2013

Java Script

satyajohnny

Green buds software Solutions

1/1/2013

JavaScript Tutorial

The <script> Tag

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

The <script> and </script> tells where the JavaScript starts and ends.

The lines between the <script> and </script> contain the JavaScript:

```
<script>
alert("My First JavaScript");
</script>
```

JavaScript in <head> or <body>

A JavaScript Function in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is called when a button is clicked:

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First JavaScript Function";
}
</script>
</head>

<body>

<h1>My Web Page</h1>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

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:

Example

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Writing to The Document Output

```
document.write("<p>My First JavaScript</p>");
```

JavaScript Variables

JavaScript variables are "containers" for storing information:

Example

```
var x=5;
var y=6;
var z=x+y;
```

When you assign a numeric value to a variable, do not put quotes around the value. If you put quotes around a numeric value, it will be treated as text.

Example

```
var pi=3.14;
var person="John Doe";
var answer='Yes I am!';
```

JavaScript Data Types

String, Number, Boolean, Array, Object, Null, Undefined.

JavaScript Has Dynamic Types

JavaScript has dynamic types. This means that the same variable can be used as different types:

Example

```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

JavaScript Arrays

The following code creates an Array called cars:

```
var cars=new Array();
cars[0]="Saab";
cars[1]="Volvo";
cars[2]="BMW";
```

or (condensed array):

```
var cars=new Array("Saab","Volvo","BMW");
```

or (literal array):

Example

```
var cars=["Saab","Volvo","BMW"];
```

JavaScript Objects

An object is delimited by curly braces. Inside the braces the object's properties are defined as name and value pairs (name : value). The properties are separated by commas:

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

The object (person) in the example above has 3 properties: firstname, lastname, and id.

Spaces and line breaks are not important. Your declaration can span multiple lines:

```
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};
```

You can address the object properties in two ways:

Example

```
name=person.lastname;
name=person["lastname"];
```

Declaring Variables as Objects

When a variable is declared with the keyword "new", the variable is declared as an object:

```
var name = new String;
var x =    new Number;
var y =    new Boolean;
```

JavaScript Objects

Almost everything in JavaScript can be an Object: Strings, Functions, Arrays, Dates....Objects are just data, with properties and methods.

Creating JavaScript Objects

Almost "everything" in JavaScript can be objects. Strings, Dates, Arrays, Functions....

You can also create your own objects.

This example creates an object called "person", and adds four properties to it:

Example

```
person=new Object();
person.firstname="John";
person.lastname="Doe";
person.age=50;
person.eyecolor="blue";
```

Accessing Object Properties

The syntax for accessing the property of an object is:

```
objectName.propertyName
```

This example uses the length property of the String object to find the length of a string:

```
var message="Hello World!";
var x=message.length;
```

The value of x, after execution of the code above will be:

```
12
```

JavaScript Functions

A function is a block of code that will be executed when "someone" calls it:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
```

```
{
alert("Hello World!");
}
</script>
</head>

<body>
<button onclick="myFunction()">Try it</button>
</body>
</html>
```

Calling a Function with Arguments

```
<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>

<script>
function myFunction(name,job)
{
alert("Welcome " + name + ", the " + job);
}
</script>
```

Functions With a Return Value

Syntax

```
function myFunction()
{
var x=5;
return x;
}
```

The function above will return the value 5.

```
var myVar=myFunction();
```

The variable myVar holds the value 5, which is what the function "myFunction()" returns.

You can also use the return value without storing it as a variable:

```
document.getElementById("demo").innerHTML=myFunction();
```

JavaScript Comparison and Logical Operators

If...else if...else Statement

JavaScript Errors - Throw and Try to Catch

JavaScript Objects"

Everything" in JavaScript is an Object.

In addition, JavaScript allows you to define your own objects.

JavaScript String Object

« Previous

Next Reference »

String Object

The String object is used to manipulate a stored piece of text.

String objects are created with new String().

Syntax

```
var txt = new String("string");

or more simply:

var txt = "string";
```

For a tutorial about the String object, read our JavaScript String Object tutorial.

String Object Properties

Property	Description
constructor	Returns the function that created the String object's prototype
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

String Object Methods

Method	Description

charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLocaleLowerCase()	Converts a string to lowercase letters, according to the host's locale
toLocaleUpperCase()	Converts a string to uppercase letters, according to the host's locale
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string

valueOf()	Returns the primitive value of a String object

String HTML Wrapper Methods

The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

These are not standard methods, and may not work as expected in all browsers.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color
fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text

The Window Object

Window Object

The window object represents an open window in a browser.

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.

Window Object Properties

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not
defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frames	Returns an array of all the frames (including iframes) in the current window
history	Returns the History object for the window (See History object)
innerHeight	Sets or returns the inner height of a window's content area
innerWidth	Sets or returns the inner width of a window's content area
length	Returns the number of frames (including iframes) in a window
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window
outerHeight	Sets or returns the outer height of a window, including toolbars/scrollbars
outerWidth	Sets or returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the x coordinate of the window relative to the screen

screenTop	Returns the y coordinate of the window relative to the screen
screenX	Returns the x coordinate of the window relative to the screen
screenY	Returns the y coordinate of the window relative to the screen
self	Returns the current window
status	Sets or returns the text in the statusbar of a window
top	Returns the topmost browser window

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
createPopup()	Creates a pop-up window
focus()	Sets focus to the current window
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window

prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the content by the specified number of pixels
scrollTo()	Scrolls the content to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

The Screen Object

« Previous

Next Reference »

Screen Object

The screen object contains information about the visitor's screen.

Note: There is no public standard that applies to the screen object, but all major browsers support it.

Screen Object Properties

Property	Description
availHeight	Returns the height of the screen (excluding the Windows Taskbar)
availWidth	Returns the width of the screen (excluding the Windows Taskbar)

colorDepth	Returns the bit depth of the color palette for displaying images
height	Returns the total height of the screen
pixelDepth	Returns the color resolution (in bits per pixel) of the screen
width	Returns the total width of the screen

JavaScript HTML DOM

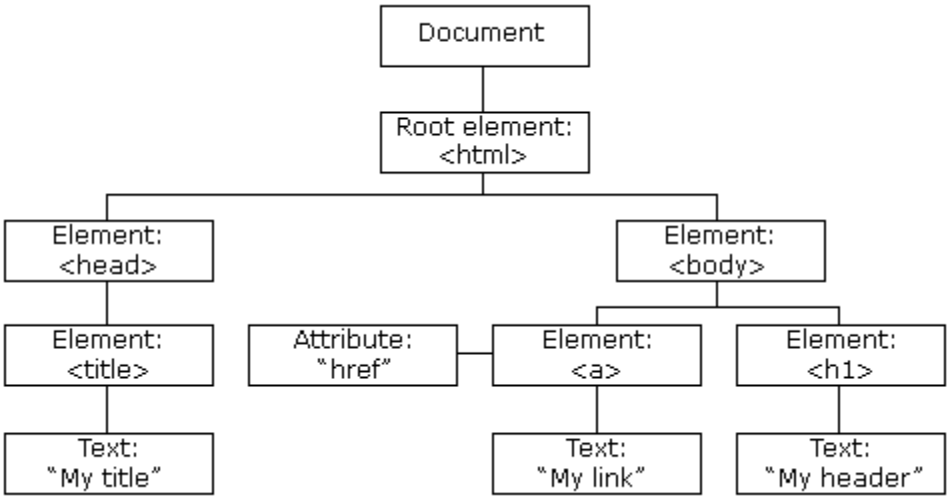
With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



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

What You Will Learn

In the next chapters of this tutorial you will learn:

- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add and delete HTML elements

What is the DOM?

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

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

« Previous

Next Chapter »

HTML DOM methods are **actions** you can perform (on HTML Elements)

HTML DOM properties are **values** (of HTML Elements) that you can set or change

The DOM Programming Interface

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

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A **property** is a value that you can get or set (like changing the content of an HTML element).

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

Example

The following code gets the content (the innerHTML) of the <p> element with id="intro":

Example

```
<html>
<body>

<p id="intro">Hello World!</p>

<script>
var txt=document.getElementById("intro").innerHTML;
document.write(txt);
</script>

</body>
</html>
```

The getElementById Method

The most common way to access an HTML element is to use the id of the element.

In the example above the getElementById method used id="intro" to find the element.

The innerHTML Property

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

The innerHTML property is useful for getting or replacing the content of HTML elements.

The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

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.

Below are some examples of how you can use the document object to access and manipulate HTML.

The next chapters demonstrate all the methods.

Finding HTML Elements

Method	Description
document.getElementById()	Finding an element by element id
document.getElementsByTagName()	Finding elements by tag name
document.getElementsByClassName()	Finding elements by class name
document.forms[]	Finding elements by HTML element objects

Changing HTML Elements

Method	Description
document.write(*text*)	Writing into the HTML output stream
document.getElementById(*id*).innerHTML=	Changing the inner HTML of an element
document.getElementById(*id*).*attribute*=	Changing the attribute of an element
document.getElementById(*id*).style.*attribute*=	Changing 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

Adding Events Handlers

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

JavaScript HTML DOM Elements

« Previous

Next Chapter »

This page teaches you how to find and access HTML elements in an HTML page.

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by HTML object collections

Finding HTML Elements by Id

The easiest way to find HTML elements in the DOM, is by using the element id.

This example finds the element with id="intro":

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

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

Example

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

Finding HTML Elements by Class Name

If you want to find all HTML elements with the same class name. Use this method:

```
document.getElementsByClassName("intro");
```

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

Finding HTML Elements by HTML Object Collections

This example finds the form element with id="frm1", in the forms collection, and displays all element values:

Example

```
<!DOCTYPE html>
<html>
<body>
```

```
<form id="frm1" action="form_action.asp">
  First name: <input type="text" name="fname" value="Donald"><br>
  Last name: <input type="text" name="lname" value="Duck"><br>
  <input type="submit" value="Submit">
</form> <p>Return the value of each element in the form:</p>
<script>
var x=document.forms["frm1"];
for (var i=0;i<x.length;i++)
  {
  document.write(x.elements[i].value);
  document.write("<br>");
  }
</script>

</body>
</html>
```

JavaScript HTML DOM - Changing HTML

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
```

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML="New text!";
</script>

</body>
</html>
```

Changing the Value of an Attribute

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

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

This example changes the value of the src attribute of an <img> element:

Example

```
<!DOCTYPE html>
<html>
<body>
```

```
<img id="image" src="smiley.gif">

<script>
document.getElementById("image").src="landscape.jpg";
</script>

</body>
</html>
```

JavaScript HTML DOM - Changing CSS

« Previous

Next Chapter »

The HTML DOM allows JavaScript to change the style of HTML elements.

Changing HTML Style

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

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

The following example changes the style of a <p> element:

Example

```
<html>
<body>

<p id="p2">Hello World!</p>

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

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

HTML DOM Style Object

« Previous

Next Reference »

Style object

The Style object represents an individual style statement.

The Style object can be accessed from the document or from the elements to which that style is applied.

Syntax for using the Style object properties:

```
document.getElementById("id").style.property="value"
```

The Style object property categories:

- Background
- Border/Outline
- Generated Content
- List
- Misc
- Margin/Padding
- Positioning/Layout
- Printing
- Table
- Text

Background properties

Property	Description
background	Sets or returns all the background properties in one declaration
backgroundAttachment	Sets or returns whether a background-image is fixed or scrolls with the page
backgroundColor	Sets or returns the background-color of an element
backgroundImage	Sets or returns the background-image for an element
backgroundPosition	Sets or returns the starting position of a background-image
backgroundRepeat	Sets or returns how to repeat (tile) a background-image

Border/Outline properties

Property	Description
border	Sets or returns border-width, border-style, and border-color in one declaration

borderBottom	Sets or returns all the borderBottom* properties in one declaration
borderBottomColor	Sets or returns the color of the bottom border
borderBottomStyle	Sets or returns the style of the bottom border
borderBottomWidth	Sets or returns the width of the bottom border
borderColor	Sets or returns the color of an element's border (can have up to four values)
borderLeft	Sets or returns all the borderLeft* properties in one declaration
borderLeftColor	Sets or returns the color of the left border
borderLeftStyle	Sets or returns the style of the left border
borderLeftWidth	Sets or returns the width of the left border
borderRight	Sets or returns all the borderRight* properties in one declaration
borderRightColor	Sets or returns the color of the right border
borderRightStyle	Sets or returns the style of the right border
borderRightWidth	Sets or returns the width of the right border
borderStyle	Sets or returns the style of an element's border (can have up to four values)
borderTop	Sets or returns all the borderTop* properties in one declaration
borderTopColor	Sets or returns the color of the top border
borderTopStyle	Sets or returns the style of the top border
borderTopWidth	Sets or returns the width of the top border
borderWidth	Sets or returns the width of an element's border (can have up to four values)
outline	Sets or returns all the outline properties in one declaration
outlineColor	Sets or returns the color of the outline around a element
outlineStyle	Sets or returns the style of the outline around an element

outlineWidth	Sets or returns the width of the outline around an element

Generated Content Properties

Property	Description
content	Sets or returns the generated content before or after the element
counterIncrement	Sets or returns the list of counters and increment values
counterReset	Sets or returns the list of counters and their initial values

List properties

Property	Description
listStyle	Sets or returns list-style-image, list-style-position, and list-style-type in one declaration
listStyleImage	Sets or returns an image as the list-item marker
listStylePosition	Sets or returns the position of the list-item marker
listStyleType	Sets or returns the list-item marker type

Margin/Padding properties

Property	Description
margin	Sets or returns the margins of an element (can have up to four values)
marginBottom	Sets or returns the bottom margin of an element
marginLeft	Sets or returns the left margin of an element
marginRight	Sets or returns the right margin of an element
marginTop	Sets or returns the top margin of an element
padding	Sets or returns the padding of an element (can have up to four values)
paddingBottom	Sets or returns the bottom padding of an element
paddingLeft	Sets or returns the left padding of an element

paddingRight	Sets or returns the right padding of an element
paddingTop	Sets or returns the top padding of an element

Misc properties

Property	Description
cssText	Sets or returns the contents of a style declaration as a string

Positioning/Layout properties

Property	Description
bottom	Sets or returns the bottom position of a positioned element
clear	Sets or returns the position of the element relative to floating objects
clip	Sets or returns which part of a positioned element is visible
cssFloat	Sets or returns the horizontal alignment of an object
cursor	Sets or returns the type of cursor to display for the mouse pointer
display	Sets or returns an element's display type
height	Sets or returns the height of an element
left	Sets or returns the left position of a positioned element
maxHeight	Sets or returns the maximum height of an element
maxWidth	Sets or returns the maximum width of an element
minHeight	Sets or returns the minimum height of an element
minWidth	Sets or returns the minimum width of an element
overflow	Sets or returns what to do with content that renders outside the element box
position	Sets or returns the type of positioning method used for an element (static, relative, absolute or fixed)

right	Sets or returns the right position of a positioned element
top	Sets or returns the top position of a positioned element
verticalAlign	Sets or returns the vertical alignment of the content in an element
visibility	Sets or returns whether an element should be visible
width	Sets or returns the width of an element
zIndex	Sets or returns the stack order of a positioned element

Printing properties

Property	Description
orphans	Sets or returns the minimum number of lines for an element that must be visible at the bottom of a page
pageBreakAfter	Sets or returns the page-break behavior after an element
pageBreakBefore	Sets or returns the page-break behavior before an element
pageBreakInside	Sets or returns the page-break behavior inside an element
widows	Sets or returns the minimum number of lines for an element that must be visible at the top of a page

Table properties

Property	Description
borderCollapse	Sets or returns whether the table border should be collapsed into a single border, or not
borderSpacing	Sets or returns the space between cells in a table
captionSide	Sets or returns the position of the table caption
emptyCells	Sets or returns whether to show the border and background of empty cells, or not
tableLayout	Sets or returns the way to lay out table cells, rows, and columns

Text properties

Property	Description
color	Sets or returns the color of the text
direction	Sets or returns the text direction
font	Sets or returns font-style, font-variant, font-weight, font-size, line-height, and font-family in one declaration
fontFamily	Sets or returns the font face for text
fontSize	Sets or returns the font size of the text
fontSizeAdjust	Sets or returns the font aspect value
fontStyle	Sets or returns whether the style of the font is normal, italic or oblique
fontVariant	Sets or returns whether the font should be displayed in small capital letters
fontWeight	Sets or returns the boldness of the font
letterSpacing	Sets or returns the space between characters in a text
lineHeight	Sets or returns the distance between lines in a text
quotes	Sets or returns the type of quotation marks for embedded quotations
textAlign	Sets or returns the horizontal alignment of text
textDecoration	Sets or returns the decoration of a text
textIndent	Sets or returns the indentation of the first line of text
textShadow	Sets or returns the shadow effect of a text
textTransform	Sets or returns the case of a text
unicodeBidi	Sets or returns whether the text should be overridden to support multiple languages in the same document
whiteSpace	Sets or returns how to handle tabs, line breaks and whitespace in a text

wordSpacing	Sets or returns the spacing between words in a text

JavaScript HTML DOM Events

Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
onclick=JavaScript
```

Examples of HTML events:

- 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

```
<!DOCTYPE html>
<html>
<head>
<script>
function changetext(id)
{
id.innerHTML="Ooops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

Assign Events Using the HTML DOM

The HTML DOM allows you to assign events to HTML elements using JavaScript:

Example

Assign an onclick event to a button element:

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

HTML DOM Events

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

Tip: The event model was standardized by the W3C in DOM Level 2.

HTML DOM Events

DOM: Indicates in which DOM Level the property was introduced.

Mouse Events

Property	Description	DOM
onclick	The event occurs when the user clicks on an element	2
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when a user presses a mouse button over an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element	2
onmouseup	The event occurs when a user releases a mouse button over an element	2

Keyboard Events

Attribute	Description	DOM
onkeydown	The event occurs when the user is pressing a key	2
onkeypress	The event occurs when the user presses a key	2
onkeyup	The event occurs when the user releases a key	2

Frame/Object Events

Attribute	Description	DOM
onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)	2
onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)	
onload	The event occurs when a document, frameset, or <object> has been loaded	2
onresize	The event occurs when a document view is resized	2
onscroll	The event occurs when a document view is scrolled	2
onunload	The event occurs once a page has unloaded (for <body> and <frameset>)	2

Form Events

Attribute	Description	DOM
onblur	The event occurs when a form element loses focus	2
onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	2
onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)	2
onreset	The event occurs when a form is reset	2
onselect	The event occurs when a user selects some text (for <input> and <textarea>)	2
onsubmit	The event occurs when a form is submitted	2

Event Object

Constants

Constant	Description	DOM
CAPTURING_PHASE	The current event phase is the capture phase (3)	1
AT_TARGET	The current event is in the target phase, i.e. it is being evaluated at the event target (1)	2

BUBBLING_PHASE	The current event phase is the bubbling phase (2)	3

Properties

Property	Description	DOM
bubbles	Returns whether or not an event is a bubbling event	2
cancelable	Returns whether or not an event can have its default action prevented	2
currentTarget	Returns the element whose event listeners triggered the event	2
eventPhase	Returns which phase of the event flow is currently being evaluated	2
target	Returns the element that triggered the event	2
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created	2
type	Returns the name of the event	2

Methods

Method	Description	DOM
initEvent()	Specifies the event type, whether or not the event can bubble, whether or not the event's default action can be prevented	2
preventDefault()	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur	2
stopPropagation()	To prevent further propagation of an event during event flow	2

EventTarget Object

Methods

Method	Description	DOM
addEventListener()	Allows the registration of event listeners on the event target (IE8 = attachEvent())	2
dispatchEvent()	Allows to send the event to the subscribed event listeners (IE8 = fireEvent())	2
removeEventListener()	Allows the removal of event listeners on the event target (IE8 = detachEvent())	2

EventListener Object

Methods

Method	Description	DOM
handleEvent()	Called whenever an event occurs of the event type for which the EventListener interface was registered	2

DocumentEvent Object

Methods

Method	Description	DOM
createEvent()		2

MouseEvent/KeyboardEvent Object

Properties

Property	Description	DOM
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	2
button	Returns which mouse button was clicked when an event was triggered	2
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when an event was triggered	2
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when an event was triggered	2
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	2
keyIdentifier	Returns the identifier of a key	3
keyLocation	Returns the location of the key on the device	3
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	2
relatedTarget	Returns the element related to the element that triggered the event	2
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered	2
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event	2

	was triggered	
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	2

Methods

Method	Description	W3C
initMouseEvent()	Initializes the value of a MouseEvent object	2
initKeyboardEvent()	Initializes the value of a KeyboardEvent object	

JavaScript HTML DOM Elements (Nodes)

« Previous

Next Chapter »

Adding and Removing Nodes (HTML Elements)

Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

Example

```
<div id="div1">
<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("div1");
element.appendChild(para);
</script>
```

Try it yourself »

Example Explained

This code creates a new <p> element:

```
var para=document.createElement("p");
```

To add text to the <p> element, you must create a text node first. This code creates a text node:

```
var node=document.createTextNode("This is a new paragraph.");
```

Then you must append the text node to the <p> element:

```
para.appendChild(node);
```

Finally you must append the new element to an existing element.

This code finds an existing element:

```
var element=document.getElementById("div1");
```

This code appends the new element to the existing element:

```
element.appendChild(para);
```

Creating new HTML Elements - insertBefore()

The appendChild() method in the previous example, appended the new element as the last child of the parent.

If you don't want that you can use the insertBefore() method:

Example

```
<div id="div1">
<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("div1");
var child=document.getElementById("p1");
element.insertBefore(para,child);
</script>
```

Try it yourself »

Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element:

Example

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.removeChild(child);
</script>
```

Try it yourself »

Example Explained

This HTML document contains a element with two child nodes (two <p> elements):

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
```

Find the element with id="div1":

```
var parent=document.getElementById("div1");
```

Find the <p> element with id="p1":

```
var child=document.getElementById("p1");
```

Remove the child from the parent:

```
parent.removeChild(child);
```

It would be nice to be able to remove an element without referring to the parent.
But sorry. The DOM needs to know both the element you want to remove, and its parent.

Here is a common workaround: Find the child you want to remove, and use its parentNode property to find the parent:

```
var child=document.getElementById("p1");
child.parentNode.removeChild(child);
```

Replacing HTML Elements

To replace an element to the HTML DOM, use the replaceChild() method:

Example

```
<div id="div1">
<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 parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.replaceChild(para,child);
</script>
```

JavaScript HTML DOM Nodelist

« Previous

Next Chapter »

A nodelist is an array of nodes (like an array of all HTML elements)

HTML DOM Node List

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

The following code selects all <p> nodes in a document:

Example

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

The nodes can be accessed by index number. To access the second <p> you can write:

```
y=x[1];
```

Try it yourself »

Note: The index starts at 0.

HTML DOM Node List Length

The length property defines the number of nodes in a node-list.

You can loop through a node-list by using the length property:

Example

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

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

Try it yourself »

Example explained:

1. Get all <p> element nodes
2. For each <p> element, output the value of its text node

JS Libraries

JS LibrariesJS jQueryJS Prototype

JS Examples

JS Basic ExamplesJS Objects ExamplesJS DOM ExamplesJS HTML ExamplesJS Events ExamplesJS Browser ExamplesJS QuizJS CertificateJS Summary