**Java Script**

Uses of Java Script:

1. Client side form validation
2. Provides the facility for handling events
3. Makes your web page/html dynamic
4. Provides the kind of programming style (object based programming language).
5. Allows you to read/update html & xml DOM tree.

Client Side Validation:

* This decrease the round trip and traffic for server.
* But not good practice to always depend upon client side validation because client side validation only depends upon the browser. Sometimes it may be possible that browser stops the Java Script.
* So, best practice is that all should depend on server side and client side both kind of validation.

Features of Java Script:

* Java Script is scripting language created by “Netscape”.

1. Variables
2. Literals
3. Operators
4. Control Statement
5. Array
6. Functions
7. JS Events
8. JS Object
9. Variables:

var variable\_name;

eg:

var x;

var str;

1. Literals:

* Literal is a value which you can assign to Java Script variables.
* Following are the Java Script literal:

1. Integer literal
2. Floating point literal
3. String Literal
4. Boolean Literal

x = 10;

x = 10.5;

x = “abc”;

* Variable declared with “var” type can contain any type of literal and any type of Java Script object.

1. Operators
2. Control Statement
3. Array
   * + JS Array automatically extends the length of any array when new array elements are initialized.

For eg:

cust\_orders = new Array();

cust\_orders[50] = “suraj”;

cust\_orders[100] = “surajkumar”;

* + - Dense arrays can be declared and initialized at the same time.

arrayName = new Array(value0, value1, value2,……., valuen);

1. Functions

function function\_name(){

}

For eg:

Definition:

function show(){

……….

}

Call

show();

1. Events:

* Java Script supports following events:

1. onClick
2. onBlure
3. onFocus
4. onSelect
5. onChange
6. onDoubleClick
7. onMouseMove
8. onMouseOver
9. onMouseOut
10. onKeyPress
11. onKeyDown
12. onKeyUp
13. onload
14. Java Script Object
15. String
16. Date
17. Math
18. Array
19. Boolean

Note:

1. There are no compilers for checking the grammatical mistake in Java Script Code.
2. JS uses interpreter for checking the errors.
3. Every browser contains Java Script interpreter to check errors and Java Script Runtime environment to execute the Java Script code.
4. Some browser allows you to enable and disable Java Script environment.

**Two ways of writing Java Script:**

<script type = “text/javascript”>

//JS Code

</script>

1. abc.js

// JS code

<script type = “text/javascript” src = “abc.js” />

JavaScript inserted into HTML pages, can be executed by all modern web browsers.

To insert a JavaScript into an HTML page, use the <script></script> tag.

Scripts can be in the <body> or in the <head> section of HTML, and/or in both, it is preferred way to keep it within the <head> tags or at the bottom of the page.

The script tag takes two important attributes:

* **language:** This attribute specifies what scripting language you are using. Typically, its value will be *javascript*. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
* **type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to *"text/javascript"*.

External JavaScripts

Scripts can also be placed in external files. External files often contain code to be used by several different web pages.

External JavaScript files have the file extension .js.

To use an external script, point to the .js file in the "src" attribute of the <script> tag:

<script src="myScript.js"></script>x

Example of a java script

<!DOCTYPE HTML>

<html>

<head>

<meta charset="UTF-8">

<title>JavaScript Page</title>

<link href="style.css"rel="stylesheet"type="text/css">

<script type="text/javascript">

window.alert("The page is loading");

</script>

</head>

<body>

<p>

<span onclick="document.bgColor='red';">Red</span> |

<span onclick="document.bgColor='white';">White</span>

</p>

<script type="text/javascript"src="script-1.js"></script>

</body>

</html>

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: **Alert box, Confirm box** and **Prompt box.**

## Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax:

window.alert("*sometext*");

e.g:

alert("I am an alert box!");

## Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

### Syntax

window.confirm("*sometext*");

The **window.confirm()** method can be written without the window prefix.

var r=confirm("Press a button");  
if (r==true)  
{  
x="You pressed OK!";  
}  
else  
{  
x="You pressed Cancel!";  
}

## Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

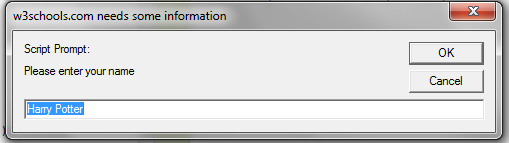
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

### Syntax

window.prompt("*sometext*","*defaultText*");

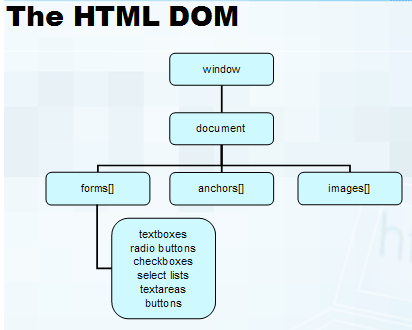
The **window.prompt()** method can be written without the window prefix.

var person=prompt("Please enter your name","Harry Potter");  
  
if (person!=null)  
{  
x="Hello " + person + "! How are you today?";  
document.getElementById("demo").innerHTML=x;  
}



# The HTML DOM

The HTML Document Object Model (DOM) is the browser's view of an HTML page as an object hierarchy, starting with the browser window itself and moving deeper into the page, including all of the elements on the page and their attributes. Below is a simplified version of the HTML DOM.



**Dom Level and DOM architecture**

DOM Levels are essentially versions or specifications which define of DOM works.

[DOM Level 1](http://www.w3.org/TR/DOM-Level-1/) defines the core elements of the Document Object Model. [DOM Level 2](http://www.w3.org/TR/DOM-Level-2-Core/) extends those elements and adds events. [DOM Level 3](http://www.w3.org/TR/DOM-Level-3-Core/) extends DOM lvl 2 and adds more elements and events.

Each new level of the DOM adds or changes specific sets of features. When browsers are said to be DOM Level X compliant developers can (hopefully) assume that the browser correctly handles the specified DOM api calls

[**https://dom.spec.whatwg.org/**](https://dom.spec.whatwg.org/)

[**https://developer.mozilla.org/en-US/docs/Web/API/Document\_Object\_Model**](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)

## The HTML DOM Document

In the HTML DOM object model, the document object represents your web page.

The document object is the owner of all other objects in your web page.

If you want to access objects in an HTML page, you always start with accessing the document object.

## Finding HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById() | Find an element by element id |
| document.getElementsByTagName() | Find elements by tag name |
| document.getElementsByClassName() | Find elements by class name |

## Changing HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| element.innerHTML= | Change the inner HTML of an element |
| element.attribute= | Change the attribute of an HTML element |
| element.setAttribute(attribute,value) | Change the attribute of an HTML element |
| element.style.property= | Change the style of an HTML element |

## Adding and Deleting Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.createElement() | Create an HTML element |
| document.removeChild() | Remove an HTML element |
| document.appendChild() | Add an HTML element |
| document.replaceChild() | Replace an HTML element |
| document.write(text) | Write into the HTML output stream |

## Adding Events Handlers

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(id).onclick=function(){code} | Adding event handler code to an onclick event |

**Questions:**

1. **How to use java script in the html page?**

**Ans:**

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

Developers can use these events to execute JavaScript coded responses

**About Event:**

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

**HTML 5 Standard Events**

The standard HTML 5 events are listed here for your reference. Here script indicates a Javascript function to be executed against that event.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Window Event Attributes Events triggered for the window object (applies to the <body> tag):   |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [onafterprint](http://www.w3schools.com/tags/ev_onafterprint.asp) | *script* | Script to be run after the document is printed | | [onbeforeprint](http://www.w3schools.com/tags/ev_onbeforeprint.asp) | *script* | Script to be run before the document is printed | | [onbeforeunload](http://www.w3schools.com/tags/ev_onbeforeunload.asp) | *script* | Script to be run when the document is about to be unloaded | | Onerror | *script* | Script to be run when an error occur | | [onhashchange](http://www.w3schools.com/tags/ev_onhashchange.asp) | *script* | Script to be run when there has been changes to the anchor part of the a URL | | [onload](http://www.w3schools.com/tags/ev_onload.asp) | *script* | Fires after the page is finished loading | | onmessage | *script* | Script to be run when the message is triggered | | [onoffline](http://www.w3schools.com/tags/ev_onoffline.asp) | *script* | Script to be run when the browser starts to work offline | | [ononline](http://www.w3schools.com/tags/ev_ononline.asp) | *script* | Script to be run when the browser starts to work online | | onpagehide | *script* | Script to be run when a user navigates away from a page | | [onpageshow](http://www.w3schools.com/tags/ev_onpageshow.asp) | *script* | Script to be run when a user navigates to a page | | onpopstate | *script* | Script to be run when the window's history changes | | [onresize](http://www.w3schools.com/tags/ev_onresize.asp) | *script* | Fires when the browser window is resized | | onstorage | *script* | Script to be run when a Web Storage area is updated | | [onunload](http://www.w3schools.com/tags/ev_onunload.asp) | *script* | Fires once a page has unloaded (or the browser window has been closed) |  Form Events Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):   |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [onblur](http://www.w3schools.com/tags/ev_onblur.asp) | *script* | Fires the moment that the element loses focus | | [onchange](http://www.w3schools.com/tags/ev_onchange.asp) | *script* | Fires the moment when the value of the element is changed | | [oncontextmenu](http://www.w3schools.com/tags/ev_oncontextmenu.asp) | *script* | Script to be run when a context menu is triggered | | [onfocus](http://www.w3schools.com/tags/ev_onfocus.asp) | *script* | Fires the moment when the element gets focus | | [oninput](http://www.w3schools.com/tags/ev_oninput.asp) | *script* | Script to be run when an element gets user input | | [oninvalid](http://www.w3schools.com/tags/ev_oninvalid.asp) | *script* | Script to be run when an element is invalid | | [onreset](http://www.w3schools.com/tags/ev_onreset.asp) | *script* | Fires when the Reset button in a form is clicked | | [onsearch](http://www.w3schools.com/tags/ev_onsearch.asp) | *script* | Fires when the user writes something in a search field (for < input="search">) | | [onselect](http://www.w3schools.com/tags/ev_onselect.asp) | *script* | Fires after some text has been selected in an element | | [onsubmit](http://www.w3schools.com/tags/ev_onsubmit.asp) | *script* | Fires when a form is submitted |  Keyboard Events  |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [onkeydown](http://www.w3schools.com/tags/ev_onkeydown.asp) | *script* | Fires when a user is pressing a key | | [onkeypress](http://www.w3schools.com/tags/ev_onkeypress.asp) | *script* | Fires when a user presses a key | | [onkeyup](http://www.w3schools.com/tags/ev_onkeyup.asp) | *script* | Fires when a user releases a key |  Mouse Events Events triggered by a mouse, or similar user actions:   |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [onclick](http://www.w3schools.com/tags/ev_onclick.asp) | *script* | Fires on a mouse click on the element | | [ondblclick](http://www.w3schools.com/tags/ev_ondblclick.asp) | *script* | Fires on a mouse double-click on the element | | [ondrag](http://www.w3schools.com/tags/ev_ondrag.asp) | *script* | Script to be run when an element is dragged | | [ondragend](http://www.w3schools.com/tags/ev_ondragend.asp) | *script* | Script to be run at the end of a drag operation | | [ondragenter](http://www.w3schools.com/tags/ev_ondragenter.asp) | *script* | Script to be run when an element has been dragged to a valid drop target | | [ondragleave](http://www.w3schools.com/tags/ev_ondragleave.asp) | *script* | Script to be run when an element leaves a valid drop target | | [ondragover](http://www.w3schools.com/tags/ev_ondragover.asp) | *script* | Script to be run when an element is being dragged over a valid drop target | | [ondragstart](http://www.w3schools.com/tags/ev_ondragstart.asp) | *script* | Script to be run at the start of a drag operation | | [ondrop](http://www.w3schools.com/tags/ev_ondrop.asp) | *script* | Script to be run when dragged element is being dropped | | [onmousedown](http://www.w3schools.com/tags/ev_onmousedown.asp) | *script* | Fires when a mouse button is pressed down on an element | | [onmousemove](http://www.w3schools.com/tags/ev_onmousemove.asp) | *script* | Fires when the mouse pointer is moving while it is over an element | | [onmouseout](http://www.w3schools.com/tags/ev_onmouseout.asp) | *script* | Fires when the mouse pointer moves out of an element | | [onmouseover](http://www.w3schools.com/tags/ev_onmouseover.asp) | *script* | Fires when the mouse pointer moves over an element | | [onmouseup](http://www.w3schools.com/tags/ev_onmouseup.asp) | *script* | Fires when a mouse button is released over an element | | onmousewheel | *script* | Deprecated. Use the [onwheel](http://www.w3schools.com/tags/ev_onwheel.asp) attribute instead | | [onscroll](http://www.w3schools.com/tags/ev_onscroll.asp) | *script* | Script to be run when an element's scrollbar is being scrolled | | [onwheel](http://www.w3schools.com/tags/ev_onwheel.asp) | *script* | Fires when the mouse wheel rolls up or down over an element |  Clipboard Events  |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [oncopy](http://www.w3schools.com/tags/ev_oncopy.asp) | *script* | Fires when the user copies the content of an element | | [oncut](http://www.w3schools.com/tags/ev_oncut.asp) | *script* | Fires when the user cuts the content of an element | | [onpaste](http://www.w3schools.com/tags/ev_onpaste.asp) | *script* | Fires when the user pastes some content in an element |  Media Events Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>):   |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | Onabort | *script* | Script to be run on abort | | oncanplay | *script* | Script to be run when a file is ready to start playing (when it has buffered enough to begin) | | oncanplaythrough | *script* | Script to be run when a file can be played all the way to the end without pausing for buffering | | oncuechange | *script* | Script to be run when the cue changes in a <track> element | | ondurationchange | *script* | Script to be run when the length of the media changes | | onemptied | *script* | Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects) | | Onended | *script* | Script to be run when the media has reach the end (a useful event for messages like "thanks for listening") | | Onerror | *script* | Script to be run when an error occurs when the file is being loaded | | onloadeddata | *script* | Script to be run when media data is loaded | | onloadedmetadata | *script* | Script to be run when meta data (like dimensions and duration) are loaded | | onloadstart | *script* | Script to be run just as the file begins to load before anything is actually loaded | | Onpause | *script* | Script to be run when the media is paused either by the user or programmatically | | Onplay | *script* | Script to be run when the media is ready to start playing | | onplaying | *script* | Script to be run when the media actually has started playing | | onprogress | *script* | Script to be run when the browser is in the process of getting the media data | | onratechange | *script* | Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode) | | onseeked | *script* | Script to be run when the seeking attribute is set to false indicating that seeking has ended | | onseeking | *script* | Script to be run when the seeking attribute is set to true indicating that seeking is active | | Onstalled | *script* | Script to be run when the browser is unable to fetch the media data for whatever reason | | onsuspend | *script* | Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason | | ontimeupdate | *script* | Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media) | | onvolumechange | *script* | Script to be run each time the volume is changed which (includes setting the volume to "mute") | | onwaiting | *script* | Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data) |  Misc Events  |  |  |  | | --- | --- | --- | | **Attribute** | **Value** | **Description** | | [onerror](http://www.w3schools.com/tags/ev_onerror.asp) | *script* | Fires when an error occurs while loading an external file | | [onshow](http://www.w3schools.com/tags/ev_onshow.asp) | *script* | Fires when a <menu> element is shown as a context menu | | [ontoggle](http://www.w3schools.com/tags/ev_ontoggle.asp) | *script* | Fires when the user opens or closes the <details> element | |  |  |

**Finding HTML Objects**

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **DOM** |
| document.anchors | Returns all <a> elements that have a name attribute | 1 |
| document.applets | Returns all <applet> elements (Deprecated in HTML5) | 1 |
| document.baseURI | Returns the absolute base URI of the document | 3 |
| document.body | Returns the <body> element | 1 |
| document.cookie | Returns the document's cookie | 1 |
| document.doctype | Returns the document's doctype | 3 |
| document.documentElement | Returns the <html> element | 3 |
| document.documentMode | Returns the mode used by the browser | 3 |
| document.documentURI | Returns the URI of the document | 3 |
| document.domain | Returns the domain name of the document server | 1 |
| document.domConfig | Obsolete. Returns the DOM configuration | 3 |
| document.embeds | Returns all <embed> elements | 3 |
| document.forms | Returns all <form> elements | 1 |
| document.head | Returns the <head> element | 3 |
| document.images | Returns all <img> elements | 1 |
| document.implementation | Returns the DOM implementation | 3 |
| document.inputEncoding | Returns the document's encoding (character set) | 3 |
| document.lastModified | Returns the date and time the document was updated | 3 |
| document.links | Returns all <area> and <a> elements that have a href attribute | 1 |
| document.readyState | Returns the (loading) status of the document | 3 |
| document.referrer | Returns the URI of the referrer (the linking document) | 1 |
| document.scripts | Returns all <script> elements | 3 |
| document.strictErrorChecking | Returns if error checking is enforced | 3 |
| document.title | Returns the <title> element | 1 |
| document.URL | Returns the complete URL of the document | 1 |

**Finding HTML Element by Id**

Example:

var x = document.getElementById("intro");

If the element is found, the method will return the element as an object (in x).

If the element is not found, x will contain null.

**Finding HTML Elements by Tag Name**

finds all <p> elements

var x = document.getElementsByTagName("p");

finds the element with id="main", and then finds all <p> elements inside "main":

var x = document.getElementById("main");  
var y = x.getElementsByTagName("p");

**Finding HTML Elements by Class Name**

var x = document.getElementsByClassName("intro");

returns a list of all elements with class="intro".

**Finding HTML Elements by CSS Selectors**

If you want to find all HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

var x = document.querySelectorAll("p.intro");

This example returns a list of all <p> elements with class="intro".

**Changing the HTML Output Stream**

In JavaScript, document.write() can be used to write directly to the HTML output stream:

<!DOCTYPEhtml>  
<html>  
<body>  
<script>  
 document.write(Date());  
</script>  
</body>  
</html>

Never use document.write() after the document is loaded. It will overwrite the document.

**Changing HTML Content**

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use this syntax:

document.getElementById(*id*).innerHTML = *new HTML*

**Changing the Value of an Attribute**

To change the value of an HTML attribute, use this syntax:

document.getElementById(*id*).*attribute=new value*

Example:

<!DOCTYPEhtml>

<html>

<head>

<script>

**var** f = **true**;

**function** change() {

**if** (f) {

document.getElementById("image").src = "landscape.jpg";

document.getElementById("na").innerHTML = "YOU";

f = **false**;

} **else** {

document.getElementById("image").src = "smiley.gif";

document.getElementById("na").innerHTML = "ME";

f = **true**;

}

}

</script>

</head>

<body>

<imgid=*"image"*src=*"smiley.gif"*width=*"160"*height=*"120"*

onclick="change()">

<pid=*"na"*>ME</p>

</body>

</html>

**Changing HTML Style**

To change the style of an HTML element, use this syntax:

document.getElementById(*id*).style.*property*=*new style*

<html>  
<body>  
  
<pid="p2">Hello World!</p>  
  
<script>  
document.getElementById("p2").style.color = "blue";  
</script>  
  
<p>The paragraph above was changed by a script.</p>  
  
</body>  
</html>

**Using Events**

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

<!DOCTYPEhtml>

<html>

<body>

<h1id=*"id1"*>My Heading 1</h1>

<buttontype=*"button"*

onclick="document.getElementById('id1').style.color = 'red'"

value=*"yahoo"*>Click Me!</button>

<br/>

<br/>

<inputtype=*"button"*

onclick="document.getElementById('id1').style.color = 'blue'"

value=*"yahoo"*> Click Me!

</input><!-- End tag (</input>) not needed -->

</body>

</html>

Output:



http://www.w3schools.com/js/js\_htmldom\_eventlistener.asp

**JavaScript HTML DOM EventListener**

Add an event listener that fires when a user clicks a button:

* The addEventListener() method attaches an event handler to the specified element.
* The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
* You can add many event handlers to one element.
* You can add many event handlers of the same type to one element, i.e two "click" events.
* You can add event listeners to any DOM object not only HTML elements. i.e the window object.
* The addEventListener() method makes it easier to control how the event reacts to bubbling.
* When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.
* You can easily remove an event listener by using the removeEventListener() method.

Syntax:

element.addEventListener(event, function, useCapture);

* The first parameter is the type of the event (like "click" or "mousedown").
* The second parameter is the function we want to call when the event occurs.
* The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

Note that you don't use the "on" prefix for the event; use "click" instead of "onclick".

Example:

document.getElementById("myBtn").addEventListener("click", displayDate);

*element*.addEventListener("click", function(){ alert("Hello World!"); });

*element*.addEventListener("click", myFunction);  
function myFunction() {  
    alert ("Hello World!");  
}

**Add Many Event Handlers to the Same Element**

*element*.addEventListener("click", myFunction);  
*element*.addEventListener("click", mySecondFunction);

*element*.addEventListener("mouseover", myFunction);  
*element*.addEventListener("click", mySecondFunction);  
*element*.addEventListener("mouseout", myThirdFunction);

**Add an Event Handler to the Window Object**

The addEventListener() method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that supports events, like the xmlHttpRequest object.

Example:

Add an event listener that fires when a user resizes the window:

window.addEventListener("resize", function(){  
    document.getElementById("demo").innerHTML = *sometext*;  
});

**Event Bubbling or Event Capturing**

* There are two ways of event propagation in the HTML DOM, bubbling and capturing.
* Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?
* In *bubbling* the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.
* In *capturing* the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.
* With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:

addEventListener(event, function, useCapture);

The default value is false, which will use the bubbling propagation, when the value is set to true, the event uses the capturing propagation.

document.getElementById("myP").addEventListener("click", myFunction, true);  
document.getElementById("myDiv").addEventListener("click", myFunction, true);

**The removeEventListener() method**

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

*element*.removeEventListener("mousemove", myFunction);

**http://www.w3schools.com/js/js\_htmldom\_navigation.asp**

**Question:** How to write or invoke events in the html code?

Every html element contains by default events.

If any event attribute is defined with the html element then when that event occurs on the page the value of the attribute which is either a java script code or function, will be executed.

The following are the most important events recognized by javascript:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Event | Detected when | HTML tags | | onfocus="" | **Form field gets focus** | select, text, textarea | | onblur="" | **Form field loses focus** | select, text, textarea | | onchange="" | **Content of a field changes** | select, text, textarea | | onselect="" | **Text is selected** | text, textarea | | onmouseover="" | **Mouse moves over a link** | A | | onmouseout="" | **Mouse moves out of a link** | A | | onclick="" | **Mouse clicks an object** | A, button, checkbox,  radio, reset, submit | | onload="" | **Page is finished loading** | body, frameset | | onunload="" | **Browser opens new document** | body, frameset | | onSubmit="" | **Submit button is clicked** | form | |

**Example for sending email after submitting the form:**

<form name="myWebForm" action="mailto:suraj06k@email.com" method="post">

<input type="checkbox" /> Checkbox 1<br />

<input type="text" /> Text Field 1<br />

<input type="submit" value="SUBMIT" />

</form>

HTML form elements rely on *action* and *method* attributes to identify where to send the form data for processing (action) and how to process the data (method).

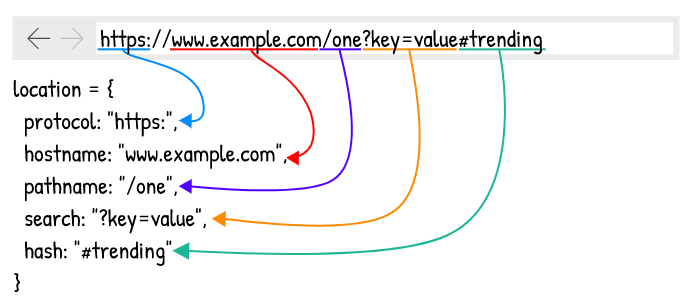
Question: What is the use of javascript?

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

There are many uses (and abuses!) for the powerful JavaScript language. Here are a few things that you may or may not have seen in your web surfing days:

* Clocks
* Mouse Trailers (an animation that follows your mouse when you surf a site)
* Drop Down Menus
* Alert Messages
* Popup Windows
* [HTML Form](http://www.tizag.com/htmlT/forms.php) Data Validation



**Java Script as the programming Language:**