CC BY-NC-SA 4.0

EcmaScript in Web Quick Reference

Versions

Version information true at 12 Aug 2004.

The Standard is ECMA - 262, 3rd Edition (December 1999).

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Both EcmaScript implementations, JavaScript (Netscape), and JScript (Microsoft), are ECMA-262 compatible.

Goodman 2004 > JavaScript Bible 5th Edition > p59

The two current dominant implementation versions of ECMA -262, 3rd Edition are: Microsoft's JScript 5.0 and Netscape's Javascript 1.5.

These versions are found in the dominant browsers as follows:

Gecko Browsers	Javascript Version
Navigator 6.0 Mozilla	JavaScript 1.5

http://devedge.netscape.com/library/manuals/2000/javascript/1.5/guide/preface.html#1003515

Microsoft Browsers and OS	JScript Version			
	5.0	5.1	5.5	5.6
Microsoft Internet Explorer 5.0	x			
Microsoft Internet Explorer 5.01		x		
Microsoft Windows 2000		x		
Microsoft Internet Explorer 5.5			x	
Microsoft Windows Millennium Edition			x	
Microsoft Internet Explorer 6.0				x
Microsoft Windows XP				x

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriVersionInformation.asp

When learning either implementation (JavaScript or Jscript) stick to the standard (ECMA 262, 3rd Edition).

Learn the language. Stick to the standard. Make your stuff portable. Avoid proprietary features and traps. Keep it clean. Look at jslint for automated advice on a reliable common subset.
http://www.crockford.com/javascript/lint.html

Douglas Crockford > ***comp.lang.javascript > Re: Which to Learn: Javascript, Jscript, JScript.NET, ECMA 262 (3rd ed)? 2004-01-30 07:43:01 PST***

Browser Popularity

BrowserID	Broswer Name	Popularity^
WinIE6	Microsoft Internet Explorer 6.0 for Windows	71.50%
WinIE5.5	Microsoft Internet Explorer 5.5 for Windows	6.60%
WinIE5.0	Microsoft Internet Explorer 5.0 for Windows	4.30%
Moz1.6	Mozilla 1.6	4.30%
NN7.x	Netscape Navigator 7.0	4.20%
Saf1.2	Safari 1.2	1.30%
Opera7.5	Opera 7.5	1.10%
MacIE5.2	Microsoft Internet Explorer 5.2 for Macintosh	0.02%

^ *Rough % use according to* *http://www.webhits.de/webhits/browser.htm* *at 18 July 2004*

Embedding

Run From

There are three ways to run script: Direct from the head or body, from an event, or the `javascript:` Psuedo-protocol.

```
<!-- From the Head or Body -->
<html>
  <head>
    <script type="text/javascript">
    <!--
      document.write("Hello World from head!")
    //-->
    </script>
  </head>
  <body>
    <script type="text/javascript">
    <!--
      document.write("Hello World from body!")
    //-->
    </script>
  </body>
</html>

<!-- From on event -->
<head>
  <script type="text/javascript">
  <!--

    function main () {
      document.write("From Main.");

    } // main
  //-->
  </script>
</head>

<body onload="main()">
  <!-- None of the following is written -->
  <p>
    First para.
  </p>
  <script type="text/javascript">
   document.writeln("Another line.");
  </script>
</body>
```

Direct from Head Or Body

*Scripts run from either section, head or body, are executed as the page loads.***Error! Reference source not found.**

Event

Event Scripts can be hooked up in the head section without any other fuss in the body:

```
<script type="text/javascript">
<!--
  window.onload = function () {
    alert("In onload");
  }
//-->
</script>
</head>
<body>
  ...
```

Script ran from an event can reference a function embedded within a head or body section ....

```
  <script type="text/javascript">
  <!--
    function doItFromHead() {
      alert("Inside doItFromHead.");
    }
  //-->
  </script>
</head>
<body>
  <form name="frmMain">
    <p>
      <input type="button"
             value="Do it from Head"
             onclick="doItFromHead()" />
      <input type="button"
             value="Do it from Body"
             onclick="doItFromBody()" />
    </p>
  </form>
  <script type="text/javascript">
  <!--
    function doItFromBody() {
      alert("Inside doItFromBody.");
    }
  //-->
  </script>
</body>
```

... Or simply run script inline

```
      <input type="button"
             value="Do it Inline"
             onclick="alert('Done inline.')" />
```

If your page has any inline scripts, that is, within element events then you must declare the language with a meta tag.

```
<meta http-equiv="Content-Script-Type"
content="text/javascript" />
```

Required by XHTML. Browsers usually tolerate the absence.

Elizabeth Castro > HTML for the World Wide Web with XHTML and CSS, 5th Edition > P 322

Window event handlers go inside the body tag.

```
<head>
  <script language="javascript">
  <!--

    function main () {
      document.write("From Main.");

    } // main
```

CC BY-NC-SA 4.0

```
  //-->
  </script>
</head>
<body onload="main()">
</body>
```

Even though you would most naturally associate events found inside <body> with the document, it's not. It is the window.

Goodman 2004 > JavaScript Bible 5th Edition > p 86

Starting execution from the Window's onload event is useful to ensure everything has been loaded into memory properly.

Goodman 2004 > JavaScript Bible 5th Edition > p 86

Javascript psuedo-protocol

From onclick

```
<script type="text/javascript">
  <!--
    function doIt() {
      var now = Date();

document.getElementById("pResult").innerHTML =
"Done. " + now;
      blankStatus();
      return true;
    }

    function blankStatus(){
      window.status = "";
      return true;
    }
  //-->
  </script>
</head>
<body>
  <a  href="javascript:%20void%20(0);"
      onmouseover="return blankStatus()"
      onclick="return doIt()">This link</a>
  <p id="pResult"></p>
</body>
```

Goodman 2004 > JavaScript Bible 5th Edition > p135, P137 & P978

Step 1: "javacript:%20void%20(0)" in the `href` or `src` attribute; Note the spaces surrounding `void`.

```
<a  href="javascript:%20void%20(0);" ..
```

This prevents a value coming back that sometimes displays users hard disk.

Goodman 2004 > JavaScript Bible 5th Edition > p978

Step 2: True *`functionName()`* in the onclick event.

```
onclick="return doIt()">
```

Step 3: Mask the default status bar messages: Return true onmouseover.

```
<a href="javascript:%20void%20(0);"
onmouseover="displayHoverText(this); return
true;>

Or

    function blankStatus(){
      window.status = "";
      return true;
    }
  //-->
  </script>
</head>
<body>
  <a  href="javascript:%20void%20(0);"
      onmouseover="return blankStatus()"
      onclick="return doIt()">This link</a>
```

From non onclick event & non href.

When not using the onclick event you should inhibit the default behaviour of the browser with a `return false` in the `onclick` event.

```
<a href="javascript:%20void%20(0);
   onmouseover="return blankStatus()"
   onmousedown="return checkModifiers(event)"
   onclick="return false">
This link</a>
```

Goodman 2004 > JavaScript Bible 5th Edition > p978

Also better?

```
<a id="foo42" href="#foo42"
onclick="func();return false">
```

But when javacript is disabled this will scroll window to link.

alt.html > Jukka K. Korpela > Sun, Jun 18 2006 4:16 pm ("name" changed to "id")

From Href

Use: voiding; and mouseover status bar blanking. Don't use: any onclick returning.

```
<a href="javascript:%20void%20doIt();"
    onmouseover="return blankStatus()">
href event</a>.
```

Script Element

Embedded and External

The script element, in XHTML, should always have a closing tag.

```
// This
<script src=... type="text/javascript"></script>

// Not this
<script src=... type="text/javascript"/>
```

This applies to both to embedded scripts and references to external javascript files.

Use type="text/javascript" not language="javascript"

```
  <!-- Use This -->
  <script type="text/javascript">
  </script>

  <!-- Not This -->
  <script language="javascript">
  </script>
```

The language attribute is deprecated.

Goodman 2004 > JavaScript Bible 5th Edition > P 48

Use <noscript> in every page that contains javascript.

```
<body>
  <noscript>
    <p><strong>Your browser has Javascript
    turned off.</strong></p>

    <p> Some functions on this
    page will not work without Javascript.
    Please turn javascript on.</p>

    <p>For example to turn Javascript on in
    Mozilla:</p>
    <ol>
      <li>Go to Edit &gt; Preferences &gt;
      Advanced &gt; Scripts &amp; Plugins &gt;
      Enable Javascript for</li>
      <li>Ensure there is a check mark next to
      <em>Navigator</em>.</li>
    </ol>
    <hr />
  </noscript>
  <p>Conten here.</p>
</body>
```

Goodman 2004 > JavaScript Bible 5th Edition > p 149

Embedded

Escape script blocks. In final XTHML and in Source XML files.

```
  <script type="text/javascript">
  /* <![CDATA[ */

  /* ]]> */
  </script>
```

Even though we add `/* <![CDATA[ */` during the transform, the CDATA in the source helps prevent the need to escape '&', '<', and '>'.

It is a choice. If you use `/* <![CDATA[ */` you must **not** escape '&', '<', and '>'.

Deprecated: Escape script blocks with comments, <!-- //-->, not <!CDATA[..]]>

```
<!-- Do This -->
<!-- 1 Web OK: IE5+, Moz1.6 -->
<script type="text/javascript">
<!--

//-->
</script>

<!-- Not these bellow -->
<!-- 2 Web OK in: IE5+, Moz1.6+ -->
<script type="text/javascript">
<!-- <!CDATA[

//]]> -->
</script>

<!-- 3 Web OK in: IE5+, Moz1.6+  -->
<script type="text/javascript">
//<![CDATA[

//]]>
</script>

<!-- 4 Web Error in: IE5+, Moz1.6  -->
<script type="text/javascript">
<![CDATA[

]]>
</script>
```

We have four candidate ways of escaping script. We decide on two criteria: that it works in browsers; and it's still well formed xml.

Being well formed xml means characters like "<" or ">", which are frequently used in javascript, need either to be: commented out; written as character references (eg <); or escaped with <![CDATA[…]]>

All four candidate ways are well formed xml.

CC BY-NC-SA 4.0

We can eliminate the fourth as it doesn't work in either browser. The third works in new browsers but doesn't comment out the whole script from an old browser, html point of view. The second works in all browsers and we could use this way.

However, the first way is traditional and, more importantly, simpler. It works in all browsers. Any > is commented out and that's ok by the conformance rules of xml. It is well formed xml.

It also is well formed and works in all broswers after HTMLTidy fucks with it by inserting it's own script escaping.

If you leave the comments out altogether script will still work in Microsoft and Gecko browsers but it will no longer be well formed xml (unless you resort to character referencing, as in >).

Therefore, use the first way, `<!-- ... //-->`

John Bentley Experimentation

External

Scripts can be place in an external file and referenced anywhere within a web page.

```
// External Header.js
document.write("External File referenced from a
header.")

// External Body.js
document.write("External File referenced from a
body.")

// Xhtml file
<html>
  <head>
    <script src="Header.js"
            type="text/javascript"></script>
  </head>
  <body>
    <script src="file:///SomeDir\Body.js"
            type="text/javascript"></script>
  </body>
</html>
```

Use **Error! Hyperlink reference not valid.** at the beginning of absolute references to files.

```
<script type="text/javascript"
src="file:///C:\Data\Dev\Code\EcmaScript\Librari
es\StandardLibrary\EcmaScriptLibrary\Maths.js"><
/script>
```

This will make the reference in Firefox. It is not necessary in IE 6

If your webpage doesn't seem to recognize the <script src=...> then check that your script source file doesn't have bugs.

Functions (and variables) in one external .js file can call functions (and variables) in another .js file as long as both are referenced in the xthml file. The order of the referencing is NOT important: place external files that depend on other external files last in the xhtml file.

```
// file1.js
function getTheMessage() {
  return getTheDeepMessage();
}

// file2.js
function getTheDeepMessage() {
  return "To initiate is to exist";
}

// xhtml file
<html>
  <head>
    <script type="text/javascript"
src="file2.js"></script>
    <script type="text/javascript"
src="file1.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      document.write("<p>" + getTheMessage() +
"</p>");
    </script>
  </body>
</html>
```

Proven not important in IE6 and FF1 1.0.7

Bentley > WebScriptExamples > ReferencingVariables ... & ReferencingFunctions

Code

Refresh

Sometimes you need to force a browser refresh with Shift + R (or F5).

Goodman 2004 > JavaScript Bible 5th Edition

Window Object

References to objects inside the document can omit the window portion.

```
// Equivalent
window.document.getElementById("elementID")
document.getElementById("elementID")
```

Goodman 2004 > JavaScript Bible 5th Edition > P 43

Write Method

Do not use document.write()'s.

Document write() DOES NOT work in XHTML when served as application/xhtml+html. So avoid writes(), at least in production apps, to future proof XHTML and getting into good habits when you write XHTML in the future. Where future = a time when serving XHTML as application/xhtml+html is the standard.

John Bentley > As demonstrated by Chris Bentley.

Event initiated document.write()'s overwrites document content and document.write()'s executed direct from the body or head.

After the page loads the stream is closed so a subsequent write() erases the page contents.

Always end your write()'s with a document.close().

Otherwise images and forms may not appear.

Goodman 2004 > JavaScript Bible 5th Edition > p 92

document.writes can be used in external .js files.

Form Object

If you use the 'name' attribute in the form element then also use 'id'.

```
<form name="frmMain" id="frmMain" action=""
method="post">
```

Under XHTML 1.1 & XHTML 1.0 Strict You must not use the 'name' attribute on the form element but you should use it on form fields.

```
<form id="frmMain" action="" method="post">
 <fieldset>
   <input type="text" id="txtInput"
name="txtInput" />

Note that you will have to refer to your form
using JavaScript of this technique:
<head>
<script type="text/javascript">
  <!--
  window.onload = function () {
    // The form has an id but no name attribute
    // under XHTML 1.1
    // So we provide a handy global reference.
    // Use Pascal case to:
    // 1. Designate it is a Global Variable;
  and
    // 2. So it doesn't conflict with the IE 6
  global form variable eg "frmMain".
    FrmMain = document.forms["frmMain"];
  //-->
</script>
</head>
<body>
<form id="frmMain" onsubmit="return false;"
action="" method="post">
...
```

http://www.w3.org/TR/xhtml-modularization/abstract_modules.html#s_extformsmodule

If you are submitting a form to a server then you will have to use the 'name' attribute on the form element (in addition to the 'id' attribute) and use XHTML 1.0 Transitional.

When submitting a form to a server accessing the form fields via the name property is the only way.

Goodman 2004 > JavaScript Bible 5th Edition > p 100

For form references use the longhand syntax.

```
// Do any of these.
document.forms["formName"] // Recommended.
document.forms[i]

// Not this
document.formName
```

The long hand syntax is the only way to refer to a form element with an 'id' attribute but no 'name' attribute.

Bentley > experiment.
See also > Goodman 2004 > JavaScript Bible 5th Edition > p 92

For form field references use any of the following syntax:

```
document.forms["formName"].elementName
document.forms["formName"].elements["elementName
"]
document.forms["formName"].elements[0]
```

Form fields will have (because you should make them have) an 'id' attribute and 'name' attribute.

Goodman 2004 > JavaScript Bible 5th Edition > p 100

CC BY-NC-SA 4.0

Prevent one field forms from submitting on pressing enter by cancelling the forms onsubmit event

```
<form name="frmMain" onsubmit="return false">
```

Goodman 2004 > JavaScript Bible 5th Edition > p101

Prevent an annoying POSTDATA popup message during development by using the get method. This is method (1) below.

```
<form action="/Php/Apps/BooksDB/thisPage.html"
id="frmMain" method="get" >
```

(1) Always post forms with the GET method unless absolutely necessary to use POST

(2) You can POST a form to a page that has GET variables (i.e. action="thispage.php?var=value" and method="post").

(3) If you decide to use the POST method then make sure the page that handles the POST redirects to itself or somewhere else immediately after you're done with the POST data.

jedhunsaker > Jul Wed 11th 2007 2:26pm > http://forums.mozillazine.org/viewtopic.php?p=3001440&

Frame Object

Reference objects between frames

```
// Parent to Child
function writeCurrentCourse() {
  // Either of these:
   fraMechanics.document.frmMain.txtResult.value
    = "Nice";
frames["fraMechanics"].document.frmMain.txtResul
t.value = "To";
   frames[0].document.frmMain.txtResult.value
    = "See you";
   }

   //-->
   </script>
 </head>
   <frameset cols="25%,75%"
             onload="writeCurrentCourse()"
             frameborder="yes" >
     <frame name="fraMechanics"
            src="history101M.htm" />
     <frame name="fraDescription"
            src="history101D.htm" />
   </frameset>
</html>
```

Events

Concepts

Events are used to either: trigger a single function; reveal information; trigger many functions (propagation).

Events are always executed serially.

Goodman 2004 > JavaScript Bible 5th Edition > p720

Events can propagate with respect to the containment hierarchy in up to three phases:

- Bottom (window, document, html, body) to target's parent (e.g. button's div) ("Event Capture");
- Target phase (e.g. button, the inner most child activated).
- Target's parent (e.g. button's div) to bottom (body, html, document, window) ("Event Bubbling").

https://www.w3.org/TR/DOM-Level-3-Events/#dom-event-architecture
https://www.quirksmode.org/js/events_order.html

Control event propagation in three ways:

1. Don't register an element's event for the capture phase;
2. Stop propagation.
3. Turn off bubbling (Set the bubbles attribute to false)

https://www.w3.org/TR/DOM-Level-3-Events/#dom-event-architecture

Event Bubbling is on by default in W3C and IE4+.

Implementation

General

Concept	IE4+	W3C Browsers
Prevent Propagation	event.cancelBubble = true;	evt.stopPropagation()
on from proceeding past current event handler.		
Prevent Element's default action.	event.returnValue = false;	evt.preventDefault();
Reroute	document.body.fireEvent("onclick",event);	document.body.dispatchEvent(evt);
Event Capture	element.setCapture(); element.releaseCapture();	// True for Event Capture. // False for Event Bubble. document.addEventListener("click", docClick, true); document.removeEventListener("click", docClick, true);

Event Object Referencing

In IE4+ An `event` object is created when an event occurs and only exists for as long as the script processes the event.

Goodman 2004 > JavaScript Bible 5th Edition > p720

Event Binding

From an element's event.

```
<input type="button" name="btn1" value="Button"
onclick="doIt()" />
```

By assigning a function to the object's event property.

```
function doIt() {
  ...
}

document.forms["frmMain"].btn1.onclick = doIt;
```

Using addEventListener (W3C Only).

```
// Add for Event Capture
document.addEventListener("click", doIt, true);

// Add for Event Bubble (the default)
document.forms['frmMain'].addEventListener('clic
k', doIt, false);
```

Event Binding and event Object Referencing

Global event object referencing in IE4+

```
window.event.propertyName;
//or
event.propertyName;
```

From an element's event.

```
<input type="button" name="btn1" value="Button"
onclick="doIt(event)" />
```

Cross Browser Function receiving events

Use "evt", not "event", as a parameter.

```
function doIt(evt) {

}
```

Using "event" will interfere with IE's global "event" object.

Goodman 2004 > JavaScript Bible 5th Edition > p740

Cross-browser "event" object referencing:

```
function checkWhich(evt) {
  var evt = (evt) ? evt : ((window.event) ?
window.event : null);
  if (evt) {
    var el = (evt.target) ? evt.target :
evt.srcElement;
  ...
  }
}
```

Goodman 2004 > JavaScript Bible 5th Edition > p741 & 742

Cross Browser event object properties and methods

Property/Action	NN4	IE4+	W3C DOM
Target element	target	srcElement	target
Event type	type	type	type
X coordinate in element	n/a†	offsetX	n/a†
Y coordinate in element	n/a†	offsetY	n/a†

CC BY-NC-SA 4.0

X coordinate in positioned element	layerX	x	layerX
Y coordinate in positioned element	layerY	y	layerY
X coordinate on page	pageX	n/a†	pageX
Y coordinate on page	pageY	n/a†	pageY
X coordinate in window	n/a	clientX	clientX
Y coordinate in window	n/a	clientY	clientY
X coordinate on screen	screenX	screenX	screenX
Y coordinate on screen	screenY	screenY	screenY
Mouse button	which	button	button
Keyboard key	which	keyCode	charCode
Shift key pressed	modifiers	shiftKey	shiftKey
Alt key pressed	modifiers	altKey	altKey
Ctrl key pressed	modifiers	ctrlKey	ctrlKey
Previous Element	n/a	fromElement	relatedTarget
Next Element	n/a	toElement	relatedTarget
Cancel bubbling	n/a	cancelBubble	preventBubble()
Prevent default action	return false	returnValue	preventDefault()

†Value can be derived by calculation from other properties.

Goodman 2004 > JavaScript Bible 5th Edition > p742

Event Propagation

Event Bubbling (and Propagation?) can occur at these container levels (as an example).

Event Handler Location	WinIE4+	MacIE4+	NN6+/Moz1+/Safari
button	yes	yes	yes
form	yes	yes	yes
body	yes	yes	yes
HTML	yes	no	yes
document	yes	yes	yes
window	no	no	yes

Goodman 2004 > JavaScript Bible 5th Edition > p731

Onload function

Ensure DOM has been loaded

```
window.addEventListener("load", function(){
  main();
});
```

https://stackoverflow.com/a/36096571/872154

Adds a function to the onload event without overwriting previous functions assignments to onload

```
Wsl.Events.addLoadEvent = function(func) {
  if (window.addEventListener) {
    window.addEventListener("load", func,
false);
  } else if (window.attachEvent) {
    window.attachEvent("onload", func);
  } else { // fallback
    var old = window.onload;
    window.onload = function() {
      if (old) old();
      func();
    };
  }
}

addLoadEvent(nameOfSomeFunctionToRunOnPageLoad);
addLoadEvent(function() {
  /* more code to run on page load */
});
```

C:\Users\John\Documents\Sda\Code\EcmaScript\Libraries\StandardLibrary\Web\events.js
https://stackoverflow.com/a/3196789/872154, gblazex's answer to "addLoadEvent is not helping with onload conflict" Based on Simon Willison > Executing JavaScript on page load https://simonwillison.net/2004/May/26/addLoadEvent/

onpageshow function

To ensure the load event is fired every time a user navigates to the page:

```
window.onpageshow = function (evt) {
  if (evt.persisted) window.onload();
}

// Your onload fucntion defined somewhere.

// Don't use
<body onpageshow="if (event.persisted)
onPageShow();">
```

From Firefox 1.5 in memory caching is used. The onload event only fires when the page first loads and not when the user navigates away and back to the page. To ensure code in the onload event fires every time you need to use the onpageshow (for firefox).

Using the mozilla recommended xhtml attribute is not a good idea: it makes the xhtml page invalid.

Bentley > Experiement
See Also > Mozilla Developer Center > Using Firefox 1.5 caching > http://developer.mozilla.org/en/docs/Using_Firefox_1.5_caching

W3C DOM

DOM Basics

The sequence of child nodes is entirely dependent upon the HTML source code order.

Goodman 2004 > JavaScript Bible 5th Edition > P 42

Element Referencing

Give an element an id.

```
<p id="pFirst">... </p>
```

Universal Referencing:

```
var elem = document.getElementById("pFirst");
var elem = document.getElementByTagName("p")[0];
```

Goodman 2004 > JavaScript Bible 5th Edition > p180

By convention you can use the "Prototype" dollar function to ease the referencing of elements:

```
function $() {
  var elements = new Array();
  for (var i = 0; i < arguments.length; i++) {
        var element = arguments[i];
        if (typeof element == 'string')
                element =
document.getElementById(element);
        if (arguments.length == 1)
                return element;
        elements.push(element);
  }
  return elements;
}

// Sample Usage:
var obj1 = document.getElementById('element1');
var obj2 = document.getElementById('element2');
function alertElements() {
  var i;
  var elements =
$('a','b','c',obj1,obj2,'d','e');
  for ( i=0;i<elements.length;i++ ) {
    alert(elements[i].id);
  }
}
```

Dustin Diaz, Top 10 Javascript functions of all time, http://www.dustindiaz.com/top-ten-javascript/

Elements on a Form:

```
document.forms[0].txtMyBox;
document.frmMain.txtMyBox;
document.forms["frmMain"].txtMyBox;
```

Node Hierarchy

The number of third-generation nodes further nested within the family tree does not influence the number of children associated with a parent.

Goodman 2004 > JavaScript Bible 5th Edition > p181

Important W3C Node Properties and Methods (All HTML Elements)

Properties	Methods
attributes[] childNodes[] firstChild id lastChild localName	appendChild(newChild) cloneNode(deep) hasChildNodes() insertBefore(new, ref) isSupported(feature, version)

CC BY-NC-SA 4.0

Properties	Methods
namespaceURI nextSibling nodeName nodeType nodeValue ownerDocument parentNode prefix previousSibling	removeChild(old) replaceChild(new, old) setAttribute("attr",val)

Goodman 2004 > JavaScript Bible 5th Edition > p183-186

Node Types

Element 1; Attribute 2 ; Text 3; Comment 8; Document 9; DocumentType 10; Fragment 11.

See also Goodman 2004 > JavaScript Bible 5th Edition > Table 14-3 W3C DOM HTML-Related Node Types > p181

W3C DOM Node object inheritance

Node
+--Document
| +--**HTMLDocument**
+--CharacterData
| +--Text
| | +--CDATASection
| +--Comment
+--Attr
+--Element
| +--**HTMLElement**
| +-- (Each specific HTML element)
+--DocumentType
+--DocumentFragment
+--Notation
+--Entity
+--Entity Reference
+--ProcessingInstruction

Goodman 2004 > JavaScript Bible 5th Edition > p185

W3C DOM Coding Techniques

Remove Node

Add this to your library and call it:

```
// Remove all the child nodes for a particular
// element.
function removeElementChildren(elementId){
  var element =
    document.getElementById(elementId);

  // We must walk backwards through the
  // collection as the length changes.
  for (i = element.childNodes.length - 1; i >=0;
    i--) {

    currentNode = element.childNodes[i]
currentNode.parentNode.removeChild(currentNode);
  }
}
```

Create & Insert Node

Important W3C Document Object properties and Methods:

Properties	Methods
	createElement(elementType) createTextNode(text)

```
function createAndInsertNode() {

var element = document.createElement("i");
element.setAttribute("id","newElement");
var textNode = document.createTextNode(" Some
new text");
element.appendChild(textNode);
document.getElementById("insertPlace").appendChi
ld(element);
}

function
createAndInsertTextNode(existingElementTagName,
text) {
  var textNode = document.createTextNode(text);

document.getElementsByTagName(existingElementTag
Name)[0].appendChild(textNode);
}
```

Replace Node Content: W3C

```
<p id="pThird">Some of the content will be
replaced: <em id="replacePlace">This old stuff
will be replaced.</em></p>

function replaceNode() {
 var textNode = document.createTextNode("New
  text and all is well");
 var newChildNode = document.createElement("i");

 newChildNode.appendChild(textNode);

 var oldChildNode =
  document.getElementById("replacePlace");

document.getElementById("pThird").replaceChild(n
ewChildNode, oldChildNode);
}


<p id="pThird">Some of the content will be
replaced: <i>New text and all is well</i></p>
```

Replace Node Content: innerHTML

Note this is a propriety property that has found widespread support.

```
document.getElementById("pThird").innerHTML =
"Some of the content will be replaced
(innerHTML): <i>New text and all is well.
(innerHTML)</i>";
```

Replace Text Content: W3C

```
document.getElementById("replaceTextPlace").chil
dNodes[0].nodeValue = "Super Groovy Text";

document.getElementsByTagName('output')[0].child
Nodes[0].nodeValue = "nice";
```

Goodman 2004 > JavaScript Bible 5th Edition > p188

Replace Text Content: innerHTML

```
document.getElementById("replaceTextPlace").inne
rHTML = "Super Groovy Text (innerHTML)";
```

Debugging

Use Firefox

To test a series of single expressions use the Firefox add ons: Console[2]; or Firebug

Firefox debugging messages are usually more informative the IE. For example they correctly identify bugs in the embedded javascript file.

In the Firefox Browser > Tools > Web Development > Javascript Console.

The Firefox Browser

Test a single expression in any web browser by prefacing your expression with "javascript:" in the address bar.

```
javascript: 1 + 1
```

Use the Firefox Venkman **Javascript debugger**.

It's better than any Microsoft debugger, including Visual Studio 2002, as of 12 Jun 2004. Why? Venkman can debug JavaScript files references within a (X)HTML page. Visual Studio 2002 can't.

In the Firefox Browser > Tools > Web Development > Javascript Debugger.

The Firefox Browser

Customize Firefox by adding these bookmarks to your toolbar:

⇒ "JS Debugger" >> x-jsd:debugger

⇒ "Current DOM" >> javascript: "<textarea cols=120 rows=40>" + document.body.parentNode.innerHTML + "</textarea>"

⇒ "JS Console" >> javascript:

Use Bookmarks > Manage Bookmarks

Goodman 2004 > JavaScript Bible 5th Edition > p 86
Venkman Faq > http://www.hacksrus.com/~ginda/venkman/faq/venkman-faq.html#2.1

The Goodman Evaluator

Use Danny Goodman's embeddable evaluator to test & evaluate document expressions & variables.

1. Copy V:\Listings\Chap45\evaluator.js from the Goodman Bonus CD to your project directory.
2. Link the library into your page with the following tag set in the HEAD portion of your document:
```
<script type="text/javascript"
src="evaluator.js"></script>
```
3. Then bring the evaluator into the rendered page by adding the following to the very bottom of your HTML document:
```
<script type="text/javascript">
printEvaluator();
</script>
```
4. Use the embedded evaluator to access various values in your page.

Goodman 2004 > JavaScript Bible 5th Edition > p BC291

Use Danny Goodman's embeddable evaluator trace function.

1. Install Goodman's embeddable evaluator as above.
2. In your source code add a traceSwitch set to true and call the trace function.
```
// In the source code:
function main() {

  // Image(width, height)
  var width = 370;
  var traceSwitch = true;
```

CC BY-NC-SA 4.0

```
   trace(traceSwitch,"width",width);

// Output to results text box when page loaded
in browser with embedded  evaluator:
In  main(): width=370
```

Other Debugging Techniques

Use alerts and "window.status =" to evaluate expressions in script along the way.

Goodman 2004 > JavaScript Bible 5th Edition > p 86

Use emergency evaluation by typing in the address bar.

```
javascript:alert(myFunction("myParam1"))
```

Cross Browser Technique

Philosophy

Always use <noscript> for browser that don't support scripting (or scripting turned off).

Use standards compatible mode by having a DOCTYPE switch at the beginning of your page.

Goodman 2004 > JavaScript Bible 5th Edition > p199

Cope with different browsers using these possible strategies:

1. Use lowest common denominator code.
2. Use Object Detection.
3. Use Browser Version detection.
4. Use a branch index page. *Goodman 2004 > JavaScript Bible 5th Edition > p150*

Use object detection over browser version detection.

> The bottom line, then, is letting your scripts decide how to perform actions based on the browser version is not a good idea. Instead, the scripts should be smart enough to act based on the capabilities of the browser that is currently running the script.

Goodman 2004 > JavaScript Bible 5th Edition > BC320
Goodman 2004 > JavaScript Bible 5th Edition > p155

Don't use object detection FOR browser version detection.

The ideal page is one that displays useful content on any browser, but whose scripting enhances the experience of the page visitor.

Goodman 2004 > JavaScript Bible 5th Edition > p155

Don't try to support all possible browsers. Limit yourself to the most popular, Eg IE5+, plus W3C compatible.

Goodman 2004 > JavaScript Bible 5th Edition > BC71

Broswer (Version) Detection

Use qualifyBroswer() from BrowserDetection.js

Bentley > WebScriptLibrary

Object Detection

Simple

```
if (document.images) {
// statements to work with image objects
}
```

Goodman 2004 > JavaScript Bible 5th Edition > BC323

If the property might return zero or zero-length string then you need to use `typeof` and "undefined".

```
if (typeof document.body.scroll != "undefined")
{ // Do stuff depending on body's scroll }
```

Goodman 2004 > JavaScript Bible 5th Edition > 155

Use this for methods too.

```
if (typeof XmlDoc.setProperty != "undefined" ) {
 XmlDoc.setProperty("SelectionLanguage",
"XPath");
}
```

When object detecting test for the first suspect object and no deeper.

```
' Do this
if (document.styleSheets[0].cssRules) { …

' Not this
if (document.styleSheets[0].cssRules[0]) {
```

Bentley > Experiment

Custom APIs

In the context of designing a cross-browser DHTML page, an API can offer single function that smoothes over the differences in object references and/or property names among several platforms.

```
function getObject(obj) {
  var theObj;
  if (document.layers) {
    if (typeof obj == "string") {
      // just one layer deep
      return document.layers[obj];
    } else {
      // can be a nested layer
      return obj;
    }
  }

  if (document.all) {
    if (typeof obj == "string") {
      return document.all(obj).style;
    } else {
      return obj.style;
    }
  }

  if (document.getElementById) {
    if (typeof obj == "string") {
      return document.getElementById(obj).style;
    } else {
      return obj.style;
    }
  }

  return null;
}
```

Your custom function provides a single access point that is consistent across all platforms.

Goodman 2004 > JavaScript Bible 5th Edition > BC324

Techniques

Data Entry Validation

The different types and places for Data validation:

1. For Each Key Press (onKeyPress).
2. Leaving the field (onChange).
3. Prior to Submission to Server. Allows field cross checking. (onSubmit)
4. On the server.

You have two opportunities to perform data-entry validation in a form:as the user enters data into a form and just before the form is submitted. I recommend you take advantage of both of these opportunities.

Goodman 2004 > JavaScript Bible 5th Edition > BC213

The most convenient time to catch an error is immediately after the user makes it [after leaving the field].

Goodman 2004 > JavaScript Bible 5th Edition > BC214

For validation "after-leaving-the-field" use `onchange`

Goodman 2004 > JavaScript Bible 5th Edition > BC214

Always use validation just prior to submission to the server as the user can defeat the `onchange` event.

Goodman 2004 > JavaScript Bible 5th Edition > BC214

My preferred sequence is to start with examinations that require less work and increase the intensity of validation detective work with succeeding functions.

Goodman 2004 > JavaScript Bible 5th Edition > BC222

You should set focus and select field that caused problem to guide the user.

Goodman 2004 > JavaScript Bible 5th Edition > BC227

Bentley > WebScriptLibrary
C:\Data\Dev\Code\EcmaScript\Libraries\StandardLibrary\WebScriptLibrary\ValidationAllClientSideTypes.html

Passing Data Between Pages

1. Search string of a URL.
2. Cookies.
3. Variables in framesetting documents.

Goodman 2004 > JavaScript Bible 5th Edition > 496

Cookies allow you to persist values between browser sessions. Search strings do not.

URL search Strings

See Bentley > WebScriptLibrary > passingDataBetweenPages_ViaUrl_01.html

& Goodman 2004 > JavaScript Bible 5th Edition > p 495

Maximum limit of URL string??

Cookies

Use Bill Dortches Cookie Functions.

Bentley > WebScriptLibrary > Cookies.html

Cookie lasts only for the session if you do not set an expiry date.

Goodman 2004 > JavaScript Bible 5th Edition > p 521

Check if user has cookies enabled:

```
navigator.cookieEnabled
```

To get around the Netscape and IE a limit of 20 cookie entries (name/value pairs) for each domain see Goodman's decision helper.

Even so there will be a limit of:
2000 characters per name/value pair and 4000 total characters for a domain's combined cookies.

Goodman 2004 > JavaScript Bible 5th Edition > p 522, 528,

More Techniques

⇒ Bentley > WebScriptLibrary
A web script Library by John Bentley.
C:\Data\Dev\Code\EcmaScript\Libraries\StandardLibrary\WebScriptLibrary\index.html

⇒ Bentley > WebScriptExamples
Web Scripting Examples by John Bentley.
C:\Data\Dev\Code\EcmaScript\ExamplesAndSamples\WebScriptExamples\

Sources

⇒ Microsoft > Jscript Guide and Reference
Official Jscript implementation source
Cited: 27 Feb 2004
http://msdn.microsoft.com/library/en-us/script56/html/js56jsorijScript.asp?frame=true

⇒ Netscape > JavaScript Guide And Reference
Official JavaScript implementation source
Cited: 27 Feb 2004
http://devedge.netscape.com/central/javascript/

⇒ Netscape > Core JavaScript Guide 1.5
Netscape, Core JavaScript Guide 1.5, Last updated 28 Sep 2000, Accessed 27 Feb 2004
http://devedge.netscape.com/central/javascript/

⇒ ECMA > Scripting Standard 262, 3rd Edition
Official ECMA scripting standard source
Last Dated: Dec 1999
http://www.ecma-international.org/publications/standards/Ecma-262.htm
Cited: 27 Feb 2004

⇒ W3Schools > JavaScript Tutorial
Terse but powerful free online tutorial. Good on showing EcmaScript in the context of the browser with many fundamental technique examples.
Last Dated: Dec1999
http://www.w3schools.com/js/default.asp
Cited: 27 Feb 2004

⇒ Goodman 2004 > JavaScript Bible 5th Edition
Danny Goodman, Mar 2004.
Excellent introduction and reference on using EcmaScript in browsers. Has a core language reference. Shows cross browser techniques that follow the standard as far as possible but use multi-propriety features where necessary.
"BC" stands for "Bonus Chapter Page"

⇒ The Firefox Browser
The leading W3C standards compliant web browser with excellent javascript debugging facilities.
http://www.mozilla.org/products/firefox/

Document Licence

EcmaScript - In Web - Quick Reference © 2021 by John Bentley is licensed under Attribution-NonCommercial-ShareAlike 4.0 International