

```

>
</DIV>
<DIV ID="ID_Div2"
      STYLE="position:absolute; left:5; top:400; z-index:-1; background-color:beige; display:none"
>
</DIV>
</BODY>

```

Zugriff:

Zeiger.eigenschaft

Eigenschaften:

sind les- und schreibbar

.bottom	untere Pixelposition des Rechteckes um ein Objekt
.left	linke Pixelposition des Rechteckes um ein Objekt
.right	rechte Pixelposition des Rechteckes um ein Objekt
.top	obere Pixelposition des Rechteckes um ein Objekt

Methoden:

keine

4.3.2.2.5. window.event Objekt

Instanz aller Ereignisse im Fenster

Ein Ereignis ist ein browserinternes Signal zu einer getätigten Aktion anhand des HTML-Elementes. Das Signal zeigt an, dass eine Aktion erfolgt ist bzw. noch andauert. Typisches Ereignis ist das Mausklicken des Users auf ein HTML-Element: Klickt der User, so wird die Aktion "Klick" per Ereignis "onclick" signalisiert, wenn das HTML-Element klickbar ist, also die Aktion "Klick" unterstützt.

Ein Ereignis kann von einem konkreten HTML-Element bzw. Objekt nur dann ausgelöst werden, wenn es Events überhaupt unterstützt. Nicht alle HTML-Elemente unterstützen Events. Die Event-Unterstützung von HTML-Elementen ist in der Scriptmaschine spezifisch zu jedem HTML-Element genau vordefiniert. Aus dieser Menge der vordefinierten Events zum HTML-Element kann der Programmierer schöpfen und Aktionen des Users zum HTML-Element zulassen oder unterbinden. Dabei gilt die Regel: Was nicht explizit zugelassen wurde, gilt als unterdrückt, falls es keine standardmäßige Zulassung gibt.

Ein HTML-Objekt kann natürlich nur dann Events auslösen, wenn es instanziiert ist. Beispiel für einen DIV: Wird dem DIV über die Eigenschaft .innerHTML nur ein Leerzeichen zugewiesen (zeiger_auf_div.innerHTML=' ' ; mit zeiger laut ID-Attribut-Wert), so gilt der DIV für die Eventerzeugung z.B. onmousedown als **nicht instanziiert**. Desweiteren darf der DIV per Style-Eigenschaft style.visibility mit Wert 'hidden' **nicht ausgeblendet** sein.

Pro Aktionsart existiert eine Eventart. Erfreulicherweise gibt es diverse Aktionen, die von vielen HTML-Elementen unterstützt werden, vorallem eben o.g. Useraktionen. Aber es gibt diverse Aktionen, von denen der User nichts mitbekommt. Die Eventbehandlung dient also nicht ausschliesslich der Aktionsfreudigkeit des Users und Programmierers, sondern ist ein Nebeneffekt in Form der interaktiven Webseite.

Viele Objekte können erst kommunizieren, wenn sie Events erzeugen bzw. ein Signal als Voraussetzung für eine Aktion bekommen. Dabei ist das Wort "Aktion" auch als "Verhalten" zu verstehen., also als Komponente zur Steuerung von HTML-Elementen während der Anzeige des HTML-Dokumentes im Browserfenster. Typisches Event ist das Signal "onload", das ausgelöst wird, wenn das HTML-Dokument komplett geparkt und falls nötig in das Browserfenster geladen **wurde**. Es gibt diverse Aktionen, die erst **nach** Auslösung von onload zulässig sind. Diese Aktionen betreffen auch die skriptgesteuerte Verwaltung des HTML-Dokumentes zu dessen Laufzeit.

Jedes instanziierte Objekt, das Events unterstützt, nutzt das Eventobjekt, mit dem die Verarbeitung aller unterstützten Events möglich ist (verschiedene Eventarten).

Per JScript kann ein Eventobjekt nur für das Objekt document erzeugt werden.

Aufgrund der Verschachtelung von HTML-Elementen müssen Events auch innerhalb der Hierarchie der HTML-Elemente **und** in der Vererbungsfolge verfügbar sein. Events können also durchgereicht werden. In Script wird üblicherweise die Punktnotation für Hierarchieebenen verwendet. Events können nur dann durchgereicht werden, wenn **beide betroffene Ebenen** diese Events unterstützen. Natürlich müssen die verschachtelten HTML-Elemente, welche Events durchreichen (und diese eventuell zuvor auswerten, wobei das Kind anders auf das Event reagieren kann als Eltern, die aber über das Ereignis immer informiert werden **sollten**), das Entstehen von Ereignissen auch überwachen, denn es muss nicht bekannt sein, wann und wo das Ereignis entsteht. Diese Überwachung kann in Verbindung mit Script genau gesteuert werden.

Standardgemäß gilt beim Internet Explorer: 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 instanziiert, 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).

Es existiert pro HTML-Element bzw. Objekt, das Events unterstützt, eine je nach Eventart vordefinierte Standardbehandlung. Aber Events können abweichend von der Standardbehandlung durch private Eventhandler in Script verarbeitet werden. Auch die Eventüberwachung kann in Script genau gesteuert werden: Es muss also nicht sein, dass ein Kind ein Event seinen Eltern mitteilt (darüber entscheidet der Programmierer).

Aufgrund dieser Tatsache sind Script-Methoden der Eventverwaltung nicht im Objekt event implementiert, sondern im Objekt, das das Event erzeugt. Die Implementierung ist objekt-spezifisch (siehe konkrete Objektbeschreibung) aber genormt. Es gibt also nicht viele jedoch wie immer browserspezifische Methoden.

Microsoft ändert fortlaufend die Active-X-Eigenschaften von Windows und somit auch des Internet Explorers

Diese fortlaufenden Änderungen muss der Programmierer in Erfahrung bringen.

Der Programmierer kann sich definitiv nicht auf Verfügbarkeit von Active-X-Controls verlassen und muss damit rechnen, dass seine Webseiten schlagartig nicht mehr komplett laufen weil u.a. Programmcode noch nicht angepasst ist. Ebenfalls muss der Programmierer Varianten von Windows und Patchzustände beachten, die prinzipiell Kostenprobleme verursachen können.

Mit anderen Worten: Wer Microsoft-Komponenten nutzt, muss wissen, was ihm blüht ... siehe nachfolgende Beispiel für Risiken.

Prinzipielle Lizenzprobleme für den Programmierer

Microsoft verlangt Lizensierung von Windows. Bezüglich Windows-Versionen gibt es die Updatestufen z.B. per Servicepacks

Ein Windows mit Servicepack fällt unter die Lizenz des geupdateten Windows.

Ein Windows mit Vorversion zum Servicepack bedarf einer anderen Lizenz.

Will man z.B. den Internet Explorer 7 und 6 parallel testen, benötigt man 2 Windowslizenzen, da beide Versionen nicht parallel installierbar. Dazu kommt, dass es den IE 6 in 2 Versionen gibt: Win SP1 und SP2 (IE 7 nur ab Win SP2).

Für 3 Browserversionen benötigt man 3 Windowslizenzen, will man parallel testen.

Ein Blick auf Browser-Konkurrenzprodukte klärt die Sachlage unschlagbar: Opera ist z.B. parallel installierbar.

Hinweis: Man suche doch mal im Internet nach einem kostenlosen HTTP-Server vom Microsoft, um IE-Seite testen zu können, die JScript nutzen (inklusive Debugger). Denn sollte kein kostenloses Angebot findbar sein, kommen die Kosten von Entwicklungssoftware zum IE hinzu. Ein Blick auf Konkurrenz-HTTP-Server klärt die Sachlage: Apache-HTTP-Server ist kostenlos, allerdings nicht einfach einzurichten (Hinweis: Der HTTP-Server sollte virtuelle Hosts einrichten können und korrekt mit der Firewall des Users zusammenarbeiten können).

Abänderungen wegen Sicherheitspatches der jeweiligen Windows-Versionen

Abschaltungen von Active-X-Controls erfolgen auch im Rahmen der Sicherheitspatches zu Windows-Versionen.

Es ist auch möglich, dass wegen Sicherheitslücken abgeschaltet wird und somit Komponenten einer Webseite je nach Windowsversion nicht mehr laufen.

Im Rahmen der Sicherheitspatches ist es Microsoft sogar gelungen, Webseiten, die den MS-Encoder zur Komprimierung von

HTML- und JScript-Code nutzen, schlagartig unnutzbar zu machen: Ein Bug in einem Patch zu Windows XP - Q918899

Das Patch verursacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit

Kompression genutzt wird z.B. bei
onlick-Handler auf IMG
klick ins Fenster per aktivem PopUp

Der Absturz ist "read"-Fehler von immer ein und derselben Speicherstelle.

User, die dieses Patch installiert haben, können ab sofort keine IE-Seiten mit codiertem Script mehr ansehen.

Microsoft stellt Abhilfe nach geraumer Zeit zur Verfügung, jedoch spezifisch nach Windows XP-Version:

Patch Q918899 für

Windows XP SP1Download für jedermann bereitgestellt

SP2 nur auf kostenpflichtige telefonische Anfrage des Users per Downloadlink bereitgestellt, da

Microsoft explizit die User registrieren will, bei denen das

Patchproblem auftritt (User muss sich Telefonnummer besorgen)

Solange also das Patch zum fehlerhaften Patch vom User nicht installiert wird,

z.B. weil der User keine Ahnung hat, dass und wo er sich die Telefonnummer

von Microsoft besorgen muss bzw. zu besorgen hat, wird der User

IE-Seiten mit komprimierten Code dauerhaft nicht nutzen können.

(Microsoft-Support ist z.T. nur in Englisch).



Abänderungen wegen Browser-Inkompatibilität*Popupblocker-Fehler*

Die Microsoft Browser-Version IE 7 ist nicht abwärtskompatibel bezüglich Popup per window.createPopup()

Popup per window-Objekt ist ein Markenzeichen des IE, das im IE 7 nicht mehr fehlerfrei nutzbar ist.

Der Fehler liegt in der Popup-Blockerverwaltung des IE und wurde mit dem IE 7 implementiert.

Der Fehler tritt nicht auf, wenn ein Fenster per window.open() erzeugt wurde.

Bedingung:

Scriptfehleranzeige ist erlaubt im IE 7

Popupblocker ist im IE abgeschaltet

ein aktives Fenster (Register) mit Dokument, dass fortlaufend (rekursiv) genau 1 window.popup per .show() erzeugt.

ein weiteres Fenster (Register) z.B. leere Seite (about:blank)

beide (Register) liegen in einer gemeinsamen IE-Instanz

Ablauf: Wird Focus auf Register der leeren Seite gehalten und wird parallel das Popup per .show() erzeugt,

bricht der Browser das Dokument mit .show() ab (Scriptfehler).

Der Popupblocker für die leere Seite verursacht den Programmfehler im Dokument mit .show(). Es wird folgende

Meldung angezeigt (in der Informationsleiste):

'Ein Popup wurde geblockt. Klicken Sie hier, um das Popup bzw. weitere Optionen anzuzeigen.'

Die Bedeutung der Meldung laut Microsoft-Hilfe im IE 7:

Der Popupblocker hat ein Populfenster geblockt. Sie können den Popupblocker deaktivieren

oder Popups temporär zulassen, indem Sie auf die Informationsleiste klicken.

Die Realität zur obigen Meldung ist völlig anders:

Linke oder rechte Maus auf die Meldung liefert z.B. Einstellungen darunter

Popupblocker einschalten

weitere Informationen

jedoch keine Möglichkeit wie laut Bedeutung

Damit gilt: Der abgeschaltete Popupblocker ist in Wirklichkeit aktiv.

Pikant: Ein Popup erscheint normalerweise auch über fremde Fenster, die nicht das Popup erzeugt haben (z.B. Fenster einer Windowsanwendung z.B. einer anderen IE-Instanz)

Der Popupblocker des IE bemerkt aber NUR Webseite, die das Popup erzeugt.

Durch das Abwürgen von Popup wird das Popup natürlich auf und für anderen Seiten nicht relevant; im Falle einer anderen IE-Instanz also auch für diese nicht relevant, obwohl diese Instanz per Popupblocker verwaltet wird.

Der Popupblocker beschneidet die Popup-Reichweite an der Wurzel, ist aber nicht objektorientiert zu den anderen Webseiten (die nicht das Popup erzeugt haben).

Der Popupblocker ist nicht als Filter aufgesetzt sondern reingestrickt worden.

Der Popupblockerfehler verändert die Eventverwaltung:

Es werden u.a. ignoriert

onfocus

onblur

onfocusin

onfocusout

und viele andere, so dass trotz Events z.B. des Body der Popupblockerfehler entsteht.

// nachfolgender Code setzt focus nicht neu: Fenstereintrag in Taskleiste blinkt eventuell

window.focus();

window.document.focus();

if(document.body!=null)

{if(document.body.style!='hidden') // wenn hidden so focus() nicht möglich (Scriptfehler erzeugt)

{document.body.focus();}

}

// wenn paralleles Fenster offen (on oder offline), so Scriptfehler erzeugt

popupzeiger.show(...);

Hinweis: Der Popupfehler ist so elementar, dass die vielen Beta-Testphasen des IE mehr als fragwürdig erscheinen, wie die Angabe von Microsoft, dass Code neu programmiert wurde, um den IE sicherer zu machen.

focus-Methode beim IE 7

windows.focus() document.focus() und body.focus() funktionieren NICHT

zwischen Register in einem IE-Fenster

zwischen Fensters z.B. in Taskleiste

Hinweis:

.focus() setzt Element aktiv, gibt dem Element den Focus und feuert dann onfocus

.setActive() ist Teilmenge von .focus(): nur das aktiv setzen

funktioniert nicht mit allen Elementen, mit denen .focus() funktioniert

animierte Gif (mit Timer)

Animierte Gifs (mit Timer), die unter IE 6 korrekt laufen, müssen unter IE 7 im Timer nicht mehr laufen:

z.B. garnicht mehr sichtbar, oder Timer nicht verwendet.

Dann müssen animierte Gif-Bilder nach IE-Version bereitgestellt werden.



Abänderungen wegen Rechtstreitigkeiten von Microsoft mit Fremdanbietern

Ein sehr bekanntes Beispiel ist die nachträglich eingeführte Einschränkung von Active-X-Controls wegen Patentwahrung durch Microsoft, wobei für den JScript-Programmierer massive Änderungen eintreten.

Wegen Patentwahrung hat Microsoft ein zunächst freiwilliges Patch herausgegeben, dass bei ActiveX-Control per APPLLET, EMBED oder OBJECT, die auf dem Bildschirm rendern (mit oder ohne Userschnittstelle), dafür sorgt, dass bei mouseover über das Control eine Sprechblase erscheint, die darauf hinweist, dass das Objekt als ActiveX-Control klickbar ist. Diese Sprechblase erscheint auch, wenn das Control keine Userschnittstelle hat, also diese gar nicht klickbar ist.

Es wurde das Eventmodell gleichzeitig geändert:

Es werden alle Events solange unterdrückt, bis der User die Sprechblase geklickt hat.
 Das Klicken muss auf das Objekt im Sprechblasenrahmen erfolgen, der so groß ist, wie die Dimension, in der gerendert wurde.
 Es muss also ERST per Mausklick das Control aktiviert werden, ehe das Control klickbar und damit die Eventsteuerung aktiviert ist.
 Ein Control, dass programmtechnisch zwar was rendert, aber ansonsten ohne sichtbare programmtechnisch startet, muss ebenfalls geklickt werden, obwohl es bereits läuft und es nichts zu klicken gäbe (wenn keine Eventsteuerung eingebaut wurde).
 Wegen blockierter Eventsteuerung ist also die Sprechblase z.B. nicht automatisch klickbar.
 Die Eventauslösung per nicht-objekteigenen Eventhandler, der für das Objekt per fireEvent() ein Event auslöst, ist solange blockiert, bis der User die Sprechblase geklickt hat.

style.visibility='hidden' wird ignoriert

Die Sprechblase erscheint auch dann, wenn das Control mit style.visibility='hidden' belegt ist, also sich unsichtbar rendert:

Der Sprechblasenrahmen hat genau die Dimension wie die des unsichtbaren Controls. Der Sprechblasenrahmen erscheint also Zusammenhangslos, und der User weiß nicht, warum er klicken soll, wenn er nichts sieht. Vor allem weiß er nicht, WAS er klickt ... ideale Basis für Schadsoftware per Script.

Diese Sprechblase erscheint nur DANN NICHT, wenn die Userschnittstelle mit Breite == Höhe == 0 gerendert wird. Sollte die Userschnittstelle in einem Container liegen, z.B. DIV, dann wird der Container, wenn er in der Dimension kleiner ist, also die Userschnittstelle, angepasst. Daher muss der Container ebenfalls mit Breite == Höhe == 0 gerendert werden. Wegen Dimensionierung auf 0 sollte style.visibility="hidden" sein. Im Falle eines Containers reicht es, den style des Containers zu ändern, da visibility normalerweise vererbt wird an Kinder, also auch an das Control.

Abänderung wegen Abschaltungen

DirectX ist wegen Abschaltung von Active-X--Controls nicht mehr abwärtskompatibel:

Z.B. wurde bei Win XP SP2 Direct Animation aus DirectX schlagartig durch Abschaltung von Bibliotheken dezimiert, die es bei Win XP SP1 aber noch gibt.

Hier ein Beispiel aus dem Jahr 2004: Abschaltungen von Active-X-Controls

ActiveX-Controls und Unterstützung/Verbot 20041215**erlaubt sind noch**

Tabular Data-Steuerelement {333C7BC4-460F-11D0-BC04-0080C7055A83} Das TDC (Tabular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformat vorliegen, beispielsweise durch Darstellung in einer Tabelle oder Sortierung. Weitere Informationen:•
http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp(http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp)

Microsoft Agent Control - Version 2.0 {D45FD31B-5C6E-11D1-9EC1-00C04FD7081F} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Weitere Informationen:•
<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>(<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

Microsoft MSChat-Steuerelement-Objekt 2.0 - 2.5 {D6526FE0-E651-11CF-99CB-00C04FD64497}
 Dieses Steuerelement wird von Webautoren verwendet, um text- und graphisch basierte Chatgemeinden für Echtzeitkonversationen im Web zu erstellen.



Microsoft ActiveX Upload-Steuerelement, Version 1.5 {886e7bf0-c867-11cf-b1ae-00aa00a3f2c3} Dieses Steuerelement kann auf vielerlei Art genutzt werden, um auf einfache Weise Webinhalte via Drag and Drop zu veröffentlichen. Weitere Informationen: • 230298 (<http://support.microsoft.com/kb/230298/DE/>) - Posting Acceptor Release Notes

• http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp
(http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp)

verboten sind

Datenbindung RDS {BD96C556-65A3-11D0-983A-00C04FC29E36} {BD96C556-65A3-11D0-983A-00C04FC29E33} Die RDS (Remote Data Service) Steuerelemente ermöglichen dem Browser, client-basierte SQL Abfragen an einen Webserver zu stellen. Inzwischen wurde RDS jedoch durch neuere Standards wie SOAP abgelöst, von einer weiteren Verwendung von RDS wird daher abgeraten. Weitere Informationen: • 184375 (<http://support.microsoft.com/kb/184375/DE/>) - Sicherheitsaspekte bei RDS 1.5, IIS 3.0 oder 4.0 und ODBC

<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>
(<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>)

http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp
(http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp)

XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer {550dda30-0541-11d2-9ca9-0060b0ec3d39} {CFC399AF-D876-11d0-9C10-00C04FC99C8E} {e54941b2-7756-11d1-bc2a-00c04fb925f3} {7108ECB4-AFDC-11D1-ADC1-00805FC752D8} XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer ermöglichen die Verarbeitung von XML Daten, etwa die Bindung von HTML Elementen an einen XML Datensatz, oder das Einlesen, Manipulieren, und Zurückschreiben von XML Daten.

Die Steuerelemente DOMDocument und XMLIslandPeer bzw. die dazugehörigen ClassIDs sind nicht mehr aktuell, so dass von einer generellen Freigabe dieser Steuerelementgruppe abgeraten wird. Weitere Informationen: • http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp (http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp)

Internet Explorer

Active Setup / IE Active Setup-Steuerelement {F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1} Dieses Steuerelement enthält die in Microsoft Security Bulletin MS99-037 beschriebene Sicherheitsanfälligkeit. Umeine weitere Ausführung zu verhindern wurde im Rahmen dieses Security Bulletins ein Kill-Bit gesetzt, so dass selbst bei einer Freigabe dieses Controls eine Ausführung blockiert wird. Weitere Informationen: •

<http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>
(<http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>)

<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>
(<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>)

240797 (<http://support.microsoft.com/kb/240797/DE/>) - So verhindern Sie die Ausführung von ActiveX-Steuerelementen in Internet Explorer

Media Player / Active Movie Runtime {A4001DE0-7075-11d0-89AB-00A0C9054129} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Runtime Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / ActiveMovie-Steuerelement {05589FA1-C356-11CE-BF01-00AA0055595A} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.



Media Player / Microsoft NetShow Player {2179C5D3-EBFF-11CF-B6FD-00AA00B4E220} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das NetShow Player Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / Windows Media Player {22D6F312-B0F6-11D0-94AB-0080C74C7E95} Dies ist das Steuerelement für Windows Media Player version 6.4 und war Installationsbestandteil bis einschließlich Windows Media Player Version 8. Ab Windows Media Player 9 wurde diese ClassID durch die neue ClassID {6BF52A52-394A-11D3-B153-00C04F79FAA6} abgelöst, deren Verwendung stattdessen empfohlen wird. Ab Windows Media Player Version 9 wird ferner die alte ClassID anhand eines Wrappers automatisch auf die neue ClassID umgeleitet. Die ClassID für Windows Media Player Version 9 ist jedoch nicht in der Liste der vom Administrator genehmigten Steuerelemente enthalten, und muss bei Bedarf manuell hinzugefügt werden.

Animierte Schaltflächen {0482B100-739C-11CF-A3A9-00A0C9034920} Dieses Steuerelement erlaubte in frühen Versionen des Internet Explorer die Verwendung animierter Schaltflächen auf Webseiten. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von der Freigabe des Steuerelements wird daher abgeraten.

IE Label-Steuerelement

{99B42120-6EC7-11CF-A6C7-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 auch kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

IE Menu-Steuerelement {74701400-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ermöglicht die Handhabung von Menüstrukturen in Webseiten, wird jedoch nicht mehr unterstützt und dürfte nur noch selten Verwendung finden. Von einer Freigabe des Steuerelements wird daher abgeraten.

IE Preloader-Steuerelement {16E349E0-702C-11CF-A3A9-00A0C9034920} Dieses Steuerelement ermöglichte das Vorladen von Webseiten, ist jedoch inzwischen nicht mehr aktuell, wird nicht mehr unterstützt und dürfte nicht mehr im Einsatz sein. Aufgrund einer potentiellen Sicherheitsanfälligkeit in diesem Steuerelement wird von einer Freigabe abgeraten. Weitere Informationen: • 231452 (<http://support.microsoft.com/kb/231452/DE/>) - Update Available for "Legacy ActiveX Control" Issue

IE Timer-Steuerelement {59CCB4A0-727D-11CF-AC36-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

MCSiMenü {275E2FE0-7486-11D0-89D6-00A0C90C9B67} Dieses Steuerelement dient der Anpassung von Popupmenüs, ist jedoch nicht mehr aktuell und wurde nach Windows 98 nicht mehr ausgeliefert. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten.

Popupmenüobjekt {7823A620-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht



mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

Microsoft Agent Control - Version 1.5 {F5BE8BD2-7DE6-11D0-91FE-00C04FD701A5} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Diese Version des Steuerelements ist jedoch nicht mehr aktuell und wird nicht mehr unterstützt. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • <http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm> (<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

Aktive Inhalte im Internet Explorer

Ab IE 6.0 ist das Blockieren aktiver Inhalte möglich, z.B. als Standardeinstellung. Es wird also dem IE verboten, JScript zu nutzen. Daher muss mit Start der Webseite auf das Blockieren von Inhalten der Webseite, die auf JScript basieren, aufmerksam gemacht werden. Bleibt die Blockierung aktiv, so muss die Webseite ALLE Elemente, die per Script angesteuert werden, inaktiv machen: Am besten garnicht erst anzeigen. Oder es wird eine scriptfreie Version der Webseite per <NOSCRIPT> aktiviert, wobei dann Browser vorzuziehbar sind, die z.B. CSS

exakter rendern als der IE (will man keine IE-spezifischen HTML-Elemente verwenden).

Wenn der IE 6.x aktive Inhalte blockiert, wird NOSCRIPT-Tag aktiviert, Ausnahme: Frameset

FRAMESET ist ein aktiver Inhalt:

Da der Frameset anstelle <BODY> kodiert sein muss, gilt:

Alle Tags, die für BODY zulässig sind, werden ignoriert, auch NOSCRIPT.

Wird neben Frameset noch BODY kodiert, so wird Frameset ignoriert.

Die Freigabe der Scriptblockierung erzeugt Ausführung aller Script-Teile inklusive der Eventauslösungen

Bsp.: Folgendes funktioniert vom Dokument, das window.open() hat im geöffneten Dokument (Quelltext im Dokument das window.open() verwendet):

```
function Y_unload(X00){X85[X00].close();}

var X85=new Array();var X86=new Array();

X85[0]=window.open(...);

var X87='parent.Y_unload(0);'; X86[0]=new Function(",X87);

X85[0].document.body.onunload=X86[0];
```

Wird die Scriptblockierung im geöffneten Fenster abgeschaltet, so wird das Fenster geschlossen, weil onunload ausgelöst wird.

Achtung: document.body.onunload funktioniert ev. nicht mehr wenn z.B. mit attachevent() aktiviert wurde

Folgende Metatags sind für den IE 6.x aktiver Inhalt:

```
<META HTTP-EQUIV="imagetoolbar" CONTENT="no">
unterdrückt NICHT IE-Kontextmenü rechte Maus auf Bild
```

```
<META HTTP-EQUIV="site-enter" CONTENT="revealtrans(duration=0.3, transition=12)">
<META HTTP-EQUIV="site-exit" CONTENT="revealtrans(duration=0.3, transition=12)">
```

Achtung: Für das Hinzufügen von Elementen in den BODY (document.body) per DOM-Funktion createElement() MUSS der Body komplett geparkt sein (document.body.readyState == 'complete').

Grund: Es wird standargemäß immer am Ende des BODY angefügt.

Für das Hinzufügen nicht an das Ende des BODY muss im HTML-Code ein Platzhalter z.B. DIV kodiert sein, innerhalb

dessen dann die neuen HTML-Elemente erzeugt werden.

Instanz-Erzeugung in HTML:

Beispiele zur Kodierung von onXXX="..." im HTML-Tag des Objektes, wobei onXXX ein im Objekt implementiertes Event ist, z.B. onclick.

Übliche Kodierung:

```
<HEAD>
<SCRIPT>
```



```

        function InputButton_Event_onclick_Handler()
        {alert("onclick erkannt");}
    </SCRIPT>
</HEAD>
<BODY>
    <INPUT  TYPE=button
           NAME="ID_Button"
           VALUE="Test"
           onclick="InputButton_Event_onclick_Handler()"
    >
</BODY>

```

Alternative 1:

```

<HEAD>
    <SCRIPT>
        function InputButton_Event_onclick_Handler()
        {alert("onclick erkannt");}
    </SCRIPT>
</HEAD>
<BODY>
    <INPUT  TYPE=button
           NAME="ID_Button"
           VALUE="Test"
    >
    <SCRIPT>
        // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
        // eine im HEAD deklarierte Funktion

        // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
        // geparkt wird, also im HEAD das INPUT-Tag nicht bekannt ist
        ID_Button.onclick = InputButton_Event_onclick_Handler; //ohne ()
    </SCRIPT>
</BODY>

```

Alternative 2:

```

<BODY>
    <INPUT  TYPE=button
           NAME="ID_Button"
           VALUE="Test"
    >
    <SCRIPT>
        // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
        // eine im BODY deklarierte Funktion

        // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
        // geparkt wird, also im HEAD das INPUT-Tag nicht bekannt ist
        function ID_Button.onclick()
        {alert("onclick erkannt");}
    </SCRIPT>
</BODY>

```

Alternative 3:

```

<HEAD>
    <SCRIPT>
        function InputButton_Event_onclick_Handler()
        {alert("onclick erkannt");}
    </SCRIPT>
</HEAD>
<BODY>
    <INPUT  TYPE=button
           NAME="ID_Button"
           VALUE="Test"
    >
    <SCRIPT>
        // anstelle der onclick-Kodierung im Tag erfolgt hier das Überschreiben des Standardeventhandlers durch
        // eine im BODY deklarierte Funktion, die eine Funktion im HEAD aktiviert

        // Überschreiben muss im BODY erfolgen, da der HEAD vor dem BODY-Teil
        // geparkt wird, also im HEAD das INPUT-Tag nicht bekannt ist
        function ID_Button.onclick()
        {InputButton_Event_onclick_Handler();}
    </SCRIPT>
</BODY>

```



Alternative 4 nur beim IE ab IE 4.x:

Das Script-Tag wird um die Attribute FOR und EVENT erweitert:

```
<SCRIPT FOR=objekt_bezeichner EVENT=event_bezeichner .....>
.....
</SCRIPT>
```

objekt_bezeichner	Standard-Objekt laut DOM z.B. window ID des instanziierten Objekt z.B. laut ID-Attribut Kodierung wahlweise mit oder ohne "" bzw. "
event_bezeichner	alle Events mit Präfix on z.B. onload siehe Objekt event Kodierung wahlweise mit oder ohne "" bzw. "

Ansonsten gilt das übliche zum Script-Tag, also auch die Lage im HEAD und/oder BODY.

Beispiel für Kodierung im BODY: Das Objekt muss während der Abarbeitung des HEAD nicht bereits bekannt sein.

```
<BODY>
  <INPUT TYPE=button
    NAME="ID_Button"
    VALUE="Test"
  >
  <SCRIPT FOR= ID_Button
    EVENT=onclick
  >
    alert("onclick erkannt");
  </SCRIPT>
</BODY>
```

Beispiel für Kodierung im HEAD: Das Objekt muss bereits während der Abarbeitung des HEAD bekannt sein.

```
<HTML>
<HEAD>
<SCRIPT FOR=window
  EVENT=onload
>
  var Filter0 = ID_Div.filters[0];
  Filter0.Apply();
  ID_Div.innerHTML= "<IMG SRC='test.jpg' WIDTH=300 HEIGHT=300>";
  Filter0.Play();
</SCRIPT>
</HEAD>
<BODY>
  <DIV ID= "ID_Div"
    STYLE= "position:absolute;width:300;height:300;top:20;left:20;
      filter:revealTrans(transition=12,duration=8)
    "
  >
    Dieser Text wird ueberblendet per Filter revealTrans (nur IE).
    <BR>
    Der Filter wird aktiv mit Laden des Dokumentes in das Fensters.
  </DIV>
</BODY>
</HTML>
```

Events bei sich überlagernden Objekten:

Beispiel: Sollte ein Image mit seiner Erzeugung ein anderes Image überlagern, so ist die Eventübergabe von und an das untere Image unterbrochen: Ereignisse onXXX kommen nicht mehr durch, solange das untere Bild nicht den **Fokus** erhält. Alternativ ist die Style-Eigenschaft z-index zu kodieren und dann der z-index auf den obersten zu setzen.

4.3.2.2.5.1. Eventarten Internet Explorer und Netscape

4.3.2.2.5.1.1. Eventarten der HTML-Tags (Auswahl)

onabort	meldet Abbruch des Ladens	IMG
onblur	meldet Verlassen/Deaktivieren eines HTML-Elementes	BODY BUTTON CHECKBOX FILEUPLOAD FRAMESET Frame-Objekt



		PASSWORD RADIO RESET SELECT SUBMIT TEXT TEXTAREA Window-Objekt
onchange	meldet Veränderung eines HTML-Elementes	FILEUPLOAD SELECT TEXT TEXTAREA
onclick	meldet Mausklick auf HTML-Element	BUTTON CHECKBOX Link per HREF <div> <div>es wird erst gemeldet, dann das</div> <div>Link angesprungen</div> <div>Handler muss return true; bzw.</div> <div>return false; besitzen</div> <div>true --> Aktion des HTML-Elementes</div> <div>wird zugelassen</div> <div>false --> Aktion des HTML-Elements</div> <div>wird nicht zugelassen</div> </div>
		RADIO RESET SUBMIT
onerror	meldet Fehler bei Ausführung der Aktion zum HTML-Element	IMG (Fehler bei Bildladen) Window-Objekt (Fehler bei Ausführung Javascript)
onfocus	meldet Aktivierung des HTML-Elementes	BODY BUTTON CHECKBOX FILEUPLOAD FRAMESET Frame-Objekt PASSWORD RADIO RESET SELECT TEXT TEXTAREA Window-Objekt
onload	meldet komplettes Laden des HTML-Elementes	BODY FRAME FRAMESET IMG Window-Objekt
onmouseout	meldet, dass Mauszeiger nicht auf HTML-Element steht	A AREA Link per HREF
onmouseover	meldet, dass Mauszeiger sich über Element bewegt	A AREA Link per HREF
onreset	meldet gedrückten Reset-Button	FORM <div> <div>Handler muss return true; oder return false; besitzen:</div> <div>true --> Reset wird ausgeführt</div> <div>false --> Reset wird nicht ausgeführt</div> </div>
onselect	meldet, dass der Benutzer einen Textausschnitt markiert hat	



TEXT
TEXTAREA

onsubmit	meldet gedrückten Submit-Button
	FORM
	Handler muss return true; oder return false; besitzen: true --> Submit wird ausgeführt false --> Submit wird nicht ausgeführt bei E-Mail erfolgt automatische Anfrage, ob wirklich gesendet werden soll
onunload	meldet Verlassen des HTML-Elementes
	BODY FRAME FRAMESET Window-Objekt

4.3.2.2.5.1.2. Eventarten des IE und NS (Auswahl)

onabort	Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
	Tag Objekt image IE ab 4.x, NS ab 3.x
onafterprint	Druck eines Fensters oder Frames: Ereignis, das direkt nach dem Druckende ausgelöst wird
	Tag <BODY>, <FRAMESET> Objekt window nur IE ab 5.x
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>, , <TEXTAREA> Objekt alle Objekte mit markierbaren Text nur IE ab 5.x
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>, , <TEXTAREA> Objekt alle Objekte mit markierbaren Text nur IE ab 5.x
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) Einfüge-Operation im Kontextmenü ermöglichen: Handler muss return false; bzw. returnValue=false; liefern
	Tag alle Tags mit markierbarem Text z.B. <A>, <DIV>, , <TEXTAREA> Objekt alle Objekte mit markierbaren Text nur IE ab 5.x
onbeforeprint	Druck eines Fensters oder Frames: Ereignis, das direkt vor dem Druckstart ausgelöst wird
	Tag <BODY>, <FRAMESET> Objekt window nur IE ab 5.x
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> Objekt window nur IE ab 4.x
onblur	Ereignis, das direkt vor dem De-Fokussieren ausgelöst wird
	Tag <FRAME>, <INPUT>, <TEXTAREA> IE ab 4.x, NS ab 3.x
onchange	Ereignis ausgelöst, wenn Inhalt von Eingabefeld Textarea Auswahlliste Radiobox (nur IE) Checkbox (nur IE) durch User verändert wurde und bei Eingabefeld, Textarea und Auswahlliste



	<p>diese de-fokussiert wurden (dagegen bei Radiobox und Checkbox sofort Ereignisauslösung)</p> <p>Handler hat innerhalb <OPTION> eines <SELECT> keine Wirkung, also nur innerhalb <SELECT> kodieren</p> <p>Tag <INPUT>, <SELECT>, <TEXTAREA></p> <p>IE ab 4.x, NS ab 3.x</p>
onclick	<p>Ereignis ausgelöst, wenn Maus das Objekt angeklickt hat: Taste drücken und loslassen</p> <p>im Handler return false; kodieren für Links, Formularelemente und bei IE auch DIV wenn eine Aktion nicht erwünscht ist Bsp.: bei Link wird dieser nicht ausgeführt, obwohl angeklickt</p> <p>Tag <A>, <BODY>, <INPUT>, nur bei IE: <DIV></p> <p>Objekt document, button, submit, reset, checkbox</p> <p>IE ab 4.x, NS ab 3.x</p>
oncontextmenu	<p>Ereignis ausgelöst direkt vor Aufruf des Kontextmenüs mit der rechten Maustaste</p> <p>Unterdrückung, wenn der Eventhandler event.returnValue=false; enthält (mit return false; erfolgt keine Unterdrückung)</p> <p>gilt standardgemäß HTML-seitenweit, also für alle Elemente der Seite, da das Ereignis standardgemäß bis nach oben weitergereicht wird</p> <p>Tag fast alle</p> <p>Objekt analog zu Tag z.B. document, link</p> <p>nur IE ab 4.x</p> <p>Hinweis für NS: Kontextmenü sperren per Ereignishandler für mousedown der rechten Maustaste: nur für Object document muss return false; liefern</p>
oncopy	<p>markierten Text in die Zwischenablage kopieren: Ereignis, das direkt nach der Kopieraktion ausgelöst wird</p> <p>Tag alle die Text definieren, <A></p> <p>Objekt analog zu Tag z.B. link</p> <p>nur IE ab 5.x</p>
oncut	<p>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 Kontextmenü-Eintrages zum Ausschneiden (standardmäßig ist Ausschneiden nicht möglich)</p> <p>Tag alle die Text definieren z.B. <A></p> <p>Objekt analog zu Tag z.B. link</p> <p>nur IE ab 5.x</p>
ondblclick	<p>Ereignis ausgelöst, wenn Doppelklick auf Objekt erfolgt 1 Klick = Maustaste drücken und loslassen bei IE: nur durch linke Maustaste</p> <p>bei NS: durch linke bzw. rechte Maustaste</p> <p>Tag fast alle</p> <p>Objekt link</p> <p>bei IE: nur durch linke Maustaste</p> <p>bei NS: durch linke bzw. rechte Maustaste</p> <p>IE ab 4.x, NS ab 4.x</p>
ondrag	<p>IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen</p> <p>Ereignis ausgelöst, immer wenn HTML-Element verschoben wird, also permanent solange gezogen wird</p> <p>nur verwenden für Erfassung der aktuellen Element-Position Quell- und Ziel-Element müssen gleicher Art sein</p> <p>Eventhandler für Quellobjekt kodieren: nur auf Ereignis reagieren, solange der Bereich des Zielobjektes noch nicht erreicht wurde private Boolean-Variable verwenden: if (false) ... reaktion per dragenter auf true setzen per dragleave auf false initialisieren</p> <p>Tag alle Tags, in denen Text markierbar sind: <DIV>, , </p> <p>Objekt analog zu Tag</p> <p>nur IE ab 5.x</p>
ondragdrop	<p>NS ermöglicht Drag&Drop von Dateien und Verknüpfungen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen</p> <p>Ereignis ausgelöst, wenn eine Datei oder Verknüpfung verschoben wurde</p> <p>Tag <BODY></p>



	Objekt window nur NS ab 4.x IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen Ereignis ausgelöst, wenn HTML-Element verschoben, eventuell abgelegt und somit Drag & Drop nun beendet werden kann aktivieren des Ablegen per dragover Quell- und Ziel-Element müssen gleicher Art sein Eventhandler für Quellobjekt kodieren Tag alle Tags, in denen Text markierbar sind: <DIV>, , Objekt analog zu Tag nur IE ab 5.x
ondragenter	IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen 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 Quell- und Ziel-Element müssen gleicher Art sein Tag alle Tags, in denen Text markierbar sind: <DIV>, , Objekt document und analog zu Tag nur IE ab 5.x
ondragleave	IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen 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 Quell- und Ziel-Element müssen gleicher Art sein Tag alle Tags, in denen Text markierbar sind: <DIV>, , Objekt document und analog zu Tag nur IE ab 5.x
ondragover	IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen 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 abgelegt 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 !) Quell- und Ziel-Element müssen gleicher Art sein Tag alle Tags, in denen Text markierbar sind: <DIV>, , Objekt document und analog zu Tag nur IE ab 5.x
ondragstart	IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen Ereignis ausgelöst, wenn Maus auf Quellobjekt dauergedrückt ist, also die Verschiebung damit beginnen kann Eventhandler für Quellobjekt kodieren Quell- und Ziel-Element müssen gleicher Art sein Tag alle Tags, in denen Text markierbar sind: <DIV>, , Objekt document und analog zu Tag nur IE ab 5.x
ondrop	IE ermöglicht Drag&Drop von HTML-Elementen auch über Fenster per Maus: Click auf Element und Maus gedrückt lassen, dann Element ziehen an gewünschte Position, dann Maus loslassen Ereignis ausgelöst, wenn Quellobjekt auf dem Bereich des Zielobjektes



	<p>abgelegt wird, falls es per dragend zugelassen wird</p> <p>Eventhandler für Zielobjekt kodieren</p> <p>Quell- und Ziel-Element müssen gleicher Art sein</p> <p>Tag alle Tags, in denen Text markierbar sind: <DIV>, , </p> <p>Objekt document und analog zu Tag</p> <p>nur IE ab 5.x</p>
onerror	<p>Ereignis ausgelöst, wenn</p> <ul style="list-style-type: none"> während Ladens eines HTML-Dokumentes ein Syntaxfehler auftritt eines Bildes ein Fehler auftritt Abarbeitung von Scripten ein Runtime-Error auftritt <p>sämtliche Fehlermeldungen unterbinden per</p> <pre>var RetteOnErrorHandler = window.onerror; window.onerror = null;</pre> <p>Eventhandler muss wie folgt kodiert werden:</p> <pre>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 }</pre> <p>pro Fehler ein Aufruf des Eventhandlers</p> <p>--> Folge von Fehlern, also Folge von Aufrufen</p> <p>Die Script-Debugger-Aufforderung ist nur in den Eigenschaften des Browsers abschaltbar</p>
onfilterchange	<p>Tag </p> <p>Objekt window, img</p> <p>IE ab 4.x und NS ab 3.x</p> <p>IE unterstützt Filter-Effekte per CSS: STYLE=".....filter:....."</p> <p>z.B. filter: revealTrans(Transition = 5, Duration = 1.0)</p> <p>Ereignis ausgelöst mit Ablaufende des Filtereffektes</p> <p>Tag <DIV>, , <INPUT>, , <TABLE>, <TD>, <TEXTAREA>, <TH>, <TR></p> <p>Objekt image</p> <p>nur IE ab 4.x</p>
onfocus	<p>Ereignis ausgelöst, wenn HTML-Element fokussiert wird, also das aktuelle wird</p> <p>in Handler keine alert(...)-Anweisung kodieren !!!</p> <p>Tag IE und NS: <BODY>, <DIV>, <INPUT>, <FRAMESET>, <TEXTAREA></p> <p>nur IE: <APPLET>, , , <FORM>, <IFRAME>, , <PRE>, <TABLE>, <TD>, <TH>, <TR>, </p> <p>Objekt window bei IE zusätzlich image</p> <p>IE ab 4.x, NS ab 3.x</p>
onhelp	<p>Ereignis ausgelöst, wenn im Browserfenster die F1-Taste gedrückt wurde</p> <p>Handler muss return true; liefern, wenn eine Reaktion auf die F1-Taste auch wirksam werden soll</p> <p>Tag <BODY>, <FRAMESET></p> <p>Objekt window</p> <p>nur IE ab 4.x</p>
onkeydown	<p>Ereignis ausgelöst</p> <ul style="list-style-type: none"> beim IE nicht für alle Tastenarten beim NS für alle Tastenarten <p>liefert ASCII-Code der Taste nach Eigenschaft</p> <ul style="list-style-type: none"> bei IE .keyCode bei NS .which <p>nach String umwandeln per string_name=fromCharCode(..)</p> <p>NS einmalig, also genau wenn Taste gedrückt wird</p> <p>IE erste Mal, wenn Taste gedrückt</p> <p>periodisch, wenn Dauerdruck der Taste</p> <p>--> siehe Eigenschaft .repeat</p> <p>Besonderheit bei Kombination Steuertaste und andere Taste</p> <p>z.B. Shift-Taste und B-Taste zugleich für Großbuchstabe B</p> <p>Ereignis wird pro Tastendruck aufgerufen, also</p> <ol style="list-style-type: none"> 1. Mal für Shift-Taste 2. Mal für B-Taste <p>ab IE 5.x und NS 4.x:</p> <p>keydown-Ereignis ruft automatisch das keypress-Ereignis auf, aber wenn keydown-Handler return false; bzw. beim IE alternativ event.returnValue=false; liefert, so wird ein kodierter keypress-Handler nicht aktiviert !</p> <p>Ereignisprogrammierung unterscheidet sich nicht nur zwischen den Browsern</p>



	sondern auch innerhalb deren Versionen !!!
Tag	<p>bei IE <BODY>, <DIV>, <FORM>, <INPUT>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TR>, alle Text-Tags wie z.B. , </p> <p>bei NS <A>, , <TEXTAREA></p>
Objekt	document
ab IE 4.x bzw. 5.x und NS 4.x	
onkeypress	<p>wird unmittelbar nach dem keydown-Ereignis ausgelöst allerdings nur für Tastenarten</p> <p>! @ # \$ % ^ & * () _ - + = < [] { } , . / ? \ ' " ` ~</p> <p>0 bis 9</p> <p>a bis z</p> <p>beim IE: egal ob Gross oder Klein, da immer der ASCII-Code des Grossbuchstaben geliefert wird</p> <p>Enter</p> <p>Leertaste</p> <p>und dient dem Erkennen von Tastaturkombinationen, die nicht vom Ereignis keydown abgefangen werden</p> <p>der Aufruf des keypress-Handlers ist nur möglich, wenn der keydown-Handler</p> <p>return true;</p> <p>bzw. beim IE alternativ event.returnValue=true;</p> <p>liefert</p>
	Ereignisprogrammierung unterscheidet sich nicht nur zwischen den Browsern sondern auch innerhalb deren Versionen !!!
Tag	<p>bei IE <BODY>, <DIV>, <FORM>, <INPUT>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TR>, alle Text-Tags wie z.B. , </p> <p>bei NS <A>, , <TEXTAREA></p>
Objekt	document
ab IE 4.x bzw. 5.x und NS 4.x	
onkeyup	<p>Ereignis ausgelöst, wenn gedrückte Taste jeder Art losgelassen wird</p> <p>wird automatisch nach Ereignis keypress ausgelöst</p> <p>Ereignis keypress automatisch nach Ereignis keydown ausgelöst</p> <p>bei NS: besitzt vordefiniertes Schlüsselwort KEYUP für captureEvents also document.captureEvent(Event.KEYUP);</p> <p>Ereignisprogrammierung unterscheidet sich nicht nur zwischen den Browsern sondern auch innerhalb deren Versionen !!!</p>
Tag	<p>bei IE <BODY>, <DIV>, <FORM>, <INPUT>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TR>, alle Text-Tags wie z.B. , </p> <p>bei NS <A>, , <TEXTAREA></p>
Objekt	document
ab IE 4.x bzw. 5.x und NS 4.x	
onload	<p>Ereignis ausgelöst, wenn vollständig geladen wurde</p> <p>HTML-Dokument mit all seinen Elementen jeder Art</p> <p>Framset (innerhalb <FRAMESET> kodieren)</p> <p>Bild</p> <p>DIV</p> <p>bei NS zusätzlich Layer</p> <p>bei IE zusätzlich Applet, Script, Link, IFRAME</p> <p>zu animiertes Bild (Gif-Datei):</p> <p>bei NS: Ereignis load bei jedem Bildwechsel der Animation ausgelöst es kann also für jedes Bild das Ereignis behandelt werden</p> <p>bei IE: Ereignis load bei jeder Animationswiederholung ausgelöst es kann nur pro komplette Animation das Ereignis behandelt werden</p>
Tag	<p><BODY>, <DIV>, <FRAMESET>, </p> <p>bei IE zusätzlich <A>, <APPLET>, <IFRAME>, <SCRIPT></p>
Objekt	window, img
ab IE 4.x und NS 3.x	
onlosecapture	<p>Ereignis ausgelöst, wenn</p> <p>releaseCapture() für mousemove, mouseover und mouseout ausgeführt wurde</p> <p>der User mit der Maus das Browserfenster verlässt</p>
Tag	<p><A>, <APPLET>, <BODY>, <DIV>, <FORM>, <INPUT>, <P>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TR></p>
nur IE 5.x	
onmousedown	<p>Ereignis ausgelöst</p> <p>bei IE mit Drücken irgendeiner Maustaste siehe Event-Eigenschaft .button</p> <p>bei NS mit Drücken der linken oder rechten Maustaste (mittlere nicht !) siehe Event-Eigenschaft .which auch in Verbindung mit Umschalt (Shift) und Alt</p>



	<p>bei IE: return false; bzw. returnValue=false; unterbinden nicht die Ausführung des Eventhandlers also Unterbindung z.B. in click oder contextmenu kodieren</p> <p>bei verschachtelt 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 cancelBubble aktiviert wurde)</p> <p>bei NS: return false; unterbindet immer die Ausführung des Eventhandlers return false bei rechter Maustaste auf Object document wird das Kontextmenü gesperrt</p> <p>Tag <A>, <DIV> bei NS zusätzlich <INPUT TYPE="button"> bei IE zusätzlich <APPLET>, <AREA>, <BODY>, <FORM>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.</p>
onmouseenter	<p>Objekt document ab IE 4.x und NS 4.x Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt</p> <p>Tag unbekannt nur für IE, wahrscheinlich ab 5.02 Empfehlung: mouseover verwenden</p>
onmouseleave	<p>Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt</p> <p>Tag unbekannt nur für IE, wahrscheinlich ab 5.02 Empfehlung: mouseout verwenden</p>
onmousemove	<p>Ereignis bei jeder Mausbewegung ausgelöst Auswertung für den Bereich eines Objektes durch dessen Eventhandler</p> <p>Tag bei NS unbekannt bei IE <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.</p>
onmouseout	<p>Objekt document ab IE 4.x und NS 4.x Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt</p> <p>Tag <A>, <DIV>, bei IE zusätzlich <APPLET>, <AREA>, <BODY>, <FORM>, <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.</p>
onmouseover	<p>Objekt document ab IE 4.x und NS 3.x Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt</p> <p>Tag <A>, <DIV>, bei IE zusätzlich <APPLET>, <AREA>, <BODY>, <FORM>, <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.</p>
onmouseup	<p>Objekt document ab IE 4.x und NS 3.x Ereignis ausgelöst</p> <p>bei IE mit Loslassen irgendeiner Maustaste siehe Event-Eigenschaft .button</p> <p>bei NS mit Loslassen der linken Maustaste (rechte und mittlere nicht !) siehe Event-Eigenschaft .which</p> <p>bei IE: return false; bzw. returnValue=false; unterbinden nicht die Ausführung des Eventhandlers also Unterbindung z.B. in click oder contextmenu kodieren</p> <p>bei verschachtelt 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 cancelBubble aktiviert wurde)</p> <p>bei NS: return false; unterbindet immer die Ausführung des Eventhandlers</p> <p>Tag <A>, <DIV> bei NS zusätzlich <INPUT TYPE="button"> bei IE zusätzlich <APPLET>, <AREA>, <BODY>, <FORM>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.</p> <p>Objekt document</p>



onmove	<p>ab IE 4.x und NS 4.x Ereignis ausgelöst solange ein Fenster verschoben wird Tag keine nur für Objekt window zulässig nur NS ab 4.x</p>
onpaste	<p>Text aus Zwischenablage einfügen: Ereignis, das direkt mit dem Einfügen ausgelöst wird verlangt im Handler event.returnValue = false; für Aktivierung des Kontextmenü-Eintrages zum Einfügen (standardmäßig ist Einfügen nicht möglich) Tag alle Tags mit markierbarem Text z.B. <A>, <DIV>, , <TEXTAREA></p>
onpropertychange	<p>Objekt link nur IE ab 5.x Ereignis, das mit der Veränderung des HTML-Elementes eintritt Abfangen des Ereignisses von verschachtelten Elementen: jedes Elementes für sich: dort je den Eventhandler per onpropertychange kodieren das übergeordnete für alle seine untergeordneten: nur im übergeordneten Element den Handler kodieren Objekt feststellbar durch event.srcElement.name laut NAME-Attribut im HTML-Element bzw. event.srcElement.id laut ID-Attribut im HTML-Element Eigenschaft feststellbar durch event.propertyName Tag <A>, <APPLET>, <AREA>, , <BODY>, <CODE>, <DIV>, <DL>, , <FORM>, <I>, , <INPUT>, , <MAP>, , <OPTION>, <P>, <PRE>, <SCRIPT>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TH>, <TR>, </p>
onreset	<p>nur IE ab 5.x Ereignis VOR dem Rücksetzen eines Formulars ausgelöst Rücksetzen per Reset-Button bzw. Neuladen des Dokumentes dient zum Abfangen der Resetaktion des Users und eventueller Unterbindung des Resets z.B. wegen falscher Ausfüllung des Formulars Reset wird nicht ausgelöst, wenn Handler return false; liefert Tag <FORM> Objekt form ab IE 4.x und NS 3.x</p>
onresize	<p>Ereignis mit ausgelöst bei Größenänderung des Fensters per Maus: Maus auf Fensterrahmen es erscheint das <> Symbol linke Maustaste drücken, gedrückt lassen und ziehen linke Maus loslassen bei IE: mit jeder Veränderung der Größen durch ziehen, also jede Veränderung einzeln behandelbar bei NS: genau für die letzte Größenänderung des Fensters, also mit Loslassen der linken Mausänderung Ereignis muss beim IE abgefangen werden per logischer_window_name.attachEvent('onresize', event_handler); Tag bei NS: keins, da nur für window-Objekt zulässig bei IE: neben dem window-Objekt auch <A>, , <CODE>, <DIV>, <FORM>, <FRAME>, <HR>, <I>, , <INPUT>, , <P>, <PRE>, <SELECT>, , <STYLE>, <TABLE>, <TEXTAREA>, <U>, , Stylesheet Objekt window ab IE 4.x und NS 3.x</p>
onresizeend	<p>Ereignis ausgelöst, wenn Größe einer Markierung verändert wurde Tag <A>, , <BODY>, <CODE>, <DIV>, <FORM>, <FRAME>, <HR>, <I>, , <INPUT>, <P>, <PRE>, <SELECT>, , <TABLE>, <TEXTAREA>, <U> Objekt window, document nur IE ab 5.5</p>
onresizestart	<p>Ereignis ausgelöst mit erster Veränderung der Größe einer Markierung Tag <A>, , <BODY>, <CODE>, <DIV>, <FORM>, <FRAME>, <HR>, <I>, , <INPUT>, <P>, <PRE>, <SELECT>, , <TABLE>, <TEXTAREA>, <U> Objekt window, document nur IE ab 5.5</p>
onscroll	<p>Ereignis ausgelöst, wenn HTML-Element gescrollt wird z.B. HTML-Dokument --> Handler innerhalb <BODY> kodieren Tag <APPLET>, <BODY>, <DIV>, <EMBED>, <MAP>, <TABLE>, <TEXTAREA> nur IE ab 5.5</p>



onselect	Ereignis ausgelöst wenn sich Textmarkierung ändert Tag <code><INPUT TYPE="text"></code> , <code><TEXTAREA></code> bei IE zusätzlich <code><BODY></code> ab IE 4.x und NS 3.x
onstop	Ereignis ausgelöst, wenn Stop-Button des Browsers gedrückt wurde Seite verlassen wird auch durch Browser-Buttons Objekt <code>document</code> nur IE ab 5.x
onsubmit	Ereignis VOR dem Abschieken eines Formulars ausgelöst Abschieken per Submit-Button dient zum Abfangen der Resetaktion des Users und eventueller Unterbindung des Submits z.B. wegen falscher Ausfüllung des Formulars Submit wird nicht ausgelöst, wenn Handler <code>return false</code> ; liefert Tag <code><FORM></code> Objekt <code>form</code> ab IE 4.x und NS 3.x
onunload	Ereignis ausgelöst, wenn das HTML-Dokument verlassen wird durch Schliessen Browserfenster Wechsel zu anderer Seite auch per Browser-Button <code>window.document.open()</code> <code>window.document.write()</code> Tag <code><BODY></code> , <code><FRAMESET></code> Objekt <code>window</code> ab IE 4.x und NS 3.x

4.3.2.5.2. **Eigenschaften im IE und NS (Auswahl)**

Die Eigenschaften gelten nur für die zugehörige Eventart:

z.B. gilt die Eigenschaft

`.button` NUR für ein Mausereignis

`.altKey` NUR für ein Tastaturereignis

oder sind teilweise auf genau eine Ereignisart zugeschnitten

z.B. `.toElement` für `onmouseover`

unterscheiden sich zwischen Internet Explorer und Netscape:

die wenigsten gelten für beide Browser

<code>.altKey</code>	true wenn ALT-Taste gedrückt, sonst false (egal ob linke oder rechte) nur IE ab 4.x
<code>.altLeft</code>	true wenn linke ALT-Taste gedrückt, sonst false nur IE ab 5.5 Hinweis: rechte Alt-Taste (AltGr) ermitteln aus (<code>altKey && (!altLeft)</code>)
<code>.button</code>	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 ($1 + 2 + 3 + 4 = 7$)
<code>.cancelBubble</code>	nur IE ab 4.x true, so Ereignis an das übergeordnete Event weiterleiten, sonst false nur IE ab 4.x
<code>.clientX</code>	horizontale Pixel-Position der Maus gegen über linke obere Ecke (0,0) des Anzeigebereiches im Browserfenster nur IE ab 4.x Hinweis: horizontale Pixel-Position der Maus gegenüber der linken oberen Ecke des HTML-Dokumentes ergibt sich aus <code>event.clientX + document.body.scrollLeft</code>
<code>.clientY</code>	vertikale Pixel-Position der Maus gegen über linke obere Ecke (0,0) des Anzeigebereiches im Browserfenster nur IE ab 4.x Hinweis: vertikale Pixel-Position der Maus gegenüber der linken oberen Ecke des HTML-Dokumentes ergibt sich aus <code>event.clientY + document.body.scrollTop</code>
<code>.ctrlKey</code>	true, so Strg-Taste gedrückt (egal ob linke oder rechte) nur IE ab 4.x
<code>.ctrlLeft</code>	true wenn linke Strg-Taste gedrückt, sonst false nur IE ab 5.5 Hinweis: rechte Strg-Taste ermitteln aus (<code>ctrlKey && (!ctrlLeft)</code>)
<code>.data[]</code>	Feld der per Urls aller per Drag & Drop auf das HTML-Dokument abgelegten Objekte nur NS ab 4.x mit signiertem Script
<code>.dataTransfer</code>	selbst Objekt für Cut & Paste-Operationen für Texte oder Urls nur IE ab 5.x mit folgenden Methoden <code>dataTransfer.setData()</code>



	<code>.dataTransfer.getData()</code> <code>.dataTransfer.clearData()</code>
<code>.fromElement</code>	enthält Zeiger desjenigen Objektes, das das Ereignis onmouseout hat nur IE ab 4.x siehe auch <code>.toElement</code>
<code>.height</code>	Höhe in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde nur NS ab 4.x
<code>.keyCode</code>	ASCII-Code der gedrückten Taste wenn 0 so keine Taste gedrückt Umwandlung in String per <code>String.fromCharCode(TastencodeASCII)</code> nur IE ab 4.x
<code>.layerX</code>	horizontale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis darf kein resize-Event sein neue Breite in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Breite verändert wurde, nur resize-Event nur NS ab 4.x
<code>.layerY</code>	vertikale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis darf kein resize-Event sein neue Höhe in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Höhe verändert wurde, nur resize-Event nur NS ab 4.x
<code>.modifiers</code>	Bitmaske für Zustand der Steuertasten Shift, Alt und Strg dient zur Ermittlung der Steuertaste per vordefinierter Maske durch bitweise UND-Verknüpfung mit der Bitmaske ergibt die Verknüpfung 1, also true, so Steuertaste gedrückt worden vordefinierte Maske für Alt-Taste "Event.ALT_MASK" Strg-Taste "Event.CTRL_MASK" Shift-Taste "Event.SHIFT_MASK" Bsp: <code>return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0);</code> // nicht logisches UND (&&) sondern bitweises UND also &
<code>.offsetX</code>	nur NS ab 4.x horizontale Pixel-Position der Maus im HTML-Element z.B. <DIV>, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberen Ecke (0,0)
<code>.offsetY</code>	nur IE ab 4.x vertikale Pixel-Position der Maus im HTML-Element z.B. <DIV>, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberen Ecke (0,0)
<code>.pageX</code>	nur IE ab 4.x horizontale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
<code>.pageY</code>	nur NS ab 4.x vertikale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
<code>.propertyName</code>	nur NS ab 4.x enthält Bezeichner der geänderten Eigenschaft nur Verwenden als Parameter des onpropertychange-Handler
<code>.repeat</code>	nur IE ab 5.x true, so Taste im Dauerdruck, sonst false
<code>.returnValue</code>	nur IE ab 5.x enthält den Boolean-Rückkercode des Eventhandlers <code>event.returnValue=true;</code> ist identisch <code>return true;</code> <code>event.returnValue=false;</code> ist identisch <code>return false;</code> Eventhändler muss liefern true, um eine Aktion aufgrund des Events zuzulassen false, um eine Aktion aufgrund des Events nicht zuzulassen Anwendung z.B. bei RESET und SUBMIT vom Formular
<code>.screenX</code>	nur IE ab 4.x horizontale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirms IE ab 4.x und NS ab 4.x
<code>.screenY</code>	vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirms IE ab 4.x und NS ab 4.x
<code>.shiftKey</code>	true, so Shift-Taste gedrückt (egal ob linke oder rechte) nur IE ab 4.x
<code>.shiftLeft</code>	true wenn linke Shift-Taste gedrückt, sonst false nur IE ab 5.5 Hinweis: rechte shift-Taste ermitteln aus <code>(shiftKey && (!shiftLeft))</code>
<code>.srcElement</code>	Zeiger auf dasjenige HTML-Element, für das das Ereignis ausgelöst wurde entweder im HTML-Tag definieren per <code>onXXX = ""</code> oder Bezug auf den logischen Namen des HTML-Elementes <code>if (logischer_name == event.srcElement)</code>
<code>.srcFilter</code>	nur IE ab 4.x Zeiger auf das Filter-Objekt, für das das Ereignis ausgelöst wurde nur verwenden innerhalb onfilterchange-Handler nur IE ab 4.x



.target	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 = Ereignis.target)		
	nur NS ab 4.x		
.toElement	enthält Zeiger desjenigen Objektes, das Ereignis onmouseover hat nur IE ab 4.x siehe auch .fromElement		
.type	enthält die Event-Art z.B.	abort	
	IE ab 4.x und NS ab 4.x		
.which	enthält bei gedrückter	Maustaste:	1 für linke Taste 2 für mittlere Taste 3 für rechte Taste
		Tastatur	ASCII-Code der Taste Umwandlung in String per String.fromCharCode(TastenKodeASCII)
	nur NS ab 4.x		
.width	Breite in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde nur NS		
.x	bei IE ab 4.x: horizontale Pixel-Position der Maus bezüglich der linken oberen Ecke (0,0) des Eltern-HTML-Elementes (parent)		
	bei NS ab 4.x: identisch mit .layerX		
.y	bei IE ab 4.x: vertikale Pixel-Position der Maus bezüglich der linken oberen Ecke (0,0) des Eltern-HTML-Elementes (parent)		
	bei NS ab 4.x: identisch mit .layerY		

4.3.2.2.5.3. Methoden im IE und NS

keine

Events können zusätzlich nur durch Methoden anderer Objekte verwaltet werden.

4.3.2.2.5.4. 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 instanziiert, 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).

4.3.2.2.5.4.1. 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()
```

4.3.2.2.5.4.1. 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 Mousrad 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

4.3.2.2.5.4.2. 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()



4.3.2.2.5.4.3. 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 !!!

4.3.2.2.5.4.4. 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 dropfen
    </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 dropfen
    </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 deaktiv 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 deaktiviert 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 dropfen
    </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.innerHTML = 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

4.3.2.2.5.4.5. 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 Mausereignis 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 direkt 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 und 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 Einfügen 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 bewogender 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 Doppelklick 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))
ondatachanged	erzeugt von der Datenquelle wenn Daten geändert wurden in der Datenquelle verwendet bei Filter-Operationen
ondatacomplete	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



ondragend	<p>Drag = Maus gedrückt halten</p> <p>erzeugt am Ende der Dragoperation also wenn Maus losgelassen wird</p> <p>erzeugt immer vom Quellobjekt</p> <p>Hinweis: Zielobjekt erzeugt anschließend ondragleave Event</p>
ondragenter	<p>erzeugt wenn Maus über Objekt gedrückt wurde UND der User das Quell-Objekt bei gedrückter Maustaste zieht</p> <p>erzeugt immer vom Ziel-Objekt wenn Ziel erreicht wurde</p> <p>ist das ERSTE Event des Ziels, also das als Erstes ausgelöste Event</p> <p>erzeugt immer vom Quell-Objekt solange Ziel nicht erreicht wurde</p> <p>wird immer direkt vor dem ondragover Event erzeugt</p>
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
ondragstart	<p>UND immer direkt nach dem ondragenter Event</p> <p>erzeugt vom Quell-Objekt wenn User die Selektion für Drag-Operationen ausführt</p> <p>auch bei Textelement</p> <p>ist ERSTES Event für Drag-und Drop</p> <p>Hinweis: Quell-Objekt nutzt Methode .setData() für Übergabe der Daten</p>
ondrop	<p>erzeugt vom Ziel-Objekt wenn Maustaste losgelassen wird über dem Ziel</p> <p>Drop = Ablegen</p> <p>direkt erzeugt vor ondragleave und ondragend Events.</p>
onend	<p>erzeugt wenn das aktive Element das Ende der aktiven Timeline erreicht inklusive aller Wiederholungen laut .repeatCount</p> <p>bzw. wenn das aktive Element gestoppt wird</p> <p>erzeugt wenn das Elternelement das Ende seiner Timeline erreicht hat</p> <p>und somit des Kindelement ebenfalls endet</p> <p>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</p> <p>siehe Methode .endElement() und Eigenschaft .end</p> <p>siehe onbegin und onrepeat</p> <p>siehe Objekt currTimeState und Behavior .style.time2</p>

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.onerror
}

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 geparkt, also IMG-Tag ausgeführt
                                     // ( anstelle von eval()
                                     // bzw. document.write() )

    ID_IMG.src="";
    ID_IMG.style.display="block";

    ID_IMG.onerror= IMGAltTextAufFehlerMeldung; // ohne ()
}

```



	<pre> function IMGAltTextAufFehlerMeldung () { ID_IMG.alt="Das Bild konnte nicht geladen werden."; return true; } </pre>
	<pre> </SCRIPT> <INPUT TYPE=button VALUE="Klicke zum Bildladen" onclick="Laden()"> <DIV ID="ID_Div"></DIV> </pre>
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	<pre> <BODY> <MARQUEE LOOP=2 onfinish="alert(event.srcElement.id + ' onfinish-Ereignis erkannt')"> Marquee Text </MARQUEE> </BODY> </pre>
onfocus	erzeugt, wenn Objekt den Focus erhält
	Fokus nur erhaltbar nach dem kompletten Laden des Dokumentes
Beispiel für Label	<pre> <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 ZeigerAufLabel=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 ZeigerAufLabel.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"> </pre>
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	<pre> <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> </pre>
onkeypress	erzeugt bei Druck einer alphanumerischen Tastatur-Taste
Beispiel	<pre> <HEAD> </pre>



```

<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
onload           setzt Eigenschaft .contentOverflow auf true
                 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 dem 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.innerHTML +=
                                                '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 den 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 **currentTimeState** 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 Objektbereich 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.innerHTML="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 Mause rad 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 VergrössernVerkleinern()
```



```

    {
        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" SRC="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>
    offsetTop = <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>
    offsetTop = <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>
offsetTop = <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 Timeline 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 onsyncstored

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 Clipboard 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 und 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 einfügen"
        onbeforepaste="EventBeforePaste()"
        onpaste="EventPaste()"
    >
</BODY>

```

onpause

erzeugt wenn das aktive Element auf der aktiven Timeline 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 Herausheben 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 Elementes 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 bewogender 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 aktuellen Spalte
aufgrund Änderung der Daten in der Datenquelle zu Spalte

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>
    &nbsp;&nbsp;&nbsp;Sekunden
    <BR>
    setze seekToFrame:
    <INPUT TYPE="text"
        VALUE=""
        NAME="ID_Input"
        SIZE="3"
    >&nbsp;&nbsp;&nbsp;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

 Art der Formularaktion laut ACTION-Attribut des Formular-Tag
 Senden unterbindbar, wenn Eventhandler false liefert als Rückkehrcode (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>
```

onunload siehe Objekt currTimeState und Behavior .style.time2
 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

4.3.2.2.5.4.6. Eventarten wichtiger Objekte (Auswahl)



a Objekt

onactivate	ondragend	onmousemove
onafterupdate	ondragenter	onmouseout
onbeforeactivate	ondragleave	onmouseover
onbeforecopy	ondragover	onmouseup
onbeforecut	ondragstart	onmousewheel
onbeforedeactivate	ondrop	onmove
onbeforeeditfocus	onerrorupdate	onmoveend
onbeforepaste	onfocus	onmovestart
onbeforeupdate	onfocusin	onpaste
onblur	onfocusout	onpropertychange
onclick	onhelp	onreadystatechange
oncontextmenu	onkeydown	onresize
oncontrolselect	onkeypress	onresizeend
oncopy	onkeyup	onresizestart
oncut	onlosecapture	onselectstart
ondblclick	onmousedown	ontimeerror
ondeactivate	onmouseenter	
ondrag	onmouseleave	

applet Objekt

onactivate	ondeactivate	onmouseup
onbeforeactivate	onfocus	onmousewheel
onbeforecut	onfocusin	onmove
onbeforedeactivate	onfocusout	onmoveend
onbeforeeditfocus	onhelp	onmovestart
onbeforepaste	onkeydown	onpaste
onblur	onkeypress	onpropertychange
oncellchange	onkeyup	onreadystatechange
onclick	onload	onresize
oncontextmenu	onlosecapture	onresizeend
oncontrolselect	onmousedown	onresizestart
oncut	onmouseenter	onrowenter
ondataavailable	onmouseleave	onrowexit
ondatachanged	onmousemove	onrowsdelete
ondatasetcomplete	onmouseout	onrowsinserted
ondblclick	onmouseover	onscroll

area Objekt

onactivate	ondragend	onmouseleave
onbeforeactivate	ondragenter	onmousemove
onbeforecopy	ondragleave	onmouseout
onbeforecut	ondragover	onmouseover
onbeforedeactivate	ondragstart	onmouseup
onbeforeeditfocus	ondrop	onmousewheel
onbeforepaste	onfocus	onmove
onblur	onfocusin	onmoveend
onclick	onfocusout	onmovestart
oncontextmenu	onhelp	onpaste
oncontrolselect	onkeydown	onpropertychange
oncopy	onkeypress	onreadystatechange
oncut	onkeyup	onresizeend
ondblclick	onlosecapture	onresizestart
ondeactivate	onmousedown	onselectstart
ondrag	onmouseenter	ontimeerror

bgsound Objekt

onlayoutcomplete	onmouseleave
onmouseenter	onreadystatechange

body Objekt

onactivate	oncontrolselect	onfilterchange
onafterprint	oncut	onfocusin
onbeforeactivate	ondblclick	onfocusout
onbeforecut	ondeactivate	onkeydown
onbeforedeactivate	ondrag	onkeypress
onbeforeeditfocus	ondragend	onkeyup
onbeforepaste	ondragenter	onload
onbeforeprint	ondragleave	onlosecapture
onbeforeunload	ondragover	onmousedown
onclick	ondragstart	onmouseenter
oncontextmenu	ondrop	onmouseleave



onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove

onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend

onresizestart
onscroll
onselect
onselectstart
onunload

button Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondragenter

ondragleave
ondragover
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

comment Objekt

onpropertychange

onreadystatechange

custom Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag

ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmousecenter

onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onscroll
onselectstart

div Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste Fires
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlayoutcomplete
onlosecapture
onmousedown
onmousecenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onscroll
onselectstart
ontimeerror

document Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate

onbeforeeditfocus
onbeforepaste
onclick
oncontextmenu

oncontrolselect
oncut
ondblclick
ondeactivate



ondrag
ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop
onfocusin
onfocusout
onhelp

onkeydown
onkeypress
onkeyup
onmousedown
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove

onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectionchange
onstop

embed Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate

onfocus
onfocusin
onfocusout
onhelp
onload
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove
onmouseout
onmouseover

onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onscroll

fieldset Objekt

onactivate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

font Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter

ondragleave
ondragover
ondragstart
ondrop
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlayoutcomplete
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

form Objekt

onactivate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect

oncopy
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop

onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave



onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove

onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onreset

onresize
onresizeend
onresizestart
onselectstart
onsubmit
ontimeerror

frame Objekt

onactivate
onafterupdate
onbeforedeactivate
onbeforeupdate
onblur
oncontrolselect

ondeactivate
onerrorupdate
onfocus
onload
onmove
onmoveend

onmovestart
onresize
onresizeend
onresizestart

frameset Objekt

onactivate
onafterprint
onbeforedeactivate
onbeforeprint
onbeforeunload
onblur

oncontrolselect
ondeactivate
onfocus
onload
onmove
onmoveend

onmovestart
onresizeend
onresizestart
onunload

head Objekt

onlayoutcomplete

onreadystatechange

html Objekt

onlayoutcomplete
onmouseenter

onmouseleave
onreadystatechange

html comment Objekt

onlayoutcomplete

iframe Objekt

onactivate
onafterupdate
onbeforedeactivate
onbeforeupdate
onblur
oncontrolselect

ondeactivate
onerrorupdate
onfocus
onload
onmove
onmoveend

onmovestart
onreadystatechange
onresizeend
onresizestart
ontimeerror

img Objekt

onabort
onactivate
onafterupdate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag

ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerror
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onload
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input Objekt

keine Events

input button Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu

oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter
ondragleave
ondragover

ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress



onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove
onmouseout

onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste

onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input checkbox Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmousecenter

onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

input file Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter

ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input hidden Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforeeditfocus
onbeforeupdate
oncontrolselect

ondeactivate
onerrorupdate
onfocus
onlosecapture
onmove
onmoveend

onmovestart
onpropertychange
onreadystatechange
onresizeend
onresizestart
ontimeerror

input image Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter

ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input password Objekt

onactivate
onafterupdate

onbeforeactivate
onbeforecut

onbeforedeactivate
onbeforeeditfocus



onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter
ondragleave
ondragover
ondragstart

ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove

onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input radio Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

input reset Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter

ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart
ontimeerror

input submit Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter
ontimeerror

ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart

input text Objekt

onactivate

onafterupdate

onbeforeactivate



onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onchange
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter
ondragleave

ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove

onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselect
onselectstart
ontimeerror

label Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag

ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter

onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onselectstart

link Objekt

onload

onreadystatechange

map Objekt

onbeforeactivate
onbeforecut
onbeforepaste
onclick
oncut
ondblclick
ondrag
ondragend
ondragenter
ondragleave
ondragover

ondragstart
ondrop
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter

onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onpaste
onpropertychange
onreadystatechange
onscroll
onselectstart

marquee Objekt

onactivate
onafterupdate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onbeforeupdate
onblur
onbounce
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondrag
ondragend
ondragenter

ondragleave
ondragover
ondragstart
ondrop
onerrorupdate
onfilterchange
onfinish
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove

onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresize
onresizeend
onresizestart
onscroll
onselectstart
onstart
ontimeerror



namespace Objekt

onreadystatechange

noframe Objekt

onreadystatechange

noscript Objekt

onreadystatechange

object Objekt

onactivate
 onbeforedeactivate
 onbeforeeditfocus
 onblur
 oncellchange
 onclick
 oncontrolselect
 ondataavailable
 ondatasetchanged
 ondatasetcomplete
 ondblclick
 ondeactivate
 ondrag

ondragend
 ondragenter
 ondragleave
 ondragover
 ondragstart
 ondrop
 onerror
 onfocus
 onkeydown
 onkeypress
 onkeyup
 onlosecapture
 onmove

onmoveend
 onmovestart
 onpropertychange
 onreadystatechange
 onresize
 onresizeend
 onresizestart
 onrowenter
 onrowexit
 onrowsdelete
 onrowsinserted
 onscroll
 onselectstart

option Objekt

onlayoutcomplete
 onlosecapture

onpropertychange
 onreadystatechange

onselectstart
 ontimeerror

popup Objekt

keine Events

scriptObjekt

onload

onpropertychange

onreadystatechange

select Objekt

onactivate
 onafterupdate
 onbeforeactivate
 onbeforecut
 onbeforedeactivate
 onbeforeeditfocus
 onbeforepaste
 onbeforeupdate
 onblur
 onchange
 onclick
 oncontextmenu
 oncontrolselect
 oncut
 ondblclick
 ondeactivate

ondragenter
 ondragleave
 ondragover
 ondrop
 onerrorupdate
 onfocus
 onfocusin
 onfocusout
 onhelp
 onkeydown
 onkeypress
 onkeyup
 onlosecapture
 onmousedown
 onmouseenter
 onmouseleave

onmousemove
 onmouseout
 onmouseover
 onmouseup
 onmousewheel
 onmove
 onmoveend
 onmovestart
 onpaste
 onpropertychange
 onreadystatechange
 onresize
 onresizeend
 onresizestart
 onselectstart

selection Objekt

ontimeerror

span Objekt

onactivate
 onafterupdate
 onbeforeactivate
 onbeforecopy
 onbeforecut
 onbeforedeactivate
 onbeforeeditfocus
 onbeforepaste
 onbeforeupdate
 onblur
 onclick
 oncontextmenu
 oncontrolselect
 oncopy
 oncut
 ondblclick
 ondeactivate
 ondrag

ondragend
 ondragenter
 ondragleave
 ondragover
 ondragstart
 ondrop
 onerrorupdate
 onfilterchange
 onfocus
 onfocusin
 onfocusout
 onhelp
 onkeydown
 onkeypress
 onkeyup
 onlosecapture
 onmousedown
 onmousecenter

onmouseleave
 onmousemove
 onmouseout
 onmouseover
 onmouseup
 onmousewheel
 onmove
 onmoveend
 onmovestart
 onpaste
 onpropertychange
 onreadystatechange
 onresize
 onresizeend
 onresizestart
 onselectstart
 ontimeerror



style Objekt

onerror onreadystatechange

table Objekt

onactivate	ondragover	onmouseover
onbeforeactivate	ondragstart	onmouseup
onbeforecut	ondrop	onmousewheel
onbeforedeactivate	onfilterchange	onmove
onbeforeeditfocus	onfocus	onmoveend
onbeforepaste	onfocusin	onmovestart
onblur	onfocusout	onpaste
onclick	onhelp	onpropertychange
oncontextmenu	onkeydown	onreadystatechange
oncontrolselect	onkeypress	onresize
oncut	onkeyup	onresizeend
ondblclick	onlosecapture	onresizestart
ondeactivate	onmousedown	onscroll
ondrag	onmouseenter	onselectstart
ondragend	onmouseleave	ontimeerror
ondragenter	onmousemove	
ondragleave	onmouseout	

table.caption Objekt

onactivate	ondragenter	onmousemove
onbeforeactivate	ondragleave	onmouseout
onbeforecopy	ondragover	onmouseover
onbeforecut	ondragstart	onmouseup
onbeforedeactivate	ondrop	onmousewheel
onbeforepaste	onfocus	onmove
onblur	onfocusin	onmoveend
onclick	onfocusout	onmovestart
oncontextmenu	onhelp	onpaste
oncontrolselect	onkeydown	onpropertychange
oncopy	onkeypress	onreadystatechange
oncut	onkeyup	onresizeend
ondblclick	onlosecapture	onresizestart
ondeactivate	onmousedown	onselectstart
ondrag	onmouseenter	ontimeerror
ondragend	onmouseleave	

table.col Objekt

keine

table.colGroup Objekt

onreadystatechange

table.tBody Objekt

onactivate	ondragleave	onmousemove
onbeforeactivate	ondragover	onmouseout
onbeforecut	ondragstart	onmouseover
onbeforedeactivate	ondrop	onmouseup
onbeforepaste	onfocus	onmousewheel
onblur	onfocusin	onmove
onclick	onfocusout	onmoveend
oncontextmenu	onhelp	onmovestart
oncontrolselect	onkeydown	onpaste
oncut	onkeypress	onpropertychange
ondblclick	onkeyup	onreadystatechange
ondeactivate	onlosecapture	onresizeend
ondrag	onmousedown	onresizestart
ondragend	onmouseenter	onselectstart
ondragenter	onmouseleave	ontimeerror

table.tFoot Objekt

onactivate	oncut	onkeydown
onbeforeactivate	ondblclick	onkeypress
onbeforecut	ondeactivate	onkeyup
onbeforedeactivate	ondragenter	onlosecapture
onbeforepaste	ondragstart	onmousedown
onblur	onfocus	onmouseenter
onclick	onfocusin	onmouseleave
oncontextmenu	onfocusout	onmousemove
oncontrolselect	onhelp	onmouseout



onmouseover	onmovestart
onmouseup	onpaste
onmousewheel	onpropertychange
onmove	onreadystatechange
onmoveend	onresizeend

onresizestart
onselectstart
ontimeerror

table.tHead Objekt

onactivate
onbeforeactivate
onbeforecut
onbeforedeactivate
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncut
ondblclick
ondeactivate
ondragenter
ondragstart

onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave
onmousemove
onmouseout
onmouseover

onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

table.tr Objekt

onactivate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

table.tr.td Objekt

onactivate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforeeditfocus
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick
ondeactivate
ondrag
ondragend

ondragenter
ondragleave
ondragover
ondragstart
ondrop
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown
onmouseenter
onmouseleave

onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange
onresizeend
onresizestart
onselectstart
ontimeerror

table.tr.th Objekt

onactivate
onbeforeactivate
onbeforecopy
onbeforecut
onbeforedeactivate
onbeforepaste
onblur
onclick
oncontextmenu
oncontrolselect
oncopy
oncut
ondblclick

ondeactivate
ondragenter
ondragstart
onfilterchange
onfocus
onfocusin
onfocusout
onhelp
onkeydown
onkeypress
onkeyup
onlosecapture
onmousedown

onmouseenter
onmouseleave
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onmove
onmoveend
onmovestart
onpaste
onpropertychange
onreadystatechange



onresizeend	onselectstart
onresizestart	ontimeerror

textarea Objekt

onactivate	ondragenter	onmouseout
onafterupdate	ondragleave	onmouseover
onbeforeactivate	ondragover	onmouseup
onbeforecopy	ondragstart	onmousewheel
onbeforecut	ondrop	onmove
onbeforedeactivate	onerrorupdate	onmoveend
onbeforeeditfocus	onfilterchange	onmovestart
onbeforepaste	onfocus	onpaste
onbeforeupdate	onfocusin	onpropertychange
onblur	onfocusout	onreadystatechange
onchange	onhelp	onresize
onclick	onkeydown	onresizeend
oncontextmenu	onkeypress	onresizestart
oncontrolselect	onkeyup	onscroll
oncut	onlosecapture	onselect
ondblclick	onmousedown	onselectstart
ondeactivate	onmouseenter	ontimeerror
ondrag	onmouseleave	
ondragend	onmousemove	

textrange Objekt

keine Events

var Objekt

onactivate	ondragleave	onmouseout
onbeforeactivate	ondragover	onmouseover
onbeforecut	ondragstart	onmouseup
onbeforedeactivate	ondrop	onmousewheel
onbeforeeditfocus	onfocus	onmove
onbeforepaste	onfocusin	onmoveend
onblur	onfocusout	onmovestart
onclick	onhelp	onpaste
oncontextmenu	onkeydown	onpropertychange
oncontrolselect	onkeypress	onreadystatechange
oncut	onkeyup	onresize
ondblclick	onlosecapture	onresizeend
ondeactivate	onmousedown	onresizestart
ondrag	onmouseenter	onselectstart
ondragend	onmouseleave	ontimeerror
ondragenter	onmousemove	

window Objekt

onactivate	ondeactivate	onmovestart
onafterprint	onerror	onresize
onbeforedeactivate	onfocus	onresizeend
onbeforeprint	onhelp	onresizestart
onbeforeunload	onload	onscroll
onblur	onmove	onunload
oncontrolselect	onmoveend	

xml Objekt

ondataavailable	onreadystatechange	onrowsdelete
ondatasetchanged	onrowenter	onrowsinserted
ondatasetcomplete	onrowexit	

4.3.2.2.5.5. Event-Behandlung beim Internet Explorer bzw. Netscape**4.3.2.2.5.5.1. 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 Tatsache nötig, dass der Netscape im Gegensatz zum Internet Explorer nicht standardgemäß über integrierte Komponenten des Betriebssystems verfügt.



Eventcapturing kann per Programmierung 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 das Ereignis nicht verarbeiten kann. Es besteht reichlich
Programmierungsaufwand
Der Eventhandler muss dem Ereignis als Parameter übergeben bekommen.

4.3.2.2.5.5.2. 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	<pre> if (document.event) { if (.document.event.eigenschaft) } </pre>
	<p>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)</p>
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 Parameters 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 Ereignisses 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;

```

4.3.2.2.5.5.3. Eventbehandlung im Netscape

Die Standard-Eventbehandlung wird beim Netscape immer vom window-Objekt selbst realisiert. Damit landen alle Events immer beim obersten Objekt. Eine davon abweichende Eventbehandlung ist zu programmieren.

4.3.2.2.5.5.3.1. Event des Netscape einer Nicht-Standardbehandlung unterziehen

4.3.2.2.5.5.3.1.1. Event und Eventhandler dem Objekt window zuordnen (captureEvents(event_liste))

Diese Methode unterbindet die Ereignisregistrierung des Browsers. Damit MUSS das Ereignis von einem Eventhandler verarbeitet werden. Der Eventhandler kann mehrere Ereignisse verarbeiten.

```
window.captureEvents(Event.event_schluessselwort_liste);
```

event_schluessselwort_liste: Folge von Eventbezeichnern mit Trennung durch | also logisches Oder

Bsp. für Eventbezeichner

<u>onXXX</u>	<u>Event.XXX</u>
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS
onload	Event.LOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER
onkeydown	Event.KEYDOWN
onkeypress	Event.KEYPRESS
onkeyup	Event.KEYUP
onresize	Event.RESIZE

```

function MouseDownHandler(Ereignis)
{..}

```

```
window.captureEvents(Event.MOUSEDOWN);
```

```
window.onmouseover=MouseDownHandler;    // Achtung: nicht () kodieren !!
```

Wenn das Ereignis nur einmal bearbeitet und danach wieder der Standardbehandlung zugeordnet werden soll, so muss die Funktion am Ende .releaseEvents() zum Ereignis aufrufen.

Wenn das Ereignis nicht durch nachgelagerte Eventhandler in der Handlerhierarchie bearbeitet werden soll, so muss die Funktion mit return false; enden (sonst return true;).

4.3.2.2.5.5.3.1.2. Event entlang der Eventhierarchie weiterreichen

Es besteht die Möglichkeit, dass ein privater Eventhandler für die Weitergabe des Events die Standardhierarchie benutzt oder einen ausgewählten Eventhandler aufruft.



4.3.2.2.5.3.1.2.1. Event entlang der Nicht-Standard- Eventhierarchie weiterreichen (handleEvent(event_objekt))

Diese Methode ruft den aktuell dem Ereignis laut Parameter event_objekt zugeordneten Eventhandler auf.

Diese Methode wird also innerhalb des Programmcodes eines Eventhandlers verwendet, der das Ereignis von einem anderen Eventhandler als Argument bekommen hat.

4.3.2.2.5.3.1.2.2. Event entlang der Standard-Eventhierarchie weiterreichen (routeEvent(ereignis_objekt))

Diese Methode aktiviert das Weiterreichen des Events zum in der Eventhierarchie untergeordneten Eventhandler.

4.3.2.2.5.3.2. Event des Netscape von Nicht-Standardbehandlung wieder der Standardbehandlung unterziehen

4.3.2.2.5.3.2.1. Standard-Eventhandler dem Objekt window zuordnen (releaseEvents(event_liste))

Diese Methode aktiviert die Ereignisregistrierung durch den Browser, ist also der Gegensatz zu .captureEvents(). Es dürfen nur Ereignisse releast werden, die auch gecaptured wurden.

```
window.releaseEvents(Event.event_schluessselwort_liste);
```

event_schluessselwort_liste: Folge von Eventbezeichnern mit Trennung durch | also logisches Oder

Bsp. für Eventbezeichner

<u>onXXX</u>	<u>Event.XXX</u>
onblur	Event.BLUR
ondragdrop	Event.DRAGDROP
onerror	Event.ERROR
onfocus	Event.FOCUS
onload	Event.LOAD
onmove	Event.MOVE
onresize	Event.RESIZE
onunload	Event.UNLOAD
onmouseover	Event.MOUSEOVER
onkeydown	Event.KEYDOWN
onkeypress	Event.KEYPRESS
onkeyup	Event.KEYUP
onresize	Event.RESIZE

Beispiel:

```
function EinmaligerMouseDownHandler()
{
    .....
    window.releaseEvents(Event.MOUSEDOWN);
}
```

```
window.captureEvents(Event.MOUSEDOWN);
```

```
window.onmouseover=EinmaligerMouseDownHandler; // Achtung: nicht () kodieren !!
```

Wenn das Ereignis nicht durch nachgelagerte Eventhandler in der Handlerhierarchie bearbeitet werden soll, so muss die Funktion mit return false; enden (sonst return true;).

4.3.2.2.5.3.2.2. Event entlang der Standard-Eventhierarchie weiterreichen (routeEvent(ereignis_objekt))

Diese Methode aktiviert das Weiterreichen des Events zum in der Eventhierarchie untergeordneten Eventhandler.

4.3.2.2.5.3.3. Eventbehandlung durch eine Fremdseite mit signiertem Script

Eventbehandlung des Netscape für eine Seite mit Frame, in dem ein fremdes HTML-Dokument angezeigt wird:

Es ist möglich, dass die fremde Seite die Ereigniskontrolle derjenigen Seite übernimmt, die das Frame enthält, in dem die Fremdseite angezeigt wird. Dieser Trojanereffekt ist nur dann realisierbar, wenn die Fremdseite ein signiertes Script enthält: Dieses Script muss sich beim Eigentümer der Seite, die den Frame besitzt, das Recht (Privileg) zur Übernahme der Ereigniskontrolle beschaffen. Dazu wird der User, in dessen Browser die Seite mit dem Frame läuft, befragt. Der Netscape hat einen Privileg-Manger integriert.

Hinweis: Beim Internet Explorer wird nur ActiveX (Betriebssystem-Schnittstelle) verwendet. Privilegien und signierte Script gibt es nicht. ActiveX wird vorallem bei der Integration von Fremderweiterungen der Browserfähigkeiten (beim NS als Plugin bezeichnet) benutzt.

4.3.2.2.5.3.3.1. Beschaffung der Rechte "UniversalBrowserWrite" bzw. "UniversalBrowserWrite"

per Privilegmanger

Die fremde Seite kann z.B. folgenden Code besitzen:

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite");
window.enableExternalCapture();
window.captureEvents(Event.CLICK | Event.MOUSEDOWN);
```

Bei Ausführung des Codes wird der Manager aktiv, um die Rechtegenehmigung durch den



User zu beschaffen, da dieser Code (Script) vom User signiert sein muss.
 Der Netscape-Privilegmanager sichert ab, dass der Browseruser gefragt wird, ob er eine Privilegänderung gegenüber den Standard-Privilegien dulden will.
 Für ein signiertes Script muss das Privileg "UniversalBrowserWrite" bzw. "UniversalBrowserWrite" vom Privilegmanager angefordert sein.

4.3.2.2.5.3.3.2. *Eventbehandlung durch Fremde Seite aktivieren (enableExternalCapture())*

Diese Methode aktiviert die Kontrolle einer Fremdseite, die in einem Frame dsrgestellt wird, auf die Seite, die den Frame inne hat. Die Ereignissüberwachung in einem Fenster wird somit aktiv.

Diese Methode benötigt ein signiertes Script.

Das Event muss mit captureEvents() dem Window-Objekt zugeordnet werden.

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

4.3.2.2.5.3.3.3. *Eventbehandlung durch Fremde Seite deaktivieren (disableExternalCapture())*

Diese Methode deaktiviert die Kontrolle einer Fremdseite, die in einem Frame dsrgestellt wird, auf die Seite, die den Frame inne hat. Die Ereignissüberwachung in einem Fenster wird somit aktiv.

Diese Methode benötigt ein signiertes Script.

Das Event muss mit captureEvents() dem Window-Objekt zugeordnet werden.

Achtung: Anzeige der fremden Seite in einem Frame ist rechtlich nur mit Einverständnis des Herstellers der Fremdseite zulässig !

4.3.2.2.5.3.4. *Beispiel*

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 Paramaters 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.

```
<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript">
<!--
    function MouseDownFuerInputButton(Ereignis)
    { }

    function OnMouseDownEventWeiterreichen(Ereignis)
    {window.routeEvent(Ereignis);} // In der Hierarchie weiterreichen

    window.captureEvents(Event.MOUSEDOWN);

    window.onmousedown= OnMouseDownEventWeiterreichen;
                        // Achtung: ohne () kodieren!!

// -->
</SCRIPT>
</HEAD>
<BODY>
    <FORM>
        <INPUT  TYPE=button
                        VALUE="Weiterreichen"
                        onmousedown= "MouseDownFuerInputButton()"
        >
    </FORM>
</BODY>
</HTML>
```

Ablauf: Head-Teil: Der Script-Teil im Head wird zuerst abgearbeitet.

Body-Teil:

Es wird auf das Input-Button gedrückt.
 Wegen captureEvents() wird das Mausereignis ZUERST von
 OnMouseDownEventWeiterreichen
 bearbeitet. Damit wird auf das Weiterleiten aktiviert.
 Das untergeordnete Element ist der Input-Button. Damit wird
 MouseDownFuerInputButton
 aktiviert.

Hinweis: Dieses Beispiel hinkt, da die Standardbehandlung letztendlich das gleiche bewirkt



aber auf anderen Wegen.

4.3.2.2.5.5.4. 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 Paramaters 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.

4.3.2.2.5.5.4.1. 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);
```

4.3.2.2.5.5.4.2. Event von Nicht-Standardbehandlung wieder der Standardbehandlung

unterziehen (detachEvent(event_objekt))

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

4.3.2.2.5.5.4.3. Ereignisbehandlung für mouseover und mouseout ein-bzw. ausschalten

(.releaseCapture() und .setCapture())

ausschalten: `releaseCapture()`
einschalten: `setCapture()`

4.3.2.2.5.5.4.4. Event bei Behavior (Verhaltensweise)

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

4.3.2.2.5.5.5. Fehlerbehandlung per onerror

4.3.2.2.5.5.5.1. onerror-Standard abschalten

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

4.3.2.2.5.5.5.2. 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.onerror;
}

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 geparkt, 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>

```

4.3.2.2.5.5.6. Eventbehandlung in einem Formular des IE und NS

Es wird das Click-Event abgefangen und mit einer Funktion bearbeitet.

Das Formular wird bei erkannter Veränderung

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE ="Javascript">
<!--
    // Browser feststellen
    var ns = document.layers ? true : false;
    var ie = document.all ? true : false;

    if (ns)
    {
        // nur für NS: Eventart CLICK wird abgefangen
        window.captureEvents(Event.ONCLICK);
    }

    // soll das Fenster selbst den Click-Eventhandler 1 bekommen, so hier kodieren
    // window.onclick=click_auswertung_1;
    // WICHTIG ist, dass das Fenster auch das Event WEITERLEITET !! nach unten

    function click_auswertung_1(Ereignis)
    {
        // folgende alternative Varianten sind möglich

        // nur für NS:
        // in der Objekt-Hierarchie weiterleiten
        if (ns)
        {routeEvent(Ereignis);}

        // für NS und IE:
        // nichts tun
        return true;

        // im aktuellen Fenster und dessen Dokument das Formular anwählen
        // und dem Eventhandler des 2. Buttons das Ereignis übergeben,
        // wobei der Eventhandler des 2. Buttons dieses Ereignis verarbeiten muss
        self.document.logischer_form_name.logischer_button_name_2. handleEvent(Ereignis);

        // die Auswertung hier kodieren --> bitte kein alert(); !!!
        if (ie)
        {
            Ereignis=event.button;
            // hier das Maus-Ereignis allgemein erfasst
            //          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)
            //          nur Internet Explorer ab 4.x
            .....
        }
        else
        {
            if (ns)
            {.....}
        }
    }

    function click_auswertung_2(Ereignis)
    { // analog zu click_auswertung_1}

```



```
// -->
</SCRIPT>
</HEAD>
<BODY ....>
    <FORM NAME="logischer_form_name">
        <INPUT          TYPE=button
                        NAME="logischer_button_name_1"
                        onclick="click_auswertung_1();"
        >
        <INPUT          TYPE=button
                        NAME="logischer_button_name_2"
                        onclick="click_auswertung_2();"
        >
    </FORM>
</BODY>
</HTML>
```

4.3.2.2.5.5.7. Eventbehandlung bei Drag & Drop

4.3.2.2.5.5.7.1. Eventbehandlung bei Drag & Drop eines HTML-Elementes beim Internet Explorer

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 !

4.3.2.2.5.5.7.1.1. 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 erreicht wurde private Boolean-Variable verwenden: if (false) ... reaktion per dragenter auf true setzen per dragleave auf false initialisieren
ondragend	Tag alle Tags, in denen Text markierbar sind: <DIV>, , 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
ondragenter	Tag alle Tags, in denen Text markierbar sind: <DIV>, , 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
ondragleave	Tag alle Tags, in denen Text markierbar sind: <DIV>, , 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
ondragover	Tag alle Tags, in denen Text markierbar sind: <DIV>, , 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 abgelegt 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 !)
ondragstart	Tag alle Tags, in denen Text markierbar sind: <DIV>, , Ereignis ausgelöst, wenn Maus auf Quellobjekt dauergedrückt ist, also die Verschiebung damit beginnen kann Eventhandler für Quellobjekt kodieren
ondrop	Tag alle Tags, in denen Text markierbar sind: <DIV>, , 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>, ,



4.3.2.2.5.5.7.1.2. 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>

```

4.3.2.2.5.5.7.2. Eventbehandlung bei Drag & Drop von Dateien und Verknüpfungen beim Netscape ab 4.x

NS ermöglicht Drag&Drop von Dateien und Verknüpfungen

auch über Fenster

per Maus: linker Maustaste auf das Datei bzw. Verknüpfung niederdrücken
und gedrückt lassen,
dann Element ziehen an gewünschte Position
dann linke Maustaste loslassen

Eventart: dragdrop

Ereignis ausgelöst, wenn eine Datei oder Verknüpfung verschoben wurde
Tag <BODY>

4.3.2.2.5.5.8. Tastatur-Eventbehandlung des IE und NS

Die Programmierung der Tastatur-Events unterscheidet sich nicht nur zwischen den Browsern IE und Netscape, sondern auch noch innerhalb deren Versionsnummern. Dieser Umstand zeugt vom divergierenden Ansatz der Browserhersteller, welcher im krassen Widerspruch zu einer Programmierungsfreundlichkeit steht.

Die bessere Konformität zur konsequent-objektorientierten Umsetzung des HTML-Dokumentes, seiner Elemente und Ereignisse hat der Internet Explorer, der damit ein weiteres Spektrum an Möglichkeiten zur Programmierung unter Javascript bietet, obwohl Microsoft weiterhin Visual Basic-Script als Komponente des Windows Script Host (Betriebssystem-Komponente) favorisiert und eine Dokumentierung der Javascript-Komponente (inklusive Debugger-Freeware-Tool zum aktuellen IE) konsequent vernachlässigt.

4.3.2.2.5.5.8.1. Tastatur-Eventbehandlung beim Internet Explorer

4.3.2.2.5.5.8.1.1. Tastatur-Eventarten

keydown Ereignis ausgelöst nicht für alle Tastenarten (siehe weiter unten)
liefert ASCII-Code der Taste nach Eigenschaft .keyCode
nach String umwandeln per string_name=fromCharCode(..)
erste Mal, wenn Taste gedrückt
periodisch, wenn Dauerdruck der Taste
--> siehe Eigenschaft .repeat
Besonderheit bei Kombination Steuertaste und andere Taste
z.B. Shift-Taste und B-Taste zugleich für Großbuchstabe B
Ereignis wird pro Tastendruck aufgerufen, also
1. Mal für Shift-Taste
2. Mal für B-Taste
ab IE 5.x: keydown-Ereignis ruft automatisch das keypress-Ereignis auf, aber
wenn keydown-Handler return false; bzw. beim IE
alternativ event.returnValue=false; liefert, so wird ein
kodierter keypress-Handler nicht aktiviert !
Tag <BODY>, <DIV>, <FORM>, <INPUT>, <SELECT>, ,
<TABLE>, <TD>, <TEXTAREA>, <TR>,
alle Text-Tags wie z.B. ,
keypress wird unmittelbar nach dem keydown-Ereignis ausgelöst allerdings nur für
Tastenarten
! @ # \$ % ^ & * () _ - + = < [] { } , . / ? \ | ' " ` ~
0 bis 9
a bis z
beim IE: egal ob Gross oder Klein, da immer der ASCII-Code des
Grossbuchstaben geliefert wird
Enter
Leertaste
und dient dem Erkennen von Tastaturkombinationen,
die nicht vom Ereignis keydown abgefangen werden
der Aufruf des keypress-Handlers ist **nur** möglich,
wenn der keydown-Handler



bzw. beim IE alternativ `return true;`
`event.returnValue=true;`
 liefert
 Tag `<BODY>`, `<DIV>`, `<FORM>`, `<INPUT>`, `<SELECT>`, ``,
`<TABLE>`, `<TD>`, `<TEXTAREA>`, `<TR>`,
 alle Text-Tags wie z.B. ``, ``
 keyup Ereignis ausgelöst, wenn gedrückte Taste jeder Art losgelassen wird
 wird automatisch nach Ereignis keypress ausgelöst
 Ereignis keypress automatisch nach Ereignis keydown ausgelöst
 Tag `<BODY>`, `<DIV>`, `<FORM>`, `<INPUT>`, `<SELECT>`, ``,
`<TABLE>`, `<TD>`, `<TEXTAREA>`, `<TR>`,
 alle Text-Tags wie z.B. ``, ``

4.3.2.2.5.8.1.2. Tastatur-Eigenschaft

4.3.2.2.5.8.1.2.1. Alle Tasten (.keyCode und .repeat)

.keyCode ASCII-Code der gedrückten Taste
 wenn 0 so keine Taste gedrückt
 Umwandlung in String per `String.fromCharCode(TastenKodeASCII)`
 .repeat true, so Taste im Dauerdruck, sonst false

4.3.2.2.5.8.1.2.2. Steuertasten Alt, Strg (Ctrl) und Umschalt (Shift) (.xxxKey und .xxxLeft)

.altKey true wenn ALT-Taste gedrückt, sonst false (egal ob linke oder rechte)
 .altLeft true wenn linke ALT-Taste gedrückt, sonst false
 Hinweis: rechte Alt-Taste (AltGr) ermitteln aus
`altKey AND NOT altLeft`

.ctrlKey true, so Strg-Taste gedrückt (egal ob linke oder rechte)
 .ctrlLeft true wenn linke Strg-Taste gedrückt, sonst false
 Hinweis: rechte Strg-Taste ermitteln aus
`ctrlKey AND NOT ctrlLeft`

.shiftKey true, so shift-Taste gedrückt (egal ob linke oder rechte)
 .shiftLeft true wenn linke shift-Taste gedrückt, sonst false
 Hinweis: rechte shift-Taste ermitteln aus
`shiftKey AND NOT shiftLeft`

4.3.2.2.5.8.1.2.3. Eventhandler-Eigenschaften (.returnValue)

.returnValue enthält den Boolean-Rückkercode des Eventhandlers
`event.returnValue=true;` ist identisch `return true;`
`event.returnValue=false;` ist identisch `return false;`
 Eventhändler muss liefern
 true, um eine Aktion aufgrund des Events zuzulassen
 false, um eine Aktion aufgrund des Events nicht zuzulassen
 Anwendung z.B. bei RESET und SUBMIT vom Formular

4.3.2.2.5.8.1.2.4. HTML-Element-Event-Eigenschaft (.srcElement)

.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.srcElement)`

4.3.2.2.5.8.1.2.5. Eventart-Eigenschaft (.type)

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

4.3.2.2.5.8.1.3. Tastatur-Ereignis-Folge onkeydown und onkeypress

Standard: erst `onkeydown`
 dann `onkeypress`

modifiziert: wenn Eventhandler für `keydown` `return false;` oder `event.returnValue=false;`
 enthält, so wird der Eventhandler für `keypress` nicht aktiviert

4.3.2.2.5.8.1.3.1. Tastatur-Ereignis onkeydown

onkeydown ausgelöst bei den Tasten
 ab IE 4.x `!@#$%^&*()_-+=<[]{}.,/?\|'\"'~`
 0 bis 9
 a bis z
 egal ob Gross oder Klein, da immer der ASCII-Code des Grossbuchstaben geliefert wird
 Cursor-Tasten
 Einfg
 Ende
 Entf
 ESC
 F1 bis F12



Leertaste
 Pos1
 Umschalt (Shift)
 Tab
 ab IE 5.x zusätzlich Bild auf und ab
 Rück
 Umschalt+Tab (Shift+Tab)

Die Enter-Taste wird hier nicht geliefert !

liefert den ASCII-Code in die Event-Eigenschaft

.keyCode ASCII-Code der gedrückten Taste
 wenn 0 so keine Taste gedrückt
 wobei bei Buchstaben nicht unterschieden wird, ob gross oder klein,
 da immer der ASCII-Code des Grossbuchstaben geliefert wird
 Umwandlung in String per String.fromCharCode(TastenKodeASCII)

liefert Dauerdruckzustand in die Eventeigenschaft

.repeat true, so Taste im Dauerdruck, sonst false

folgende Tasten werden auch vom Ereignis keypress geliefert:

ab IE 4.x ! @ # \$ % ^ & * () _ - + = < [] { } , . / ? \ | ' " ` ~
 0 bis 9
 a bis z
 egal ob Gross oder Klein, da immer der ASCII-Code des Grossbuchstaben
 geliefert wird
 Enter
 Leertaste

folgende Tasten werden nicht vom Ereignis onkeypress geliefert:

Cursor-Tasten
 Einfg
 Ende
 Entf
 ESC
 F1 bis F12
 Pos1
 Umschalt (Shift)
 Tab
 Bild auf und ab
 Rück
 Umschalt+Tab (Shift+Tab)

Zusatzanalyse möglich per Event-Eigenschaften

.altKey true wenn ALT-Taste gedrückt, sonst false (egal ob linke oder rechte)
 .altLeft true wenn linke ALT-Taste gedrückt, sonst false
 Hinweis: rechte Alt-Taste (AltGr) ermitteln aus
 altKey AND NOT altLeft
 .ctrlKey true, so Strg-Taste gedrückt (egal ob linke oder rechte)
 .ctrlLeft true wenn linke Strg-Taste gedrückt, sonst false
 Hinweis: rechte Strg-Taste ermitteln aus
 ctrlKey AND NOT ctrlLeft
 .shiftKey true, so shift-Taste gedrückt (egal ob linke oder rechte)
 .shiftLeft true wenn linke shift-Taste gedrückt, sonst false
 Hinweis: rechte shift-Taste ermitteln aus
 shiftKey AND NOT shiftLeft

Unterbindung des Aufrufes des keypress-Handlers:

keydown-Handler muss return false; bzw. event.returnValue=false; liefern

4.3.2.2.5.8.1.3.2. Tastatur-Ereignis onkeypress

wird unmittelbar nach dem onkeydown-Ereignis ausgelöst und dient dem Erkennen von Tastaturkombinationen,
 die nicht vom Ereignis keydown abgefangen werden

der Aufruf des keypress-Handlers ist **nur** möglich,

wenn der keydown-Handler return true; bzw. event.returnValue=true; liefert

wird ausgelöst bei den Tasten

ab IE 4.x ! @ # \$ % ^ & * () _ - + = < [] { } , . / ? \ | ' " ` ~
 0 bis 9
 a bis z
 egal ob Gross oder Klein, da immer der ASCII-Code des Grossbuchstaben
 geliefert wird
 Enter
 Leertaste



folgende Tasten sind hier nicht auslesbar (nur per keydown):

Cursor-Tasten
Einfüg
Ende
Entf
ESC
F1 bis F12
Pos1
Umschalt (Shift)
Tab
Bild auf und ab
Rück
Umschalt+Tab (Shift+Tab)

bereits mit dem Ereignis keydown sind abfragbar:

.altKey true wenn ALT-Taste gedrückt, sonst false (egal ob linke oder rechte)
.altLeft true wenn linke ALT-Taste gedrückt, sonst false
Hinweis: rechte Alt-Taste (AltGr) ermitteln aus
altKey AND NOT altLeft

.ctrlKey true, so Strg-Taste gedrückt (egal ob linke oder rechte)
.ctrlLeft true wenn linke Strg-Taste gedrückt, sonst false
Hinweis: rechte Strg-Taste ermitteln aus
ctrlKey AND NOT ctrlLeft

.shiftKey true, so shift-Taste gedrückt (egal ob linke oder rechte)
.shiftLeft true wenn linke shift-Taste gedrückt, sonst false
Hinweis: rechte shift-Taste ermitteln aus
shiftKey AND NOT shiftLeft

.repeat liefert Dauerdruckzustand in die Eventeigenschaft
true, so Taste im Dauerdruck, sonst false

4.3.2.2.5.5.8.2. Tastatur-Eventbehandlung beim Netscape

Der Event-Handler muss als Parameter das gewünschte Event übergeben bekommen

4.3.2.2.5.5.8.2.1. Tastatur-Eventarten

onkeydown Ereignis ausgelöst für alle Tastenarten
liefert ASCII-Code der Taste nach Eigenschaft.which
nach String umwandeln per string_name=fromCharCode(..)
einmalig, also genau wenn Taste gedrückt wird
Besonderheit bei Kombination Steuertaste und andere Taste
z.B. Shift-Taste und B-Taste zugleich für Großbuchstabe B
Ereignis wird pro Tastendruck aufgerufen, also
1. Mal für Shift-Taste
2. Mal für B-Taste
ab NS 4.x: keydown-Ereignis ruft automatisch das keypress-Ereignis auf, aber
wenn keydown-Handler return false; bzw. beim IE
alternativ event.returnValue=false; liefert, so wird ein
kodierter keypress-Handler nicht aktiviert !

onkeypress Tag <A>, , <TEXTAREA>
wird unmittelbar nach dem keydown-Ereignis ausgelöst allerdings nur für
Tastenarten
! @ # \$ % ^ & * () _ - + = < [] { } , . / ? \ | ' " ` ~
0 bis 9
a bis z mit Unterscheidung Gross oder Klein
Enter
Leertaste
und dient dem Erkennen von Tastaturkombinationen,
die nicht vom Ereignis keydown abgefangen werden
der Aufruf des keypress-Handlers ist **nur** möglich,
wenn der keydown-Handler
return true;
bzw. beim IE alternativ event.returnValue=true;
liefert
Tag <A>, , <TEXTAREA>
onkeyup Ereignis ausgelöst, wenn gedrückte Taste jeder Art losgelassen wird
wird automatisch nach Ereignis keypress ausgelöst
Ereignis keypress automatisch nach Ereignis keydown ausgelöst
besitzt vordefiniertes Schlüsselwort KEYUP für captureEvents
also document.captureEvent(Event.KEYUP);
Tag <A>, , <TEXTAREA>

4.3.2.2.5.5.8.2.2. Tastatur-Eventeigenschaften

4.3.2.2.5.5.8.2.2.1. Steuertasten Alt, Strg (Ctrl) und Umschalt (Shift) (.modifiers)



.modifiers Bitmaske für Zustand der Steuertasten Shift, Alt und Strg
 dient zur Ermittlung der Steuertaste per vordefinierter Maske
 durch bitweise UND-Verknüpfung mit der Bitmaske
 ergibt die Verknüpfung 1, also true, so Steuertaste gedrückt worden
 vordefinierte Maske für

```
Alt-Taste "Event.ALT_MASK"
Strg-Taste "Event.CTRL_MASK"
Shift-Taste "Event.SHIFT_MASK"
Bsp: return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0);
// nicht logisches UND && sondern bitweises UND &
```

4.3.2.2.5.5.8.2.2.2. Eventwert-Eigenschaft (.which)

.which enthält ASCII-Code der Taste
 Umwandlung in String per String.fromCharCode(TastenKodeASCII)

4.3.2.2.5.5.8.2.2.3. Eventart-Eigenschaft (.type)

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

4.3.2.2.5.5.8.2.2.4. Eventquelle-Eigenschaft (.target)

.target 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 = Ereignis.target)

4.3.2.2.5.5.8.2.3. Tastatur-Ereignis-Folge onkeydown und onkeypress

Standard: erst onkeydown
 dann onkeypress

modifiziert: wenn Eventhandler für keydown return false; liefert, so wird der Eventhandler für
 keypress nicht aktiviert

4.3.2.2.5.5.8.2.3.1. Tastatur-Ereignis onkeydown

onkeydown ausgelöst bei allen Tasten
 liefert den ASCII-Code in die Event-Eigenschaft
 .which ASCII-Code der gedrückten Taste
 wenn 0 so keine Taste gedrückt
 wobei bei Buchstaben unterschieden wird
 Umwandlung in String per String.fromCharCode(TastenKodeASCII)

Zusatzanalyse möglich per Event-Eigenschaft

.modifiers Bitmaske für Zustand der Steuertasten Shift, Alt und Strg
 dient zur Ermittlung der Steuertaste per vordefinierter Maske
 durch bitweise UND-Verknüpfung mit der Bitmaske
 ergibt die Verknüpfung 1, also true, so Steuertaste gedrückt worden
 vordefinierte Maske für
 Alt-Taste "Event.ALT_MASK"
 Strg-Taste "Event.CTRL_MASK"
 Shift-Taste "Event.SHIFT_MASK"

Bsp: return ((Ereignis.modifiers & Event.SHIFT_MASK) != 0);
 // nicht logisches UND && sondern bitweises UND &

Unterbindung des Aufrufes des keypress-Handlers:
 keydown-Handler muss return false; bzw. event.returnValue=false; liefern

4.3.2.2.5.5.8.2.3.2. Tastatur-Ereignis onkeypress

wird unmittelbar nach dem keydown-Ereignis ausgelöst und dient dem Erkennen von Tastaturkombinationen,
 die nicht vom Ereignis keydown abgefangen werden

der Aufruf des keypress-Handlers ist **nur** möglich,
 wenn der keydown-Handler return true; bzw. event.returnValue=true; liefert

wird ausgelöst **nur** bei den Tasten

```
! @ # $ % ^ & * ( ) _ - + = < [ ] { } , . / ? \ | ' " ` ~
0 bis 9
a bis z mit Unterscheidung Gross oder Klein
Enter
Leertaste
```

alle anderen Tasten mit dem Ereignis keydown abfangen

4.3.2.2.5.5.8.3. Beispiel zur Tastatur-Eventbehandlung beim IE und Netscape

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

```
var ns = document.layers ? true : false;
```



```

var ie = document.all ? true : false;

// ***** Funktionen für Tasten aller Art holen
function NS_IE_TastencodeHolen_ASCII(Ereignis)
{
    if (ie)
    {return event.keyCode;}
    else
    {
        if (ns)
        {return Ereignis.which;}
    }
}
function NS_IE_TastencodeHolen_String(TastencodeASCII)
{
    if (TastencodeASCII == 0)
    {return "";}
    else
    {return String.fromCharCode(TastencodeASCII);}
}

function IE_TasteDauerdruck_StatusHolen()
{return event.repeat;} // true, so Taste im Dauerdruck, sonst false

// ***** Funktionen für Shift-Taste
// liefern true für Shift-Taste gedrückt; sonst false

function NS_IE_ShiftTaste_StatusHolen(Ereignis)
{
    if (ie)
    {return event.shiftKey;}
    else
    {
        if (ns)
        {return (Ereignis.modifiers & Event.SHIFT_MASK)
            != 0;}
    }
}

function IE_ShiftLeftTaste_StatusHolen()
{return event.shiftLeft;}

function IE_ShiftRightTaste_StatusHolen()
{return (event.shiftKey & (!event.shiftLeft));}

// ***** Funktionen für Alt-Taste analog zu Shift aber für NS mit Event.ALT_MASK

// ***** Funktionen für Strg-Taste analog zu Shift aber für NS mit Event.CTRL_MASK

// -->
</SCRIPT>

```

4.3.2.2.5.5.9. Mouse-Eventbehandlung des IE und NS

Die Programmierung der Tastatur-Events unterscheidet sich nicht nur zwischen den Browsern IE und Netscape, sondern auch noch innerhalb deren Versionsnummern. Dieser Umstand zeugt vom divergierenden Ansatz der Browserhersteller, welcher im krassen Widerspruch zu einer Programmierungsfreundlichkeit steht.

Die bessere Konformität zur konsequent-objektorientierten Umsetzung des HTML-Dokumentes, seiner Elemente und Ereignisse hat der Internet Explorer, der damit ein weiteres Spektrum an Möglichkeiten zur Programmierung unter Javascript bietet, obwohl Microsoft weiterhin Visual Basic-Script als Komponente des Windows Script Host (Betriebssystem-Komponente) favorisiert und eine Dokumentierung der Javascript-Komponente (inklusive Debugger-Freeware-Tool zum aktuellen IE) konsequent vernachlässigt.

4.3.2.2.5.5.9.1. Mouse-Eventbehandlung beim Internet Explorer ab 4.x

4.3.2.2.5.5.9.1.1. 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 rechten Maustaste



		Unterdrückung, wenn der Eventhandler 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 verschachtelt 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 cancelBubble aktiviert wurde)
	Tag	<A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie 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>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.
onmouseout		Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.
onmouseover		Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt Tag <A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie 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 verschachtelt 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 cancelBubble aktiviert wurde)
	Tag	<A>, <APPLET>, <AREA>, <BODY>, <DIV>, <FORM>, , <INPUT>, <MAP>, <SELECT>, , <TEXTAREA>, die meisten Textelemente wie etc.
4.3.2.2.5.5.9.1.2. 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	
aus	Hinweis: horizontale Pixel-Position der Maus gegenüber der linken oberen Ecke des HTML-Dokumentes ergibt sich	
	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 Ereignis 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 oberer Ecke (0,0)
.offsetY	vertikale Pixel-Position der Maus im HTML-Element z.B. <DIV>, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer 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 Bildschirms
.screenY	vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirms
.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.srcElement)
.toElement	enthält Zeiger desjenigen Objektes, das Ereignis 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)

4.3.2.2.5.9.2. **Mouse-Eventbehandlung beim Netscape ab 4.x**

Der Event-Handler muss als Parameter das gewünschte Event übergeben bekommen

4.3.2.2.5.9.2.1. **Mouse-Eventarten**

onclick	Ereignis ausgelöst, wenn Maus das Objekt angeklickt hat: Taste drücken und loslassen im Handler return false; kodieren für Links, Formularelemente wenn eine Aktion nicht erwünscht ist Bsp.: bei Link wird dieser nicht ausgeführt, obwohl angeklickt
ondblclick	Tag <A>, <BODY>, <INPUT> Ereignis ausgelöst, wenn durch linke bzw. rechte Maustaste ein Doppelklick auf Objekt erfolgt 1 Klick = Maustaste drücken und loslassen
onlosecapture	Tag fast alle Ereignis ausgelöst, wenn releaseCapture() für mousemove, mouseover und mouseout ausgeführt wurde der User mit der Maus das Browserfenster verlässt
onmousedown	Tag <A>, <APPLET>, <BODY>, <DIV>, <FORM>, <INPUT>, <P>, <SELECT>, , <TABLE>, <TD>, <TEXTAREA>, <TR> Ereignis ausgelöst mit Drücken der linken oder rechten Maustaste (mittlere nicht !) siehe Event-Eigenschaft .which auch in Verbindung mit Umschalt (Shift) und Alt return false; unterbindet immer die die Ausführung des Eventhandlers return false bei rechter Maustase auf Object document wird das Kontextmenü gesperrt
onmousemove	Tag <A>, <DIV>, <INPUT TYPE="button"> Ereignis bei jeder Mausbewegung ausgelöst Auswertung für den Bereich eines Objektes durch dessen Eventhandler
onmouseout	Tag bei NS unbekannt Ereignis ausgelöst, wenn Maus den Bereich des Objektes verlässt
onmouseover	Tag <A>, <DIV>, Ereignis ausgelöst, wenn Maus den Bereich des Objektes betritt
onmouseup	Tag <A>, <DIV>, Ereignis ausgelöst mit Loslassen der linken Maustaste (rechte und mittlere nicht !) siehe Event-Eigenschaft .which return false; unterbindet immer die die Ausführung des Eventhandlers

4.3.2.2.5.9.2.2. **Mouse-Event-Eigenschaften**

.height	Höhe in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde
.layerX	horizontale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis darf kein resize-Event sein
.layerY	neue Breite in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Breite verändert wurde, nur resize-Event vertikale Pixel-Position der Maus im Layer, in dem das Ereignis ausgelöst wurde, bezüglich dessen linker oberer Ecke (0,0) Ereignis darf kein resize-Event sein
.pageX	neue Höhe in Pixel desjenigen Fensters in Pixel, das durch Mausziehen in der Höhe verändert wurde, nur resize-Event
.pageY	horizontale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
.screenX	vertikale Position der Maus bezüglich der linken oberen Ecke (0,0) des HTML-Dokumentes
.screenY	horizontale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirms



.screenY	vertikale Pixel-Position der Maus gegenüber linker, oberer Ecke (0,0) des Bildschirms		
.target	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.srcElement)		
.type	enthält die Event-Art z.B. abort		
.which	enthält bei gedrückter	Maustaste:	1 für linke Taste 2 für mittlere Taste 3 für rechte Taste
.width	Breite in Pixel vom Fenster bzw. Frame, in dem ein Event ausgelöst wurde		
.x	identisch mit layerX		
.y	identisch mit layerY		

4.3.2.2.5.5.10. Eventbehandlung für Textoperationen mit der Windows-Zwischenablage (Clipboard)

nur Internet Explorer ab 5.x

4.3.2.2.5.5.10.1. 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>, , <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>, , <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>, , <TEXTAREA>		
oncopy	Einfüge-Operation im Kontextmenü ermöglichen: Handler muss return false; bzw. returnValue=false; liefern markierten Text in die Zwischenablage kopieren: Ereignis, das direkt nach der Kopieraktion ausgelöst wird		
oncut	Tag alle die Text definieren, <A> 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 Kontextmenü-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 Kontextmenü-Eintrages zum Einfügen (standardmäßig ist Einfügen nicht möglich) Tag alle Tags mit markierbarem Text z.B. <A>, <DIV>, , <TEXTAREA>		

4.3.2.2.5.5.10.2. Beispiel

```

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

function AusschneidenUndInDieZwischenAblageEinfuegen()
{
    event.returnValue=false; // Kontextmenü-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 !

Es wird damit eine 3. Zeile erzeugt !


```
</SPAN>
```

```
</BODY>
```

```
</HTML>
```

4.3.2.2.5.5.11. 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>

4.3.2.2.5.5.12. HTML-Element-Lade-Eventbehandlung des IE und NS

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

4.3.2.2.5.5.12.1.1. Lade-Ereignisse für Bild

onabort Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
Tag

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>, , <SCRIPT>

4.3.2.2.5.5.12.1.2. 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

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>, , <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()

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

4.3.2.2.5.5.12.2. Lade-Ereignisse des Netscape ab 3.x

4.3.2.2.5.5.12.2.1. Lade-Ereignisse für Bild



onabort Ladevorgang eines Bildes bricht ab z.B. wegen Useraktion Stop
 Tag
 onload Ereignis ausgelöst, wenn Bild vollständig geladen wurde
 zu animiertes Bild (Gif-Datei):
 Ereignis load bei jedem Bildwechsel der Animation ausgelöst
 es kann also für jedes Bild das Ereignis behandelt werden
 Tag <BODY>, <DIV>, <FRAMESET>,

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

onload Ereignis ausgelöst, wenn vollständig geladen wurde
 HTML-Dokument mit all seinen Elementen jeder Art
 Framset (innerhalb <FRAMESET> kodieren)
 DIV
 Layer
 Tag <BODY>, <DIV>, <FRAMESET>,
 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>

4.3.2.2.6. window.external Objekt des Internet Explorer

Dieses Objekt betrifft direkt die Privatsphäre des Users und ist mit Sorgfalt zu verwenden ! Jeder Interessenkonflikt zwischen dem Webseitenanbieter, der dieses Objekt nutzt, und dem User ist zu vermeiden. Die Nutzung des Objektes ist dem User transparent zu machen, so dass er die Wahl hat, ob das Objekt wirksam werden soll oder nicht.

Das Objekt ermöglicht den Zugriff auf

- das Kontextmenü
- die Favoritenliste
- den Microsoft Active Desktop (Desktop muss vom User per Desktop-Eigenschaften als Active Desktop eingestellt worden sein)
- die Microsoft Active Channel (Channel muss vom User per Desktop-Eigenschaften als Active Channel eingestellt worden sein)
- das Autocomplete bei Formularen (Autokomplete kann vom User in den Browser-Optionen eingerichtet werden)
- das permanente Speichern von User-Daten
- das Autoscan von Urls
- die Suche in Webseiten
- die Einstellungen zu Sprache, Favoriten, Sicherheit

kompletter Zugriff erst ab IE 5.x

Eigenschaften:

.menuArguments Zeiger auf dasjenige offene Fenster, indem ein Eintrag aus dem Kontextmenü ausgeführt wurde

Beispiel:

```
<SCRIPT LANGUAGE = "JavaScript">
var ZeigeraufWindow = window.external.menuArguments;

// beispielsweise einen selektierten Text im Fenster verarbeiten
var ZeigerAufDokuemntImWindow = ZeigeraufWindow.document;
var SelektionImDokument = ZeigerAufDokuemntImWindow.selection;
var TextBereichImDokument = SelektionImDokument.createRange();
var SelektierterTextImTextBereich = TextBereichImDokument.text;

if (SelektierterTextImTextBereich.length == 0)
{ TextBereichImDokument.text = "Einfuege Text";}
else
{TextBereichImDokument.text = SelektierterTextImTextBereich.toUpperCase();}
</SCRIPT>
```

Methoden:

.AddChannel() Dialogbox zur Channel-Einstellung öffnen um einen Channel zu installieren (Microsoft Active Channel)
 erzeugt im Fehlerfall eine Dialogbox und das Event onerror

Beispiel:

```
window.external.AddChannel("http://test /file.cdf");
```

.AddDesktop() eine Webseite oder Bild zum installierten Microsoft Active Desktop hinzufügen
 wenn Active Desktop nicht installiert, so passiert nichts

Beispiel:

```
window.external.AddDesktopComponent("http://www.test.de","website",100,100,200,200);
```

.AddFavorite() Dialogbox zum Hinzufügen einer Url in die Favoritenliste für den User öffnen
 Syntax:

```
logischer_window_name.external.AddFavorite(Kette1 [, Kette2])
```

Kette1

String

URL des Favoriteneintrages, also diejenige Url
 die gebokkmarkt werden soll

