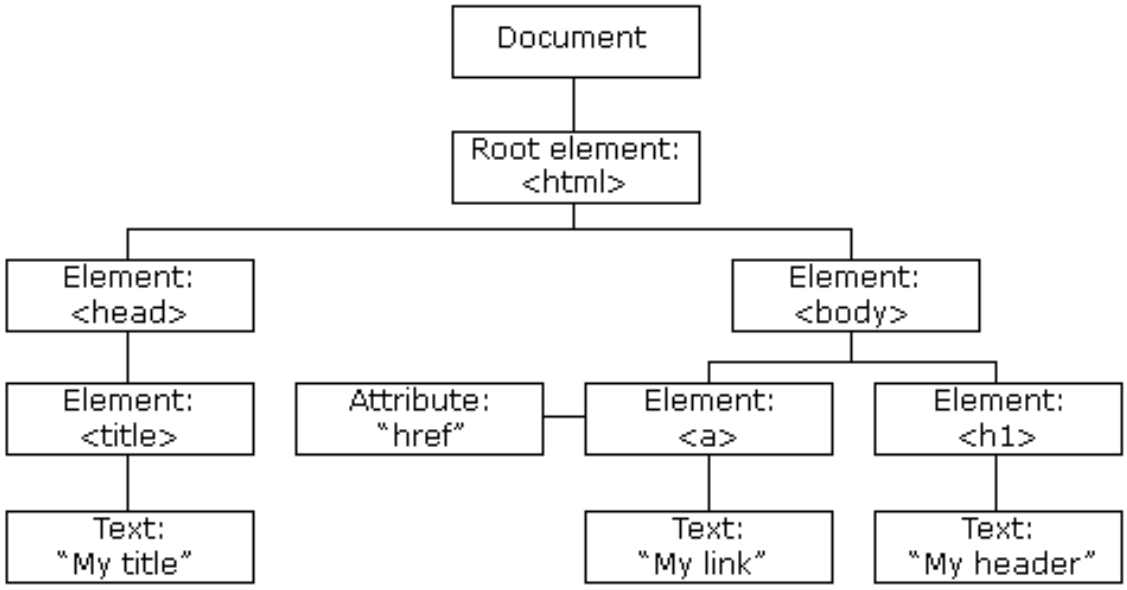
What is the DOM
The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents like HTML and XML.

The DOM is separated into 3 different parts / levels:
- Core DOM - standard model for any structured document
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

The HTML DOM defines a standard way for accessing and manipulating HTML documents.
The DOM presents an HTML document as a tree-structure.



The DOM defines the **objects and properties** of all document elements, and the **methods** (interface) to access them.

Hierarchy of objects in an example HTML DOM - Document Object Model
The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Aspects of the DOM (such as its "Elements") may be addressed and manipulated within the syntax of the programming language in use. The public interface of a DOM is specified in its Application Programming Interface (API).

Standardization
The World Wide Web Consortium (W3C), founded in 1994 to promote open standards for the World Wide Web, brought Netscape Communications and Microsoft together with other companies to develop a standard for browser scripting languages, called "ECMAScript".

DOM Level 2 was published in late 2000. It introduced the "getElementById" function as well as an event model and support for XML namespaces and CSS. DOM Level 3, the current release of the

DOM specification, published in April 2004, added support for XPath and keyboard event handling, as well as an interface for serializing documents as XML.

By 2005, large parts of W3C DOM were well-supported by common ECMAScript-enabled browsers, including Microsoft Internet Explorer version 6 (2001), Opera, Safari and Gecko-based browsers (like Mozilla, Firefox, SeaMonkey Application Suite and Camino)

Layout engines
Web browsers rely on layout engines to parse HTML into a DOM. Some layout engines such as Trident/MSHTML and Presto are associated primarily or exclusively with a particular browser such as Internet Explorer and Opera respectively. Others, such as WebKit and Gecko, are shared by a number of browsers, such as Safari, Google Chrome, Firefox or Flock. The different layout engines implement the DOM standards to varying degrees of compliance.

Programming Interface
In the DOM, HTML documents consist of a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we will use JavaScript.

The programming interface of the DOM is defined by standard properties and methods.
Properties are often referred to as something that is (i.e. the name of a node).
Methods are often referred to as something that is done (i.e. remove a node).

HTML DOM Properties
Some DOM properties:

- x.innerHTML - the text value of x
- x.nodeName - the name of x
- x.nodeValue - the value of x
- x.parentNode - the parent node of x
- x.childNodes - the child nodes of x
- x.attributes - the attributes nodes of x

Note: In the list above, x is a node object (HTML element).

HTML DOM Methods
Some DOM methods:

- x.getElementById(*id*) - get the element with a specified id
- x.getElementsByTagName(*name*) - get all elements with a specified tag name
- x.appendChild(*node*) - insert a child node to x
- x.removeChild(*node*) - remove a child node from x

Note: In the list above, x is a node object (HTML element).

The innerHTML Property
The easiest way to get or modify the content of an element is by using the innerHTML property. innerHTML is not a part of the W3C DOM specification. However, it is supported by all major browsers.
The innerHTML property is useful for returning or replacing the content of HTML elements (including <html> and <body>), it can also be used to view the source of a page that has been dynamically modified.

DOM (Document Object Model) Reference
Updated: December 9th, 2009

The DOM (Document Object Model) gives you generic access to most elements, their styles and attributes in a document. This is a no-nonsense, easy to follow DOM reference for JavaScript.

DOM Window Reference
- Window properties
- Window methods

DOM Document Object Reference
- Document object properties
- Document object methods

DOM Elements Object reference
- Dom Element properties
- Dom Element methods

DOM Event Object reference
- Event Object (redirects to Event object page in the "JavaScript Reference section").

DOM Style Reference
- CSS Rule object
- Dom stylesheet object
- Inline "Style" object

DOM Table Reference
- Dom Table properties
- Dom Table methods

Miscellaneous
- Ajax (XMLHttpRequest)
- Element nodeType values
- TreeWalker object
- DOM scripting forums

Window object methods

Methods	Description
innerWidth, innerHeight	Read/write property that specifies the width and height, in pixels, of the window's content area respectively. Does not include the toolbar, scrollbars etc. **NS/Firefox exclusive properties.** **Note:** IE equivalents are "document.body.clientWidth" and "document.body.clientHeight"
length	Returns the number of frames contained in the window.
outerWidth, outerHeight	Read/write property that specifies the total width and height, in pixels, of the window's content area respectively, including any toolbar, scrollbars etc. **NS/Firefox exclusive properties** with no IE4+ equivalent.
pageXOffset, pageYOffset	Returns an integer representing the pixels the current document has been scrolled from the upper left corner of the window, horizontally and vertically, respectively. Typically used to provide the needed calculations to keep an element in view even when the page is scrolled. **NS/Firefox exclusive properties.** **Note:** IE equivalents are "document.body.scrollLeft" and "document.body.scrollTop"
window.screen	References the screen object, which provides information about the user's screen/ monitor.
screen.availWidth	Returns the height of the screen, in pixels, minus interface features such as the taskbar in Windows. In other words, the usable height available to your browser window.
screen.availHeight	Returns the width of the screen, in pixels, minus interface features such as the taskbar in Windows. In other words, the usable width available to your browser window.
screen.colorDepth	The bit depth of the color palette available for displaying images in bits per pixel.
screen.height	The total height of the screen, in pixels.
screen.pixelDepth	Display screen color resolution (bits per pixel). **NS/Firefox exclusive property.**
screen.width	The total width of the screen, in pixels.
screenX, screenY	Read/write property that specifies the x and y coordinates of the window relative to the user's monitor screen. **NS/Firefox exclusive properties.**
screenLeft, screenTop	Specifies the x and y coordinates of the window relative to the user's monitor screen. **IE only.**
scrollX, scrollY	Returns an integer representing the pixels the current document has been scrolled from the upper left corner of the window, horizontally and vertically, respectively. **NS/Firefox exclusive properties.** Equivalent to pageXOffset and pageYOffset, and in IE, "document.body.scrollLeft" and "document.body.scrollTop"

JavaScript Archive Network is a comprehensive resource for Open Source JavaScript libraries and software.
http://openjsan.org/

This specification defines the Document Object Model Core Level 3, a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model Core Level 3 builds on the Document Object Model Core Level 2 [DOM Level 2 Core].

This version enhances DOM Level 2 Core by completing the mapping between DOM and the XML Information Set [XML Information Set], including the support for XML Base [XML Base], adding the ability to attach user information to DOM Nodes or to bootstrap a DOM implementation, providing mechanisms to resolve namespace prefixes or to manipulate "ID" attributes, giving to type information, etc.

http://www.w3.org/TR/DOM-Level-3-Core/

DOM objects and methods

This gives all properties, collections and methods of the W3C DOM that can be reliably used in all major DOM browsers, as well as `document.styleSheets`, which is much less reliable, but useful if it is available.

— Key

Parent object

- **Child object**
- **Child property**
- **Child object being accessed through a collection[]**
- **Event**
- **Method()**

Each can be clicked for more information.

Collections will also have the `length` property, giving the number of entries in that collection.

— The document node

Creating nodes

document The document object for the current document - - read-only

- **createDocumentFragment()** Returns a document fragment that can be used to add multiple nodes to the document
- **createElement('element_name')** Returns a new element of type element_name
- **createTextNode('text content')** Returns a text node with a string value matching the text content

To walk the DOM tree

document The document object for the current document - - read-only

- **body** The BODY element node - - read-only
- **documentElement** The HTML element node - - read-only
- **getElementById('element id')** Returns a reference to the element with the specified id
- **getElementsByTagName('element_name')** Returns a collection of all elements with the specified element name - NOTE: does not return a true collection; - returns an object with .0 ... .n and .length properties

DOM support

document The document object for the current document - - read-only

implementation An object giving information about the DOM implementation - (Also has other methods in good browsers [not covered here]) - - read-only

hasFeature(string: feature,string: domVersion) Returns true if the browser supports that feature of the DOM specification - Do not rely on browsers to accurately represent their support for features

— All nodes

To walk the DOM tree

- IE incorrectly provides element methods like `getElementsByTagName` on comment nodes.

node An element, text node, document node, or any other node type - - read-only

childNodes[] A collection of all text nodes and element nodes the node contains directly if node is an element node or document node - - read-only

parentNode A reference to the containing node - - read-only

firstChild The first entry in the childNodes collection if node is an element node or document node - - read-only

getElementsByTagName('element_name') Returns a collection of all elements inside the current element - or document node, with the specified element name - NOTE: does not return a true collection; - returns an object with .0 ... .n and .length properties

lastChild The last entry in the childNodes collection if node is an element node or document node - - read-only

nextSibling The next entry in the childNodes collection of the parent node - - read-only

offsetParent The node that the element's offset can be calculated from - - read-only

previousSibling The previous entry in the childNodes collection of the parent node - - read-only

Attaching, copying or removing nodes

node An element, text or attribute node - - read-only

appendChild(nodeReference) Appends the referenced node to the end of the childNodes collection, and displays it on the page - If it is already attached to the document, it will be moved to this new location

cloneNode(bool: copyChildrenToo) Returns a clone of a node or branch of the DOM tree, event handlers may or may not be cloned

innerHTML The HTML contained by the element - Small amounts of HTML will be inserted into the DOM tree immediately - Large amounts of HTML may take as much as 1/2 second to be inserted into the DOM tree - Does not work with XHTML in some browsers - - read/write

insertBefore(nodeReferenceX,nodeReferenceY) Inserts or moves node nodeReferenceX as a childNode of

the element before nodeReferenceY

removeChild(nodeReference) Removes the specified child from the childNodes collection (deletes it) - If childNodes.length becomes 0, the parentNode is also deleted

replaceChild(nodeReferenceX,nodeReferenceY) Replaces child nodeReferenceY with node nodeReferenceX - Returns nodeReferenceY

splitText(index) Splits a text node into two at the specified index creating a new entry in the childNodes collection

Node information

- IE incorrectly provides element properties like `tagName` on comment nodes.

node An element, text or attribute node - - read-only

data Equivalent to nodeValue where node is a text node - - read/write

hasChildNodes() Returns boolean if the node has child nodes

id The id of the element - - read/write

nodeName Name of the node (tag name, attribute name or '#text') - - read-only

nodeType 1 for an element/tag, 2 or undefined for an attribute or 3 for a text node - - read-only

nodeValue The string of a text node or attribute node - - read/write

specified Boolean: says if the node has a value - - read-only

tagName An uppercase representation of the tag name if node is an element node - - In theory, this should be lower case for XHTML (when served as XHTML, not HTML), but some browsers will still use upper case - - This property is incorrectly also provided on comment nodes in IE - - read-only

title The title attribute of the element - - read/write

Element node attributes

node An element node - - read-only

attributes[] A numerical collection of all attributes defined or that could be defined for the element - Difficult to use because of browser incompatibilities - - read/write

getAttribute('attributeName') Returns the value of the specified attribute [not class, style or event handler]

removeAttribute('attributeName') Removes the specified attribute [not align, class, style or event handler]

setAttribute('attribute_name','attribute_value') Sets the name=value pair for the specified attribute of the element [not name, class, style or event handler] - Use along with style.textAlign to make align work in Opera 7.0-7.1

Element node style

node An element node - - read-only

className The class name of the element - - read/write

currentStyle A style object that gives the cascaded and inherited styles applied to the element - - read-only -

- IE compatible browsers only - For cross-browser scripting, see also - window.getComputedStyle
    styleName Text representation of the named style value - - read/only
  style An object containing all inline styles of the element - - read-only
    styleName Text representation of the named style value - - read/write

window The window object for the current document - - read-only
  getComputedStyle(nodeReference,string: pseudoElement) Returns a style object that gives the cascaded and inherited styles applied to the element - - pseudoElement parameter is either a pseudo element name, or null - - Standards compliant browsers only - For cross-browser scripting, see also - elementNode.currentStyle
    styleName Text representation of the named style value - - read/only

Element node size and position (not standardised)

node An element node - - read-only
- **clientHeight** The height of the element inside its border, minus any scrollbar height - - read-only
- **clientWidth** The width of the element inside its border, minus any scrollbar width - - read-only
- **offsetHeight** The height of the element outside its border - - read-only
- **offsetLeft** The distance between the left edge of the node and the left edge of the offsetParent node - - read-only
- **offsetParent** The parent element that the browser has chosen to be the offsetParent - Only reliable cross-browser if all chained offsets are added together - - read-only
- **offsetTop** The distance between the top edge of the node and the top edge of the offsetParent node - - read-only
- **offsetWidth** The width of the element outside its border - - read-only
- **scrollHeight** The height of the element's contents and padding - Only reliable if it has a horizontal scrollbar - - read-only
- **scrollLeft** The horizontal distance that the element has been scrolled - - read-write
- **scrollTop** The vertical distance that the element has been scrolled - - read-write
- **scrollWidth** The width of the element's contents and padding - Only reliable if it has a vertical scrollbar - - read-only

— *Table and associated nodes*

In addition to the normal node functionality, nodes belonging to tables have extra functionality, specially designed to work with the layouts of tables.

To walk the DOM tree

node A table (or related) node - - read-only
- **caption** The caption of the table if element is a table - - read-only

cells[] A collection of all cells in the element if the element is a tr - - read-only

rows[] A collection of all rows in the element if the element is a thead, tbody or tfoot - - read-only

tBodies[] A collection of all tbodies in the table if element is a table - - read-only

tfoot The tfoot element of the table if element is a table - - read-only

thead The thead element of the table if element is a table - - read-only

Attaching, copying or removing nodes

node A table (or related) node - - read-only

deleteCell(cellIndex) Deletes the cell at the specified index if the element is a tr

deleteRow(rowIndex) Deletes the row at the specified index if the element is a thead, tbody or tfoot

deleteTFoot() Deletes the tfoot element if the element is a table

deleteTHead() Deletes the tfoot element if the element is a table

Node information

node A table (or related) node - - read-only

cellIndex The index of the cell within the row, if element is a td or th - - read-only

rowIndex The index of the row within the tbody, thead or tfoot (only reliable if there is no thead or tfoot) - - read-only

— Stylesheets (if supported)

document The document object for the current document - - read-only

styleSheets[] Collection of all stylesheets in the document - - read-only

addRule(string selectors,string styles) Adds a new rule to the end of the stylesheet - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].insertRule

cssRules[] Collection of all CSS rules in the stylesheet, that are not inside a @media block - includes normal style rules, @charset rules, @import rules, @font-face rules, @media blocks, @page rules - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules

cssRules[] Gives all child rules, if the rule is a @media rule - Children as document.styleSheets[int numberOfStyleSheet].cssRules - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules

cssText The textual representation of the rule, including the selector and styles - - read/write (read-only in Mozilla) - - Standards compliant browsers only

deleteRule(int index) Deletes the CSS rule at the specified index within the media block, - if the parent rule is a @media rule - - Standards compliant browsers only

encoding The encoding specified by the rule if the rule is a @charset rule - - read/write - - Standards compliant browsers only

href The URL specified by the rule if the rule is an @import rule - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].imports[j].href

insertRule(string rule,int index) Inserts a new rule at the specified index within the media block, - if the parent rule is a @media rule - If the index is the same as the cssRules.length, the new rule will be added at the end - - Standards compliant browsers only

media Provides information about what media types the rule applies to, - if the rule is a @media or @import rule - Children as document.styleSheets[int numberOfStyleSheet].media - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].imports[j].media

parentRule Refers to the @media rule that the current rule is contained inside (if there is one) - - read/only - - Standards compliant browsers only

parentStyleSheet Refers to the styleSheet that the rule is inside - - read/only

selectorText The selector part of the rule, if it is a normal style rule or a @page rule - - read/write

style Refers to the styles in the CSS rule, if the rule is a normal - style rule, @font-face rule or @page rule - - read/only

- **cssText** The textual representation of the style part of the rule - - read/write
- **item(int index)** Returns the style at the specified index within the rule - - Standards compliant browsers only
- **length** Gives the number of styles that the browser sees inside the rule - - read/only - - Standards compliant browsers only
- **getPropertyValue(string style-name)** Returns the value of the named style - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules[j].style.nameOfStyle
- **setProperty(string style-name,string styleValue,string priority)** Creates or replaces the named style inside the rule, assigning it the specified value - The priority is typically an empty string or 'important' - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules[j].style.nameOfStyle
- **getPropertyPriority(string style-name)** Returns the priority of the named style - Typically an empty string or 'important' - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules[j].style.nameOfStyle
- **removeProperty(string style-name)** Removes the specified style from the rule - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].rules[j].style.nameOfStyle
- **nameOfStyle** Text representation of the named style value - - read/write

styleSheet The stylesheet object of the stylesheet imported by the rule, if the rule is an @import rule - Children as document.styleSheets[int numberOfStyleSheet] - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].imports[j]

type Number that says what type of rule this is: - 0: unknown @ rule - 1: normal style rule - 2: @charset rule - 3: @import rule - 4: @media rule - 5: @font-face rule - 6: @page rule - - read/only - For cross-browser scripting, see also - document.styleSheets[i].insertRule

deleteRule(int index) Deletes the CSS rule at the specified index - - Standards compliant browsers only -

For cross-browser scripting, see also - document.styleSheets[i].removeRule

disabled Says if the stylesheet is disabled - - read/write

href The href of the stylesheet - - read-only

imports[] The stylesheet object of the stylesheet imported by an @import rule - Children as document.styleSheets[int numberOfStyleSheet] - - read/only - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].cssRules[j].styleSheet

insertRule(string rule,int index) Inserts a new rule at the specified index within the stylesheet - If the index is the same as the cssRules.length, the new rule will be added at the end - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].addRule

media Says what media types the stylesheet applies to - - read/write - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].media.mediaText

media Provides information about what media types the stylesheet applies to - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].media as property

- **appendMedium(string mediaType)** Adds the new media type to the list of media types that the stylesheet applies to - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].media as property
- **deleteMedium(string mediaType)** Removes the new media type from the list of media types that the stylesheet applies to - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].media as property
- **item(int index)** Returns the name of the media type at the specified index - - Standards compliant browsers only
- **length** Gives the number of media types that the stylesheet applies to - (As interpreted by the browser) - - read/only - - Standards compliant browsers only
- **mediaText** String representation of what media types the stylesheet applies to - - read/write - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].media as property

ownerNode A reference to the STYLE or LINK element that creates the stylesheet - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].owningElement

- **sheet** References the StyleSheet object created by the STYLE or LINK element - - read/only - - Standards compliant browsers only - For cross-browser scripting, see also - document.styleSheets[i].owningElement.styleSheet

ownerRule A reference to the @import rule that imported this stylesheet - - read/only - - Standards compliant browsers only

owningElement A reference to the STYLE or LINK element that creates the stylesheet - - read/only - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].ownerNode

- **styleSheet** References the StyleSheet object created by the STYLE or LINK element - - read/only - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].ownerNode.sheet

removeRule(int index) Deletes the CSS rule at the specified index - - Internet Explorer only - For cross-

browser scripting, see also - document.styleSheets[i].deleteRule

rules[] Collection of all CSS normal style rules in the stylesheet - - read/only - - Internet Explorer only - For cross-browser scripting, see also - document.styleSheets[i].cssRules

parentStyleSheet Refers to the styleSheet that the rule is inside - - read/only

selectorText The selector part of the rule - - read/write

style Refers to the styles in the CSS rule - - read/only

cssText The textual representation of the style part of the rule - - read/write

nameOfStyle Text representation of the named style value - - read/write

title Reflects the TITLE attribute of the STYLE or LINK element that creates the stylesheet - - read/only

type Reflects the TYPE attribute of the STYLE or LINK element that creates the stylesheet - (Typically 'text/css') - - read/only

— Events

Adding and removing event listeners

node An element node, the document node, or the window object - - read-only

addEventListener(string: event,function,bool: phase) Adds the specified function as a handler for the named event - Phase can be true for capture, or false for bubble - - Standards compliant browsers only - For cross-browser scripting, see also - node.attachEvent

attachEvent(string: onevent,function) Adds the specified function as a handler for the named event - - IE compatible browsers only - For cross-browser scripting, see also - node.addEventListener

detachEvent(string: onevent,function) Removes the specified function as a handler for the named event - - IE compatible browsers only - For cross-browser scripting, see also - node.removeEventListener

removeEventListener(string: event,function,bool: phase) Removes the specified function as a handler for the named event - Phase can be true for capture, or false for bubble - - Standards compliant browsers only - For cross-browser scripting, see also - node.detachEvent

Creating event objects

document The document object for the current document - - read-only

createEvent(string: EventModule) Creates an uninitialised event from the specified event module - Event modules are: - - Events: non-standard events that do not fit into other categories - - HTMLEvents: 'abort', 'blur', 'change', 'error', 'focus', 'load', 'reset', 'resize', 'scroll', 'select', 'submit', 'unload' - - KeyEvents: 'keydown', 'keypress', and 'keyup' (Mozilla only, for cross browser scripting see UIEvents) - - MouseEvents: 'click', 'mousedown', 'mousemove', 'mouseout', 'mouseover', and 'mouseup' - - MutationEvents: 'DOMAttrModified', 'DOMNodeInserted', 'DOMNodeRemoved', 'DOMCharacterDataModified', 'DOMNodeInsertedIntoDocument', 'DOMNodeRemovedFromDocument', and 'DOMSubtreeModified' - - UIEvents: 'DOMActivate', 'DOMFocusIn', 'DOMFocusOut', also in non-Mozilla 'keydown', 'keypress', and 'keyup' (for cross browser scripting see KeyEvents) - - Standards compliant browsers only - For cross-

browser scripting, see also - document.createEventObject

createEventObject(optional templateObject) Creates an uninitialised event - If an existing event object is passed as a parameter, - it will be used as a template to create the new object - - IE compatible browsers only - For cross-browser scripting, see also - document.createEvent

Preparing event objects

eventObject The event object created by document.createEvent - - read-only - - Standards compliant browsers only

initEvent('type', bubbles, cancelable) Initialises the event as a generic event, without defining additional properties - Available for all event types - - Standards compliant browsers only

initKeyEvent('type', bubbles, cancelable, window, ctrlKey, altKey, shiftKey, metaKey, keyCode, charCode) Initialises the event as a key event - Available for 'KeyEvents' event types - - Mozilla only

initMouseEvent('type', bubbles, cancelable, window, detail, screenX, screenY, clientX, clientY, ctrlKey, altKey, shiftKey, metaKey, button, relatedTarget) Initialises the event as a mouse event - Available for 'MouseEvents' event types - - Standards compliant browsers only

initMutationEvent('type', bubbles, cancelable, relatedNode, prevValue, newValue, attrName, attrChange) Initialises the event as a mutation event - Available for 'MutationEvents' event types - - Standards compliant browsers only

initUIEvent('type', bubbles, cancelable, window, detail) Initialises the event as a generic UI event, without defining additional properties - Available for 'MouseEvents', 'UIEvents', and Mozilla's 'KeyEvents' event types - - Standards compliant browsers only

Firing events

node An element node - - read-only

dispatchEvent(eventObject) Fires the event, with the element node as the target - - Standards compliant browsers only - For cross-browser scripting, see also - node.fireEvent

fireEvent('ontype',eventObject) Fires the specified event, with the element node as the srcElement - - IE compatible browsers only - For cross-browser scripting, see also - node.dispatchEvent

Additional event object methods and properties

As well as the traditional event object properties, some more are provided by DOM events (or IE events) capable browsers.

eventObject The event object - - read-only - - Standards compliant browsers only

attrChange Says the type of change for DOMAttrModified mutation events - 1: modification - 2: addition - 3: removal - - read-only - - Standards compliant browsers only

attrName The name of the attribute for DOMAttrModified mutation events - - read-only - - Standards compliant browsers only

bubbles Boolean value says if the event is a type that can bubble - - read-only - - Standards compliant browsers only

cancelable Boolean value says if the event can be cancelled - - read-only - - Standards compliant browsers only

cancelBubble Set to true to prevent the event from being processed by any more event hanler stages - - read-write - - IE compatible browsers only - For cross-browser scripting, see also - eventObject.stopPropagation

charCode The value of the Unicode character associated with key events - - read-only - - Mozilla compatible browsers only

currentTarget A reference to the element that is currently processing the event - - read-only - - Standards compliant browsers only

detail Extra information depending on the event - for example a click event may give the number of consecutive clicks - - read-only - - Standards compliant browsers only

eventPhase Says what phase of the event is currently being processed - 1: capture phase - 2: bubble phase on the target element itself - 3: during the bubbling phase on the target's ancestors - 0: for a manually created event object that has not yet been fired - - read-only - - Standards compliant browsers only

fromElement Gives the previous element the mouse was over for mouseover events - - read-only - - IE compatible browsers only - For cross-browser scripting, see also - eventObject.relatedTarget

metaKey Boolean value says if the system meta key is pressed - For Windows this is the window key - For Mac this is the apple/command key - Other systems may have their own meta keys - - read-only - - Standards compliant browsers only

newValue The new nodeValue after a mutation event - - read-only - - Standards compliant browsers only

preventDefault() Prevents the default action from occuring after event processing is complete - - Standards compliant browsers only - For cross-browser scripting, see also - eventObject.returnValue

prevValue The previous nodeValue before a mutation event - - read-only - - Standards compliant browsers only

relatedNode Performs a similar function to relatedTarget, for mutation events - - read-only - - Standards compliant browsers only

relatedTarget A reference to the related element for relevant events - such as the element the mouse used to be over before a mouseover event - - read-only - - Standards compliant browsers only - For cross-browser scripting, see also - eventObject.fromElement - eventObject.toElement

returnValue Set to false to prevent the default action from occuring after event processing is complete - - read-write - - IE compatible browsers only - For cross-browser scripting, see also - eventObject.preventDefault

stopPropagation() Prevents the event being processed by any - more handlers at further stages in the event flow - - read-only - - Standards compliant browsers only - For cross-browser scripting, see also - eventObject.cancelBubble

toElement Gives the element the mouse is about to move over for mouseout events - - read-only - - IE compatible browsers only - For cross-browser scripting, see also - eventObject.relatedTarget