link Objekt des Internet Explorer

siehe auch Objektes a

Erzeugung unter HTML:

```
<LINK
        HREF="url"                                  //  protocol://[host_name[:port]/]dateiname
                                                    //  protocol://[host_name[:port]/]dateiname#hashtext
                                                    //  protocol://[host_name[:port]/]dateiname?serachtext
                                                    // Bsp:  http://www.test.com:8888/test.html
                NAME="anker_name_ohne_#"            // ist nicht der logische link_name !!
                                                    //          Netscape + Internet Explorer
        oder    ID="anker_name_ohne_#"              // ist        der logische link_name !!
                                                    //          nur Internet Explorer ab 4.x
        TARGET="logischer_window_name"
        onClick="eventhandler1"
        onDblClick="eventhandler2"
        onMouseOut="eventhandler3"
        onMouseOver="eventhandler4"
        onMouseDown="eventhandler5"
        onMouseUp="eventhandler6"
>
        link_text_immer_schreiben
</LINK>
```

Zugriff: **document**.links[index]

index: ab 0
Nummer des Links im Dokument
Anzahl aller Links lautet document.links.length

Beispiel für globalen Style per HEAD:

```
<HEAD>
<STYLE>
        .an   {text-decoration: underline overline; color:blue;}
        .aus {text-decoration: none; color:black;}
</STYLE>
</HEAD>
<BODY>
        <A      HREF="test.htm"
                CLASS="aus"
                onmouseover="this.className='an';"
                onmouseout="this.className='aus';"
        >
        </A>
</BODY>
```

Eigenschaften (ausgewählte):

.className	Klassenreferenz
.hash	entspricht **#hash**text zum Anspringen eines Ankers hier den Ankernamen mit vorgesetztem # ablegen
.host	entspricht hostname:port
.hostname	entspricht nur hostname
.href	gesamter Url (kompletter Url) zum Anspringen eines Ankers hier den Ankernamen ohne vorgesetztem # ablegen
.id	entspricht ID nur IE ab 4.x
.innerText	ist der link_text nur IE ab 4.x lesen und schreiben
.name	entspricht NAME
.offsetHeight	Pixelhöhe nur IE ab4.x
.offsetLeft	Pixelpos bezüglich linken Rand des HTML-Dokumentes nur IE ab 4.x
.offsetTop	Pixelpos bezüglich oberen Rand des HTML-Dokumentes nur IE ab 4.x
.offsetWidth	Pixelbreite nur IE ab4.x
.offsetLeft	Pixelpos bezüglich linken Rand des HTML-Dokumentes
.pathname	entspricht aus url /dateiname bzw. /dateiname**#hash**text bzw. /dateiname**?search**text

lesen und schreiben

.port entspricht port; lesen und schreiben

.protocol entspricht Protokoll mit Doppelpunkt z.B. "http:"
lesen und schreiben

.search entspricht **?search**text
lesen und schreiben

.style Zeiger auf das style-Objekt
nur IE ab 4.x

.tagName enthält "A"
nur IE ab 4.x

.target entspricht TARGET
kann auch sein _blank
_parent
_search
_self
_top
lesen und schreiben

.text enthält den link_text
nur lesen
nur NS ab 4.x

.x horizontale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes
nur lesen
nur NS ab 4.x

.y vertikale Pixelpos gegenüber linke, obere Ecke (0,0) des HTML-Dokumentes
nur lesen
nur NS ab 4.x

Methoden (ausgewählte):

.attachEvent() siehe Eventbehandlung des IE ab 5.x
nur IE ab 5.x

.blur() Element den Focus wegnehmen und Event onblur auslösen
Der Focus wird nicht automatisch auf irgend ein anderes Element gesetzt !
vor IE 5.0 TABINDEX-Attribut muss kodiert sein
ab IE 5.0 TABINDEX-Attribut muss nicht kodiert sein

.detachEvent() siehe Eventbehandlung des IE ab 5.x
nur IE ab 5.x

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

.getBoundingClientRect() liefert Kooerinaten des Rechtecktes um den Link
Funktionswert ist Zeiger auf die Rechteck-Instanz vom Objekt-Typ TextRectangle
mit den Eigenschaften .left
.top
.right
.bottom
alles Pixelpositionen
nur IE ab 5.x

.handleEvent() siehe Eventbehandlung zum NS ab 4.x
nur NS ab 4.x

.releaseCapture() siehe Eventbehandlung zum IE ab 5.x
nur IE ab 5.x

.reload([true]) wenn true entfällt: Dokument vom Cache auf Festplatte laden
wenn true kodiert: Dokument vom Server und nicht Cache laden
Achtung: Server kann selbst Cache haben und
daraus das Dokument liefern

.replace(url) aktuelle Seite mit neuer geladener Seite überschreiben sowie den History-Eintrag zur aktuellen Seite
überschreiben (keinen neuen bilden) --> BACK-Button wechselt danach nicht zur
überschriebenen Seite, da diese in der History nicht mehr bekannt ist

.scollIntoView(true oder false) Oberkante des Links in den sichtbaren Bereich scrollen
true, so bis zum oberen Fensterrand
false, so bis zum unteren Fensterrand
nur IE ab 5.x

.setCapture() siehe Eventbehandlung zum IE ab 5.x
nur IE ab 5.x

link Objekt des Internet Explorer:

Das Objekt kann nur innerhalb <HEAD> ... <HEAD> benutzt werden und wird nicht angezeigt. Es dient zum Einbinden externer Dateien, die den Quellcode des Dokumentes manipulieren sollen.

Beispiel:

```
<LINK REL=stylesheet HREF="styles.css" type="text/css">
```

Eigenschaften:

.canHaveHTML prüfen ob Objekt HTML-Tags enthalten darf

.charset Zeichensatz zum Encoden eines Objektes im Dokument

.disabled Interaktionsfähigkeit

	nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.href	Ziel-Url oder Anker als Sprungziel (z.B. <A>-Tag) siehe auch Eigenschaften .rel und .rev
.hreflang	Sprachcode des Objektes laut RFC1766
.id	Bezeichner des Objektes für Referenzierung des Objektes (Zeiger, ID, logischer Objektname) Hinweis: Browser erzeugt pro Objekt ein internes ID, das per Eigenschaft .uniqueID ermittelt und anstelle der Eigenschaft .id verwendet werden kann (falls Browser und betroffenes Objekt die Eigenschaft .uniqueID kennen). Zeiger aus ID bilden var Zeiger = eval(object.id);
.isContentEditable	Editierbarkeit des Objekt-Content (auch wenn kein Layout hat) Content = Beziehung des Objektes zum Umfeld z.B. bezüglich Layout etc.
.isDisabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.isMultiLine	Mehrzeiligkeit des Objektinhaltes
.isTextEdit	Erzeugbarkeit eines Textbereiches
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.name	Name des Objektes (nicht ID !!!) muss beim Formular für alle zu sendenden Felder kodiert sein !! Element darf nicht per Methode .createElement() erzeugt worden sein
.nextSibling	Zeiger auf das NACHFOLGENDE Kind laut childNodes collection eines Objektes
.nodeName	String als Name des Kindes (Knoten, Node, Element) also TAG-Bezeichner, Attribut-Name; #text für Anker
.nodeType	Knotentyp laut attributes Collection
.nodeValue	Knotenwert (Wert des Kindes, Node, Elementes) nur für Text- und Attribut-Elemente nicht für Element-Knoten (Knotentyp 1)
.ownerDocument	Referenz auf das document Objekt zu dem der Koten gehört, also in dem der Knoten erzeugt wurde
.parentElement	Referenz auf das Elternobjekt, also nicht Elternknoten innerhalb DOM
.parentNode	Referenz auf Elternknoten innerhalb der DOM-Hierarchie
.parentTextEdit	Textbereich des Elternobjektes referenzieren
.previousSibling	Referenz auf das Vorgängerkind
.readyState	aktueller Status des Objektes beim Füllen des Objektes mit Daten
.rel	Beziehung zwischen Objekt in Quellseite und Nachfolger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steurerungshinweis für Suchmaschinen beim durchforsten der Seiten und Nachfolgerseiten vorrangig kodiert in <A> oder <LINK> es **muss** zugleich die Eigenschaft .href kodiert und mit **gültigen** Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)
.rev	Beziehung zwischen Objekt in Quellseite und Vorgänger-Zielseite laut Objekt Objekt ist ein Link z.B. A-Tag oder Link-Tag nur wichtig für Suchmaschine: dient als Steurerungshinweis für Suchmaschinen beim durchforsten der Seiten und Vorgängerseiten vorrangig kodiert in <A> oder <LINK> es **muss** zugleich die Eigenschaft .href kodiert und mit **gültigen** Wert belegt sein nur für einen Anker ist zugleich Eigenschaft .rev kodierbar (falls sinnvoll)
.scopeName	Namensraum laut XMLNS-Attribut
.sourceIndex	Index des Objektes in der Collection document.all
.tagName	Tag-Bezeichner des Objektes
.tagUrn	Uniform Resource Name (URN) laut Namensraum laut XMLNS-Attribut
.target	Name des Ziel-Fenster bzw. Ziel-Frame
.type	MIME-Typ des Objektes (Multipurpose Internet Mail Extension) wenn die Eigenschaft .classid nicht kodiert wird, dann immer .type kodieren bzw. die im Browserstandard enthaltenenen MIME verwenden Hinweis: MIME wird von Simple Mail Transfer Protocol (SMTP) benutzt z.B. bei Audio, Images, Video, Texten
.uniqueID	durch den Browser automatisch-genriertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiede ID, wenn Objekt mehrmals geladen wurde kann anstelle eines privat vergebenen ID als ID-Attributwert weiterverwendet werden

Methoden:

.addBehavior()	DHTML-Verhaltenseigenschaft einem Element hinzufügen Empfehlung: Standard-IE-Eigenschaften nutzen, da diese mit "#default#behaviorName" komplett erfasst werden und bereits im Browser implementiert sind (keine HTC-Datei nötig). ab IE 5.x bis unter IE 5.5
.applyElement()	Elementeigenschaft "Kind sein" oder "Eltern sein" festlegen, also die Lage im DOM, und danach Referenz laut DOM liefern DOM wird geändert Element kann selbst Kinder haben Element erst sichtbar, wenn Endetag (falls vorhanden) des Elementes geparst wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht im im Dokumentenbaum eingebunden ist, so wird die Eigenschaft .innerHTML gelöscht !
.attachEvent()	Einschalten des Registrieren eines Events durch Eventhandler

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

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

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

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

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

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

.fireEvent() ein Event auslösen

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

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

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

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

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

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

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

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

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

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

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

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

.removeBehavior()	per Methode .addBehavior() einem Element hinzugefügte Verhaltenseigenschaft entfernen (stets VOR dem Entfernen des Elementes mit der zugeordneten Eigenschaft aus der Dokument-Hierarchie) DOM wird geändert
.replaceAdjacentText()	Plain-Text (ohne HTML und Script) eines Elementes durch anderen Text ersetzen und Referenz auf den zu ersetzenden Text liefern DOM wird nicht geändert
.setAttribute()	Wert von vorhandenem Attribut setzen wenn Attribut nicht vorhanden, so wird es automatisch erzeugt und mit dem Wert gefüllt DOM wird nur bei Erzeugung geändert
.setAttributeNode()	Attribut einem Knoten zuweisen und Referenz liefern DOM wird geändert
.swapNode()	Positionen von 2 Knoten im DOM tauschen (Zeigertausch) nur sichtbar wenn Endetag geparst DOM wird geändert