**Javascript cheatsheet**

**Chapter 1 BOM Operations**

****

****

BOM is browser object model. Everything you can think of for a page implemented in browser, such as window, screen, location, history, navigator, timing, cookies.

**1.1 Window Object**

The window object represents an open window in a browser. It is one of the most useful object and very easy to understand and to learn as a beginner.

If a document contain frames (<frame> or <iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

**1.1.1 Window Object Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [alert()](http://www.w3schools.com/jsref/met_win_alert.asp) | Displays an alert box with a message and an OK button |
| [blur()](http://www.w3schools.com/jsref/met_win_blur.asp) | Removes focus from the current window |
| [clearInterval()](http://www.w3schools.com/jsref/met_win_clearinterval.asp) | Clears a timer set with setInterval() |
| [clearTimeout()](http://www.w3schools.com/jsref/met_win_cleartimeout.asp) | Clears a timer set with setTimeout() |
| [close()](http://www.w3schools.com/jsref/met_win_close.asp) | Closes the current window |
| [confirm()](http://www.w3schools.com/jsref/met_win_confirm.asp) | Displays a dialog box with a message and an OK and a Cancel button |
| [createPopup()](http://www.w3schools.com/jsref/met_win_createpopup.asp) | Creates a pop-up window |
| [focus()](http://www.w3schools.com/jsref/met_win_focus.asp) | Sets focus to the current window |
| [moveBy()](http://www.w3schools.com/jsref/met_win_moveby.asp) | Moves a window relative to its current position |
| [moveTo()](http://www.w3schools.com/jsref/met_win_moveto.asp) | Moves a window to the specified position |
| [open()](http://www.w3schools.com/jsref/met_win_open.asp) | Opens a new browser window |
| [print()](http://www.w3schools.com/jsref/met_win_print.asp) | Prints the content of the current window |
| [prompt()](http://www.w3schools.com/jsref/met_win_prompt.asp) | Displays a dialog box that prompts the visitor for input |
| [resizeBy()](http://www.w3schools.com/jsref/met_win_resizeby.asp) | Resizes the window by the specified pixels |
| [resizeTo()](http://www.w3schools.com/jsref/met_win_resizeto.asp) | Resizes the window to the specified width and height |
| scroll() |  |
| [scrollBy()](http://www.w3schools.com/jsref/met_win_scrollby.asp) | Scrolls the content by the specified number of pixels |
| [scrollTo()](http://www.w3schools.com/jsref/met_win_scrollto.asp) | Scrolls the content to the specified coordinates |
| [setInterval()](http://www.w3schools.com/jsref/met_win_setinterval.asp) | Calls a function or evaluates an expression at specified intervals (in milliseconds) |
| [setTimeout()](http://www.w3schools.com/jsref/met_win_settimeout.asp) | Calls a function or evaluates an expression after a specified number of milliseconds |

1. Alert(), prompt(), confirm(), open(), close(), blur(), focus() we might have seen them before. They are skipped here.

2. [print()](http://www.w3schools.com/jsref/met_win_print.asp), brings up the file print dialog from the operating system.

3. setInterval(function, time), clearInternal(var)

*Ex1.*

*<!DOCTYPE html>*

*<html>*

*<body>*

*<input type="text" id="clock" />*

*<script>*

*var int=self.setInterval(function(){clock()},1000);*

*function clock()*

*{*

*var d=new Date();*

*var t=d.toLocaleTimeString();*

*document.getElementById("clock").value=t;*

*}*

*</script>*

*<button onclick="int=window.clearInterval(int)">Stop</button>*

*</body>*

*</html>*

4. setTimeout and clearTimeout(var);

*Ex. 1*

*var t=setTimeout(function(){*

*alert("Hello")*

*},3000);*

*Ex 2.*

*<html>  
<head>  
<script>  
var c=0;  
var t;  
var timer\_is\_on=0;  
  
function timedCount() {  
 document.getElementById('txt').value=c;  
 c=c+1;  
 t=setTimeout(function(){timedCount()},1000);  
 }*

*function doTimer() {  
 if (!timer\_is\_on) {  
 timer\_is\_on=1;  
 timedCount();  
 }  
}*

*function stopCount() {  
 clearTimeout(t);  
 timer\_is\_on=0;  
}  
</script>  
</head>  
<body>  
<form>  
<input type="button" value="Start count!" onclick="doTimer()">  
<input type="text" id="txt">  
<input type="button" value="Stop count!" onclick="stopCount()">  
</form>  
</body>  
</html>*

5. resizeBy(), resizeTo(), scrollBy() and scrollTo()

They are easy to understand, and they specify pixels.

6. moveBy(), moveTo()

They specify coordinates.

7. createPopup() only works in IE, so you can forget it now.

**1.1.2 Window Object Properties**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [self](http://www.w3schools.com/jsref/prop_win_self.asp) | | | | Returns the current window | | | | |
| [parent](http://www.w3schools.com/jsref/prop_win_parent.asp) | | | | Returns the parent window of the current window | | | | |
| [opener](http://www.w3schools.com/jsref/prop_win_opener.asp) | | | | Returns a reference to the window that created the window | | | | |
| [document](http://www.w3schools.com/jsref/dom_obj_document.asp) | | | | | | Returns the Document object for the window ([See Document object](http://www.w3schools.com/jsref/dom_obj_document.asp)) | | |
| [top](http://www.w3schools.com/jsref/prop_win_top.asp) | Returns the topmost browser window | | | | | | | |
| [history](http://www.w3schools.com/jsref/obj_history.asp) | | | | Returns the History object for the window ([See History object](http://www.w3schools.com/jsref/obj_history.asp)) | | | | |
| [location](http://www.w3schools.com/jsref/obj_location.asp) | | | Returns the Location object for the window ([See Location object](http://www.w3schools.com/jsref/obj_location.asp)) | | | | | |
| [navigator](http://www.w3schools.com/jsref/obj_navigator.asp) | | | | | Returns the Navigator object for the window ([See Navigator object](http://www.w3schools.com/jsref/obj_navigator.asp)) | | | |
| [screen](http://www.w3schools.com/jsref/obj_screen.asp) | | | | Returns the Screen object for the window [(See Screen object)](http://www.w3schools.com/jsref/obj_screen.asp) | | | | |
| [defaultStatus](http://www.w3schools.com/jsref/prop_win_defaultstatus.asp) | | | | | | | | Sets or returns the default text in the statusbar of a window |
| [status](http://www.w3schools.com/jsref/prop_win_status.asp) | | | | Sets the text in the statusbar of a window | | | | |
| [name](http://www.w3schools.com/jsref/prop_win_name.asp) | | Sets or returns the name of a window | | | | | | |
| [frames](http://www.w3schools.com/jsref/prop_win_frames.asp) | | | | Returns an array of all the frames (including iframes) in the current window | | | | |
| [closed](http://www.w3schools.com/jsref/prop_win_closed.asp) | | | | Returns a Boolean value indicating whether a window has been closed or not | | | | |
| [length](http://www.w3schools.com/jsref/prop_win_length.asp) | | | | Returns the number of frames (including iframes) in a window | | | | |
| [innerHeight](http://www.w3schools.com/jsref/prop_win_innerheight.asp) | | | | | | | Sets or returns the inner height of a window's content area | |
| [innerWidth](http://www.w3schools.com/jsref/prop_win_innerheight.asp) | | | | | | | Sets or returns the inner width of a window's content area | |
| [outerHeight](http://www.w3schools.com/jsref/prop_win_outerheight.asp) | | | | | | | Sets or returns the outer height of a window, including toolbars/scrollbars | |
| [outerWidth](http://www.w3schools.com/jsref/prop_win_outerheight.asp) | | | | | | | Sets or returns the outer width of a window, including toolbars/scrollbars | |
| [pageXOffset](http://www.w3schools.com/jsref/prop_win_pagexoffset.asp) | | | | | | | Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window | |
| [pageYOffset](http://www.w3schools.com/jsref/prop_win_pagexoffset.asp) | | | | | | | Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window | |
| [screenLeft](http://www.w3schools.com/jsref/prop_win_screenleft.asp) | | | | | | | Returns the x coordinate of the window relative to the screen | |
| [screenTop](http://www.w3schools.com/jsref/prop_win_screenleft.asp) | | | | | | | Returns the y coordinate of the window relative to the screen | |
| [screenX](http://www.w3schools.com/jsref/prop_win_screenx.asp) | | | | | | | Returns the x coordinate of the window relative to the screen | |
| [screenY](http://www.w3schools.com/jsref/prop_win_screenx.asp) | | | | | | | Returns the y coordinate of the window relative to the screen | |

1. History object

History object has length property and three methods, forward(), back(), and go().

For example, Window.history.forward()

Ex. Go back 2 pages.

*<html>  
<head>  
<script>  
function goBack() {  
 window.history.go(-2)  
}  
</script>  
</head>  
<body>  
<input type="button" value="Go back 2 pages" onclick="goBack()">  
</body>  
</html>*

**1.2 Document object**

Please look at the spec of document object. It can control the html document root node.

**1.3 Screen Object**

The screen object contains information about the visitor's screen which is the OS physical screen.

**1.3.1 Screen Object Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [availHeight](http://www.w3schools.com/jsref/prop_screen_availheight.asp) | Returns the height of the screen (excluding the Windows Taskbar) |
| [availWidth](http://www.w3schools.com/jsref/prop_screen_availwidth.asp) | Returns the width of the screen (excluding the Windows Taskbar) |
| [colorDepth](http://www.w3schools.com/jsref/prop_screen_colordepth.asp) | Returns the bit depth of the color palette for displaying images |
| [height](http://www.w3schools.com/jsref/prop_screen_height.asp) | Returns the total height of the screen |
| [pixelDepth](http://www.w3schools.com/jsref/prop_screen_pixeldepth.asp) | Returns the color resolution (in bits per pixel) of the screen |
| [width](http://www.w3schools.com/jsref/prop_screen_width.asp) | Returns the total width of the screen |

**1.4 Location Object**

The location object contains information about the current URL.

The location object is part of the window object and is accessed through the window.location property.

**1.4.1 Location Object Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [hash](http://www.w3schools.com/jsref/prop_loc_hash.asp) | Returns the anchor portion of a URL |
| [host](http://www.w3schools.com/jsref/prop_loc_host.asp) | Returns the hostname and port of a URL |
| [hostname](http://www.w3schools.com/jsref/prop_loc_hostname.asp) | Returns the hostname of a URL |
| [href](http://www.w3schools.com/jsref/prop_loc_href.asp) | Returns the entire URL |
| [pathname](http://www.w3schools.com/jsref/prop_loc_pathname.asp) | Returns the path name of a URL |
| [port](http://www.w3schools.com/jsref/prop_loc_port.asp) | Returns the port number the server uses for a URL |
| [protocol](http://www.w3schools.com/jsref/prop_loc_protocol.asp) | Returns the protocol of a URL |
| [search](http://www.w3schools.com/jsref/prop_loc_search.asp) | Returns the query portion of a URL |

**1.4.2 Location Object Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [assign()](http://www.w3schools.com/jsref/met_loc_assign.asp) | Loads a new document |
| [reload()](http://www.w3schools.com/jsref/met_loc_reload.asp) | Reloads the current document |
| [replace()](http://www.w3schools.com/jsref/met_loc_replace.asp) | Replaces the current document with a new one |

**1.5. Navigator Object**

The navigator object contains information about the browser.

**1.5.1 Navigator Object Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [appCodeName](http://www.w3schools.com/jsref/prop_nav_appcodename.asp) | Returns the code name of the browser |
| [appName](http://www.w3schools.com/jsref/prop_nav_appname.asp) | Returns the name of the browser |
| [appVersion](http://www.w3schools.com/jsref/prop_nav_appversion.asp) | Returns the version information of the browser |
| [cookieEnabled](http://www.w3schools.com/jsref/prop_nav_cookieenabled.asp) | Determines whether cookies are enabled in the browser |
| [onLine](http://www.w3schools.com/jsref/prop_nav_online.asp) | Boolean, returns *true* if the browser is on line, otherwise *false*. |
| [platform](http://www.w3schools.com/jsref/prop_nav_platform.asp) | Returns for which platform the browser is compiled |
| [userAgent](http://www.w3schools.com/jsref/prop_nav_useragent.asp) | Returns the user-agent header sent by the browser to the server |

**1.5.2 Navigator Object Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [javaEnabled()](http://www.w3schools.com/jsref/met_nav_javaenabled.asp) | Specifies whether or not the browser has Java enabled |
| [taintEnabled()](http://www.w3schools.com/jsref/met_nav_taintenabled.asp) | Specifies whether or not the browser has data tainting enabled |

Finish this chapter. And write some simple demo for chapter 1.

Create a small window and let your mouse move and let the window go with your mouse.

Reference:

HTML DOM Document Object

<http://www.w3schools.com/jsref/dom_obj_document.asp>

HTML DOM Element Object

<http://www.w3schools.com/jsref/dom_obj_all.asp>

HTML DOM Attributes Object

<http://www.w3schools.com/jsref/dom_obj_attributes.asp>

HTML DOM Events

<http://www.w3schools.com/jsref/dom_obj_attributes.asp>

HTML Objects

<http://www.w3schools.com/jsref/dom_obj_anchor.asp>

Window Object

http://www.w3schools.com/jsref/obj\_window.asp

**Chapter 2 HTML DOM OBJECT**

We will only introduce a few HTML DOM Objects.

**2.1 Anchor Object**

* The Anchor object represents an HTML hyperlink.
* For each <a> tag in an HTML document, an Anchor object is created.
* An anchor allows you to create a link to another document (with the href attribute), or to a different point in the same document (with the name attribute).
* You can access an anchor by using getElementById(), or by searching through the anchors collection property of the Document object.

**2.1.1 Anchor Object Properties**

**W3C:** W3C Standard.

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **W3C** |
| [charset](http://www.w3schools.com/jsref/prop_anchor_charset.asp) | Sets or returns the value of the charset attribute of a link | Yes |
| [href](http://www.w3schools.com/jsref/prop_anchor_href.asp) | Sets or returns the value of the href attribute of a link | Yes |
| [hreflang](http://www.w3schools.com/jsref/prop_anchor_hreflang.asp) | Sets or returns the value of the hreflang attribute of a link | Yes |
| [name](http://www.w3schools.com/jsref/prop_anchor_name.asp) | Sets or returns the value of the name attribute of a link | Yes |
| [rel](http://www.w3schools.com/jsref/prop_anchor_rel.asp) | Sets or returns the value of the rel attribute of a link | Yes |
| [rev](http://www.w3schools.com/jsref/prop_anchor_rel.asp) | Sets or returns the value of the rev attribute of a link | Yes |
| [target](http://www.w3schools.com/jsref/prop_anchor_target.asp) | Sets or returns the value of the target attribute of a link | Yes |
| [type](http://www.w3schools.com/jsref/prop_anchor_type.asp) | Sets or returns the value of the type attribute of a link | Yes |

**Ex.**

*<html>  
<body>  
<h2>Chapter 1</h2>  
<p>This chapter explains ba bla bla</p>  
  
<h2>Chapter 2</h2>  
<p>This chapter explains ba bla bla</p>  
  
<h2><a id="c3" name="C3">Chapter 3</a></h2>  
<p>This chapter explains ba bla bla</p>  
  
<script>  
document.write("Return name of anchor: ");  
document.write(document.getElementById("c3").name);  
</script>  
</body>  
</html>*

**2.2 Object Object**

* The Object object represents an HTML object element.
* The <object> tag is used to include objects such as images, audio, videos, Java applets, ActiveX, PDF, and Flash into a webpage.

**2.2.1 Object Properties**

**W3C:** W3C Standard.

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **W3C** |
| align | Sets or returns the alignment of the object according to the surrounding text | Yes |
| archive | Sets or returns a string that can be used to implement your own archive functionality for the object | Yes |
| border | Sets or returns the border around the object | Yes |
| code | Sets or returns the URL of the file that contains the compiled Java class | Yes |
| codeBase | Sets or returns the URL of the component | Yes |
| codeType |  | Yes |
| data |  | Yes |
| declare |  | Yes |
| form | Returns a reference to the object's parent form | Yes |
| height | Sets or returns the height of the object | Yes |
| hspace | Sets or returns the horizontal margin of the object | Yes |
| name | Sets or returns the name of the object | Yes |
| standby | Sets or returns a message when loading the object | Yes |
| type | Sets or returns the content type for data downloaded via the data attribute | Yes |
| useMap |  | Yes |
| vspace | Sets or returns the vertical margin of the object | Yes |
| width | Sets or returns the width of the object | Yes |

**2.3 Frame Object and IFrame Object**

* The Frame object represents an HTML frame.
* The <frame> tag defines one particular window (frame) within a frameset.
* For each <frame> tag in an HTML document, a Frame object is created.
* The IFrame object represents an HTML inline frame.
* The <iframe> tag defines an inline frame that contains another document.
* For each <iframe> tag in an HTML document, an IFrame object is created.

**2.4 DOM Elements**

Every tag in html is an element. Once you retrieve one element. You want to operate on it. Here is what you can do, I will just list a few common ones.

|  |  |
| --- | --- |
| *[element](http://www.w3schools.com/jsref/prop_html_innerhtml.asp)*[.innerHTML](http://www.w3schools.com/jsref/prop_html_innerhtml.asp) | Sets or returns the content of an element |
| *[element](http://www.w3schools.com/jsref/prop_html_id.asp)*[.id](http://www.w3schools.com/jsref/prop_html_id.asp) | Sets or returns the id of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_nodename.asp)*[.nodeName](http://www.w3schools.com/jsref/prop_node_nodename.asp) | Returns the name of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_nodetype.asp)*[.nodeType](http://www.w3schools.com/jsref/prop_node_nodetype.asp) | Returns the node type of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_nodevalue.asp)*[.nodeValue](http://www.w3schools.com/jsref/prop_node_nodevalue.asp) | Sets or returns the value of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_ownerdocument.asp)*[.ownerDocument](http://www.w3schools.com/jsref/prop_node_ownerdocument.asp) | Returns the root element (document object) for an element |
| *[element](http://www.w3schools.com/jsref/met_element_removeattribute.asp)*[.removeAttribute()](http://www.w3schools.com/jsref/met_element_removeattribute.asp) | Removes a specified attribute from an element |
| *[element](http://www.w3schools.com/jsref/met_element_removeattributenode.asp)*[.removeAttributeNode()](http://www.w3schools.com/jsref/met_element_removeattributenode.asp) | Removes a specified attribute node, and returns the removed node |
| *[element](http://www.w3schools.com/jsref/met_node_removechild.asp)*[.removeChild()](http://www.w3schools.com/jsref/met_node_removechild.asp) | Removes a child node from an element |
| *[element](http://www.w3schools.com/jsref/met_node_replacechild.asp)*[.replaceChild()](http://www.w3schools.com/jsref/met_node_replacechild.asp) | Replaces a child node in an element |
| *[element](http://www.w3schools.com/jsref/prop_node_firstchild.asp)*[.firstChild](http://www.w3schools.com/jsref/prop_node_firstchild.asp) | Returns the first child of an element |
| *[element](http://www.w3schools.com/jsref/prop_html_title.asp)*[.title](http://www.w3schools.com/jsref/prop_html_title.asp) | Sets or returns the title attribute of an element |
| *element*.toString() | Converts an element to a string |
| *[element](http://www.w3schools.com/jsref/prop_element_tagname.asp)*[.tagName](http://www.w3schools.com/jsref/prop_element_tagname.asp) | Returns the tag name of an element |
| *element*.style | Sets or returns the style attribute of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_previoussibling.asp)*[.previousSibling](http://www.w3schools.com/jsref/prop_node_previoussibling.asp) | Returns the previous element at the same node tree level |
| *[element](http://www.w3schools.com/jsref/prop_node_parentnode.asp)*[.parentNode](http://www.w3schools.com/jsref/prop_node_parentnode.asp) | Returns the parent node of an element |
| *[element](http://www.w3schools.com/jsref/prop_node_childnodes.asp)*[.childNodes](http://www.w3schools.com/jsref/prop_node_childnodes.asp) | Returns a NodeList of child nodes for an element |
| *[element](http://www.w3schools.com/jsref/prop_html_classname.asp)*[.className](http://www.w3schools.com/jsref/prop_html_classname.asp) | Sets or returns the class attribute of an element |
| *[element](http://www.w3schools.com/jsref/met_node_clonenode.asp)*[.cloneNode()](http://www.w3schools.com/jsref/met_node_clonenode.asp) | Clones an element |
| *[element](http://www.w3schools.com/jsref/met_node_appendchild.asp)*[.appendChild()](http://www.w3schools.com/jsref/met_node_appendchild.asp) | Adds a new child node, to an element, as the last child node |
| *[element](http://www.w3schools.com/jsref/prop_html_dir.asp)*[.dir](http://www.w3schools.com/jsref/prop_html_dir.asp) | Sets or returns the text direction of an element |
| *[element](http://www.w3schools.com/jsref/met_element_setattribute.asp)*[.setAttribute()](http://www.w3schools.com/jsref/met_element_setattribute.asp) | Sets or changes the specified attribute, to the specified value |
| *[element](http://www.w3schools.com/jsref/met_element_setattributenode.asp)*[.setAttributeNode()](http://www.w3schools.com/jsref/met_element_setattributenode.asp) | Sets or changes the specified attribute node |

**Chapter 3 HTML DOM and XML DOM**

3.1 DOM overview

The DOM defines a standard for accessing HTML and XML documents:

The W3C DOM standard is separated into 3 different parts:

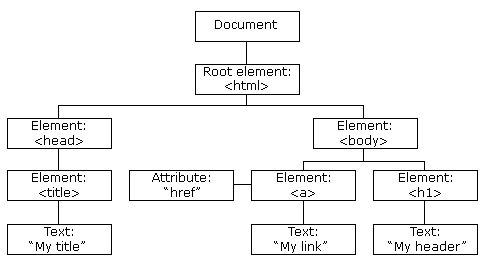
* Core DOM - standard model for any structured document
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

The XML DOM defines the **objects** and **properties** of all XML elements, and the **methods** to access them. The HTML DOM is a standard for how to get, change, add, or delete HTML elements. Here you need to understand document object, object object, anchors, links and forms and images collection.

**DOM Nodes**

According to the W3C HTML DOM standard, everything in an HTML document is a node:

* The entire document is a document node
* Every HTML element is an element node
* The text inside HTML elements are text nodes
* Every HTML attribute is an attribute node
* Comments are comment nodes



You should understand Node Parents, Children, and Siblings

**HTML DOM Methods**

The HTML DOM can be accessed with JavaScript (and other programming languages).

All HTML elements are defined as objects, and the programming interface is the object methods and object properties .

A **method** is an action you can do (like add or modify an element).

A **property** is a value that you can get or set (like the name or content of a node).

Ex.

*var element=document.getElementById("intro");*

**HTML DOM Objects - Methods and Properties**

Here are some of the (most common) methods you will learn about in this tutorial:

|  |  |
| --- | --- |
| **Method** | **Description** |
| getElementById() | Returns the element that has an ID attribute with the a value |
| getElementsByTagName() | Returns a node list (collection/array of nodes) containing all elements with a specified tag name |
| getElementsByClassName() | Returns a node list containing all elements with a specified class |
|  |  |
| appendChild() | Adds a new child node to a specified node |
| removeChild() | Removes a child node |
| replaceChild() | Replaces a child node |
| insertBefore() | Inserts a new child node before a specified child node |
|  |  |
| createAttribute() | Creates an Attribute node |
| createElement() | Creates an Element node |
| createTextNode() | Creates a Text node |
|  |  |
| getAttribute() | Returns the specified attribute value |
| setAttribute() | Sets or changes the specified attribute, to the specified value |

Ex.

*<html>  
<body>  
<p id="intro">Hello World!</p>  
  
<script>  
var txt=document.getElementById("intro").innerHTML;  
document.write(txt);  
</script>  
  
</body>  
</html>*

**The nodeName Property**

The nodeName property specifies the name of a node.

* nodeName is read-only
* nodeName of an element node is the same as the tag name
* nodeName of an attribute node is the attribute name
* nodeName of a text node is always #text
* nodeName of the document node is always #document

**Note:** nodeName always contains the uppercase tag name of an HTML element.

**The nodeValue Property**

The nodeValue property specifies the value of a node.

* nodeValue for element nodes is undefined
* nodeValue for text nodes is the text itself
* nodeValue for attribute nodes is the attribute value

Ex.1

*<html>  
<body>  
<p id="intro">Hello World!</p>  
<script type="text/javascript">  
x=document.getElementById("intro");  
document.write(x.firstChild.nodeValue);  
</script>  
</body>  
</html>*

**The nodeType Property**

The nodeType property returns the type of node. nodeType is read only.

The most important node types are:

|  |  |
| --- | --- |
| **Element type** | **NodeType** |
| Element | 1 |
| Attribute | 2 |
| Text | 3 |
| Comment | 8 |
| Document | 9 |

**DOM access**

EX1.The following example returns a list of all <p> elements that are descendants (children, grand-children, etc.) of the element with id="main":

*document.getElementsByTagName("p");*

The following example returns a list of all <p> elements that are descendants (children, grand-children, etc.) of the element with id="main":

Ex 2.

*document.getElementById("main").getElementsByTagName("p");*

Ex 3.

*document.getElementsByClassName("intro");*

**Changing HTML Content**

Ex 4.

*<html>  
<body>  
<p id="p1">Hello World!</p>  
<script>  
document.getElementById("p1").innerHTML="New text!";  
</script>  
</body>  
</html>*

**Changing HTML Style**

Ex 5

*<html>  
<body>*

*<p id="p2">Hello world!</p>*

*<script>  
document.getElementById("p2").style.color="blue";  
</script>*

*</body>  
</html>*

**Creating New HTML Elements**

Ex 6.

*<div id="d1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
<script>  
var para=document.createElement("p");  
var node=document.createTextNode("This is new.");  
para.appendChild(node);  
var element=document.getElementById("d1");  
element.appendChild(para);  
</script>*

**Using Events**

Events are generated by the browser when "things happen" to HTML elements:

* An element is clicked on
* When a user clicks the mouse
* When a web page has loaded
* When an image has been loaded
* When the mouse moves over an element
* When an input field is changed
* When an HTML form is submitted
* When a user strokes a key

Ex 7

*<html>  
<body>  
<input type="button" onclick="document.body.style.backgroundColor='lavender';"  
value="Change background color" />  
</body>  
</html>*

Ex 8

*<html>  
<body>  
<script>  
function ChangeBackground() {  
 document.body.style.backgroundColor="lavender";  
}  
</script>  
<input type="button" onclick="ChangeBackground()"  
value="Change background color" />  
</body>  
</html>*

Ex 9

*<html>  
<body>  
<p id="p1">Hello world!</p>  
<script>  
function ChangeText() {  
 document.getElementById("p1").innerHTML="New text!";  
}  
</script>  
  
<input type="button" onclick="ChangeText()" value="Change text">  
</body>  
</html>*

**HTML Event Attributes**

To assign events to HTML elements you can use event attributes.

Ex

*<button onclick="displayDate()">Try it</button>*

**Assign Events Using the HTML DOM**

*<script>  
document.getElementById("myBtn").onclick=function(){displayDate()};  
 </script>*

**The onload and onunload Events**

The onload and onunload events are triggered when the user enters or leaves the page.

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

Ex.

*<body onload="checkCookies()">*

**The onchange Event**

*<input type="text" id="fname" onchange="upperCase()">*

**The onmouseover and onmouseout Events**

Ex

*<!DOCTYPE html>*

*<html>*

*<body>*

*<div onmouseover="mOver(this)" onmouseout="mOut(this)" style="background-color:#D94A38;width:120px;height:20px;padding:40px;">Mouse Over Me</div>*

*<script>*

*function mOver(obj) {*

*obj.innerHTML="Thank You"*

*}*

*function mOut(obj) {*

*obj.innerHTML="Mouse Over Me"*

*}*

*</script>*

*</body>*

*</html>*

**The onmousedown, onmouseup and onclick Events**

*<!DOCTYPE html>*

*<html>*

*<body>*

*<div onmousedown="mDown(this)" onmouseup="mUp(this)" style="background-color:#D94A38;width:90px;height:20px;padding:40px;">Click Me</div>*

*<script>*

*function mDown(obj) {*

*obj.style.backgroundColor="#1ec5e5";*

*obj.innerHTML="Release Me"*

*}*

*function mUp(obj) {*

*obj.style.backgroundColor="#D94A38";*

*obj.innerHTML="Thank You"*

*}*

*</script>*

*</body>*

*</html>*

**HTML DOM Navigation**

The getElementsByTagName() method returns a **node list**. A node list is an array of nodes.

The length property defines the number of nodes in a node-list. You can loop through a node-list by using the length property:

Ex.

*x=document.getElementsByTagName("p");  
  
for (i=0;i<x.length;i++) {  
 document.write(x[i].innerHTML);  
 document.write("<br />");  
}*

**DOM Root Nodes**

There are two special properties that allow access to the full document:

* document.documentElement - The full document
* document.body - The body of the document

*<html>  
<body>  
  
<p>Hello World!</p>  
<div>  
<p>The DOM is very useful!</p>  
<p>This example demonstrates the <b>document.body</b> property.</p>  
</div>  
<script>  
alert(document.body.innerHTML);  
</script>  
  
</body>  
</html>*

**childNodes and nodeValue**

In addition to the innerHTML property, you can also use the childNodes and nodeValue properties to get the content of an element.

The following code gets the value of the <p> element with id="intro":

*<html>  
<body>  
<p id="intro">Hello World!</p>  
<script>  
var txt=document.getElementById("intro").childNodes[0].nodeValue;  
document.write(txt);  
</script>  
</body>  
</html>*

In the example above, getElementById is a method, while childNodes and nodeValue are properties.

In this tutorial we will use the innerHTML property. However, learning the method above is useful for understanding the tree structure and the navigation of the DOM.

**3.2 Revisit document object**

This is probably one of the most important object you need to know. And we will see what we can achieve by using it.

1. display the current document's URL: document.referrer?

2. display the title of a document : document.title

3. display the full URL of a document: document.URL

4. display the domain name of the server that loaded the document: document.domain

5. display all the name/values pairs of cookies in a document: document.cookie;

6. replace the content of a document

eg:

*<script>*

*function myFunction() {*

*document.open("text/html","replace");*

*document.write("<h2>Learning about the HTML DOM is fun!</h2>");*

*document.close();*

*}*

*</script>*

7. open a new window, and add some content

*function myFunction() {*

*var w = window.open();*

*w.document.open();*

*w.document.write("<h1>Hello World!</h1>");*

*w.document.close();*

*}*

8. display the number of elements with a specific name

*<!DOCTYPE html>*

*<html>*

*<head>*

*<script>*

*function getElements() {*

*var x = document.getElementsByName("x");*

*document.getElementById("demo").innerHTML = x.length;*

*}*

*</script>*

*</head>*

*<body>*

*<p>*

*Cats: <input name="x" type="radio" value="Cats">*

*Dogs: <input name="x" type="radio" value="Dogs">*

*</p>*

*<p>*

*<input type="button" onclick="getElements()" value="How many elements named x?">*

*</p>*

*<p id="demo"></p>*

*</body>*

*</html>*

9. display the number of elements with a specific tag name

*<!DOCTYPE html>*

*<html>*

*<head>*

*<script>*

*function getElements() {*

*var x = document.getElementsByTagName("input");*

*document.getElementById("demo").innerHTML = x.length;*

*}*

*</script>*

*</head>*

*<body>*

*<input type="text" size="20"><br>*

*<input type="text" size="20"><br>*

*<input type="text" size="20"><br>*

*<p>*

*<input type="button" onclick="getElements()" value="How many input elements?">*

*</p>*

*<p id="demo"></p>*

*</body>*

*</html>*

3.3 Other DOM collections

anchors collections and links collections are the same stuff.

Eg,

*<!DOCTYPE html>*

*<html>*

*<body>*

*<a name="html">HTML Tutorial</a><br>*

*<a name="css">CSS Tutorial</a><br>*

*<a name="xml">XML Tutorial</a><br>*

*<p id="demo"></p>*

*<script>*

*document.getElementById("demo").innerHTML =*

*"Number of anchors are: " +document.anchors.length + "</br>" + document.anchors[0].innerHTML + “</br>” + document.anchors[0].name;*

*</script>*

*</body>*

*</html>*

eg, likewise, you can get document.anchors[0].name and you can get document.links[0].href.

*<!DOCTYPE html>*

*<html>*

*<body>*

*<p>*

*<a href="/html/default.asp">HTML</a>*

*<br>*

*<a href="/css/default.asp">CSS</a>*

*</p>*

*<p id="demo"></p>*

*<script>*

*document.getElementById("demo").innerHTML =*

*"The href of the first link is " + document.links[0].href;*

*</script>*

*</body>*

*</html>*

Likewise, forms in one page can be multiple,

*<!DOCTYPE html>*

*<html>*

*<body>*

*<form action="">*

*First name: <input type="text" name="fname" value="Donald">*

*Last name: <input type="text" name="lname" value="Donald">*

*<input type="submit" value="Submit">*

*</form>*

*<p id="demo"></p>*

*<script>*

*document.getElementById("demo").innerHTML =*

*"Number of forms: " + document.forms.length;*

*</script>*

*</body>*

*</html>*

another example,

*<!DOCTYPE html>*

*<html>*

*<body>*

*<form name="Form1"></form>*

*<form name="Form2"></form>*

*<form name="Form3"></form>*

*<p id="demo"></p>*

*<script>*

*document.getElementById("demo").innerHTML =*

*"The name of the first for is " + document.forms[0].name;*

*</script>*

*</body>*

*</html>*

Likewise for other tags are the same,

*<!DOCTYPE html>*

*<html>*

*<body>*

*<img id = “img1” src="pic\_htmltree.gif">*

*<img id=”img2” src="pic\_navigate.gif">*

*<p id="demo"></p>*

*<script>*

*document.getElementById("demo").innerHTML =*

*"Number of images: " + document.images.length + “</br>” + document.images[0].id;*

*</script>*

*</body>*

*</html>*

Now how about divs? This is pretty important!

Document.style is also very very important!

*<!DOCTYPE html>*

*<html>*

*<body>*

*<p id="p1">*

*This is a text.*

*This is a text.*

*This is a text.*

*</p>*

*<input type="button" value="Hide text"*

*onclick="document.getElementById('p1').style.visibility='hidden'">*

*<input type="button" value="Show text"*

*onclick="document.getElementById('p1').style.visibility='visible'">*

*</body>*

*</html>*

3.4 Navigator object

appName, appCodeName : will return the browser

javaEnabled(), cookieEnabled(): boolean

platform, OS

product: browser engine

userAgent: A long string of everything

3.5 Cookie object

document.cookie will get and set cookies, cookies will be separated by “key:value; key:value”.

3.6 History object

window.history.back() and window.history.forward()

3.7 Location object

window.location will return the info of the current URL of your page.

Location.href, hostname, protocol, pathname and assign ..etc

3.8 Timing Event

This is one of the most important object.

The following methods belong to the window object.

setTimeout(function, milliseconds)

Executes a function, after waiting a specified number of milliseconds.

setInterval(function, milliseconds)  
Same as setTimeout(), but repeats the execution of the function continuously.

window.clearTimeout(timeoutVariable)

eg:

myVar = setTimeout(*function*, *milliseconds*);  
clearTimeout(myVar);

eg:

*<!DOCTYPE html>*

*<html>*

*<body>*

*<p>A script on this page starts this clock:</p>*

*<p id="demo"></p>*

*<script>*

*var myVar = setInterval(myTimer, 1000);*

*function myTimer() {*

*var d = new Date();*

*document.getElementById("demo").innerHTML = d.toLocaleTimeString();*

*}*

*</script>*

*</body>*

*</html>*

**Chapter 4 JQuery**

**4.1 How to load Jquery**

1. jQuery is a lightweight, "write less, do more", JavaScript library, a single file. It simplifies js programming.

You can download include the file or include cdn js file.

<script src="jquery-1.10.2.min.js"></script>  
or

..  
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">  
</script>  
..

2. Load multiple js files and load your own file

<head>  
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">  
</script>  
<script src="my\_jquery\_functions.js"></script>  
</head>

**4.2 jQuery Syntax ($(*selector*).*action*())**

The jQuery syntax is tailor made for selecting HTML elements and performing some action on the element(s).

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

Selector

It's based on the existing [CSS Selectors](http://www.w3schools.com/cssref/css_selectors.asp), and in addition, it has some own custom selectors.

It could be id, classes, types, attributes, values of attributes and much more. It is used to find HTML element.

**4.3 Jquery Event (The Document Ready Event)**

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

*$(document).ready(function(){  
 // jQuery methods go here...  
});*

Or

*$(function(){  
 // jQuery methods go here...  
});*

This is to prevent any jQuery code from running before the document is finished loading (is ready). This also allows you to have your JavaScript code before the body of your document, in the head section.

Eg: type, class, id all can be used. Pay attention to the self-defined anonymous function() {} inside ().

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="UTF-8">*

*<title>Insert title here</title>*

*<script src="<http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js>"></script>*

*<script>*

*$(document).ready(****function****() {*

*$("button:first").click(****function****() {*

*$("p").hide();*

*$("#test").hide();*

*});*

*$("#myBtn").dblclick(* ***function****() {*

*$(".test").text("changed to this");*

*}*

*);*

*}*

*);*

*</script>*

*</head>*

*<body>*

*<p>test me </p>*

*<div id="test">will disappear </div>*

*<div class="test">content will disappear after double click</div>*

*<button type="button" id="firstBtn" >Click Me!</button>*

*<button type="button" id = "myBtn">Double click testing</button>*

*</body>*

*</html>*

**More Examples of jQuery Selectors**

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Description** | **Example** |
| $("\*") | Selects all elements | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_all2) |
| $(this) | Selects the current HTML element | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_this) |
| $("p.intro") | Selects all <p> elements with class="intro" | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_pclass) |
| $("p:first") | Selects the first <p> element | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_pfirst) |
| $("ul li:first") | Selects the first <li> element of the first <ul> | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_ullifirst) |
| $("ul li:first-child") | Selects the first <li> element of every <ul> | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_ullifirstchild) |
| $("[href]") | Selects all elements with an href attribute | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_hrefattr) |
| $("a[target='\_blank']") | Selects all <a> elements with a target attribute value equal to "\_blank" | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_hrefattrblank) |
| $("a[target!='\_blank']") | Selects all <a> elements with a target attribute value NOT equal to "\_blank" | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_hrefattrnotblank) |
| $(":button") | Selects all <button> elements and <input> elements of type="button" | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_button2) |
| $("tr:even") | Selects all even <tr> elements | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_even) |
| $("tr:odd") | Selects all odd <tr> elements | [Try it](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_sel_odd) |

**Events (much less than c++ and other languages)**

In $(selector).action( ); the followings are the ‘action’, there are many more actions.

Here are some common DOM events:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mouse Events** | **Keyboard Events** | **Form Events** | **Document/Window Events** |
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave |  | blur | unload |

**4.4 Commonly Used jQuery Event Methods**

**$(document).ready()**

The $(document).ready() method allows us to execute a function when the document is fully loaded. This event is already explained in the [jQuery Syntax](http://www.w3schools.com/jquery/jquery_syntax.asp) chapter.

**click()**

$("p").click(function(){  
 $(this).hide();  
});

**dblclick()**

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

$("p").dblclick(function(){  
 $(this).hide();  
});

mouseenter()

$("#p1").mouseenter(function(){  
 alert("You entered p1!");  
});

mouseleave()

$("#p1").mouseleave(function(){  
 alert("Bye! You now leave p1!");  
});

mousedown()

$("#p1").mousedown(function(){  
 alert("Mouse down over p1!");  
});

mouseup()

$("#p1").mouseup(function(){  
 alert("Mouse up over p1!");  
});

hover()

$("#p1").hover(function(){  
 alert("You entered p1!");  
 },  
 function(){  
 alert("Bye! You now leave p1!");  
});

focus()

$("input").focus(function(){  
 $(this).css("background-color","#cccccc");  
});

blur()

$("input").blur(function(){  
 $(this).css("background-color","#ffffff");  
});

**4.5 jQuery Effects Actions**

Note all these effects action can have param for speed control and callback, some are even all optional.

**Show:** [.hide(speed,callback)](http://api.jquery.com/hide/), [.show(speed,callback)](http://api.jquery.com/show/), [.toggle(speed,callback)](http://api.jquery.com/toggle/)

**Slide:** [.slideDown(speed,callback))](http://api.jquery.com/slideDown/) , [.slideUp(speed,callback))](http://api.jquery.com/slideUp/), [.slideToggle(speed,callback))](http://api.jquery.com/slideToggle/)

**Fading:**

$(selector).fadeIn([speed],[callback]);

$(selector).fadeOut([speed],[callback]);

$(selector).fadeToggle([speed],[callback]);

$(selector).fadeTo(speed,opacity,[callback]);

Eg:

*$(document).ready(function(){*

*$("#touchMe").mouseover(function(){*

*$("#div1").fadeIn();*

*$("#div2").fadeIn("slow");*

*$("#div3").fadeIn(3000);*

*});*

*});*

*</script>*

**Callback :** is execuated after the action is done.

*$("#div3").fadeTo("slow", 0.7,function() {*

*alert("hello");*

*});*

It is different from the following, as the following alert will show before the animation is finished.

*$("#div3").fadeTo("slow", 0.7,function() {*

*});*

*alert("hello");*

**Animate:**

$(selector).animate({params},speed,callback);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

Eg: css property and values

*$("button").click(function(){  
    $("div").animate({  
        left: '250px',  
        height: '+=150px',  
        width: '+=150px'  
    });  
});*

Eg: use jquery effect action

$("button").click(function(){  
    $("div").animate({  
        height: 'toggle'  
    });  
});

**.Stop()** , Syntax:$(selector).stop(stopAll,goToEnd);

stops all the effect actions before they finished.

Eg:

*<head>*

*$(document).ready(function(){*

*$("#flip").click(function(){*

*$("#panel").slideDown(5000);*

*});*

*$("#stop").click(function(){*

*$("#panel").stop();*

*});*

*});*

*</head>*

*<body>*

*<button id="stop">Stop sliding</button>*

*<div id="flip">Click to slide down panel</div>*

*<div id="panel">Hello world!</div>*

*</body>*

**Method chaining**

$("#p1").css("color", "red").slideUp(2000).slideDown(2000);

**4.6 jQuery HTML**

**4.6.1 Get Content - text(), html(), and val()**  
**html()**

Obtains the HTML content of the first element in the matched set. This will get the physical html content that is inside of the “#Container”.

$('#Container').html()

# **text()**

Concatenates all text content of the wrapped elements and returns it as the result of the method. That means this method return all the string which display on our browser.

$('#Container').text()

# **val()**

Returns the value attribute of the first element in the matched set. When the element is a multiselect element, the returned value is an array of all selections. This method only work with control like input, select, button etc. It not work with div, span, p etc.

$('#txtCountry').val()

Don't use $(“selector”).innerHTML= “xxx”;, it will not work!

**4.6.2 Set Content - text(), html(), and val()**

$("#test1").text("Hello world!");  
$("#test2").html("<b>Hello world!</b>");  
$("#test3").val("Dolly Duck");

Example:

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>jQuery With Example</title>*

*<script src="http://code.jquery.com/jquery-1.9.1.js" type="text/javascript"></script>*

*<script type="text/javascript">*

*$(****function*** *() {*

*$("#btnClick").click(****function*** *() {*

*alert("HTML: " + $("#Container").html());*

*alert("Text: " + $("#Container").text());*

*alert("Value: " + $("#txtCountry").val());*

*alert("Value: " + $("#Container").text());*

*});*

*});*

*</script>*

*</head>*

*<body>*

*<div id="Container">*

*<b>Country</b>*

*<input type="text" id="txtCountry" value="India" />*

*</div>*

*<button id="btnClick">Click</button>*

*</body>*

*</html>*

Question: what can you do with Document.getElementById(“#test3”).innerHTML?

It works more like html() function in jquery.

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>jQuery With Example</title>*

*<script src="http://code.jquery.com/jquery-1.9.1.js" type="text/javascript"></script>*

*<script type="text/javascript">*

*$(****function*** *() {*

*$("#btnClick").click(****function*** *() {*

***var*** *content = document.getElementById("Container");*

*alert("HTML: " + content.innerHTML);*

*content.innerHTML = "hello world";*

*});*

*});*

*</script>*

*</head>*

*<body>*

*<div id="Container">*

*<b>Country</b>*

*<input type="text" id="txtCountry" value="India" />*

*</div>*

*<button id="btnClick">Click</button>*

*</body>*

*</html>*

**4.6.3 Add New HTML Content**

* append() - Inserts content at the end of the selected elements
* prepend() - Inserts content at the beginning of the selected elements
* after() - Inserts content after the selected elements
* before() - Inserts content before the selected elements

$("p").append("Some appended text.");

Eg:

*function appendText() {  
var txt1="<p>Text.</p>"; // Create element with HTML   
var txt2=$("<p></p>").text("Text."); // Create with jQuery  
var txt3=document.createElement("p"); // Create with DOM  
txt3.innerHTML="Text.";  
$("p").append(txt1,txt2,txt3); // Append the new elements   
}*

*$("img").after("Some text after");  
$("img").before("Some text before");*

**4.6.4 Remove Elements/Content**

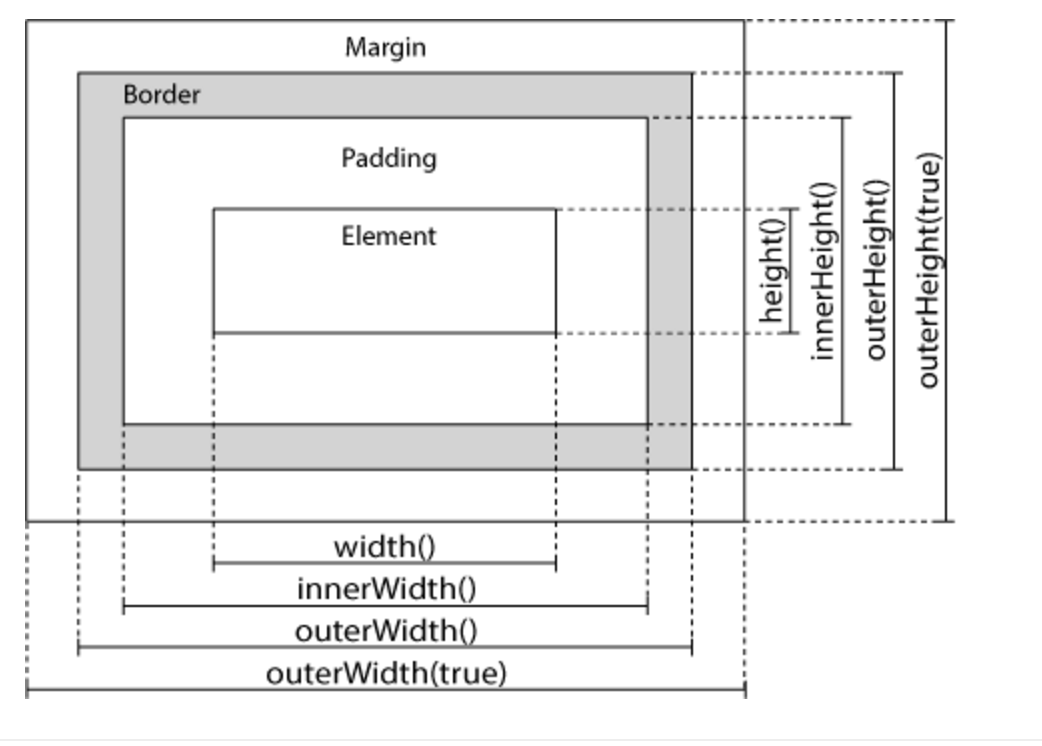
To remove elements and content, there are mainly two jQuery methods:

* remove() - Removes the selected element (and its child elements)
* empty() - Removes the child elements from the selected element

$("#div1").remove();

$("#div1").empty();

**4.6.5 Dimensions and css**

****

**eg: (Replaced this example with the ones I had in my directory)**

*<!DOCTYPE html>*

*<html>*

*<head>*

*<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>*

*<script>*

*$(document).ready(function(){*

*$("button").click(function(){*

*var txt = "";*

*txt += "Width of div: " + $("#div1").width() + "</br>";*

*txt += "Height of div: " + $("#div1").height() + "</br>";*

*txt += "Outer width of div (margin included): " + $("#div1").outerWidth(true) + "</br>";*

*txt += "Outer height of div (margin included): " + $("#div1").outerHeight(true);*

*$("#div1").html(txt);*

*});*

*});*

*</script>*

*<style>*

*#div1 {*

*height: 100px;*

*width: 300px;*

*padding: 10px;*

*margin: 3px;*

*border: 1px solid blue;*

*background-color: lightblue;*

*}*

*</style>*

*</head>*

*<body>*

*<div id="div1"></div>*

*<br>*

*<button>Display dimensions of div</button>*

*<p>outerWidth(true) - returns the width of an element (includes padding, border, and margin).</p>*

*<p>outerHeight(true) - returns the height of an element (includes padding, border, and margin).</p>*

*</body>*

*</html>*

**4.7Jquery with Ajax**

AJAX = Asynchronous JavaScript and XML, it is also js.

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. Jquery is good about this.

The jQuery load() method is a simple, but powerful AJAX method.

The load() method loads data from a server and puts the returned data into the selected element.

Syntax:

$(selector).load(URL,data,callback);

* The required URL parameter specifies the URL you wish to load.
* The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
* The optional callback parameter is the name of a function to be executed after the load() method is completed.

The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters:

* responseTxt - contains the resulting content if the call succeeds
* statusTxt - contains the status of the call
* xhr - contains the XMLHttpRequest object

Eg:

*$("button").click(function(){  
    $("#div1").load("demo\_test.txt", function(responseTxt, statusTxt, xhr){  
        if(statusTxt == "success")  
            alert("External content loaded successfully!");  
        if(statusTxt == "error")  
            alert("Error: " + xhr.status + ": " + xhr.statusText);  
    });*  
*});*

## $.get()

The $.get() method requests data from the server with an HTTP GET request.

$.get(URL,callback);

The required URL parameter specifies the URL you wish to request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.get() method to retrieve data from a file on the server:

Eg:

*$("button").click(function(){  
    $.get("demo\_test.asp", function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});*

The first parameter of $.get() is the URL we wish to request ("demo\_test.asp").

The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

## $.post()

The $.post() method requests data from the server using an HTTP POST request.

Syntax:

$.post(URL,data,callback);

The required URL parameter specifies the URL you wish to request.

The optional data parameter specifies some data to send along with the request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.post() method to send some data along with the request:

Eg:

*$("button").click(function(){  
    $.post("demo\_test\_post.asp",  
    {  
        name: "Donald Duck",  
        city: "Duckburg"  
    },  
    function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});*

**$.ajax() (get this part later)**

jQuery - Get and Set CSS Classes

jQuery - css() Method

jQuery - Dimensions

Ref:

More needed to be learned from:

<http://www.w3schools.com/jquery/jquery_ref_ajax.asp>

<http://www.w3schools.com/jquery/ajax_ajax.asp>

**4. 8 jQuery Traversing - Ancestors**

**Traversing Up the DOM Tree**

Three useful jQuery methods for traversing up the DOM tree are:

* parent(), direct parents
* parents(), all ancestors
* parentsUntil(“component”), returns all ancestor elements between a <span> and a <div> element:

$(document).ready(function(){  
 $("span").parent();  
});

$(document).ready(function(){  
 $("span").parents();  
});

$(document).ready(function(){  
 $("span").parents("ul");  
});

$(document).ready(function(){  
 $("span").parents("ul");  
});

$(document).ready(function(){  
 $("span").parentsUntil("div");  
});

**4.9 jQuery Traversing - Descendants**

ref:

<http://w3schools.com/jquery/jquery_traversing_descendants.asp>

**4.10 [jQuery Traversing - Siblings](http://w3schools.com/jquery/jquery_traversing_descendants.asp)**

http://w3schools.com/jquery/jquery\_traversing\_siblings.asp

**4.11 jQuery Callback Functions**

**Example**

*<!DOCTYPE html>*

*<html>*

*<head>*

*<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">*

*</script>*

*<script>*

*$(document).ready(function(){*

*$("#flip").click(function(){*

*$("#panel").slideUp("slow");*

*});*

*});*

*</script>*

*<style type="text/css">*

*#panel,#flip {*

*padding:5px;*

*text-align:center;*

*background-color:#e5eecc;*

*border:solid 1px #c3c3c3;*

*}*

*#panel {*

*padding:50px;*

*}*

*</style>*

*</head>*

*<body>*

*<div id="flip">Click to slide up panel</div>*

*<div id="panel">Hello world!</div>*

*</body>*

*</html>*

ref:

http://api.jquery.com/

**Chapter 5 RequireJS**

**It is an asynchronous script loader, but it is very useful in modular design.**

**Example 1: RequireJS with underscore**

RequireJS implements the AMD (Asynchronous Module Definition) spec, which means we can write our own modules and load them with RequireJS, allowing it to manage dependencies for us. Have you ever had multiple script tags and had to load them in a particular order as one relied on the other? I have, and it’s a nightmare. Here is the complete scenario.

1/ you need to load a lot of .js in <scripts>, and each has dependency

2/ If loading too many large files , it will block the rest of the page, but we can minify it, but then we have to maintain the order.

├── app.js

├── index.html

├── lib

│   ├── template

│   │   └── template.js

│   ├── require.js

│   └── underscore.js

a/ app.js is my main file, we will look into this shortly.

b/ lib/template is where all my self-written modules will go. With RequireJS all our code gets split into modules. I’ll explain further in a moment.

c/ Files immediately within lib are external libraries, in this case the RequireJS source and also under.

The following shows how we load in dependencies. This is done through the require function. To load in some code to run after a script, you use it like so:

*require(['myfile'], function(myFile) {*

*myFile.init();*

*});*

['a','b','c'] is an array of global modules/libraries you registered in config js file. The names in the conf's file's paths section on the lhs. Whatever you lists to import in this array, you must pass them as params in your callback function.

That would look for myfile.js within the same directory as your main JS file, and whatever myfile returns will be referenced within the callback as myFile, as that’s the variable name I passed into the callback. With libraries like jQuery and Underscore that register global objects, you don’t need to do this.

For example, requirejs will configure what libs you will be using app.js like the following,

*require.config({*

*paths: {*

*"https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min",*

*"underscore": "lib/underscore",*

*}*

*});*

*index.html*

*<html>*

*<head>*

*<script data-main="app" src="lib/require.js"></script>*

*</head>*

*<body>*

*<script>*

*。。。。*

*</script>*

*</body>*

*<html>*

now requirejs knows where to imports these libs, in our own template.js where we will implement our logic,

*define(['underscore', 'jquery'], function() {*

*var showName = function(n) {*

*var temp = \_.template("Hello <%= name %>");*

*$("body").html(temp({name: n}));*

*};*

*return {*

*showName: showName*

*};*

*});*

In the above code, for underscore and jquery because it is global library, so you don't not need to pass them in the callback function and use it directly. For other defined methods, you can use ［］to include them, and for importing your own lib, you can specify relative directory. In require(['mydir/a']) method, you can also specify full path.

Index.html has the following in it,

*<script>*

*require(['lib/modules/template'], function(template) {*

*template.showName("Jack");*

*});*

*</script>*

*In nodejs, you can use require() and exports.symbol= func(some function you defined earlier).*

*Note that these require are supported in ECMA 2016.*

Example2: RequireJS integration with Jquery

Ref:

<http://javascriptplayground.com/blog/2012/07/requirejs-amd-tutorial-introduction/> (done)

<https://www.sitepoint.com/understanding-requirejs-for-effective-javascript-module-loading/>

<http://www.ringabell.org/en/un-simple-guide-pour-debuter-avec-requirejs/>

<https://coderwall.com/p/u8xgvq/requirejs-basic-introduction>

https://cdnjs.com/libraries/backbone.js/tutorials/organizing-backbone-using-modules

**Chapter 6 Node.js and NPM**

NodeJS uses chrome v8 engine/vm to parse javascript. It is a vm and it is lib. For this chapter, please install NodeJS, NPM and Atom code editor.

6.1 npm install xxx

xxx is something called module and you can write it in js or c++.

eg: npm install -g npm@

6.2 Load module

*var MyModule = require('myModule');*

NodeJS will search in the following location.

./node\_modules

../node\_modules

$HOME/.node\_modules

$HOME/.node\_libraries

$PREFIX/lib/node

the following load the module in the same directory,

*var MyModule = require('./mymodule');*

Or

*var MyModule = require('./mymodule.js');*

6.3 Write your first module

eg,

*module.exports = function() {*

*console.log("hello world");*

*}*

eg,

*module.exports = function() {*

*var self = this;*

*this.counter = 0;*

*this.pump = function() {*

*self.counter++;*

*};*

*};*

6.4 Publish your module

6.4.1 To publish your module, you have to

1/ have npm account: npm adduser,

2/ generate your package like the following: npm init

3/ upload package: 'npm publish .'

note that the core concept of the package management is the meta info in package.json.

eg: Creating NPM account

xiaofengmaclap:myApp xiaofeng$ npm adduser

Username: xlics05

Password:

Email: (this IS public) xlics05@gmail.com

Logged in as xlics05 on https://registry.npmjs.org/.

Eg: create a npm package (package.json)

xiaofengmaclap:NodeJS xiaofeng$ cd myApp/

xiaofengmaclap:myApp xiaofeng$ npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields

and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and

save it as a dependency in the package.json file.

Press ^C at any time to quit.

name: (myApp)

Sorry, name can no longer contain capital letters.

name: (myApp) myapp

version: (1.0.0)

description: just some testing

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to /Users/xiaofeng/cloud/mockingtest/JavascriptExamples/WebContent/NodeJS/myApp/package.json:

{

"name": "myapp",

"version": "1.0.0",

"description": "just some testing",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

Is this ok? (yes) yes

eg: publish your project,

xiaofengmaclap:myApp xiaofeng$ npm publish .

npm ERR! publish Failed PUT 403

npm ERR! Darwin 15.6.0

npm ERR! argv "/usr/local/Cellar/node5/5.12.0/bin/node" "/usr/local/bin/npm" "publish" "."

npm ERR! node v5.12.0

npm ERR! npm v3.8.6

npm ERR! code E403

npm ERR! you do not have permission to publish "myapp". Are you logged in as the correct user? : myapp

npm ERR!

npm ERR! If you need help, you may report this error at:

npm ERR! <https://github.com/npm/npm/issues>

npm ERR! Please include the following file with any support request:

npm ERR! /Users/xiaofeng/cloud/mockingtest/JavascriptExamples/WebContent/NodeJS/myApp/npm-debug.log

In package.json, there is a "main": "index.js", if you want to change the entry point, you can change index.js to demo.js, then before you change, your project normally looks like this,

● index.js

● package.json

● README (README.md)

● LICENSE

● lib/hello1.js

● lib/hello2.js

● tests/test1.js

● tests/test2.js

after the change, it looks like this,

● demo.js

● package.json

● README (README.md)

● LICENSE

● lib/hello1.js

● lib/hello2.js

● tests/test1.js

● tests/test2.js

6.4.2 index.js and index.node

./example/index.js

var ex = require('./example')

If read directory, preset index.js or index.node. (??? check the reference again regarding this)

Ref:

<https://docs.npmjs.com/>

<https://www.linkedin.com/jobs2/view/96524616?refId=&trk=jobs_home_click_jymbii>

How to write nodejs module.pdf

6.5 Use NodeJS to test and debug JS

NodeJS by default will support RequireJS framework internally.

Nodejs will provide the ability to help us build a server as well. The following example will show us how to build a http server running on a tcp port.

1/ build the following server.js,

var http = require("http");

http.createServer(

function(request, response) {

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello World");

response.end();

}).listen(8888);

2/ Run it in NodeJS

Node server.js

3/ Test the server by doing,

open browser, and see <http://localhost:8888/> .

4/ Rewrite the above JS code,

we could also write the above code like this,

|  |
| --- |
| var http = require("http");  function onRequest(request, response) { |
| response.writeHead(200, {"Content-Type": "text/plain"});  response.write("Hello World");  response.end();  }  http.createServer(onRequest).listen(8888); |

but createServer take a function() {}, and actually let us take a look at this javascript code,

function say(word) {

console.log(word);

}

function execute(someFunction, value) {

someFunction(value);

}

execute(say, "Hello");

Output:

*xiaofengmaclap:tmp xiaofeng$ node test.js*

*hello*

and we can change the following,

function execute(someFunction, value) {

someFunction(value);

}

to just,

execute(function(word) {

console.log(word)

}, "Hello"

);

the someFunction is like function pointer in C. Here it is an anonymous function.

5/ Rewrite the above code again and use it in a packaged way,

Let us rewrite the server.js like the following,

|  |
| --- |
| var http = require("http");  function start() { function onRequest(request, response) {  console.log("Request received.");  response.writeHead(200, {"Content-Type": "text/plain"});  response.write("Hello World"); response.end();  } |
| http.createServer(onRequest).listen(8888);  console.log("Server has started."); }  exports.start = start; |

and index.js like the following,

var server = require("./server");

server.start();

require and exports looks like requireJS syntax. What it does is that it defines a start symbol in exports context, and that exports.start has the definition of function start. When you

var server = require (x.js), then you can call that server.start symbol from its exported context. And the start function of course wraps up everything. Think about exports context as a symbol in Wolfram language.

5/ querystring module in NodeJS

If we would like to parse the following URL,

http://localhost:8888/start?foo=bar&hello=world

Server.js is rewritten like the following,

var http = require("http");

var url = require("url");

function start() { function onRequest(request, response) {

console.log("Request received.");

var pathname = url.parse(request.url).pathname;

console.log("Request for " + pathname + " received.");

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello World"); response.end();

}

http.createServer(onRequest).listen(8888);

console.log("Server has started."); }

exports.start = start;

When we enter this in the browse,

http://localhost:8888/start?foo=value

Output is,

C:\Users\linfeng\myApp>node index

Server has started.

Request received.

Request for / received.

Request received.

Request for / received.

Request received.

Request for /start received.

Request received.

Request for /start received.

6/ An even more complicated example,

Here assuming we have a few js files as lib files, one lib function is calling the other lib function in the entry js file.

Now the following are the files,

Router.js

function route(handle, pathname) {

console.log("About to route a request for " + pathname);

if (typeof handle[pathname] === 'function') {

handle[pathname]();

} else {

console.log("No request handler found for " + pathname);

}

}

exports.route = route;

**requestHandlers.js**

function start() {

console.log("Request handler 'start' was called.");

}

function upload() {

console.log("Request handler 'upload' was called.");

}

exports.start = start;

exports.upload = upload;

**Server.js**

var http = require("http");

var url = require("url");

function start(route,handler) { function onRequest(request, response) {

console.log("Request received.");

var pathname = url.parse(request.url).pathname;

console.log("Request for " + pathname + " received.");

route(handler,pathname);

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello World"); response.end();

}

http.createServer(onRequest).listen(8888);

console.log("Server has started."); }

exports.start = start;

**Index.js,**

var server = require("./server");

var router = require("./router");

var requestHandlers = require("./requestHandlers");

var handle = {};

handle["/"] = requestHandlers.start;

handle["/start"] = requestHandlers.start;

handle["/upload"] = requestHandlers.upload;

server.start(router.route, handle);

To run it, the output is,

C:\Users\linfeng\myApp>node index

Server has started.

Request received.

Request for /start received.

About to route a request for /start

Request handler 'start' was called.

Request received.

Request for /upload received.

About to route a request for /upload

Request handler 'upload' was called.

TODO: need to read more, Page 41 of Node introduction.

ref:

<http://debuggable.com/posts/understanding-node-js:4bd98440-45e4-4a9a-8ef7-0f7ecbdd56cb>

<http://martinfowler.com/articles/injection.html>

<http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html>

<http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/>

<http://nodebeginner.org/index-zh-cn.html#a-word-of-warning> (done)

**Chapter 7 underscoreJS**

**Chapter 8 BackBoneJS**

**Chapter 9 React.js**

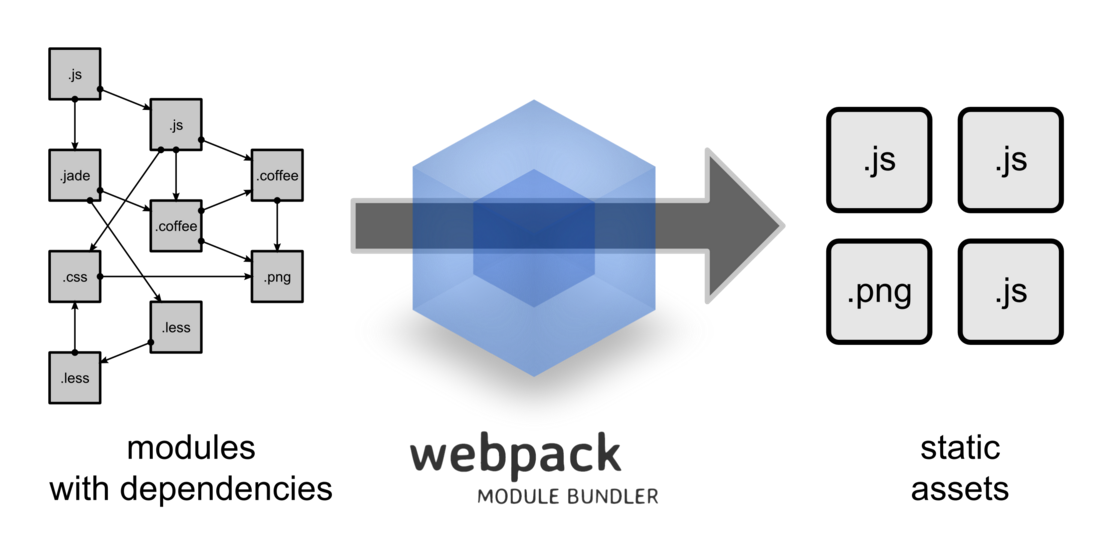
Please see sperate react.js document.

**Chapter 11 WebPack**

Webpack has two key features:

1. It analyzes dependencies and builds static assets for use in the browser, without much other tooling (no need to explicitly concatenate CSS files anymore).
2. Every source file is run through a loader that can do some preprocessing on it. The [Babel](http://babeljs.io/) [loader](https://github.com/babel/babel-loader) allows us to write modern JavaScript ([ES6](https://github.com/lukehoban/es6features) and beyond) which is "transpiled" down to ES5 that today's browsers understand.

Together with its various plugins, it's like a more modern [RequireJS](http://requirejs.org/) with a JS transpiler, a SASS compiler, and all sorts of other build tools built-in.



**Chapter 12 Authoring SASS (jruby-complete-1.7.19.jar)**

SASS for authoring CSS, CSS files are generated during runtime from the respective \*.scss files.