignisbehandlung, so dass Ereignisse z.B. onMouseOver verschiedenartig behandelt werden MÜSSEN. Das bedarf eines enormen Programmierungsaufwandes.

Internet Explorer arbeitet standardgemäß immer mit Eventbubbling:

Ereignis der aktuellen Stufe wird in die nächst höhere Stufe solange weitergereicht, bis das Ereignis verarbeitet wurde. Diese Vorgehensweise ist streng objektorientiert. Vorteil: Spätestens der Browser selber, kann auf das Ereignis reagieren. Selbstverständlich ist ebenfalls Eventcapturing möglich (Abfangen von Ereignissen).

Netscape arbeitet mit Eventcapturing:

Ereignis wird standardgemäß immer in höchster Ebene, also dem Browser registriert. Diese Vorgehensweise ist nicht streng objektorientiert, aber aufgrund der Tasache nötig, dass der Netscape im Gegensatz zum Internet Explorer nicht standardgemäß über integrierte Komponenten des Betriebssytemes verfügt.

Eventcapturing kann per Porgrammmierung geändert werden:

Sollte die aktuelle Ebenen das Ereignis nicht verarbeiten können, so kann das Ereignis solange in die nächst tiefere Ebene durchgereicht werden, bis das Ereignis auch verarbeitet wurde. Nachteile sind: Die letzte Ebene muss unbedingt das Ereignis an das
window.objekt weiterleiten, wenn diese letzte Ebene dar Ereignis nicht verarbeiten kann. Es besteht reichlich Programmierungsaufwand
Der Eventhandler muss dem Ereignis als Parameter übergeben bekommen.

Ereignis und Eventhandler

Die Implementation der Events unterscheiden sich nicht nur zwischen den Browsern Internet Explorer und Netscape sondern auch innerhalb deren Versionen. Es ist daher immer zu prüfen, ob das Event überhaupt und dann auch mit den Eigenschaften implementiert ist:

bei IE
```
if (document.event)
{
        if (.document.event.eigenschaft) ....
}
```
für event ist das das konkrete Event einzutragen
z.B. document.onmouseover
für eigenschaft ist eine konkrete zu kodieren
z.B. if (document.onmouseover.button) .....

bei NS nicht möglich, da kein direkter Zugriff auf das Objekt Event erlaubt ist

Es ist zu beachten, dass beim Netscape im Gegensatz zum Internet Explorer das Ereignis nicht als Objekt direkt angesprochen werden darf, sondern immer nur über den Parameter zum Eventhandler. Der Bezeichner des Paramateters ist frei wählbar. NS weiss, dass dieser Parameter das Eventobjekt referenziert. Es ist also auch der Bezeichner "event" wählbar, der beim Internet Explorer allerdings das Event-Objekt als vordefinierten Bezeichner darstellt. Damit ist klar, dass Bezeichner "event" nur in Eventhandlern des Netscape kodiert werden kann. Für Eventhandler, die IE **und** NS verwalten sollen, darf der Bezeichner nicht "event" lauten !!!

Ein Eventhandler für den Internet Explorer UND den Netscape muss prinzipiell wie folgt aufgebaut sein:

```
// Browser feststellen
var ns = document.layers ? true : false;
var ie = document.all ? true : false;

function EvenhandlerFuerIEundNS(Ereignis_Paramater_der_nur_vom_NS_benoetigt_wird)
{
        if (ie)
        {
                // hier den Parameter IE-gerecht füllen
                Ereignis_Paramater_der_nur_vom_NS_benoetigt_wird = event;
        }
        // hier die Aktionen kodieren, wobei nun NS und IE den Parameter nutzen können
        // beachte: NS und IE haben erheblich unterschiedliche Event-Eigenschaften
        //                  so dass wieder mit if (ns)   bzw. if (ie) gearbeitet werden muss !!

if (ns)
{
        // für den NS muss das Abfangen des Ereignisses XXX anstelle des Abfangens durch den Browser erst aktiviert
        //          werden
        document.captureEvents(Event.XXX);
}
```


```
var RetteOnxxx= onxxx;                                  //  retten nicht vergessen !!
document.onxxx =  EvenhandlerFuerIEundNS;               //  immer OHNE () kodieren !!!!, sonst
                                                        //          wird kein Zeiger übergeben werden !!!
```

Hinweis: XXX bzw.xxx ist das Event

XXX beim NS: vordefiniertes Ereignis-Schlüsselwörter z.B. ONCLICK

xxx z.B. click

In einem Eventhandler für Tastatur- und Mausereignisse sollte keine alert()-Anweisung kodiert werden, da sonst sich der Browser aufhängen kann !!! Dafür die Statuszeile des Browsers benutzen: window.status = '.....';

Beim Internet Explorer muss das Objekt event ausgewertet werden, also welche Art von Ereignis das Objekt event gerade anbietet. Das wird im Eventhandler getan.

Bei Netscape muss das Ereignis als Parameter dem Eventhandlers übergeben werden.

Bestimmte Eventhandler können nachfolgende Ereignisse oder Aktionen ein- oder abschalten. Dazu muss der Eventhandler entweder return true; oder return false; besitzen. Alternativ wäre nur beim IE auch event.returnValue = true; bzw. event.returnValue = false; möglich.

Beispiel für einen Eventhandler für das Event onmouseover

```
function EventHandlerRoutineFuerOnMouseOver(Ereignis)
{
        if (document.all)
        {
                // IE
                // aktuelles Ereignis zuweisen, um es auswerten zu können
                Ereignis = event;       // Ereignis wird damit zum Zeiger !!

                //  Netscape erhält sein Ereignis über den Parameter, der
                //          immer ein Zeiger auf das Eventobjekt ist
        }

        //  hier die Reaktion auf das Ereignis onmouseover kodieren
        //          also die Eigenschaft von Ereignis anwenden

        // nur beim NS muss das Abfangen des Ereignisses aktiviert werden
        if (document.layers)
        {
                //  NS
                // Abfangen des Ereingisses für den Eventhandler ermöglichen,
                //          da standardgemäß das Ereignis vom
                //          window-Objekt bearbeitet wird
                document.captureEvents(Event.MOUSEOVER);
        }
}
// Eventhandler dem Ereignis onmouseover zuweisen

document.onmouseover = EventHandlerRoutineFuerOnMouseOver;
```

Eventbehandlung beim Internet Explorer

Es ist zu beachten, dass beim Netscape im Gegensatz zum Internet Explorer das Ereignis nicht als Objekt direkt angesprochen werden darf, sondern immer nur über den Parameter zum Eventhandler. Der Bezeichner des Paramateters ist frei wählbar. NS weiss, dass dieser Parameter das Eventobjekt referenziert. Es ist also auch der Bezeichner "event" wählbar, der beim Internet Explorer allerdings das Event-Objekt als vordefinierten Bezeichner darstellt. Damit ist klar, dass Bezeichner "event" nur in Eventhandlern des Netscape kodiert werden kann. Für Eventhandler, die IE und NS verwalten sollen, darf der Bezeichner nicht "event" lauten !!!

Beim Internet Explorer muss das Ereignis im Eventhandler immer direkt über das Objekt event (Unterobjekt des Objektes window) behandelt werden. Daher ist für den Eventhandler kein Parameter nötig.

Event des IE einer Nicht-Standardbehandlung unterziehen (attachEvent(event_objekt, funktions_referenz))

Diese Methode ordnet dem Ereignis-Objekt laut Parameter event_bezeichner diejenige Funktion zu, die das Event behandeln soll.
Soll das Event nur einmalig behandelt werden, so ist in der Funktion die Eventbehandlung per .detachEvent() aufzuheben.

Bsp.:
```
function handler_funktion(){..}
attachEvent('onmouseover', handler_funktion);
```


Event von Nicht-Standardbehandlung wieder der Standardbehandlung unterziehen (detachEvent(event_objekt))

Diese Methode ordnet dem Ereingis-Objekt laut Parameter event_bezeichner wieder die Standard-Eventverarbeitung zu und hebt somit attachEvent() auf.

Ereignisbehandlung für mouseover und mouseout ein-bzw. ausschalten (.releaseCapture() und .setCapture())

aussschalten:	releaseCapture()
einschalten:	setCapture()

Event bei Behavior (Verhaltensweise)

siehe Objekt	element für Eventsteuerung eines Behavior per *.htc-Datei .style.time2 für Standard-Behavior des IE

Fehlerbehandlung per onerror

onerror-Standard abschalten

```
<SCRIPT ...>
<!--
        var OnErrorRetten = window.onerror;
        window.onerror=null;    // null ist Schlüsselwort
// -->
</SCRIPT>
```

onerror-Routine privater Art (eigene Fehlerbehandlung einrichten)

onerror Ereignis ausgelöst, wenn
- während Ladens eines HTML-Dokumentes ein Syntaxfehler auftritt
- eines Bildes ein Fehler auftritt
- Abarbeitung von Scripten ein Runtime-Error auftritt

sämtliche Fehlermeldungen unterbinden per
- var RetteOnErrorHandler = window.onerror;
- window.onerror = null;

Eventhandler muss wie folgt kodiert werden:

```
function freier_name_fuer_onerror_behandlung
(   error_erklaerung_string,
    url_des_html_dokumentes_als_string,
    zeilen_nr_des_errors_im_html_dokument
)
{
        .....
        return true;        //  nur wenn true geliefert, dann
                                //      wird die Browsereigenen
                                //      onerror-Behandlung
                            //      unterdrückt
}
```

pro Fehler ein Aufruf des Eventhandlers
- --> Folge von Fehlern, also Folge von Aufrufen

Die Script-Debugger-Aufforderung ist nur in den Eigenschaften des Browsers abschaltbar

Beispiel 1:

```
<SCRIPT ...>
<!--
    function eigenes_fehler_fenster(meldungs_text,url,zeilen_nr)
    {
        // Fehlermeldung bilden
        var url_als_kette=url.toString();
        var zeilen_nr_als_kette=zeilen_nr.toString();
        var meldung_komplett=medlungs_text + url_als_kette + zeilen_nr_als_kette;

        // Meldungsfenster
        var fenster=window.open();
        with (fenster.document)
        {
            open("text/html");   // HTML-Dokument im Fenster erzeugen

            writeln("<HTML><HEAD><TITLE>Private Errormeldung</TITLE></HEAD>");
            writeln("<BODY><H1>Fehlermeldung</H1>");

            writeln(meldung_komplett);

                writeln("<FORM><INPUT TYPE=BUTTON VALUE='OK'
              onClick='self.close()'>"
            );          // ist EINE Zeile

            // Button anklicken, damit das Meldungs-Fenster geschlossen wird
            close();  //HTML-Dokument schliessen
        }
        return true;        // muss true liefern für window.onerrer
    }

    var RetteOnError=window.onerror;

    window.onerror= eigenes_fehler_fenster;     // NICHT onError kodieren !
```


```
                                                    // keine () kodieren, da sonst die Funktion
                                                    //          sofort ausgeführt wird !!
                                                    // window.onerror verlangt die Zuweisung
                                                    //          von true für Abschaltung des
                                                    //       Standard-onerror
        // -->
        </SCRIPT>
        ...
        <BODY>
        ...
        </BODY>
```

Beispiel 2:

```
        <SCRIPT>
                function FehlerAufspueren (MeldungString, UrlString, ZielenNummerString)
                {
                        ID_Div.innerHTML  ="<B>Fehler erkannt</B>";
                        ID_Div.innerHTML+="Error: "  + MeldungString         + "<BR>";
                        ID_Div.innerHTML+="Line: "   + ZielenNummerString + "<BR>";
                        ID_Div.innerHTML+="URL: "    + UrlString                + "<BR>";

                        return false;
                }

                window.onerror=FehlerAufspueren;          // ohne () kodieren
        </SCRIPT>
        <BODY>
        <DIV ID="ID_Div"></DIV>
        <BODY>
```

Beispiel 3 für Bild:

```
        <SCRIPT>
                var Kette = '<IMG STYLE="display: none;" ID="ID_IMG" ALT="Das ist ein Bild ">';

                function Laden()
                {
                        ID_Div.innerHTML=Kette;                  // wird sofort geparst, also IMG-Tag ausgeführt
                                                                 //          ( anstelle von eval()
                                                                 //            bzw. document.write() )
                        ID_IMG.src="";
                        ID_IMG.style.display="block";

                        ID_IMG.onerror= IMGAltTextAufFehlerMeldung;     // ohne  ()

                }

                function IMGAltTextAufFehlerMeldung ()
                {
                        ID_IMG.alt="Das Bild konnte nicht geladen werden.";
                        return true;
                }
        </SCRIPT>
        <INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()">
        <DIV ID="ID_Div"></DIV>
```


Eventbehandlung bei Drag & Drop eines HTML-Elementes beim IE

IE ermöglicht Drag&Drop von HTML-Elementen
auch über Fenster
per Maus: linke Maustaste auf dem Element drücken und gedrückt lassen,
dann Element ziehen an gewünschte Position,
dann linke Maustaste loslassen

Das Quell- und Ziel-Element müssen identischer Art sein !

Eventarten für Drag & Drop

ondrag — Ereignis ausgelöst, immer wenn HTML-Element verschoben wird, also permanent solange gezogen wird
nur verwenden für Erfassung der aktuellen Element-Position
Eventhandler für Quellobjekt kodieren:
nur auf Ereignis reagieren, solange der Bereich des Zielobjektes noch nicht errreicht wurde
private Boolean-Variable verwenden: if (false) ... reaktion
per dragenter auf true setzen
per dragleave auf false initialisieren
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondragend — Ereignis ausgelöst, wenn HTML-Element verschoben, eventuell abgelegt und somit Drag & Drop nun beendet werden kann
aktivieren des Ablegen per dragover
Eventhandler für Quellobjekt kodieren
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondragenter — Ereignis ausgelöst, wenn der Bereich des Ziel-Elementes das 1. Mal betreten wurde
Eventhandler für Zielobjekt kodieren:
muss die private Boolean-Variable auf true setzen
per drag ausgewertet (dort auf false prüfen)
per dragleave auf false initialisieren
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondragleave — Ereignis ausgelöst, wenn während des Ziehens die Maustaste losgelassen wurde, wobei somit Drag & Drop zugleich abgebrochen wurde
Eventhandler für Quellobjekt kodieren:
muss die private Boolean-Variable auf false initialisieren
per drag ausgewertet (dort auf false prüfen)
per dragenter auf true gesetzt
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondragover — Ereignis ausgelöst, wenn der Bereich des Ziel-Elementes ab 2. Mal betreten wurde also permanent fortlaufend, solange Maus im Bereich des Zieles ist und nicht abgelelgt wurde
Eventhandler für Zielobjekt kodieren:
muss event.returnValue=false; enthalten, wenn auch Ablegen im Zielobjekt gewollt ist, also der Einfüge-Cursor mit Pfeil und Pluszeichen angezeigt werden soll (standardgemäß ist keine Ablage möglich , also Cursor mit dem durchgestrichenen Kreis sichtbar !)
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondragstart — Ereignis ausgelöst, wenn Maus auf Quellobjekt dauergedrückt ist, also die Verschiebung damit beginnen kann
Eventhandler für Quellobjekt kodieren
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

ondrop — Ereignis ausgelöst, wenn Quellobjekt auf dem Bereich des Zielobjektes abgelegt wird, falls es per dragend zugelassen wird
Eventhandler für Zielobjekt kodieren
Tag alle Tags, in denen Text markierbar sind: <DIV>, <IMG>, <SPAN>

Beispiel für Drag & Drop

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--

// Eine Grafik per Drag & Drop in das INPUT-Element verschieben und danach einen Text zur Grafik
//                    in die INPUT-Textzeile ablegen

// Quellobjekt und Zielobjekt müssen gleicher Elementart sein !

var ZielObjektErreicht=false;
```


```
var TextZurGrafik="Tanzende Frau";          // Text der durch Drag & Drop der Grafik eingefügt wird

// ###################### Funktionen für das Quellobjekt

function Drag_Start_Ereignis_Routine()
{
        window.status =   'ondragstart ausgelöst für '+ event.srcElement.name
                        + '. Verschieben ist soeben gestartet worden ! Bitte Grafik in das Input-Textfeld
                                                                                verschieben !';
}

function Drag_Ereignis_Routine()
{
        if (!ZielObjektErreicht)
        {
                window.status =   'ondrag ausgelöst für '+ event.srcElement.name
                                + '. Es wird gezogen ! Maus ist an der Position X,Y = '
                                + event.x + ',' + event.y;
        }
}

function Drag_End_Ereignis_Routine()
{
        window.status =   'ondragend ausgelöst für '+ event.srcElement.name
                                + '. Verschieben ist soeben beendet worden !';
}

function Drag_Leave_Ereignis_Routine()
{
        window.status =   'ondragleave ausgelöst für '+ event.srcElement.name
                                + '. Während ziehen wurde Maus losgelassen !';

        ZielObjektErreicht=false;
}
// ###################### Funktionen für das Zielobjekt

function Drag_Enter_Ereignis_Routine()
{
        window.status =   'ondragenter ausgelöst für Zielobjekt ! Bereich des Zielobjektes wurde soeben betreten !';
        ZielObjektErreicht=true;
}

function Drag_Over_Ereignis_Routine()
{
        window.status =   'ondragover ausgelöst für Zielobjekt ! Maus im Bereich des Zielobjektes !
                                                Ablegen nun möglich !';

        event.returnValue=false; // Zeigt den Einfüge-Cursor an, also Pfeil mit Pluszeichen
}

function Drop_Ereignis_Routine()
{
        window.status =   'ondrop ausgelöst für Zielobjekt ! Text vom Quellobjekt wurde eben eingefügt !';
        document.all.ZielObjekt.value=TextZurGrafik;
}
//-->
</SCRIPT>
</HEAD>
<BODY>
Bitte Grafik per Drag und Drop langsam in die Textzeile verschieben und dabei die Statuszeile beobachten !
<BR>
<IMG NAME="QuellObjekt" SRC="picture.gif" ALT="dein bild" HEIGHT=49 WIDTH=30
                                ondragstart ="Drag_Start_Ereignis_Routine()"
                                ondrag      ="Drag_Ereignis_Routine()"
                                ondragend ="Drag_End_Ereignis_Routine()"
                                ondragleave             ="Drag_Leave_Ereignis_Routine()"
>
<BR>
Zeile<BR>
Zeile<BR>
Zeile<BR>
Zeile<BR>
<INPUT TYPE="text" NAME="ZielObjekt" VALUE=""
```


```
                        ondragenter="Drag_Enter_Ereignis_Routine()"
                        ondragover="Drag_Over_Ereignis_Routine()"
                        ondrop="Drop_Ereignis_Routine()"
>
</BODY>
</HTML>
```


Mouse-Eventbehandlung beim Internet Explorer ab 4.x

Mouse-Eventarten

onclick — Ereignis ausgelöst, wenn Maus das Objekt angeklickt hat:
Taste drücken und loslassen
im Handler return false; kodieren für Links, Formularelemente und DIV
wenn eine Aktion nicht erwünscht ist
Bsp.: bei Link wird dieser nicht ausgeführt, obwohl angeklickt
Tag <A>, <BODY>, <INPUT>, <DIV>

oncontextmenu — Ereignis ausgelöst direkt vor Aufruf des Kontextmenüs mit der rechte Maustaste
Unterdrückung, wenn der Eventhhandler event.returnValue=false; enthält
(mit return false; erfolgt keine Unterdrückung)
gilt standardgemäß HTML-seitenweit, also für alle Elemente der Seite, da
das Ereignis standardgemäß bis nach oben weitergereicht wird
Tag fast alle

ondblclick — Ereignis ausgelöst, wenn mit linker Maustaste ein Doppelklick auf Objekt erfolgt
1 Klick = Maustaste drücken und loslassen
Tag fast alle

onmousedown — Ereignis ausgelöst mit Drücken irgendeiner Maustaste
siehe Event-Eigenschaft .button
return false; bzw. returnValue=false; unterbinden **nicht** die
die Ausführung des Eventhandlers
also Unterbindung z.B. in click oder contextmenu kodieren
bei verschachteltet Objekten
wird immer vom innen nach außen (unten nach oben) das Ereignis
geliefert, solange bis das Event behandelt wurde
kann per event.cancelBubble das Weiterreichen nach oben
verhindert werden (in allen Ebene, die oberhalb
derjenigen Ebene liegt, in der cancleBubble aktiviert
wurde)
Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <IMG>,
<INPUT>, <MAP>, <SELECT>, <SPAN>, <TEXTAREA>,
die meisten Textelemente wie <B> etc.

onmouseenter — Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt
Tag unbekannt
wahrscheinlich ab IE 5.02
Empfehlung: mouseover verwenden

onmouseleave — Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt
Tag unbekannt
wahrscheinlich ab IE 5.02
Empfehlung: mouseout verwenden

onmousemove — Ereignis bei **jeder** Mausbewegung ausgelöst
Auswertung für den Bereich eines Objektes durch dessen Eventhandler
Tag <APPLET>, <AREA>, <BODY>, <DIV>,<FORM>, <IMG>,
<INPUT>, <MAP>, <SELECT>, <SPAN>, <TEXTAREA>,
die meisten Textelemente wie <B> etc.

onmouseout — Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt
Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <IMG>
<INPUT>, <MAP>, <SELECT>, <SPAN>, <TEXTAREA>,
die meisten Textelemente wie <B> etc.

onmouseover — Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt
Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <IMG>
<INPUT>, <MAP>, <SELECT>, <SPAN>, <TEXTAREA>,
die meisten Textelemente wie <B> etc.

onmouseup — Ereignis ausgelöst mit Loslassen irgendeiner Maustaste
siehe Event-Eigenschaft .button
return false; bzw. returnValue=false; unterbinden **nicht** die
die Ausführung des Eventhandlers
also Unterbindung z.B. in click oder contextmenu kodieren
bei verschachteltet Objekten
wird immer vom innen nach außen (unten nach oben) das Ereignis
geliefert, solange bis das Event behandelt wurde
kann per event.cancelBubble das Weiterreichen nach oben
verhindert werden (in allen Ebene, die oberhalb
derjenigen Ebene liegt, in der cancleBubble aktiviert
wurde)
Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <IMG>,
<INPUT>, <MAP>, <SELECT>, <SPAN>, <TEXTAREA>,
die meisten Textelemente wie <B> etc.

Mouse-Event-Eigenschaften

.button
0 keine Maustaste gedrückt
1 linke Maustaste gedrückt
2 rechte Maustaste gedrückt
4 mittlere Maustaste gedrückt
Kombination aus 1 bis 4 für Maustastenkombination
z.B. 3 = linke UND rechte Maustaste gedrückt (1 + 2 = 3)
7 = alle Maustasten gedrückt (1 +2 +3 + 4 = 7)

.clientX
horizontale Pixel-Position der Maus gegen über linke obere Ecke (0,0) des Anzeigebereiches im Browserfenster
Hinweis: horizontale Pixel-Position der Maus gegenüber der linken oberen Ecke des HTML-Dokumentes ergibt sich aus
event.clientX + document.body.scrollLeft

.clientY
vertikale Pixel-Position der Maus gegen über linke obere Ecke (0,0) des Anzeigebereiches im Browserfenster
Hinweis: vertikale Pixel-Position der Maus gegenüber der linken oberen Ecke des HTML-Dokumentes ergibt sich aus
event.clientY + document.body.scrollTop

.fromElement
enthält Zeiger desjenigen Objektes, das das Ereingnis onmouseout hat
siehe auch .toElement

.offsetX
horizontale Pixel-Position der Maus im HTML-Element z.B. <DIV>, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberen Ecke (0,0)

.offsetY
vertikale Pixel-Position der Maus im HTML-Element z.B. <DIV>, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberen Ecke (0,0)

.returnValue
enthält den Rückkercode des Eventhandlers
Eventhändler muss true liefern (return true;) , um eine Aktion aufgrund des Events zuzulassen
false liefern (return false;), um eine Aktion aufgrund des Events nicht zuzulassen
Anwendung z.B. bei RESET und SUBMIT vom Formular

.screenX
horizontale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes

.screenY
vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirmes

.srcElement
Zeiger auf dasjenige HTML-Element, für das das Ereignis ausgelöst wurde
entweder im HTML-Tag definieren per onXXX = " ....."
oder Bezug auf den logischen Namen des HTML-Elementes
if (logischer_name = event.scrElement) ....

.toElement
enthält Zeiger desjenigen Objektes, das Ereingnis onmouseover hat
siehe auch .fromElement

.type
enthält die Event-Art z.B. abort

.x
horizontale Pixel-Position der Maus bezüglich der linken oberen Ecke (0,0) des Eltern-HTML-Elementes (parent)

.y
vertikale Pixel-Position der Maus bezüglich der linken oberen Ecke (0,0) des Eltern-HTML-Elementes (parent)

Eventbehandlung für Textoperationen mit der Windows-Zwischenablage (Clipboard) ab IE 5.x

Eventarten

onbeforecopy — markierten Text in die Zwischenablage kopieren:
Ereignis, das direkt vor der Kopieraktion ausgelöst wird
Tag alle Tags mit markierbarem Text
z.B. <A>, <DIV>, <SPAN>, <TEXTAREA>

onbeforecut — markierten Text ausschneiden und in die Zwischenablage einfügen:
Ereignis, das direkt vor dem Ausschneiden ausgelöst wird
verlangt im Händler event.returnValue = false;
für Erzeugung der Kontextmenü-Eintrages zur Ausschneide-Operation
(standardmäßig ist Ausschneiden nicht möglich)
Tag alle Tags mit markierbarem Text
z.B. <A>, <DIV>, <SPAN>, <TEXTAREA>

onbeforepaste — Text aus Zwischenablage einfügen:
Ereignis, das direkt vor dem Einfügen ausgelöst wird
verlangt im Händler event.returnValue = false;
für Erzeugung der Kontextmenü-Eintrages zur Einfüge-Operation
(standardmäßig ist Einfügen nicht möglich)
Tag alle Tags mit markierbarem Text
z.B. <A>, <DIV>, <SPAN>, <TEXTAREA>
Einfüge-Operation im Kontextmenü ermöglichen:
Handler muss return false; bzw. returnValue=false; liefern

oncopy — markierten Text in die Zwischenablage kopieren:
Ereingis, das direkt nach der Kopieraktion
ausgelöst wird
Tag alle die Text definieren, <A>

oncut — markierten Text ausschneiden und in die Zwischenablage einfügen:
Ereignis, das direkt mit dem Ausschneiden ausgelöst wird
verlangt im Handler event.returnValue = false;
für Aktivierung des Kontextmeü-Eintrages zum Ausschneiden
(standardmäßig ist Ausschneiden nicht möglich)
Tag alle die Text definieren, <A>

onpaste — Text aus Zwischenablage einfügen:
Ereignis, das direkt mit dem Einfügen ausgelöst wird
verlangt im Handler event.returnValue = false;
für Aktivierung des Kontextmeü-Eintrages zum Einfügen
(standardmäßig ist Einfügen nicht möglich)
Tag alle Tags mit markierbarem Text
z.B. <A>, <DIV>, <SPAN>, <TEXTAREA>

Beispiel

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
var Text = "";

function AusschneidenUndInDieZwischenAblageEinfuegen()
{
	event.returnValue=false; // Kontexmenü-Eintrag aktivieren
	Text=Span_ID_Quellobjekt.innerText;
	Span_ID_Quellobjekt.innerText = ".... ups, der alte Text ist tatsaechlich weg ! ....";
}

function AusDerZwischenAblageEinfuegenInFormVonAnhaengen()
{
	event.returnValue = false; // Kontextmenü-Eintrag aktivieren
	Span_ID_Zielobjekt.innerText = Span_ID_Zielobjekt.innerText + Text + "<BR>";
}

function VorDemAusschneidenKontextMenuEintragErzeugen()
{event.returnValue=false;}

function VorDemEinfuegenKontextMenuEintragErzeugen()
{event.returnValue=false;}
//-->
```

```
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="Span_ID_Quellobjekt"  onbeforecut="VorDemAusschneidenKontextMenuEintragErzeugen()"
                                oncut="AusschneidenUndInDieZwischenAblageEinfuegen()"
>
Diesen Text mit linker Maus markieren und mit Kontext-Menue (rechte Maus) ausschneiden !
</SPAN>
<BR>
<BR>
<BR>
<SPAN ID="Span_ID_Zielobjekt"  onbeforepaste= "VorDemEinfuegenKontextMenuEintragErzeugen()"
                                onpaste="AusDerZwischenAblageEinfuegenInFormVonAnhaengen()"
>
Bitte die Maus auf diesen Text setzen und Einfügen mit Kontext-Menue (rechte Maus) ausführen !<BR>
Es wird damit eine 3. Zeile erzeugt !<BR>
</SPAN>
</BODY>
</HTML>
```


Druck-Eventbehandlung nur Internet Explorer ab 5.x

onbeforeprint	Druck eines Fensters oder Frames: Ereignis, das direkt vor dem Druckstart ausgelöst wird Tag <BODY>, <FRAMESET>
onafterprint	Druck eines Fensters oder Frames: Ereignis, das direkt nach dem Druckende ausgelöst wird Tag <BODY>, <FRAMESET>

Lade-Ereignisse des Internet Explorer ab 4.x bzw. 5.x - Beispiele

Lade-Ereignisse für Bild

onabort — Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
Tag <IMG>

onload — Ereignis ausgelöst, wenn Bild vollständig geladen wurde
zu animiertes Bild (Gif-Datei):
Ereignis wird bei jeder Animationswiederholung ausgelöst
es kann nur pro komplette Animation das Ereignis behandelt werden
Tag <A>, <APPLET>, <BODY>, <DIV>, <FRAMESET>, <IFRAME>, <IMG>, <SCRIPT>

Lade-Ereignisse für HTML-Dokument und dessen Elemente (außer Bild)

onbeforeunload — Verlassen der HTML-Seite: Ereignis, das direkt vor dem Verlassen ausgelöst wird
wenn im Handler event.returnValue = 'freier_melde_text'; so wird automatisch
vor dem Verlassen ein Anfragefenster geöffnet z.B. Anfrage, ob die HTML-Seite wirklich verlassen werden soll
Tag <BODY>, <FRAMESET>

onload — Ereignis ausgelöst, wenn vollständig geladen wurde
HTML-Dokument mit all seinen Elementen jeder Art
Framset (innerhalb <FRAMESET> kodieren)
DIV
Applet, Script, Link, IFRAME
Tag <A>, <APPLET>, <BODY>, <DIV>, <FRAMESET>, <IFRAME>, <IMG>, <SCRIPT>

onunload — Ereignis ausgelöst, wenn das HTML-Dokument verlassen wird durch
Schliessen Browserfenster
Wechsel zu anderer Seite auch per Browser-Button
window.document.open()
window.document.write()
Tag <BODY>, <FRAMESET>

onstop — Ereingis ausgelöst, wenn
Stop-Button des Browsers gedrückt wurde
Seite verlassen wird auch durch Browser-Buttons
nur ab 5.x

Ladeereignisse anhand readyState

onreadystatechange

IE	NS	MO	OP	SA
4.0+	no	no	no	no

Objekte:
a
acronym
address
applet
area
b
base
basefont
bdo
bgsound
big
blockquote
body
br
button
caption
center
cite
code
col
colgroup
comment
dd
del
dfn
dir

div
dl
dt
em
embed
fieldset
font
form
h1
h2
h3
h4
h5
h6
head
hr
html
i
iframe
img
input
ins
isindex
kbd
label
legend
li
link
map
marquee
nobr
noframes
noscript
object
ol
optgroup
option
p
pre
q
rt
ruby
s
samp
script
select
small
span
strike
strong
style
sub
sup
table
tbody
td
textarea
tfoot
th
thead
title
tr
tt
u
ul
var
xml
xmp