

comment Objekt des Internet Explorer

Kommentar-Objekt

Kommentar wird nicht angezeigt (gerendert)

Eigenschaften:

.canHaveChildren	prüfen ob Kind möglich ist, also ob Objekt Parent sein kann
.canHaveHTML	prüfen ob Objekt HTML-Tags enthalten darf
.data	Url der Daten des Objektes
.disabled	Interaktionsfähigkeit nur wenn sichtbar so User-Interaktion möglich
.firstChild	Zeiger auf das ERSTE Kind laut childNodes-Collection eines Objektes
.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
.lang	Sprache für Anzeige von Sonderzeichen etc.
.lastChild	Zeiger auf das LETZTE Kind laut childNodes collection eines Objektes
.length	Anzahl der Zeichen im Kommentar (comment-Objekt), ab IE 6.x
.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)
.offsetParent	Referenz der Eltern für Nutzung von .offsetHeight, .offsetLeft, .offsetTop und .offsetWidth
.outerText	Referenz auf den gesamten Plain-Text im Objekt nur nach kompletten einlesen des Dokumentes nutzbar
.ownerDocument	Referenz auf das document Objekt zu dem der Knoten 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
.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
.text	Text eines Objektes
.uniqueID	durch den Browser automatisch-generiertes ID des Objektes Browser generiert zu verschiedenen Zeitpunkten auch verschiedene 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
.appendChild()	Knoten als Kind an die DOM-Hierarchie anhängen und danach den Zeiger laut DOM liefern DOM wird geändert Zeiger wird zugleich immer am Ende der Collection childNodes angehängen Anhängen wird danach nur sichtbar, wenn zusätzlich dem BODY-Objekt angehängen wird UND Ende-Tag (falls vorhanden) des Knoten geparkt wurde
.appendData()	String an das Ende des Objektes anhängen
.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 geparkt wurde Achtung: Wenn Element per Methode .createElement() erzeugt wurde, aber nicht 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



.clearAttributes()	<p>die Handler prüfen ihre Aufruffolge (muss programmiert werden) alle HTML-Attribute eines Objektes entfernen außer ID, STYLE und per Script definierte Attribute Script-erzeugte Attribute nicht entfernbar DOM wird geändert</p>
.cloneNode()	<p>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)</p>
.componentFromPoint()	<p>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.</p>
.deleteData()	Teilkette aus einem Objekt entfernen
.detachEvent()	<p>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)</p>
.dragDrop()	prüfen des Status der letzten Drag-Manipulation (anklicken, ziehen, ablegen) auf Element
.fireEvent()	ein Event auslösen
.getAdjacentText()	<p>Text eines Objektes liefern, wobei Textlage im Objekt definiert werden kann Text kann HTML-Tags enthalten, muss aber nicht DOM nicht geändert</p>
.getAttribute()	<p>Wert eines per HTML erzeugten Attributes liefern DOM nicht geändert</p>
.getAttributeNode()	<p>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</p>
.getBoundingClientRect()	Referenz auf TextRectangle-Objekt im Element holen
.getClientRects()	<p>Referenz auf Feld der Zeiger auf TextRectangle-Objekte im Fenster Feld mit Index als Integer ab 0 pro Eintrag ein Rectangle</p>
.hasChildNodes()	<p>prüfen auf Existenz von Kinder(n) als HTML-Elemente oder Textknoten (Textelemente) in einem Objekt DOM nicht geändert</p>
.insertAdjacentElement()	<p>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</p>
.insertAdjacentHTML()	<p>HTML-Code und/oder Script-Code in ein Element einfügen, wobei die Lage definiert sein kann nur nach dem kompletten Laden des Dokumentes möglich HTML- und Script-Code müssen syntaktisch korrekt sein wenn nicht, so wird das Einfügen nicht ausgeführt eingefügter Code wird nur dann sofort geparkt und ausgeführt, wenn syntaktisch korrekt ist bei Script-Code: <SCRIPT DEFER> muss kodiert werden DOM wird geändert</p>
.insertAdjacentText()	<p>Plain-Text (ohne HTML und Script) in ein Element einfügen, wobei die Lage definiert werden kann nur nach dem kompletten Laden des Dokumentes DOM wird geändert</p>
.insertBefore()	<p>Objekt als Kindknoten VOR dem einem anderen Kind-Objekt einfügen und Zeiger liefern einzufügendes Objekt muss mit Methode createElement() erzeugt worden sein Achtung: NICHT anwenden für einfügen VON bzw. VOR obersten Kindknoten Sichtbarkeit erst wenn Ende-Tag geparkt wurde DOM wird geändert</p>
.insertData()	Teilkette in ein Objekt einfügen
.mergeAttributes()	<p>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</p>
.normalize()	<p>Normalisierung des DOM zur Erreichung einer konsistenten Struktur Achtung: CDATA-Sections dürfen nicht enthalten sein, da diese immer Inkonsistenz erzeugen</p>
.removeAttribute()	<p>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</p>



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

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

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

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

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

.replaceData() Teilkette in einem Objekt ersetzen

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

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

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

.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

.substringData() Teilkette aus einem Objekt lesen

.swapNode() Positionen von 2 Knoten im DOM tauschen (Zeigertausch)
nur sichtbar wenn Endetag geparkt
DOM wird geändert

