JAVASCRIPT

Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the Same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

The Real Name is ECMAScript

JavaScript's official name is "ECMAScript". The standard is developed and maintained by the ECMA organisation.

ECMA-262 is the official JavaScript standard. The standard is based on JavaScript (Netscape) and JScript (Microsoft).

The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The development of ECMA-262 started in 1996, and the first edition of was adopted by the ECMA General Assembly in June 1997.

The standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development of the standard is still in progress.

JavaScript How To

The HTML <script> tag is used to insert a JavaScript into an HTML page.

Examples

Write text with Javascript
The example demonstrates how to use JavaSript to write text on a web page.

```
<html>
	<body>
		<script type="text/javascript">
			document.write("Hello World!");
		</script>
	</body>
</html>
```

Write HTML with Javascript
The example demonstrates how to use JavaScript to write HTML tags on a web page.

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<h1>This is a header</h1>");
		</script>
	</body>
</html>
```

How to Put a JavaScript Into an HTML Page

```
<html>
   <body>
      <script type="text/javascript">
         document.write("Hello World!");
      </script>
   </body>
</html>
```

The code above will produce this output on an HTML page:

```
Hello World!
```

```
```

Example Explained

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:

```
<html>
   <body>
      <script type="text/javascript">
         ...
      </script>
   </body>
</html>
```

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

```
<html>
   <body>
      <script type="text/javascript">
         document.write("Hello World!");
      </script>
   </body>
</html>
```

Try it yourself.

Note: If we had not entered the <script> tag, the browser would have treated the document.write("Hello World!") command as pure text, and just write the entire line on the page.

Try it yourself.

HTML Comments to Handle Simple Browsers

Browsers that do not support JavaScript will display JavaScript as page content.

To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag can be used to "hide" the JavaScript. Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement.

```
<html>
   <body>
      <script type="text/javascript">
         <!--
         document.write("Hello World!");
         //-->
      </script>
   </body>
</html>
```

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

JavaScript Where To

JavaScripts in the body section will be executed WHILE the page loads.

JavaScripts in the head section will be executed when CALLED.

Examples

Head section
Scripts that contain functions go in the head section of the document. Then we can be sure that the script is loaded before the function is called.

```
<html>
	<head>
		<script type="text/javascript">
			function message() {
				alert("This alert box was called with the onload event");
			}
		</script>
	</head>
	<body onload="message()">
	</body>
</html>
```

Body section
Execute a script that is placed in the body section.

```
<html>
	<head>
	</head>
	<body>
		<script type="text/javascript">
			document.write("This message is written by JavaScript");
		</script>
	</body>
</html>
```

External script
How to access an external script.

```
<html>
	<head>
	</head>
	<body>
		<script src="xxx.js">
		</script>
		<p>The actual script is in an external script file called "xxx.js".</p>
	</body>
</html>
```

Where to Put the JavaScript

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Scripts in the head section: Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
   <head>
      <script type="text/javascript">
         ....
      </script>
   </head>
</html>
```

Scripts in the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

```
<html>
   <head>
   </head>
   <body>
      <script type="text/javascript">
         ....
      </script>
   </body>
</html>
```

Scripts in both the body and the head section: You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
   <head>
      <script type="text/javascript">
         ....
      </script>
   </head>
   <body>
      <script type="text/javascript">
         ....
      </script>
   </body>
</html>
```

Using an External JavaScript

Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script> tag!

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

```
<html>
   <head>
      <script src="xxx.js">
      </script>
   </head>
   <body>
   </body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```
document.write("Hello Dolly");
```

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
   document.write("<h1>This is a header</h1>");
   document.write("<p>This is a paragraph</p>");
   document.write("<p>This is another paragraph</p>");
</script>
```

Try it yourself.

JavaScript Blocks

JavaScript statements can be grouped together in blocks.

Blocks start with a left curly bracket {, and ends with a right curly bracket }.

The purpose of a block is to make the sequence of statements execute together.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
   {
      document.write("<h1>This is a header</h1>");
      document.write("<p>This is a paragraph</p>");
      document.write("<p>This is another paragraph</p>");
   }
</script>
```

Try it yourself.

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

You will learn more about functions and conditions in later chapters.

JavaScript Comments

JavaScript comments can be used to make the code more readable.

JavaScript Comments

Comments can be added to explain the JavaScript, or to make it more readable.

Single line comments start with //.

This example uses single line comments to explain the code:

```
<script type="text/javascript">
   // This will write a header:
   document.write("<h1>This is a header</h1>");
   // This will write two paragraphs:
   document.write("<p>This is a paragraph</p>");
   document.write("<p>This is another paragraph</p>");
</script>
```

Try it yourself.

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */.

This example uses a multi line comment to explain the code:

```
<script type="text/javascript">
   /*
   The code below will write
   one header and two paragraphs
   */
   document.write("<h1>This is a header</h1>");
   document.write("<p>This is a paragraph</p>");
   document.write("<p>This is another paragraph</p>");
</script>
```

Try it yourself.

Using Comments to Prevent Execution

In this example the comment is used to prevent the execution of a single code line:

```
<script type="text/javascript">
   document.write("<h1>This is a header</h1>");
   document.write("<p>This is a paragraph</p>");
   //document.write("<p>This is another paragraph</p>");
</script>
```

Try it yourself.

In this example the comments is used to prevent the execution of multiple code lines:

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

```
   document.write("<h1>This is a header</h1>");
   document.write("<p>This is a paragraph</p>");
   document.write("<p>This is another paragraph</p>");
   */
</script>
```

Try it yourself.

Using Comments at the End of a Line

In this example the comment is placed at the end of a line:

```
<script type="text/javascript">
   document.write("Hello"); // This will write "Hello"
   document.write("Dolly"); // This will write "Dolly"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

Do You Remember Algebra From School?

Do you remember algebra from school? x=5, y=6, z=x+y

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

This example will show you how

```
<html>
	<body>
		<script type="text/javascript">
			var firstname;
			firstname="Hege";
			document.write(firstname);
			document.write("<br />");
```

```
                    firstname="Tove";
                    document.write(firstname);
                </script>
                <p>The script above declares a variable,
                assigns a value to it, displays the value, change the value,
                and displays the value again.</p>
        </body>
</html>
```

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
carname="Volvo";
```

have the same effect as:

```
var x=5;
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;
z=y+5;
```

You will learn more about the operators that can be used in the next chapter of this tutorial.

JavaScript Operators

The operator = is used to assign values.

The operator + is used to add values.

The assignment operator **=** is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

Look at these examples:

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

Try it yourself.

The rule is:

If you add a number and a string, the result will be a string.

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

JavaScript If...Else Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

If statement
How to write an if statement.

```
<html>
<body>
	<script type="text/javascript">
	var d = new Date();
	var time = d.getHours();

	if (time < 10)  {
		document.write("<b>Good morning</b>");
	}
	</script>

	<p>This example demonstrates the If statement.</p>
	<p>If the time on your browser is less than 10, you will get a "Good morning" greeting.</p>
</body>
</html>
```

If...else statement
How to write an if...else statement.

```
<html>
<body>
	<script type="text/javascript">
		var d = new Date();
		var time = d.getHours();

		if (time < 10) {
			document.write("<b>Good morning</b>");
		}
	</script>
	<p>This example demonstrates the If statement.</p>
	<p>If the time on your browser is less than 10, you will get a "Good morning" greeting.</p>
</body>
</html>
```

If..else if...else statement
How to write an if..else if...else statement.

```
<html>
<body>

	<script type="text/javascript">
	var d = new Date();
	var time = d.getHours();
	if (time<10) {
		document.write("<b>Good morning</b>");
	} else if (time>=10 && time<16) {
		document.write("<b>Good day</b>");
	} else {
		document.write("<b>Hello World!</b>");
	}
	</script>

	<p>This example demonstrates the if..else if...else statement.</p>
</body>
</html>
```

Random link
This example demonstrates a link, when you click on the link it will take you to W3Schools.com OR to RefsnesData.no. There is a 50% chance for each of them.

```
<html>
	<body>
		<script type="text/javascript">
		var r=Math.random();
```

```
            if (r>0.5) {
                document.write("<a href='http://www.w3schools.com'>Learn Web Development!</a>");
            } else {
                document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>");
            }
            </script>
        </body>
</html>
```

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

If Statement

You should use the if statement if you want to execute some code only if a specified condition is true.

Syntax

```
if (condition) {
   code to be executed if condition is true
}
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example 1

```
<script type="text/javascript">
   //Write a "Good morning" greeting if
   //the time is less than 10
   var d=new Date();
   var time=d.getHours();
   if (time<10){
      document.write("<b>Good morning</b>");
   }
</script>
```

Example 2

```
<script type="text/javascript">
   //Write "Lunch-time!" if the time is 11
   var d=new Date();
   var time=d.getHours();
   if (time==11) {
      document.write("<b>Lunch-time!</b>");
   }
</script>
```

Note: When **comparing** variables you must always use two equals signs next to each other (==)!

Notice that there is no ..else.. in this syntax. You just tell the code to execute some code **only if the specified condition is true**.

If...else Statement

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

Syntax

```
if (condition) {
   code to be executed if condition is true
} else {
   code to be executed if condition is not true
}
```

Example

```
<script type="text/javascript">
   //If the time is less than 10,
   //you will get a "Good morning" greeting.
   //Otherwise you will get a "Good day" greeting.
   var d = new Date();
   var time = d.getHours();
   if (time < 10) {
      document.write("Good morning!");
   } else {
      document.write("Good day!");
   }
</script>
```

If...else if...else Statement

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

Syntax

```
if (condition1) {
   code to be executed if condition1 is true
} else if (condition2) {
   code to be executed if condition2 is true
} else {
   code to be executed if condition1 and
   condition2 are not true
}
```

Example

```
<script type="text/javascript">
   var d = new Date()
   var time = d.getHours()
   if (time<10) {
      document.write("<b>Good morning</b>");
   } else if (time>10 && time<16) {
      document.write("<b>Good day</b>");
   } else {
      document.write("<b>Hello World!</b>");
   }
</script>
```

JavaScript Switch Statement

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

Switch statement
How to write a switch statement.

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			theDay=d.getDay();
			switch (theDay) {
				case 5:
					document.write("<b>Finally Friday</b>");
					break;
				case 6:
					document.write("<b>Super Saturday</b>");
					break;
				case 0:
					document.write("<b>Sleepy Sunday</b>");
					break;
				default:
					document.write("<b>I'm really looking forward to this weekend!</b>");
			}
		</script>
		<p>This JavaScript will generate a different greeting based on what day it is.
		Note that Sunday=0, Monday=1, Tuesday=2, etc.</p>
	</body>
</html>
```

The JavaScript Switch Statement

You should use the switch statement if you want to select one of many blocks of code to be executed.

Syntax

```
switch(n) {
   case 1:
      execute code block 1
      break;
   case 2:
      execute code block 2
      break;
   default:
      code to be executed if n is
      different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
   //You will receive a different greeting based
   //on what day it is. Note that Sunday=0,
   //Monday=1, Tuesday=2, etc.
   var d=new Date();
   theDay=d.getDay();
   switch (theDay) {
      case 5:
         document.write("Finally Friday");
         break;
      case 6:
         document.write("Super Saturday");
         break;
      case 0:
         document.write("Sleepy Sunday");
         break;
      default:
         document.write("I'm looking forward to this weekend!");
   }
</script>
```

JavaScript Popup Boxes

In JavaScript we can create three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

Examples

Alert box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("I am an alert box!!");
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
</html>
```

Alert box with line breaks

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!");
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
</html>
```

Confirm box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_confirm() {
				var r=confirm("Press a button");
				if (r==true) {
					document.write("You pressed OK!");
				} else {
					document.write("You pressed Cancel!");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_confirm()" value="Display a confirm box" />
	</body>
</html>
```

Prompt box

```
<html>
	<head>
		<script type="text/javascript">
```

```
                function disp_prompt() {
                        var name=prompt("Please enter your name","Harry Potter");
                        if (name!=null && name!="") {
                                document.write("Hello " + name + "! How are you today?");
                        }
                }
        </script>
</head>
<body>
        <input type="button" onclick="disp_prompt()" value="Display a prompt box" />
</body>
</html>
```

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:

```
alert("sometext");
```

Example-1

```
<html>
        <head>
                <script type="text/javascript">
                        function disp_alert() {
                                alert("I am an alert box!!");
                        }
                </script>
        </head>
        <body>
                <input type="button" onclick="disp_alert()" value="Display alert box" />
        </body>
</html>
```

Example-2

```
<html>
        <head>
                <script type="text/javascript">
                        function disp_alert() {
                                alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!");
                        }
                </script>
        </head>
        <body>
                <input type="button" onclick="disp_alert()" value="Display alert box" />
        </body>
</html>
```

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:

```
confirm("sometext");
```

```
<html>
```

```
<head>
	<script type="text/javascript">
	function disp_confirm() {
		var r=confirm("Press a button");
		if (r==true) {
			document.write("You pressed OK!");
		} else {
			document.write("You pressed Cancel!");
		}
	}
	</script>
</head>
<body>
	<input type="button" onclick="disp_confirm()" value="Display a confirm box" />
</body>
</html>
```

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:

```
prompt("sometext","defaultvalue");
```

```
<html>
	<head>
		<script type="text/javascript">
			function disp_prompt() {
				var name=prompt("Please enter your name","Harry Potter");
				if (name!=null && name!="")  {
					document.write("Hello " + name + "! How are you today?");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_prompt()" value="Display a prompt box" />
	</body>
</html>
```

JavaScript Functions

A function is a reusable code-block that will be executed by an event, or when the function is called.

Examples

Function
How to call a function.

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction() {
				alert("HELLO");
			}
		</script>
	</head>
```

```
	<body>
		<form>
			<input type="button" onclick="myfunction()" value="Call function">
		</form>
		<p>By pressing the button, a function will be called. The function will alert a message.</p>
	</body>
</html>
```

Function with arguments

How to pass a variable to a function, and use the variable in the function.

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction(txt) {
				alert(txt);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" onclick="myfunction('Hello')" value="Call function">
		</form>
		<p>By pressing the button, a function with an argument will be called. The function will alert
		this argument.</p>
	</body>
</html>
```

Function with arguments 2

How to pass variables to a function, and use these variables in the function.

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction(txt) {
				alert(txt);
			}
		</script>
	</head>

	<body>
		<form>
			<input type="button" onclick="myfunction('Good Morning!')" value="In the Morning">
			<input type="button" onclick="myfunction('Good Evening!')" value="In the Evening">
		</form>
		<p>When you click on one of the buttons, a function will be called. The function will alert
		the argument that is passed to it.</p>
	</body>
</html>
```

Function that returns a value

How to let the function return a value.

```
<html>
	<head>
		<script type="text/javascript">
			function myFunction() {
				return ("Hello, have a nice day!");
			}
		</script>
	</head>
	<body>
		<script type="text/javascript">
			document.write(myFunction())
		</script>
		<p>The script in the body section calls a function.</p>
		<p>The function returns a text.</p>
```

```
        </body>
</html>
```

A function with arguments, that returns a value
How to let the function find the product of two arguments and return the result.

```
<html>
        <head>
                <script type="text/javascript">
                        function product(a,b) {
                                return a*b;
                        }
                </script>
        </head>
        <body>
                <script type="text/javascript">
                        document.write(product(4,3));
                </script>
                <p>The script in the body section calls a function with two parameters (4 and 3).</p>
                <p>The function will return the product of these two parameters.</p>
        </body>
</html>
```

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to that function.

You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

Example

```
<html>
   <head>
      <script type="text/javascript">
         function displaymessage(){
            alert("Hello World!");
         }
      </script>
   </head>
   <body>
      <form>
         <input type="button" value="Click me!" onclick="displaymessage()" >
      </form>
   </body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an onClick event to the button that will execute the function displaymessage() when the button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

How to Define a Function

The syntax for creating a function is:

```
function functionname(var1,var2,...,varX){
    some code
```

```
}
```

var1, var2, etc are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name:

```
function functionname(){
    some code
}
```

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

Example

The function below should return the product of two numbers (a and b):

```
function prod(a,b) {
   x=a*b;
   return x;
}
```

When you call the function above, you must pass along two parameters:

```
product=prod(2,3);
```

The returned value from the prod() function is 6, and it will be stored in the variable called product.

The Lifetime of JavaScript Variables

When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript For Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Examples

For loop
How to write a for loop. Use a For loop to run the same block of code a specified number of times.

```
<html>
<body>
        <script type="text/javascript">
                for (i = 0; i <= 5; i++) {
                        document.write("The number is " + i);
```

```
                document.write("<br />");
            }
        </script>
        <p>Explanation:</p>
        <p>This for loop starts with i=0.</p>
        <p>As long as <b>i</b> is less than, or equal to 5, the loop will continue to run.</p>
        <p><b>i</b> will increase by 1 each time the loop runs.</p>
</body>
</html>
```

Looping through HTML headers
How to use the for loop to loop through the different HTML headers.

```
<html>
        <body>
                <script type="text/javascript">
                        for (i = 1; i <= 6; i++) {
                                document.write("<h" + i + ">This is header " + i);
                                document.write("</h" + i + ">");
                        }
                </script>
        </body>
</html>
```

JavaScript Loops

Very often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment) {
    code to be executed
}
```

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

```
<html>
   <body>
      <script type="text/javascript">
         var i=0;
         for (i=0;i<=10;i++) {
            document.write("The number is " + i);
            document.write("<br />");
         }
      </script>
   </body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

JavaScript While Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Examples

While loop
How to write a while loop. Use a while loop to run the same block of code while a specified condition is true.

```
<html>
	<body>
		<script type="text/javascript">
			i=0;
			while (i<=5) {
				document.write("The number is " + i);
				document.write("<br />");
				i++;
			}
		</script>
		<p>Explanation:</p>
		<p><b>i</b> is equal to 0.</p>
		<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>
		<p><b>i</b> will increase by1 each time the loop runs.</p>
	</body>
</html>
```

Do while loop
How to write a do...while loop. Use a do...while loop to run the same block of code while a specified condition is true. This loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested.

```
<html>
	<body>
		<script type="text/javascript">
			i = 0;
			do {
				document.write("The number is " + i);
				document.write("<br />");
				i++;
			} while (i <= 5)
		</script>
		<p>Explanation:</p>
		<p><b>i</b>  equal to 0.</p>
		<p>The loop will run</p>
		<p><b>i</b> will increase by 1 each time the loop runs.</p>
		<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>
	</body>
</html>
```

The while loop

The while loop is used when you want the loop to execute and continue executing while the specified condition is true.

```
while (var<=endvalue){
    code to be executed
}
```

Note: The <= could be any comparing statement.

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

```
<html>
   <body>
      <script type="text/javascript">
         var i=0;
         while (i<=10){
            document.write("The number is " + i);
            document.write("<br />");
            i=i+1;
         }
      </script>
   </body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

```
do {
    code to be executed
} while (var<=endvalue);
```

Example

```
<html>
   <body>
      <script type="text/javascript">
         var i=0;
         do {
            document.write("The number is " + i);
            document.write("<br />");
            i=i+1;
         } while (i<0);
      </script>
   </body>
</html>
```

Result

```
The number is 0
```

JavaScript Break and Continue

There are two special statements that can be used inside loops: break and continue.

Examples

Break statement
Use the break statement to break the loop.

```
<html>
	<body>
		<script type="text/javascript">
			var i=0;
			for (i=0;i<=10;i++) {
				if (i==3) {
					break;
				}
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation: The loop will break when i=3.</p>
	</body>
</html>
```

Continue statement
Use the continue statement to break the current loop and continue with the next value.

```
<html>
	<body>
		<script type="text/javascript">
			var i=0;
			for (i=0;i<=10;i++) {
				if (i==3) {
					continue;
				}
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation: The loop will break the current loop and continue with the next value when i=3.</p>
	</body>
</html>
```

JavaScript break and continue Statements

There are two special statements that can be used inside loops: break and continue.

Break

The break command will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
   <body>
      <script type="text/javascript">
         var i=0;
         for (i=0;i<=10;i++) {
```

```
            if (i==3) {
               break;
            }
            document.write("The number is " + i);
            document.write("<br />");
         }
      </script>
   </body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
```

Continue

The continue command will break the current loop and continue with the next value.

Example

```
<html>
   <body>
      <script type="text/javascript">
         var i=0
         for (i=0;i<=10;i++) {
            if (i==3) {
               continue;
            }
            document.write("The number is " + i);
            document.write("<br />");
         }
      </script>
   </body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

Examples

For...In statement
How to use a for...in statement to loop through the elements of an array.

```
<html>
	<body>
		<script type="text/javascript">
			var x;
			var mycars = new Array();
			mycars[0] = "Saab";
			mycars[1] = "Volvo";
			mycars[2] = "BMW";
			for (x in mycars) {
				document.write(mycars[x] + "<br />");
			}
		</script>
	</body>
</html>
```

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

The code in the body of the for ... in loop is executed once for each element/property.

Syntax

```
for (variable in object) {
    code to be executed
}
```

The variable argument can be a named variable, an array element, or a property of an object.

Example

Using for...in to loop through an array:

```
<html>
   <body>
      <script type="text/javascript">
         var x;
         var mycars = new Array();
         mycars[0] = "Saab";
         mycars[1] = "Volvo";
         mycars[2] = "BMW";
         for (x in mycars) {
            document.write(mycars[x] + "<br />");
         }
      </script>
   </body>
</html>
```

JavaScript Events

Events are actions that can be detected by JavaScript.

Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

For a complete reference of the events recognized by JavaScript, go to our complete Event reference.

onload and onUnload

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

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

Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://w3schools.com" onmouseover="alert('An onMouseOver event');return false">
    <img src="w3schools.gif" width="100" height="30">
</a>
```

JavaScript Try...Catch Statement

The try...catch statement allows you to test a block of code for errors.

Examples

The try...catch statement
How to write a try...catch statement.

```
<html>
	<head>
		<script type="text/javascript">
			var txt="";
			function message() {
				try {
					adddlert("Welcome guest!");
				} catch(err) {
					txt="There was an error on this page.\n\n";
					txt+="Error description: " + err.description + "\n\n";
					txt+="Click OK to continue.\n\n";
					alert(txt);
				}
			}
		</script>
	</head>
	<body>
		<input type="button" value="View message" onclick="message()" />
	</body>
</html>
```

The try...catch statement with a confirm box
Another example of how to write a try...catch statement.

```
<html>
	<head>
		<script type="text/javascript">
			var txt=""
			function message() {
				try {
					adddlert("Welcome guest!");
				} catch(err) {
					txt="There was an error on this page.\n\n";
					txt+="Click OK to continue viewing this page,\n";
					txt+="or Cancel to return to the home page.\n\n";
					if (!confirm(txt)) {
						document.location.href="http://www.w3schools.com/";
					}
				}
			}
		</script>
	</head>
	<body>
		<input type="button" value="View message" onclick="message()" />
	</body>
</html>
```

JavaScript - Catching Errors

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page.

This chapter will teach you how to trap and handle JavaScript error messages, so you don't lose your audience.

There are two ways of catching errors in a Web page:

- By using the **try...catch** statement (available in IE5+, Mozilla 1.0, and Netscape 6)
- By using the **onerror** event. This is the old standard solution to catch errors (available since Netscape 3)

Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
try {
   //Run some code here
} catch(err) {
   //Handle errors here
}
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

The example below contains a script that is supposed to display the message "Welcome guest!" when you click on a button. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs:

```
<html>
   <head>
      <script type="text/javascript">
         function message() {
            adddlert("Welcome guest!");
         }
      </script>
   </head>
   <body>
      <input type="button" value="View message" onclick="message()" />
   </body>
</html>
```

To take more appropriate action when an error occurs, you can add a try...catch statement.

The example below contains the "Welcome guest!" example rewritten to use the try...catch statement. Since alert() is misspelled, a JavaScript error occurs. However, this time, the catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

```
<html>
   <head>
      <script type="text/javascript">
         var txt=""
         function message() {
            try {
               adddlert("Welcome guest!");
            } catch(err) {
               txt="There was an error on this page.\n\n";
               txt+="Error description: " + err.description + "\n\n";
               txt+="Click OK to continue.\n\n";
               alert(txt);
            }
         }
      </script>
   </head>
   <body>
      <input type="button" value="View message" onclick="message()" />
   </body>
</html>
```

Example 2

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

```
<html>
   <head>
      <script type="text/javascript">
         var txt=""
         function message() {
            try {
               adddlert("Welcome guest!");
            } catch(err) {
               txt="There was an error on this page.\n\n";
               txt+="Click OK to continue viewing this page,\n";
               txt+="or Cancel to return to the home page.\n\n";
               if(!confirm(txt)) {
                  document.location.href="http://www.w3schools.com/";
               }
            }
         }
      </script>
   </head>
   <body>
      <input type="button" value="View message" onclick="message()" />
   </body>
</html>
```

The onerror Event

The onerror event will be explained soon, but first you will learn how to use the throw statement to create an exception. The throw statement can be used together with the try...catch statement.

JavaScript Throw Statement

The throw statement allows you to create an exception.

Examples

The throw statement
How to use the throw statement.

```
<html>
	<body>
		<script type="text/javascript">
			var x=prompt("Enter a number between 0 and 10:","");
			try {
				if (x>10)		{ throw "Err1";
				} else if(x<0)		{ throw "Err2";
				} else if(isNaN(x))	{ throw "Err3";
				}
			} catch(er) {
				if(er=="Err1") {
					alert("Error! The value is too high");
				}
				if(er=="Err2") {
					alert("Error! The value is too low");
```

```
                        }
                        if(er=="Err3") {
                                alert("Error! The value is not a number");
                        }
                }
        </script>
    </body>
</html>
```

The Throw Statement

The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

```
throw(exception)
```

The exception can be a string, integer, Boolean or an object.

Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

The example below determines the value of a variable called x. If the value of x is higher than 10 or lower than 0 we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

```
<html>
   <body>
      <script type="text/javascript">
         var x=prompt("Enter a number between 0 and 10:","");
         try {
            if(x>10)      throw "Err1";
            else if(x<0) throw "Err2";
         }catch(er) {
            if(er=="Err1")    alert("Error! The value is too high");
            if(er == "Err2") alert("Error! The value is too low");
         }
      </script>
   </body>
</html>
```

JavaScript The onerror Event

Using the onerror event is the old standard solution to catch errors in a web page.

Examples

The onerror event (an example with an error)
How to use the onerror event to catch errors in a web page.

```
<html>
        <head>
                <script type="text/javascript">
                        onerror=handleErr;
                        var txt="";

                        function handleErr(msg,url,l) {
                                txt="There was an error on this page.\n\n";
                                txt+="Error: " + msg + "\n";
                                txt+="URL: " + url + "\n";
                                txt+="Line: " + l + "\n\n";
                                txt+="Click OK to continue.\n\n";
                                alert(txt);
```

```
                        return true;
                }
                function message() {
                        adddlert("Welcome guest!");
                }
            </script>
        </head>
        <body>
            <input type="button" value="View message" onclick="message()" />
        </body>
</html>
```

The onerror Event

We have just explained how to use the try...catch statement to catch errors in a web page. Now we are going to explain how to use the onerror event for the same purpose.

The onerror event is fired whenever there is a script error in the page.

To use the onerror event, you must create a function to handle the errors. Then you call the function with the onerror event handler. The event handler is called with three arguments: msg (error message), url (the url of the page that caused the error) and line (the line where the error occurred).

Syntax

```
onerror=handleErr
function handleErr(msg,url,l) {
   //Handle the error here
   return true or false
}
```

The value returned by onerror determines whether the browser displays a standard error message. If you return false, the browser displays the standard error message in the JavaScript console. If you return true, the browser does not display the standard error message.

Example

The following example shows how to catch the error with the onerror event:

```
<html>
   <head>
      <script type="text/javascript">
         onerror=handleErr;
         var txt="";
         function handleErr(msg,url,l) {
            txt="There was an error on this page.\n\n";
            txt+="Error: " + msg + "\n";
            txt+="URL: " + url + "\n";
            txt+="Line: " + l + "\n\n";
            txt+="Click OK to continue.\n\n";
            alert(txt);
            return true;
         }
         function message() {
            adddlert("Welcome guest!");
         }
      </script>
   </head>
   <body>
      <input type="button" value="View message" onclick="message()" />
   </body>
</html>
```

Milan Das, SARC

JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

```
document.write ("You \& I are singing!");
```

The example above will produce the following output:

```
You & I are singing!
```

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	Tab
\b	backspace
\f	form feed

JavaScript Guidelines

Some other important things to know when scripting with JavaScript.

JavaScript is Case Sensitive

A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar".

JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

White Space

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines are equivalent:

```
name="Hege";
name = "Hege";
```

Break up a Code Line

You can break up a code line **within a text string** with a backslash. The example below will be displayed properly:

```
document.write("Hello \
World!");
```

However, you cannot break up a code line like this:

```
document.write \
("Hello World!");
```

JavaScript Objects Introduction

JavaScript is an Object Oriented Programming (OOP) language.

An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Note that an object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
   var txt="Hello World!";
   document.write(txt.length);
</script>
```

The output of the code above will be:

```
12
```

Methods

Methods are the actions that can be performed on objects.

In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
   var str="Hello world!";
   document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

```
HELLO WORLD!
```

JavaScript String Object

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

Examples

Return the length of a string
How to use the length property to find the length of a string.

```
<html>
	<body>
		<script type="text/javascript">
			var txt="Hello World!";
			document.write(txt.length);
		</script>
	</body>
</html>
```

Style strings
How to style strings.

```
<html>
	<body>
		<script type="text/javascript">
			var txt="Hello World!";

			document.write("<p>Big: " + txt.big() + "</p>");
			document.write("<p>Small: " + txt.small() + "</p>");

			document.write("<p>Bold: " + txt.bold() + "</p>");
			document.write("<p>Italic: " + txt.italics() + "</p>");

			document.write("<p>Blink: " + txt.blink() + " (does not work in IE)</p>");
			document.write("<p>Fixed: " + txt.fixed() + "</p>");
			document.write("<p>Strike: " + txt.strike() + "</p>");

			document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>");
			document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>");

			document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>");
			document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>");

			document.write("<p>Subscript: " + txt.sub() + "</p>");
			document.write("<p>Superscript: " + txt.sup() + "</p>");

			document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>");
		</script>
```

```
	</body>
</html>
```

The indexOf() method
How to use the indexOf() method to return the position of the first occurrence of a specified string value in a string.

```
<html>
	<body>
		<script type="text/javascript">
			var str="Hello world!";
			document.write(str.indexOf("Hello") + "<br />");
			document.write(str.indexOf("World") + "<br />");
			document.write(str.indexOf("world"));
		</script>
	</body>
</html>
```

The match() method
How to use the match() method to search for a specified string value within a string and return the string value if found

```
<html>
	<body>
		<script type="text/javascript">
			var str="Hello world!";
			document.write(str.match("world") + "<br />");
			document.write(str.match("World") + "<br />");
			document.write(str.match("worlld") + "<br />");
			document.write(str.match("world!"));
		</script>
	</body>
</html>
```

Replace characters in a string - replace()
How to use the replace() method to replace some characters with some other characters in a string.

```
<html>
	<body>
		<script type="text/javascript">
			var str="Visit Microsoft!";
			document.write(str.replace(/Microsoft/,"W3Schools"));
		</script>
	</body>
</html>
```

Complete String Object Reference

For a complete reference of all the properties and methods that can be used with the String object, go to our refer to the annexure given below.

The reference contains a brief description and examples of use for each property and method!

String object

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

Examples of use:

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

```
var txt="Hello world!";
document.write(txt.length);
```

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";
document.write(txt.toUpperCase());
```

The code above will result in the following output:

```
HELLO WORLD!
```

JavaScript Date Object

The Date object is used to work with dates and times.

Examples

Return today's date and time
How to use the Date() method to get today's date.

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Date());
		</script>
	</body>
</html>
```

getTime()
Use getTime() to calculate the years since 1970.

```
<html>
	<body>
		<script type="text/javascript">
			var minutes = 1000*60;
			var hours = minutes*60;
			var days = hours*24;
			var years = days*365;
			var d = new Date();
			var t = d.getTime();
			var y = t/years;
			document.write("It's been: " + y + " years since 1970/01/01!");
		</script>
	</body>
</html>
```

setFullYear()
How to use setFullYear() to set a specific date.

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			d.setFullYear(1992,10,3);
			document.write(d);
		</script>
	</body>
</html>
```

toUTCString()
How to use toUTCString() to convert today's date (according to UTC) to a string.

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			document.write (d.toUTCString());
		</script>
	</body>
</html>
```

getDay()
Use getDay() and an array to write a weekday, and not just a number.

```
<html>
	<body>
		<script type="text/javascript">
			var d=new Date();
			var weekday=new Array(7);
			weekday[0]="Sunday";
			weekday[1]="Monday";
			weekday[2]="Tuesday";
			weekday[3]="Wednesday";
			weekday[4]="Thursday";
			weekday[5]="Friday";
			weekday[6]="Saturday";
			document.write("Today it is " + weekday[d.getDay()]);
		</script>
	</body>
</html>
```

Display a clock
How to display a clock on your web page.

```
<html>
	<head>
		<script type="text/javascript">
			function startTime() {
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				// add a zero in front of numbers<10
				m=checkTime(m);
				s=checkTime(s);
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}
			function checkTime(i) {
				if (i<10) {
					i="0" + i;
				}
				return i;
			}
		</script>
	</head>

	<body onload="startTime()">
		<div id="txt"></div>
	</body>
</html>
```

Complete Date Object Reference

For a complete reference of all the properties and methods that can be used with the Date object, refer to the annexure given below.

The reference contains a brief description and examples of use for each property and method!

Create a Date Object

The Date object is used to work with dates and times.

The following code create a Date object called myDate:

```
var myDate=new Date()
```

Note: The Date object will automatically hold the current date and time as its initial value!

Set Dates

We can easily manipulate the date by using the methods available for the Date object.

In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates

The Date object is also used to compare two dates.

The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();
if (myDate>today) {
   alert("Today is before 14th January 2010");
} else {
   alert("Today is after 14th January 2010");
}
```

JavaScript Array Object

The Array object is used to store multiple values in a single variable.

Examples

Create an array
Create an array, assign values to it, and write the values to the output.

<html>

```
	<body>
		<script type="text/javascript">
			var mycars = new Array();
			mycars[0] = "Saab";
			mycars[1] = "Volvo";
			mycars[2] = "BMW";

			for (i=0;i<mycars.length;i++) {
				document.write(mycars[i] + "<br />");
			}
		</script>
	</body>
</html>
```

For...In Statement

How to use a for...in statement to loop through the elements of an array.

```
<html>
	<body>
		<script type="text/javascript">
			var x;
			var mycars = new Array();
			mycars[0] = "Saab";
			mycars[1] = "Volvo";
			mycars[2] = "BMW";

			for (x in mycars) {
				document.write(mycars[x] + "<br />");
			}
		</script>
	</body>
</html>
```

Join two arrays - concat()

How to use the concat() method to join two arrays.

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(3);
			arr[0] = "Jani";
			arr[1] = "Tove";
			arr[2] = "Hege";

			var arr2 = new Array(3);
			arr2[0] = "John";
			arr2[1] = "Andy";
			arr2[2] = "Wendy";

			document.write(arr.concat(arr2));
		</script>
	</body>
</html>
```

Put array elements into a string - join()

How to use the join() method to put all the elements of an array into a string.

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(3);
			arr[0] = "Jani";
			arr[1] = "Hege";
			arr[2] = "Stale";

			document.write(arr.join() + "<br />");
			document.write(arr.join("."));
```

```
		</script>
	</body>
</html>
```

Literal array - sort()
How to use the sort() method to sort a literal array.

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(6);
			arr[0] = "Jani";
			arr[1] = "Hege";
			arr[2] = "Stale";
			arr[3] = "Kai Jim";
			arr[4] = "Borge";
			arr[5] = "Tove";

			document.write(arr + "<br />");
			document.write(arr.sort());
		</script>
	</body>
</html>
```

Numeric array - sort()
How to use the sort() method to sort a numeric array.

```
<html>
	<body>
		<script type="text/javascript">
			function sortNumber(a, b) {
				return a - b;
			}
			var arr = new Array(6);
			arr[0] = "10";
			arr[1] = "5";
			arr[2] = "40";
			arr[3] = "25";
			arr[4] = "1000";
			arr[5] = "1";

			document.write(arr + "<br />");
			document.write(arr.sort(sortNumber));
		</script>
	</body>
</html>
```

Complete Array Object Reference

For a complete reference of all the properties and methods that can be used with the Array object, refer to the annexure given below.

The reference contains a brief description and examples of use for each property and method!

Create an Array

The following code creates an Array object called myCars:

```
var myCars=new Array()
```

There are two ways of adding values to an array (you can add as many values as you need to define as many variables you require).

1:

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

You could also pass an integer argument to control the array's size:

```
var myCars=new Array(3);
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

2:

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

Note: If you specify numbers or true/false values inside the array then the type of variables will be numeric or Boolean instead of string.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

JavaScript Boolean Object

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

Examples

Check Boolean value
Check if a Boolean object is true or false.

```
<html>
	<body>
		<script type="text/javascript">
			var b1=new Boolean( 0);
			var b2=new Boolean(1);
			var b3=new Boolean("");
			var b4=new Boolean(null);
			var b5=new Boolean(NaN);
			var b6=new Boolean("false");

			document.write("0 is boolean "+ b1 +"<br />");
			document.write("1 is boolean "+ b2 +"<br />");
			document.write("An empty string is boolean "+ b3 + "<br />");
			document.write("null is boolean "+ b4+ "<br />");
			document.write("NaN is boolean "+ b5 +"<br />");
			document.write("The string 'false' is boolean "+ b6 +"<br />");
		</script>
	</body>
</html>
```

Complete Boolean Object Reference

For a complete reference of all the properties and methods that can be used with the Boolean object, refer to the annexure given in the end of the book.

The reference contains a brief description and examples of use for each property and method!

Create a Boolean Object

The Boolean object represents two values: "true" or "false".

The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

Note: If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!

All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

And all the following lines of code create Boolean objects with an initial value of true:

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");
```

JavaScript Math Object

The Math object allows you to perform mathematical tasks.

Examples

round()
How to use round().

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.round(0.60) + "<br />");
			document.write(Math.round(0.50) + "<br />");
			document.write(Math.round(0.49) + "<br />");
			document.write(Math.round(-4.40) + "<br />");
			document.write(Math.round(-4.60));
		</script>
	</body>
</html>
```

random()
How to use random() to return a random number between 0 and 1.

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.random());
		</script>
	</body>
</html>
```

max()
How to use max() to return the number with the highest value of two specified numbers.

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.max(5,7) + "<br />");
			document.write(Math.max(-3,5) + "<br />");
			document.write(Math.max(-3,-5) + "<br />");
			document.write(Math.max(7.25,7.30));
		</script>
	</body>
</html>
```

min()
How to use min() to return the number with the lowest value of two specified numbers.

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.min(5,7) + "<br />");
			document.write(Math.min(-3,5) + "<br />");
			document.write(Math.min(-3,-5) + "<br />");
			document.write(Math.min(7.25,7.30));
		</script>
	</body>
</html>
```

Complete Math Object Reference

For a complete reference of all the properties and methods that can be used with the Math object, go to our refer to the annexure given in the end of the book.

The reference contains a brief description and examples of use for each property and method!

Math Object

The Math object allows you to perform mathematical tasks.

The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Mathematical Constants

JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

You may reference these constants from your JavaScript like this:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Mathematical Methods

In addition to the mathematical constants that can be accessed from the Math object there are also several methods available.

The following example uses the round() method of the Math object to round a number to the nearest integer:

```
document.write(Math.round(4.7));
```

The code above will result in the following output:

```
5
```

The following example uses the random() method of the Math object to return a random number between 0 and 1:

```
document.write(Math.random());
```

The code above can result in the following output:

```
0.542325422806283
```

The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

```
document.write(Math.floor(Math.random()*11));
```

The code above can result in the following output:

```
3
```

JavaScript RegExp Object

The RegExp object is used to specify what to search for in a text

What is RegExp

RegExp, is short for regular expression. When you search in a text, you can use a pattern to describe what you are searching for. **RegExp IS this pattern**.

A simple pattern can be a single character. A more complicated pattern consists of more characters, and can be used for parsing, format checking, substitution and more.

You can specify where in the string to search, what type of characters to search for, and more.

Defining RegExp

The RegExp object is used to store the search pattern.

We define a RegExp object with the *new* keyword. The following code line defines a RegExp object called patt1 with the pattern "e":

```
var patt1=new RegExp("e");
```

When you use this RegExp object to search in a string, you will find the letter "e".

Methods of the RegExp Object

The RegExp Object has 3 methods: test(), exec(), and compile().

test()

The test() method searches a string for a specified value. Returns true or false

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
true
```

Try it yourself

exec()

The exec() method searches a string for a specified value. Returns the text of the found value. If no match is found, it returns *null*

Example 1:

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
E
```

Try it yourself

Example 2:

You can add a second parameter to the RegExp object, to specify your search. For example; if you want to find all occurrences of a character, you can use the "g" parameter ("global").

For a complete list of how to modify your search, refer to the RegExp object reference given in the annexure.

When using the "g" parameter, the exec() method works like this:

- Finds the first occurrence of "e", and stores its position
- If you run exec() again, it starts at the stored position, and finds the next occurence of "e", and stores its position

```
var patt1=new RegExp("e","g");
do {
   result=patt1.exec("The best things in life are free");
   document.write(result);
} while (result!=null)
```

Since there is six "e" letters in the string, the output of the code above will be:

```
eeeeeenull
```

Try it yourself

compile()

The compile() method is used to change the RegExp.

compile() can change both the search pattern, and add or remove the second parameter.

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
patt1.compile("d");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, but not a "d", the output of the code above will be:

```
Truefalse
```

Try it yourself

Complete RegExp Object Reference

For a complete reference of all the properties and methods that can be used with the RegExp object, refer to the RegExp object reference given in the annexure.

The reference contains a brief description and examples of use for each property and method including the string object

Milan Das, SARC

JavaScript HTML DOM Objects

In addition to the built-in JavaScript objects, you can also access and manipulate all of the HTML DOM objects with JavaScript.

More JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events.

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

The HTML DOM

The HTML DOM is a W3C standard and it is an abbreviation for the Document Object Model for HTML.

The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents.

All HTML elements, along with their containing text and attributes, can be accessed through the DOM. The contents can be modified or deleted, and new elements can be created.

The HTML DOM is platform and language independent. It can be used by any programming language like Java, JavaScript, and VBScript.

Follow the links below to learn more about how to access and manipulate each DOM object with JavaScript:

Object	Description
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element
Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form

Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element
Meta	Represents a <meta> element
Option	Represents an <option> element
Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

JavaScript Browser Detection

The JavaScript Navigator object contains information about the visitor's browser.

Examples

Detect the visitor's browser and browser version

```
<html>
	<body>
		<script type="text/javascript">
			var browser=navigator.appName;
			var b_version=navigator.appVersion;
			var version=parseFloat(b_version);
			document.write("Browser name: "+ browser);
			document.write("<br />");
			document.write("Browser version: "+ version);
		</script>
	</body>
</html>
```

More details about the visitor's browser

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<p>Browser: ");
			document.write(navigator.appName + "</p>");

			document.write("<p>Browserversion: ");
			document.write(navigator.appVersion + "</p>");

			document.write("<p>Code: ");
			document.write(navigator.appCodeName + "</p>");

			document.write("<p>Platform: ");
			document.write(navigator.platform + "</p>");

			document.write("<p>Cookies enabled: ");
			document.write(navigator.cookieEnabled + "</p>");

			document.write("<p>Browser's user agent header: ");
```

```
                document.write(navigator.userAgent + "</p>");
            </script>
        </body>
</html>
```

All details about the visitor's browser

```
<html>
        <body>
            <script type="text/javascript">
                var x = navigator;
                document.write("CodeName=" + x.appCodeName);
                document.write("<br />");
                document.write("MinorVersion=" + x.appMinorVersion);
                document.write("<br />");
                document.write("Name=" + x.appName);
                document.write("<br />");
                document.write("Version=" + x.appVersion);
                document.write("<br />");
                document.write("CookieEnabled=" + x.cookieEnabled);
                document.write("<br />");
                document.write("CPUClass=" + x.cpuClass);
                document.write("<br />");
                document.write("OnLine=" + x.onLine);
                document.write("<br />");
                document.write("Platform=" + x.platform);
                document.write("<br />");
                document.write("UA=" + x.userAgent);
                document.write("<br />");
                document.write("BrowserLanguage=" + x.browserLanguage);
                document.write("<br />");
                document.write("SystemLanguage=" + x.systemLanguage);
                document.write("<br />");
                document.write("UserLanguage=" + x.userLanguage);
            </script>
        </body>
</html>
```

Alert user, depending on browser

```
<html>
        <head>
            <script type="text/javascript">
                function detectBrowser() {
                    var browser=navigator.appName;
                    var b_version=navigator.appVersion;
                    var version=parseFloat(b_version);
                    if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") && (version>=4))
                    {
                        alert("Your browser is good enough!");
                    } else {
                        alert("It's time to upgrade your browser!");
                    }
                }
            </script>
        </head>
        <body onload="detectBrowser()">
        </body>
</html>
```

Browser Detection

Almost everything in this tutorial works on all JavaScript-enabled browsers. However, there are some things that just don't work on certain browsers - especially on older browsers.

So, sometimes it can be very useful to detect the visitor's browser type and version, and then serve up the appropriate information.

The best way to do this is to make your web pages smart enough to look one way to some browsers and another way to other browsers.

JavaScript includes an object called the Navigator object, that can be used for this purpose.

The Navigator object contains information about the visitor's browser name, browser version, and more.

The Navigator Object

The JavaScript Navigator object contains all information about the visitor's browser. We are going to look at two properties of the Navigator object:

- appName - holds the name of the browser
- appVersion - holds, among other things, the version of the browser

Example

```
<html>
   <body>
      <script type="text/javascript">
         var browser=navigator.appName;
         var b_version=navigator.appVersion;
         var version=parseFloat(b_version);
         document.write("Browser name: "+ browser);
         document.write("<br />");
         document.write("Browser version: "+ version);
      </script>
   </body>
</html>
```

The variable browser in the example above holds the name of the browser, i.e. "Netscape" or "Microsoft Internet Explorer".

The appVersion property in the example above returns a string that contains much more information than just the version number, but for now we are only interested in the version number. To pull the version number out of the string we are using a function called parseFloat(), which pulls the first thing that looks like a decimal number out of a string and returns it.

IMPORTANT! The version number is WRONG in IE 5.0 or later! Microsoft starts the appVersion string with the number 4.0. in IE 5.0 and IE 6.0!!! Why did they do that??? However, JavaScript is the same in IE6, IE5 and IE4, so for most scripts it is ok.

Example

The script below displays a different alert, depending on the visitor's browser:

```
<html>
   <head>
      <script type="text/javascript">
         function detectBrowser(){
            var browser=navigator.appName;
            var b_version=navigator.appVersion;
            var version=parseFloat(b_version);
            if ((browser=="Netscape"||browser=="Microsoft Internet Explorer")
               && (version>=4)) {
               alert("Your browser is good enough!");
            } else {
               alert("It's time to upgrade your browser!");
            }
         }
      </script>
   </head>
   <body onload="detectBrowser()">
   </body>
</html>
```

JavaScript Cookies

A cookie is often used to identify a user.

Examples

Create a welcome cookie

```
<html>
	<head>
		<script type="text/javascript">
			function getCookie(c_name) {
				if (document.cookie.length>0) {
					c_start=document.cookie.indexOf(c_name + "=");
					if (c_start!=-1) {
						c_start=c_start + c_name.length+1 ;
						c_end=document.cookie.indexOf(";",c_start);
						if (c_end==-1) c_end=document.cookie.length
						return unescape(document.cookie.substring(c_start,c_end));
					}
				}
				return ""
			}
			function setCookie(c_name,value,expiredays) {
				var exdate=new Date();
				exdate.setDate(exdate.getDate()+expiredays);
				document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" : ";
				expires="+exdate.toGMTString());
			}
			function checkCookie() {
				username=getCookie('username');
				if (username!=null && username!="") {
					alert('Welcome again '+username+'!');
				} else {
					username=prompt('Please enter your name:',"");
					if (username!=null && username!="") {
						setCookie('username',username,365);
					}
				}
			}
		</script>
	</head>
	<body onLoad="checkCookie()">
	</body>
</html>
```

What is a Cookie?

A cookie is a variable that is stored on the visitor's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With JavaScript, you can both create and retrieve cookie values.

Examples of cookies:

- Name cookie - The first time a visitor arrives to your web page, he or she must fill in her/his name. The name is then stored in a cookie. Next time the visitor arrives at your page, he or she could get a welcome message like "Welcome John Doe!" The name is retrieved from the stored cookie
- Password cookie - The first time a visitor arrives to your web page, he or she must fill in a password. The password is then stored in a cookie. Next time the visitor arrives at your page, the password is retrieved from the cookie
- Date cookie - The first time a visitor arrives to your web page, the current date is stored in a cookie. Next time the visitor arrives at your page, he or she could get a message like "Your last visit was on Tuesday August 11, 2005!" The date is retrieved from the stored cookie

Milan Das, SARC

Create and Store a Cookie

In this example we will create a cookie that stores the name of a visitor. The first time a visitor arrives to the web page, he or she will be asked to fill in her/his name. The name is then stored in a cookie. The next time the visitor arrives at the same page, he or she will get welcome message.

First, we create a function that stores the name of the visitor in a cookie variable:

```
function setCookie(c_name,value,expiredays) {
   var exdate=new Date();
   exdate.setDate(exdate.getDate()+expiredays);
   document.cookie=c_name+ "=" +escape(value)+
                          ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
```

The parameters of the function above hold the name of the cookie, the value of the cookie, and the number of days until the cookie expires.

In the function above we first convert the number of days to a valid date, then we add the number of days until the cookie should expire. After that we store the cookie name, cookie value and the expiration date in the document.cookie object.

Then, we create another function that checks if the cookie has been set:

```
function getCookie(c_name) {
   if (document.cookie.length>0) {
      c_start=document.cookie.indexOf(c_name + "=");
   if (c_start!=-1) {
      c_start=c_start + c_name.length+1;
      c_end=document.cookie.indexOf(";",c_start);
      if (c_end==-1) c_end=document.cookie.length;
         return unescape(document.cookie.substring(c_start,c_end));
      }
   }
   return "";
}
```

The function above first checks if a cookie is stored at all in the document.cookie object. If the document.cookie object holds some cookies, then check to see if our specific cookie is stored. If our cookie is found, then return the value, if not - return an empty string.

Last, we create the function that displays a welcome message if the cookie is set, and if the cookie is not set it will display a prompt box, asking for the name of the user:

```
function checkCookie() {
   username=getCookie('username');
   if (username!=null && username!="") {
      alert('Welcome again '+username+'!');
   } else {
      username=prompt('Please enter your name:',"");
      if (username!=null && username!="") {
         setCookie('username',username,365);
      }
   }
}
```

All together now:

```
<html>
   <head>
      <script type="text/javascript">
         function getCookie(c_name) {
            if (document.cookie.length>0) {
               c_start=document.cookie.indexOf(c_name + "=");
               if (c_start!=-1) {
                  c_start=c_start + c_name.length+1;
```

```
                c_end=document.cookie.indexOf(";",c_start);
                if (c_end==-1) c_end=document.cookie.length;
                return unescape(document.cookie.substring(c_start,c_end));
             }
          }
          return "";
       }
       function setCookie(c_name,value,expiredays) {
          var exdate=new Date();
          exdate.setDate(exdate.getDate()+expiredays);
          document.cookie=c_name+ "=" +escape(value)+
                       ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
       }
       function checkCookie() {
          username=getCookie('username');
          if (username!=null && username!="") {
             alert('Welcome again '+username+'!');
          } else {
             username=prompt('Please enter your name:',"");
             if (username!=null && username!="") {
                setCookie('username',username,365);
             }
          }
       }
    </script>
 </head>
 <body onLoad="checkCookie()">
 </body>
</html>
```

The example above runs the checkCookie() function when the page loads.

JavaScript Form Validation

JavaScript can be used to validate input data in HTML forms before sending off the content to a server.

JavaScript Form Validation

JavaScript can be used to validate input data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Required Fields

The function below checks if a required field has been left empty. If the required field is blank, an alert box alerts a message and the function returns false. If a value is entered, the function returns true (means that data is OK):

```
function validate_required(field,alerttxt) {
   with (field) {
      if (value==null||value=="") {
         alert(alerttxt);return false;
      } else {
         return true;
      }
   }
```

```
}
```

The entire script, with the HTML form could look something like this:

```
<html>
   <head>
      <script type="text/javascript">
         function validate_required(field,alerttxt) {
            with (field) {
               if (value==null||value=="") {
                  alert(alerttxt);return false;
               } else {
                  return true
               }
            }
         }
         function validate_form(thisform) {
            with (thisform) {
               if (validate_required(email,"Email must be filled out!")==false) {
                  email.focus();
                  return false;
               }
            }
         }
      </script>
   </head>
   <body>
      <form action="submitpage.htm" onsubmit="return validate_form(this)" method="post">
         Email: <input type="text" name="email" size="30">
         <input type="submit" value="Submit">
      </form>
   </body>
</html>
```

E-mail Validation

The function below checks if the content has the general syntax of an email.

This means that the input data must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

```
function validate_email(field,alerttxt) {
   with (field) {
      apos=value.indexOf("@");
      dotpos=value.lastIndexOf(".");
      if (apos<1||dotpos-apos<2) {
         alert(alerttxt);return false;
      } else {
         return true;
      }
   }
}
```

The entire script, with the HTML form could look something like this:

```
<html>
   <head>
      <script type="text/javascript">
         function validate_email(field,alerttxt) {
            with (field) {
               apos=value.indexOf("@");
               dotpos=value.lastIndexOf(".");
               if (apos<1||dotpos-apos<2) {
                  alert(alerttxt);return false;
               } else {
```

```
                return true;
             }
          }
       }
       function validate_form(thisform) {
          with (thisform) {
             if (validate_email(email,"Not a valid e-mail address!")==false) {
                email.focus();return false;
             }
          }
       }
    </script>
  </head>
  <body>
    <form action="submitpage.htm" onsubmit="return validate_form(this);" method="post">
       Email: <input type="text" name="email" size="30">
       <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

JavaScript Animation

With JavaScript we can create animated images.

Examples

Button animation

```
<html>
	<head>
		<script type="text/javascript">
			function mouseOver() {
				document.b1.src ="b_blue.gif";
			}
			function mouseOut() {
				document.b1.src ="b_pink.gif";
			}
		</script>
	</head>
	<body>
		<a href="http://www.w3schools.com" target="_blank">
		<img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1" width="26" height="26"
		onmouseover="mouseOver()" onmouseout="mouseOut()" /></a>
	</body>
</html>
```

JavaScript Animation

It is possible to use JavaScript to create animated images.

The trick is to let a JavaScript change between different images on different events.

In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between the images.

The HTML Code

The HTML code looks like this:

```
<a href="http://www.w3schools.com" target="_blank">
   <img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1"
        onmouseOver="mouseOver()" onmouseOut="mouseOut()" />
</a>
```

Note that we have given the image a name to make it possible for JavaScript to address it later.

The onMouseOver event tells the browser that once a mouse is rolled over the image, the browser should execute a function that will replace the image with another image.

The onMouseOut event tells the browser that once a mouse is rolled away from the image, another JavaScript function should be executed. This function will insert the original image again.

The JavaScript Code

The changing between the images is done with the following JavaScript:

```
<script type="text/javascript">
   function mouseOver() {
      document.b1.src ="b_blue.gif";
   }
   function mouseOut() {
      document.b1.src ="b_pink.gif";
   }
</script>
```

The function mouseOver() causes the image to shift to "b_blue.gif".

The function mouseOut() causes the image to shift to "b_pink.gif".

The Entire Code

```
<html>
   <head>
      <script type="text/javascript">
         function mouseOver() {
            document.b1.src ="b_blue.gif";
         }
         function mouseOut() {
            document.b1.src ="b_pink.gif";
         }
      </script>
   </head>
   <body>
      <a href="http://www.w3schools.com" target="_blank">
         <img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1"
              onmouseOver="mouseOver()" onmouseOut="mouseOut()" />
      </a>
   </body>
</html>
```

JavaScript Image Maps

An image-map is an image with clickable regions.

Examples

Simple HTML Image map

```
<html>
	<body>
		<img src ="planets.gif" width ="145" height ="126" alt="Planets" usemap="#planetmap" />
		<map id ="planetmap" name="planetmap">
			<area shape ="rect" coords ="0,0,82,126" href ="sun.htm" target ="_blank" alt="Sun" />
			<area shape ="circle" coords ="90,58,3" href ="mercur.htm" target ="_blank"
			alt="Mercury" />
			<area shape ="circle" coords ="124,58,8" href ="venus.htm" target ="_blank" alt="Venus" />
		</map>
	</body>
</html>
```

Image map with added JavaScript

```
<html>
	<body>
		<img src ="planets.gif" width ="145" height ="126" alt="Planets" usemap="#planetmap" />

		<map id ="planetmap" name="planetmap">
			<area shape ="rect" coords ="0,0,82,126" href ="sun.htm" target ="_blank" alt="Sun" />
			<area shape ="circle" coords ="90,58,3" href ="mercur.htm" target ="_blank"
			alt="Mercury" />
			<area shape ="circle" coords ="124,58,8" href ="venus.htm" target ="_blank" alt="Venus" />
		</map>
	</body>
</html>
```

HTML Image Maps

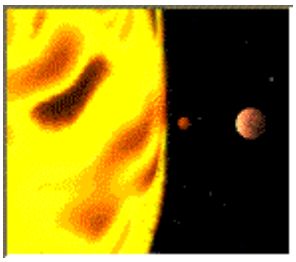
From our HTML tutorial we have learned that an image-map is an image with clickable regions. Normally, each region has an associated hyperlink. Clicking on one of the regions takes you to the associated link.

Example

The example below demonstrates how to create an HTML image map, with clickable regions. Each of the regions is a hyperlink:

```
<img src ="planets.gif" width ="145" height ="126" alt="Planets" usemap ="#planetmap" />
<map id ="planetmap" name="planetmap">
  <area shape="rect" coords ="0,0,82,126" href ="sun.htm"  target="_blank" alt="Sun" />
  <area shape="circle" coords="90,58,3" href ="mercur.htm" target="_blank" alt="Mercury" />
  <area shape="circle" coords="124,58,8" href ="venus.htm" target ="_blank" alt="Venus" />
</map>
```

Result



Adding some JavaScript

We can add events (that can call a JavaScript) to the <area> tags inside the image map. The <area> tag supports the onClick, onDblClick, onMouseDown, onMouseUp, onMouseOver, onMouseMove, onMouseOut, onKeyPress, onKeyDown, onKeyUp, onFocus, and onBlur events.

Here's the above example, with some JavaScript added:

```
<html>
```

```
   <head>
      <script type="text/javascript">
         function writeText(txt) {
            document.getElementById("desc").innerHTML=txt;
         }
      </script>
   </head>
   <body>
      <img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap" />
      <map id ="planetmap" name="planetmap">
         <area shape ="rect" coords ="0,0,82,126" onMouseOver="writeText('The Sun and the
               gas giant planets like Jupiter are by far the largest objects in our Solar
               System.')" href ="sun.htm" target ="_blank" alt="Sun" />
         <area shape ="circle" coords ="90,58,3" onMouseOver="writeText('The planet Mercury
               is very difficult to study from the Earth because it is always so close to
               the Sun.')" href ="mercur.htm" target ="_blank" alt="Mercury" />
         <area shape ="circle" coords ="124,58,8" onMouseOver="writeText('Until the 1960s,
               Venus was often considered a twin sister to the Earth because Venus is the
               nearest planet to us, and because the two planets seem to share many
               characteristics.')" href ="venus.htm" target ="_blank" alt="Venus" />
      </map>
      <p id="desc"></p>
   </body>
</html>
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called timing events.

Examples

Simple timing

```
<html>
	<head>
		<script type="text/javascript">
			function timedMsg() {
				var t=setTimeout("alert('5 seconds!')",5000);
			}
		</script>
	</head>

	<body>
		<form>
			<input type="button" value="Display timed alertbox!" onClick = "timedMsg()">
		</form>
		<p>Click on the button above. An alert box will be displayed after 5 seconds.</p>
	</body>
</html>
```

Another simple timing

```
<html>
	<head>
		<script type="text/javascript">
			function timedText() {
				var t1=setTimeout("document.getElementById('txt').value='2 seconds!'",2000);
				var t2=setTimeout("document.getElementById('txt').value='4 seconds!'",4000);
				var t3=setTimeout("document.getElementById('txt').value='6 seconds!'",6000);
			}
		</script>
	</head>
```

```
	<body>
		<form>
			<input type="button" value="Display timed text!" onClick="timedText()">
			<input type="text" id="txt">
		</form>
		<p>Click on the button above. The input field will tell you when two, four, and six seconds have
		passed.</p>
	</body>
</html>
```

Timing event in an infinite loop

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}
		</script>
	</head>

	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
		</form>
		<p>Click on the button above. The input field will count for ever, starting at 0.</p>
	</body>
</html>
```

Timing event in an infinite loop - with a Stop button

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}

			function stopCount() {
				clearTimeout(t);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
			<input type="button" value="Stop count!" onClick="stopCount()">
		</form>
		<p>Click on the "Start count!" button above to start the timer. The input field will count forever,
		starting at 0. Click on the "Stop count!" button to stop the counting.</p>
	</body>
</html>
```

A clock created with a timing event

```
<html>
	<head>
```

```
		<script type="text/javascript">
			function startTime() {
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				// add a zero in front of numbers<10
				m=checkTime(m);
				s=checkTime(s);
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}
			function checkTime(i) {
				if (i<10) {
					i="0" + i;
				}
				return i;
			}
		</script>
	</head>
	<body onload="startTime()">
		<div id="txt"></div>
	</body>
</html>
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called timing events.

It's very easy to time events in JavaScript. The two key methods that are used are:

- setTimeout() - executes a code some time in the future
- clearTimeout() - cancels the setTimeout()

Note: The setTimeout() and clearTimeout() are both methods of the HTML DOM Window object.

setTimeout()

Syntax

```
var t=setTimeout("javascript statement",milliseconds);
```

The setTimeout() method returns a value - In the statement above, the value is stored in a variable called t. If you want to cancel this setTimeout(), you can refer to it using the variable name.

The first parameter of setTimeout() is a string that contains a JavaScript statement. This statement could be a statement like "alert('5 seconds!')" or a call to a function, like "alertMsg()".

The second parameter indicates how many milliseconds from now you want to execute the first parameter.

Note: There are 1000 milliseconds in one second.

Example

When the button is clicked in the example below, an alert box will be displayed after 5 seconds.

```
<html>
   <head>
      <script type="text/javascript">
         function timedMsg() {
```

```
            var t=setTimeout("alert('5 seconds!')",5000);
         }
      </script>
   </head>
   <body>
      <form>
         <input type="button" value="Display timed alertbox!" onClick="timedMsg()">
      </form>
   </body>
</html>
```

Example - Infinite Loop

To get a timer to work in an infinite loop, we must write a function that calls itself. In the example below, when the button is clicked, the input field will start to count (for ever), starting at 0:

```
<html>
   <head>
      <script type="text/javascript">
         var c=0
         var t
         function timedCount() {
            document.getElementById('txt').value=c;
            c=c+1;
            t=setTimeout("timedCount()",1000);
         }
      </script>
   </head>
   <body>
      <form>
         <input type="button" value="Start count!" onClick="timedCount()">
         <input type="text" id="txt">
      </form>
   </body>
</html>
```

clearTimeout()

Syntax

```
clearTimeout(setTimeout_variable)
```

Example

The example below is the same as the "Infinite Loop" example above. The only difference is that we have now added a "Stop Count!" button that stops the timer:

```
<html>
   <head>
      <script type="text/javascript">
         var c=0
         var t
         function timedCount() {
            document.getElementById('txt').value=c;
            c=c+1;
            t=setTimeout("timedCount()",1000);
         }
         function stopCount() {
            clearTimeout(t);
         }
      </script>
   </head>
   <body>
      <form>
         <input type="button" value="Start count!" onClick="timedCount()">
```

```
        <input type="text" id="txt">
        <input type="button" value="Stop count!" onClick="stopCount()">
      </form>
   </body>
</html>
```

JavaScript Create Your Own Objects

Objects are useful to organize information.

Examples

Create a direct instance of an object

```
<html>
	<body>
		<script type="text/javascript">
			personObj=new Object();
			personObj.firstname="John";
			personObj.lastname="Doe";
			personObj.age=50;
			personObj.eyecolor="blue";
			document.write(personObj.firstname + " is " + personObj.age + " years old.");
		</script>
	</body>
</html>
```

Create a template for an object

```
<html>
	<body>
		<script type="text/javascript">
			function person(firstname,lastname,age,eyecolor) {
				this.firstname=firstname;
				this.lastname=lastname;
				this.age=age;
				this.eyecolor=eyecolor;
			}
			myFather=new person("John","Doe",50,"blue");
			document.write(myFather.firstname + " is " + myFather.age + " years old.");
		</script>
	</body>
</html>
```

JavaScript Objects

Earlier in this tutorial we have seen that JavaScript has several built-in objects, like String, Date, Array, and more. In addition to these built-in objects, you can also create your own.

An object is just a special kind of data, with a collection of properties and methods.

Let's illustrate with an example: A person is an object. Properties are the values associated with the object. The persons' properties include name, height, weight, age, skin tone, eye color, etc. All persons have these properties, but the values of those properties will differ from person to person. Objects also have methods. Methods are the actions that can be performed on objects. The persons' methods could be eat(), sleep(), work(), play(), etc.

Properties

The syntax for accessing a property of an object is:

```
objName.propName
```

You can add properties to an object by simply giving it a value. Assume that the personObj already exists - you can give it properties named firstname, lastname, age, and eyecolor as follows:

```
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=30;
personObj.eyecolor="blue";
document.write(personObj.firstname);
```

The code above will generate the following output:

```
John
```

Methods

An object can also contain methods.

You can call a method with the following syntax:

```
objName.methodName()
```

Note: Parameters required for the method can be passed between the parentheses.

To call a method called sleep() for the personObj:

```
personObj.sleep();
```

Creating Your Own Objects

There are different ways to create a new object:

1. Create a direct instance of an object

The following code creates an instance of an object and adds four properties to it:

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

Adding a method to the personObj is also simple. The following code adds a method called eat() to the personObj:

```
personObj.eat=eat;
```

2. Create a template of an object

The template defines the structure of an object:

```
function person(firstname,lastname,age,eyecolor) {
   this.firstname=firstname;
   this.lastname=lastname;
   this.age=age;
   this.eyecolor=eyecolor;
}
```

Notice that the template is just a function. Inside the function you need to assign things to this.propertyName. The reason for all the "this" stuff is that you're going to have more than one person at a time (which person you're dealing with must be clear). That's what "this" is: the instance of the object at hand.

Once you have the template, you can create new instances of the object, like this:

```
myFather=new person("John","Doe",50,"blue");
myMother=new person("Sally","Rally",48,"green");
```

You can also add some methods to the person object. This is also done inside the template:

```
function person(firstname,lastname,age,eyecolor) {
   this.firstname=firstname;
   this.lastname=lastname;
   this.age=age;
   this.eyecolor=eyecolor;
   this.newlastname=newlastname;
}
```

Note that methods are just functions attached to objects. Then we will have to write the newlastname() function:

```
function newlastname(new_lastname) {
   this.lastname=new_lastname;
}
```

The newlastname() function defines the person's new last name and assigns that to the person. JavaScript knows which person you're talking about by using "this.". So, now you can write: myMother.newlastname("Doe").

You Have Learned JavaScript, Now What?

JavaScript Summary

This tutorial has taught you how to add JavaScript to your HTML pages, to make your web site more dynamic and interactive.

You have learned how to create responses to events, validate forms and how to make different scripts run in response to different scenarios.

You have also learned how to create and use objects, and how to use JavaScript's built-in objects.

For more information on JavaScript, please look at our JavaScript examples and our JavaScript reference.

Now You Know JavaScript, What's Next?

The next step is to learn about the HTML DOM and DHTML.

If you want to learn about server-side scripting, the next step is to learn ASP.

HTML DOM

The HTML DOM defines a standard way for accessing and manipulating HTML documents.

The HTML DOM is platform and language independent and can be used by any programming language like Java, JavaScript, and VBScript.

If you want to learn more about the DOM, please visit our HTML DOM tutorial.

DHTML

DHTML is a combination of HTML, CSS, and JavaScript. DHTML is used to create dynamic and interactive Web sites.

W3C once said: "Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

If you want to learn more about DHTML, please visit our DHTML tutorial.

ASP

While scripts in an HTML file are executed on the client (in the browser), scripts in an ASP file are executed on the server.

With ASP you can dynamically edit, change or add any content of a Web page, respond to data submitted from HTML forms, access any data or databases and return the results to a browser, customize a Web page to make it more useful for individual users.

Since ASP files are returned as plain HTML, they can be viewed in any browser.

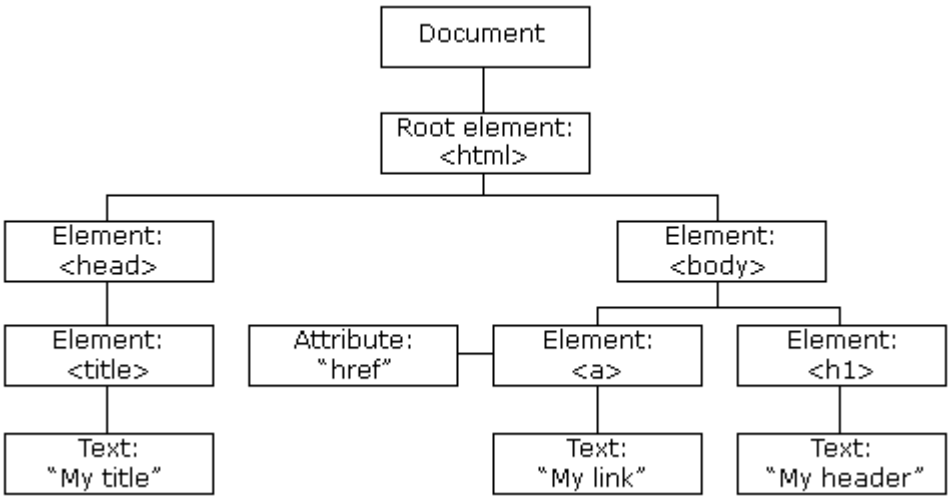
If you want to learn more about ASP, please visit our ASP tutorial.

HTML DOM Tutorial

HTML DOM Tutorial

The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents.

The DOM presents an HTML document as a tree-structure (a node tree), with elements, attributes, and text.



HTML DOM Examples

Learn by 100 examples! With our editor, you can edit the HTML, and click on a test button to view the result.

Try-It-Yourself!

HTML DOM Reference

At W3Schools you will find a complete HTML DOM reference with lots of examples.

HTML DOM Introduction

The HTML DOM defines a standard for accessing and manipulating HTML documents.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML
- JavaScript

If you want to study these subjects first, find the tutorials on our http://www.w3schools.com/default.asp.

What is the DOM?

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

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

"The 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 DOM is separated into 3 different parts / levels:

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

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

What is the XML DOM?

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

If you want to study the XML DOM, find the XML DOM tutorial on http://www.w3schools.com/default.asp.

What is the HTML DOM?

The HTML DOM is:

- A standard object model for HTML
- A standard programming interface for HTML
- Platform- and language-independent
- A W3C standard

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

In other words:

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

HTML DOM Nodes

In the DOM, everything in an HTML document is a node.

Nodes

According to the DOM, everything in an HTML document is a node.

The DOM says:

- The entire document is a document node
- Every HTML tag is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

DOM Example

Look at the following HTML document:

```
<html>
   <head>
      <title>DOM Tutorial</title>
  </head>
  <body>
      <h1>DOM Lesson one</h1>
      <p>Hello world!</p>
  </body>
</html>
```

The root node in the HTML above is <html>. All other nodes in the document are contained within <html>.

The <html> node has two child nodes; <head> and <body>.

The <head> node holds a <title> node. The <body> node holds a <h1> and <p> node.

Text is Always Stored in Text Nodes

A common error in DOM processing is to expect an element node to contain text.

However, the text of an element node is stored in a text node.

In this example: **<title>DOM Tutorial</title>**, the element node <title>, holds a text node with the value "DOM Tutorial".

"DOM Tutorial" is **not** the value of the <title> element!

However, in the HTML DOM the value of the text node can be accessed by the **innerHTML** property.

You can read more about the innerHTML property in a later chapter.

HTML DOM Node Tree

The HTML DOM views a HTML document as a node-tree.

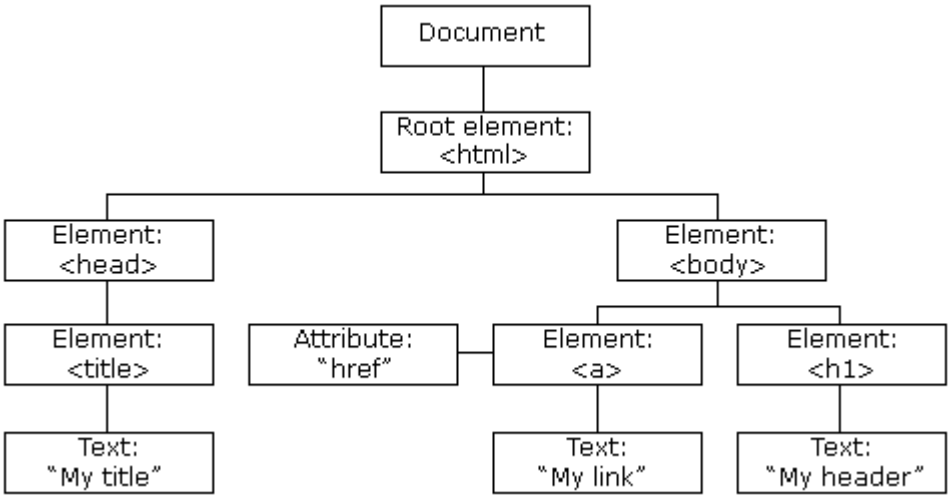
All the nodes in a node tree have relationships to each other.

The HTML DOM Node Tree (Document Tree)

The HTML DOM views a HTML document as a tree-structure. The tree structure is called a **node-tree.**

All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created.

The node tree below shows the set of nodes, and the connections between them. The tree starts at the root node and branches out to the text nodes at the lowest level of the tree:



Node Parents, Children, and Siblings

The nodes in the node tree have a hierarchical relationship to each other.

The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).

- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

Look at the following HTML fragment:

```
<html>
   <head>
      <title>DOM Tutorial</title>
   </head>
   <body>
      <h1>DOM Lesson one</h1>
      <p>Hello world!</p>
   </body>
</html>
```

In the HTML above, every node except for the document node has a parent node:

- The <html> node has no parent node; the root node
- The parent node of the <head> and <body> nodes is the <html> node
- The parent node of the "Hello world!" text node is the <p> node

Most element nodes have child nodes:

- The <html> node has two child nodes; <head> and <body>
- The <head> node has one child node; the <title> node
- The <title> node also has one child node; the text node "DOM Tutorial"
- The <h1> and <p> nodes are siblings, and both child nodes of <body>

HTML DOM Properties and Methods

Properties and methods define the programming interface to the HTML DOM.

Programming Interface

The DOM models HTML as a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we use JavaScript.

The programming interface to the DOM is defined by a set standard properties and methods.

Properties are often referred to as something that is (i.e. nodename is "p").

Methods are often referred to as something that is done (i.e. delete "p").

HTML DOM Properties

These are some typical DOM properties:

- x.innerHTML - the inner text value of x (a HTML element)
- x.nodeName - the name of x
- x.nodeValue - the value of x
- x.parentNode - the parent node of x
- x.childNodes - the child nodes of x
- x.attributes - the attributes nodes of x

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

HTML DOM Methods

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

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

innerHTML

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

innerHTML is not a part of the W3C DOM specification. However, it is supported by all major browsers.

The innerHTML property is useful for returning or replacing the content of HTML elements (including <html> and <body>), it can also be used to view the source of a page that has been dynamically modified.

Example

The JavaScript code to get the text from a <p> element with the id "intro" in a HTML document:

txt=document.getElementById("intro").innerHTML

After the execution of the statement, txt will hold the value "Hello World!"

Explained:

- **document** - the current HTML document
- **getElementById("intro")** - the <p> element with the id "intro"
- **innerHTML** - the inner text of the HTML element

In the example above, getElementById is a method, while innerHMTL is a property.

```
<html>
	<body>
		<p id="intro">Hello World!</p>
		<p id="main">This is an example for the HTML DOM tutorial.</p>
		<script type="text/javascript">
			txt=document.getElementById("intro").innerHTML;
			document.write("The text from the intro paragraph: " + txt);
		</script>
	</body>
</html>
```

Result:

Hello World!

This is an example for the HTML DOM tutorial.

The text from the intro paragraph: Hello World!

childNodes and nodeValue

The W3C DOM specified way of getting to the content of an element works like this:

The JavaScript code to get the text from a <p> element with the id "intro" in a HTML document:

txt=document.getElementById("intro").childNodes[0].nodeValue

After the execution of the statement, txt will hold the value "Hello World!"

Explained:

- **document** - the current HTML document
- **getElementById("intro")** - the <p> element with the id "intro"
- **childNodes[0]** - the first child of the <p> element (the text node)
- **nodeValue** - the value of the node (the text itself)

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

```
<html>
	<body>

		<p id="intro">Hello World!</p>
		<p id="main">This is an example for the HTML DOM tutorial.</p>

		<script type="text/javascript">
			txt=document.getElementById("intro").childNodes[0].nodeValue;
```

```
                document.write("The text from the intro paragraph: " + txt);
            </script>
        </body>
</html>
```

Result:

Hello World!

This is an example for the HTML DOM tutorial.

The text from the intro paragraph: Hello World!

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

HTML DOM Access Nodes

With the DOM, you can access every node in an HTML document.

Accessing Nodes

You can access a node in three ways:

1. By using the getElementById() method

2. By using the getElementsByTagName() method

3. By navigating the node tree, using the node relationships.

The getElementById() Method

The getElementById() method returns the element with the specified ID:

Syntax

```
document.getElementById("someID");
```

```
<html>
        <body>
                <p id="intro">W3Schools example</p>
                <div id="main">
                        <p id="main1">The DOM is very useful</p>
                        <p id="main2">This example demonstrates how to use the <b>getElementById</b>
                        method</p>
                </div>
                <script type="text/javascript">
                        x=document.getElementById("intro");
                        document.write("Intro paragraph text: " + x.innerHTML);
                </script>
        </body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **getElementById** method

Intro paragraph text: W3Schools example

Note: The getElementById() method doesn't work in XML.

The getElementsByTagName() Method

getElementsByTagName() returns all elements with a specified tag name.

Syntax

```
node.getElementsByTagName("tagname");
```

Example 1

The following example returns a nodeList of all <p> elements in the document:

```
document.getElementsByTagName("p");
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
			<p id="main1">The DOM is very useful</p>
			<p id="main2">This example demonstrates how to use the <b>getElementsByTagName</b>
			method</p>
		</div>
		<script type="text/javascript">
			x=document.getElementsByTagName("p");
			document.write("First paragraph text: " + x[0].innerHTML);
		</script>
	</body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **getElementsByTagName** method

First paragraph text: W3Schools example

Example 2

The following example returns a nodeList of all <p> elements that are descendants of the element with id="main":

```
document.getElementById('main').getElementsByTagName("p");
```

```
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
		<p id="main1">The DOM is very useful</p>
		<p id="main2">This example demonstrates how to use the <b>getElementsByTagName</b>
		method</p>
		</div>
		<script type="text/javascript">
```

```
                x=document.getElementById("main").getElementsByTagName("p");
                document.write("First paragraph inside the main div: " + x[0].innerHTML);
            </script>
        </body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **getElementsByTagName** method

First paragraph inside the main div: The DOM is very useful

DOM Node List

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

The following code stores a list of <p> nodes (a node list) in the variable x:

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

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

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

```
<html>
        <body>
                <p id="intro">W3Schools example</p>
                <div id="main">
                        <p id="main1">The DOM is very useful</p>
                        <p id="main2">This example demonstrates how to use the <b>getElementsByTagName and
                        node list</b></p>
                </div>
                <script type="text/javascript">
                        x=document.getElementsByTagName("p");
                        document.write("Second paragraph text: " + x[1].innerHTML);
                </script>
        </body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **getElementsByTagName and node list**

Second paragraph text: The DOM is very useful

Note: The index starts at 0.

You will learn more about node lists in a later chapter of this tutorial.

DOM Node List Length

The length property defines the length of a node list (the number of nodes).

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

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

Example explained:

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

```
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
			<p id="main1">The DOM is very useful</p>
			<p id="main2">This example demonstrates how to use the <b>nodeList length</b>
			property</p>
		</div>
		<script type="text/javascript">
			x=document.getElementsByTagName("p");
			for (i=0;i<x.length;i++) {
				document.write(x[i].innerHTML);
				document.write("<br />");
			}
		</script>
	</body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **nodeList length** property

W3Schools example
The DOM is very useful
This example demonstrates how to use the **nodeList length** property

Navigating Node Relationships

The three properties parentNode, firstChild, and lastChild follow the document structure and allow short-distance travel in the document.

Look at the following HTML fragment:

```
<html>
   <body>
      <p id="intro">W3Schools example</p>
      <div id="main">
         <p id="main1">The DOM is very useful</p>
         <p id="main2">This example demonstrates <b>node relationships</b></p>
      </div>
   </body>
</html>
```

In the HTML code above, the <p id="intro"> is the first child node (firstChild) of the <body> element, and the element is the last child node (lastChild) of the <body> element.

Furthermore, the <body> is the parent (parentNode) of the every <p> element.

The firstChild property can be used to access the text of an element:

```
var x=document.getElementById("intro");
var text=x.firstChild.nodeValue;
```

```
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
			<p id="main1">The DOM is very useful</p>
			<p id="main2">This example demonstrates how to use the <b>firstChild</b> property</p>
		</div>
		<script type="text/javascript">
			x=document.getElementById("intro");
			document.write("Text inside the first paragraph: " + x.firstChild.nodeValue);
		</script>
	</body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use the **firstChild** property

Text inside the first paragraph: W3Schools example

Root Nodes

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

- document.documentElement
- document.body

The first property returns the root node of the document and exists in all XML and HTML documents.

The second property is a special addition for HTML pages, and gives direct access to the <body> tag.

```
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
			<p id="main1">The DOM is very useful</p>
			<p id="main2">This example demonstrates how to use <b>document.body</b></p>
		</div>
		<script type="text/javascript">
			x=document.body;
			alert(x.innerHTML);
		</script>
	</body>
</html>
```

Result:

W3Schools example

The DOM is very useful

This example demonstrates how to use **document.body**

HTML DOM Node Information

The nodeName, nodeValue, and nodeType properties contain information about nodes.

Node Properties

In the HTML Document Object Model (DOM), each node is an **object**.

Objects have methods (functions) and properties (information about the object), that can be accessed and manipulated by JavaScript.

Three important HTML DOM node properties are:

- nodeName
- nodeValue
- nodeType

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.

```
<html>
	<head>
		<script type="text/javascript" src="loadxmldoc.js"></script>
	</head>
	<body>
		<script type="text/javascript">
			xmlDoc=loadXMLDoc("books.xml");
			document.write(xmlDoc.documentElement.nodeName);
		</script>
	</body>
</html>
```

Result:

bookstore

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

Example1: Get the Value of an Element

The following code fragment retrieves the text node value of the first <p> element:

```
x=document.getElementById("intro").firstChild;
txt=x.nodeValue;
```

Result: txt = "W3Schools example"

Example explained:

1. Get text node of the first <p> element node
2. Set the txt variable to be the value of the text node

```
<html>
	<body>
		<p id="intro">W3Schools example</p>
		<div id="main">
			<p id="main1">The DOM is very useful</p>
			<p id="main2">This example demonstrates how to use the <b>nodeValue</b> property</p>
		</div>
		<script type="text/javascript">
			x=document.getElementById("intro").firstChild;
			document.write("First paragraph text: " + x.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

HTML DOM - How to Change HTML Elements

HTML elements are changed using JavaScript, the HTML DOM and events.

Changing an HTML Element

The HTML DOM and JavaScript can be used to change the inner content and attributes of HTML elements dynamically.

Example 1 - Change the Background Color

The following example changes the background color of the <body> element:

```
<html>
   <body>
      <script type="text/javascript">
         document.body.bgColor="yellow";
      </script>
   </body>
</html>
```

```
<html>
	<body>
		<script type="text/javascript">
			document.body.bgColor="yellow";
		</script>
		<p>The background color was changed by the script.</p>
	</body>
</html>
```

Changing the Text HTML Element - innerHTML

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

Example 2 - Change the Text of an Element

The following example changes the text of the <p> element:

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

```
<html>
	<body>
		<p id="p1">Hello World!</p>
		<script type="text/javascript">
			document.getElementById("p1").innerHTML="New text!";
		</script>
		<p>The paragraph above was changed by the script.</p>
	</body>
</html>
```

Changing an HTML Element Using Events

An event handler allows you to execute code when an event occurs.

Events are generated by the browser when the user clicks an element, when the page loads, when a form is submitted, etc.

You can read more about events in the next chapter.

Example 3 - Change the Background Color Using the onclick Event

The following example changes the background color of the <body> element when the button is clicked:

```
<html>
   <body>
      <input type="button" onclick="document.body.bgColor='yellow';"
             value="Click me to change background color">
```

```
   </body>
</html>
```

Example 4 - Create a Function to Change the Text of an Element

The following example uses a function to change the text of the <p> element when the button is clicked:

```
<html>
   <head>
      <script type="text/javascript">
         function ChangeText() {
            document.getElementById("p1").innerHTML="New text!";
         }
      </script>
   </head>
   <body>
      <p id="p1">Hello world!</p>
      <input type="button" onclick="ChangeText()" value="Click me to change text above">
   </body>
</html>
```

Using the Style Object

The Style object represents of each HTML element represents its individual style.

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

Example 5 - Change the Background Color

The following example uses a function to change the style of the <body> element when the button is clicked:

```
<html>
    <head>
        <script type="text/javascript">
            function ChangeBackground() {
                document.body.style.backgroundColor="yellow";
            }
        </script>
    </head>
    <body>
        <p id="p1">Hello world!</p>
        <input type="button" onclick="ChangeBackground()" value="Click me to change
        background color">
    </body>
</html>
```

Example 6 - Change the Text font and color of an Element

The following example uses a function to change the style of the <p> element when the button is clicked:

```
<html>
   <head>
      <script type="text/javascript">
         function ChangeText() {
            document.getElementById("p1").style.color="blue";
            document.getElementById("p1").style.fontFamily="Arial";
         }
      </script>
   </head>
```

```
   <body>
      <p id="p1">Hello world!</p>
      <input type="button" onclick="ChangeText()" value="Click me to change text above">
   </body>
</html>
```

HTML DOM - Events

Events are actions that can be detected by JavaScript.

Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

For a complete reference of the events supported by the HTML DOM, see our HTML DOM Event reference.

onload and onUnload

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

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

Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver event');
                                                return false">
<img src="w3schools.gif" width="100" height="30">
</a>
```

JavaScript HTML DOM Objects

In addition to the built-in JavaScript objects, you can also access and manipulate all of the HTML DOM objects with JavaScript.

More JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events.

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

The HTML DOM

The HTML DOM is a W3C standard and it is an abbreviation for the Document Object Model for HTML.

The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents.

All HTML elements, along with their containing text and attributes, can be accessed through the DOM. The contents can be modified or deleted, and new elements can be created.

The HTML DOM is platform and language independent. It can be used by any programming language like Java, JavaScript, and VBScript.

Follow the links below to learn more about how to access and manipulate each DOM object with JavaScript:

Object	Description
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element

Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form
Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element
Meta	Represents a <meta> element
Option	Represents an <option> element
Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

You Have Learned HTML DOM, Now What?

HTML DOM Summary

This tutorial has taught you how to use the HTML DOM to make your web site more dynamic and interactive.

You have learned how manipulate HTML elements in response of different scenarios.

For more information on HTML DOM, please look at our HTML DOM examples and our HTML DOM reference in the annexure given behind.

Now You Know HTML DOM, What's Next?

The next step is to learn about ASP.

While scripts in an HTML file are executed on the client (in the browser), scripts in an ASP file are executed on the server.

With ASP you can dynamically edit, change or add any content of a Web page, respond to data submitted from HTML forms, access any data or databases and return the results to a browser, customize a Web page to make it more useful for individual users.

Since ASP files are returned as plain HTML, they can be viewed in any browser.

ANNEXURE

The String Object

The String object let's you work with text.

Syntax for creating a String object:

```
var myStr=new String(string);
```

String Object Properties

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Property	Description	FF	N	IE
constructor	A reference to the function that created the object	1	4	4
length	Returns the number of characters in a string	1	2	3
prototype	Allows you to add properties and methods to the object	1	2	4

String Object Methods

Method	Description	FF	N	IE
anchor()	Creates an HTML anchor	1	2	3
big()	Displays a string in a big font	1	2	3
blink()	Displays a blinking string	1	2	
bold()	Displays a string in bold	1	2	3
charAt()	Returns the character at a specified position	1	2	3
charCodeAt()	Returns the Unicode of the character at a specified position	1	4	4
concat()	Joins two or more strings	1	4	4
fixed()	Displays a string as teletype text	1	2	3
fontcolor()	Displays a string in a specified color	1	2	3
fontsize()	Displays a string in a specified size	1	2	3
fromCharCode()	Takes the specified Unicode values and returns a string	1	4	4
indexOf()	Returns the position of the first occurrence of a specified string value in a string	1	2	3
italics()	Displays a string in italic	1	2	3
lastIndexOf()	Returns the position of the last occurrence of a specified string value, searching backwards from the specified position in a string	1	2	3
link()	Displays a string as a hyperlink	1	2	3
match()	Searches for a specified value in a string	1	4	4
replace()	Replaces some characters with some other characters in a string	1	4	4
search()	Searches a string for a specified value	1	4	4
slice()	Extracts a part of a string and returns the extracted part in a new string	1	4	4
small()	Displays a string in a small font	1	2	3
split()	Splits a string into an array of strings	1	4	4
strike()	Displays a string with a strikethrough	1	2	3
sub()	Displays a string as subscript	1	2	3
substr()	Extracts a specified number of characters in a string, from a start index	1	4	4
substring()	Extracts the characters in a string between two specified indices	1	2	3
sup()	Displays a string as superscript	1	2	3
toLowerCase()	Displays a string in lowercase letters	1	2	3
toUpperCase()	Displays a string in uppercase letters	1	2	3
toSource()	Represents the source code of an object	1	4	-
valueOf()	Returns the primitive value of a String object	1	2	4

Milan Das, SARC

The Date Object

The Date object is used to work with dates and times.

Syntax for creating a Date object:

```
var myDate=new Date()
```

Note: The Date object will automatically hold the current date and time as its initial value!

Date Object Properties

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Property	Description	FF	N	IE
constructor	Returns a reference to the Date function that created the object	1	4	4
prototype	Allows you to add properties and methods to the object	1	3	4

Date Object Methods

Method	Description	FF	N	IE
Date()	Returns today's date and time	1	2	3
getDate()	Returns the day of the month from a Date object (from 1-31)	1	2	3
getDay()	Returns the day of the week from a Date object (from 0-6)	1	2	3
getFullYear()	Returns the year, as a four-digit number, from a Date object	1	4	4
getHours()	Returns the hour of a Date object (from 0-23)	1	2	3
getMilliseconds()	Returns the milliseconds of a Date object (from 0-999)	1	4	4
getMinutes()	Returns the minutes of a Date object (from 0-59)	1	2	3
getMonth()	Returns the month from a Date object (from 0-11)	1	2	3
getSeconds()	Returns the seconds of a Date object (from 0-59)	1	2	3
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970	1	2	3
getTimezoneOffset()	Returns the difference in minutes between local time and Greenwich Mean Time (GMT)	1	2	3
getUTCDate()	Returns the day of the month from a Date object according to universal time (from 1-31)	1	4	4
getUTCDay()	Returns the day of the week from a Date object according to universal time (from 0-6)	1	4	4
getUTCMonth()	Returns the month from a Date object according to universal time (from 0-11)	1	4	4
getUTCFullYear()	Returns the four-digit year from a Date object according to universal time	1	4	4
getUTCHours()	Returns the hour of a Date object according to universal time (from 0-23)	1	4	4
getUTCMinutes()	Returns the minutes of a Date object according to universal time (from 0-59)	1	4	4
getUTCSeconds()	Returns the seconds of a Date object according to universal time (from 0-59)	1	4	4
getUTCMilliseconds()	Returns the milliseconds of a Date object according to universal time (from 0-999)	1	4	4
getYear()	Returns the year, as a two-digit or a three/four-digit number, depending on the browser. Use getFullYear() instead !!	1	2	3
parse()	Takes a date string and returns the number of milliseconds since midnight of January 1, 1970	1	2	3
setDate()	Sets the day of the month in a Date object (from 1-31)	1	2	3
setFullYear()	Sets the year in a Date object (four digits)	1	4	4
setHours()	Sets the hour in a Date object (from 0-23)	1	2	3
setMilliseconds()	Sets the milliseconds in a Date object (from 0-999)	1	4	4
setMinutes()	Set the minutes in a Date object (from 0-59)	1	2	3
setMonth()	Sets the month in a Date object (from 0-11)	1	2	3
setSeconds()	Sets the seconds in a Date object (from 0-59)	1	2	3
setTime()	Calculates a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970	1	2	3
setUTCDate()	Sets the day of the month in a Date object according to universal time (from 1-31)	1	4	4
setUTCMonth()	Sets the month in a Date object according to universal time (from 0-11)	1	4	4
setUTCFullYear()	Sets the year in a Date object according to universal time (four digits)	1	4	4
setUTCHours()	Sets the hour in a Date object according to universal time (from 0-23)	1	4	4
setUTCMinutes()	Set the minutes in a Date object according to universal time (from 0-59)	1	4	4
setUTCSeconds()	Set the seconds in a Date object according to universal time (from 0-59)	1	4	4
setUTCMilliseconds()	Sets the milliseconds in a Date object according to universal time (from 0-	1	4	4

	999)			
setYear()	Sets the year in the Date object (two or four digits). Use setFullYear() instead !!	1	2	3
toDateString()	Returns the date portion of a Date object in readable form			
toGMTString()	Converts a Date object, according to Greenwich time, to a string. Use toUTCString() instead !!	1	2	3
toLocaleDateString()	Converts a Date object, according to local time, to a string and returns the date portion	1	4	4
toLocaleTimeString()	Converts a Date object, according to local time, to a string and returns the time portion	1	4	4
toLocaleString()	Converts a Date object, according to local time, to a string	1	2	3
toSource()	Represents the source code of an object	1	4	-
toString()	Converts a Date object to a string	1	2	4
toTimeString()	Returns the time portion of a Date object in readable form			
toUTCString()	Converts a Date object, according to universal time, to a string	1	4	4
UTC()	Takes a date and returns the number of milliseconds since midnight of January 1, 1970 according to universal time	1	2	3
valueOf()	Returns the primitive value of a Date object	1	2	4

Milan Das, SARC

The Array Object

The Array object is used to store multiple values in a single variable.

Syntax for creating an Array object:

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

To access and to set values inside an array, you must use the index numbers as follows:

- myCars[0] is the first element
- myCars[1] is the second element
- myCars[2] is the third element

Array Object Properties

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Property	Description	FF	N	IE
constructor	Returns a reference to the array function that created the object	1	2	4
Index		1	3	4
Input		1	3	4
length	Sets or returns the number of elements in an array	1	2	4
prototype	Allows you to add properties and methods to the object	1	2	4

Array Object Methods

Method	Description	FF	N	IE
concat()	Joins two or more arrays and returns the result	1	4	4
join()	Puts all the elements of an array into a string. The elements are separated by a specified delimiter	1	3	4
pop()	Removes and returns the last element of an array	1	4	5.5
push()	Adds one or more elements to the end of an array and returns the new length	1	4	5.5
reverse()	Reverses the order of the elements in an array	1	3	4
shift()	Removes and returns the first element of an array	1	4	5.5
slice()	Returns selected elements from an existing array	1	4	4
sort()	Sorts the elements of an array	1	3	4
splice()	Removes and adds new elements to an array	1	4	5.5
toSource()	Represents the source code of an object	1	4	-
toString()	Converts an array to a string and returns the result	1	3	4
unshift()	Adds one or more elements to the beginning of an array and returns the new length	1	4	6
valueOf()	Returns the primitive value of an Array object	1	2	4

Milan Das, SARC

The Boolean Object

The Boolean object represents two values: "true" or "false".

Syntax for creating a Boolean object:

```
var myBool=new Boolean(value)
```

Note: If the value parameter is omitted, or is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is set to true (even with the string "false")!

Boolean Object Properties

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Property	Description	FF	N	IE
constructor	Returns a reference to the Boolean function that created the object	1	2	4
prototype	Allows you to add properties and methods to the object	1	2	4

Boolean Object Methods

Method	Description	FF	N	IE
toSource()	Returns the source code of the object	1	4	-
toString()	Converts a Boolean value to a string and returns the result	1	4	4
valueOf()	Returns the primitive value of a Boolean object	1	4	4

Milan Das, SARC

The Math Object

The Math object allows you to perform mathematical tasks.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Math Object Properties

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Property	Description	FF	N	IE
E	Returns Euler's constant (approx. 2.718)	1	2	3
LN2	Returns the natural logarithm of 2 (approx. 0.693)	1	2	3
LN10	Returns the natural logarithm of 10 (approx. 2.302)	1	2	3
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)	1	2	3
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)	1	2	3
PI	Returns PI (approx. 3.14159)	1	2	3
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)	1	2	3
SQRT2	Returns the square root of 2 (approx. 1.414)	1	2	3

Math Object Methods

Method	Description	FF	N	IE
abs(x)	Returns the absolute value of a number	1	2	3
acos(x)	Returns the arccosine of a number	1	2	3
asin(x)	Returns the arcsine of a number	1	2	3
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians	1	2	3
atan2(y,x)	Returns the angle theta of an (x,y) point as a numeric value between -PI and PI radians	1	2	3
ceil(x)	Returns the value of a number rounded upwards to the nearest integer	1	2	3
cos(x)	Returns the cosine of a number	1	2	3
exp(x)	Returns the value of E^x	1	2	3
floor(x)	Returns the value of a number rounded downwards to the nearest integer	1	2	3
log(x)	Returns the natural logarithm (base E) of a number	1	2	3
max(x,y)	Returns the number with the highest value of x and y	1	2	3
min(x,y)	Returns the number with the lowest value of x and y	1	2	3
pow(x,y)	Returns the value of x to the power of y	1	2	3
random()	Returns a random number between 0 and 1	1	2	3
round(x)	Rounds a number to the nearest integer	1	2	3
sin(x)	Returns the sine of a number	1	2	3
sqrt(x)	Returns the square root of a number	1	2	3
tan(x)	Returns the tangent of an angle	1	2	3
toSource()	Represents the source code of an object	1	4	-
valueOf()	Returns the primitive value of a Math object	1	2	4

Milan Das, SARC

The Number Object

The Number object is an object wrapper for primitive numeric values.

Syntax for creating a Number object:

```
var myNum=new Number(number);
```

Note: If the number parameter cannot be converted into a number, it returns NaN.

Number Object Properties

FF: Firefox, **IE**: Internet Explorer

Property	Description	FF	IE
constructor	Returns a reference to the Number function that created the object	1	4
MAX_VALUE	Returns the largest possible value in JavaScript	1	4
MIN_VALUE	Returns the smallest possible value in JavaScript	1	4
NaN	Represents "Not-a-number" value	1	4
NEGATIVE_INFINITY	Represents a value that is less than MIN_VALUE	1	4
POSITIVE_INFINITY	Represents a value that is greater than MAX_VALUE	1	4
Prototype	Allows you to add properties and methods to the object	1	4

Number Object Methods

Method	Description	FF	IE
toExponential()	Converts the value of the object into an exponential notation	1	5.5
toFixed()	Formats a number to the specified number of decimals	1	5.5
toLocaleString()			
toPrecision()	Converts a number into a number with a specified number of digits	1	5.5
toString()	Converts the Number object into a string	1	4
valueOf()	Returns the value of the Number object	1	4

Milan Das, SARC

Top-level Functions

The top-level properties and functions can be used on all of the built-in JavaScript objects.

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Function	Description	FF	N	IE
decodeURI()	Decodes an encoded URI	1	4	5.5
decodeURIComponent()	Decodes an encoded URI component	1	4	5.5
encodeURI()	Encodes a string as a URI	1	4	5.5
encodeURIComponent()	Encodes a string as a URI component	1	4	5.5
escape()	Encodes a string	1	-	3
eval()	Evaluates a string and executes it as if it was script code	1	2	3
isFinite()	Checks if a value is a finite number	1	4	4
isNaN()	Checks if a value is not a number	1	2	3
Number()	Converts an object's value to a number	1		
parseFloat()	Parses a string and returns a floating point number	1	2	3
parseInt()	Parses a string and returns an integer	1	2	3
String()	Converts an object's value to a string	1		
unescape()	Decodes a string encoded by escape()	1	-	3

Top-level Properties

Property	Description	FF	N	IE
Infinity	A numeric value that represents positive or negative infinity	1	4	4
NaN	Indicates that a value is "Not a Number"	1	4	4
undefined	Indicates that a variable has not been assigned a value	1	4	5.5

Milan Das, SARC

Event Handlers

Events are normally used in combination with functions, and the function will not be executed before the event occurs!

New to HTML 4.0 was the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element. Below is a list of the attributes that can be inserted into HTML tags to define event actions.

FF: Firefox, **N**: Netscape, **IE**: Internet Explorer

Attribute	The event occurs when...	FF	N	IE
onabort	Loading of an image is interrupted	1	3	4
onblur	An element loses focus	1	2	3
onchange	The user changes the content of a field	1	2	3
onclick	Mouse clicks an object	1	2	3
ondblclick	Mouse double-clicks an object	1	4	4
onerror	An error occurs when loading a document or an image	1	3	4
onfocus	An element gets focus	1	2	3
onkeydown	A keyboard key is pressed	1	4	3
onkeypress	A keyboard key is pressed or held down	1	4	3
onkeyup	A keyboard key is released	1	4	3
onload	A page or an image is finished loading	1	2	3
onmousedown	A mouse button is pressed	1	4	4
onmousemove	The mouse is moved	1	6	3
onmouseout	The mouse is moved off an element	1	4	4
onmouseover	The mouse is moved over an element	1	2	3
onmouseup	A mouse button is released	1	4	4
onreset	The reset button is clicked	1	3	4
onresize	A window or frame is resized	1	4	4
onselect	Text is selected	1	2	3
onsubmit	The submit button is clicked	1	2	3
onunload	The user exits the page	1	2	3

Milan Das, SARC

HTML DOM Reference

JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events. **Contain lots of examples!**

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

HTML DOM Objects

Follow the links to learn more about the objects and their collections, properties, methods and events. **Contain lots of examples!**

Object	Description
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element
Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form
Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element
Meta	Represents a <meta> element
Option	Represents an <option> element
Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

Milan Das, SARC

HTML DOM Window Object

Window Object

The Window object is the top level object in the JavaScript hierarchy.

The Window object represents a browser window.

A Window object is created automatically with every instance of a <body> or <frameset> tag.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Window Object Collections

Collection	Description	IE	F	O
frames[]	Returns all named frames in the window	4	1	9

Window Object Properties

Property	Description	IE	F	O
closed	Returns whether or not a window has been closed	4	1	9
defaultStatus	Sets or returns the default text in the statusbar of the window	4	No	9
document	See Document object	4	1	9
history	See History object	4	1	9
length	Sets or returns the number of frames in the window	4	1	9
location	See Location object	4	1	9
name	Sets or returns the name of the window	4	1	9
opener	Returns a reference to the window that created the window	4	1	9
outerHeight	Sets or returns the outer height of a window	No	1	No
outerWidth	Sets or returns the outer width of a window	No	1	No
pageXOffset	Sets or returns the X position of the current page in relation to the upper left	No	No	No
pageYOffset	Sets or returns the Y position of the current page in relation to the upper left	No	No	No
parent	Returns the parent window	4	1	9
personalbar	Sets whether or not the browser's personal bar (or directories bar) should be			
scrollbars	Sets whether or not the scrollbars should be visible			
self	Returns a reference to the current window	4	1	9
status	Sets the text in the statusbar of a window	4	No	9
statusbar	Sets whether or not the browser's statusbar should be visible			
toolbar	Sets whether or not the browser's tool bar is visible or not (can only be set			
top	Returns the topmost ancestor window	4	1	9

Window Object Methods

Method	Description	IE	F	O
alert()	Displays an alert box with a message and an OK button	4	1	9
blur()	Removes focus from the current window	4	1	9
clearInterval()	Cancels a timeout set with setInterval()	4	1	9
clearTimeout()	Cancels a timeout set with setTimeout()	4	1	9
close()	Closes the current window	4	1	9
confirm()	Displays a dialog box with a message and an OK and a Cancel button	4	1	9
createPopup()	Creates a pop-up window	4	No	No
focus()	Sets focus to the current window	4	1	9
moveBy()	Moves a window relative to its current position	4	1	9
moveTo()	Moves a window to the specified position	4	1	9

open()	Opens a new browser window	4	1	9
print()	Prints the contents of the current window	5	1	9
prompt()	Displays a dialog box that prompts the user for input	4	1	9
resizeBy()	Resizes a window by the specified pixels	4	1	9
resizeTo()	Resizes a window to the specified width and height	4	1.5	9
scrollBy()	Scrolls the content by the specified number of pixels	4	1	9
scrollTo()	Scrolls the content to the specified coordinates	4	1	9
setInterval()	Evaluates an expression at specified intervals	4	1	9
setTimeout()	Evaluates an expression after a specified number of milliseconds	4	1	9

HTML DOM Navigator Object

Navigator Object

The Navigator object is actually a JavaScript object, not an HTML DOM object.

The Navigator object is automatically created by the JavaScript runtime engine and contains information about the client browser.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Navigator Object Collections

Collection	Description	IE	F	O
plugins[]	Returns a reference to all embedded objects in the document	4	1	9

Navigator Object Properties

Property	Description	IE	F	O
appCodeName	Returns the code name of the browser	4	1	9
appMinorVersion	Returns the minor version of the browser	4	No	No
appName	Returns the name of the browser	4	1	9
appVersion	Returns the platform and version of the browser	4	1	9
browserLanguage	Returns the current browser language	4	No	9
cookieEnabled	Returns a Boolean value that specifies whether cookies are enabled in the	4	1	9
cpuClass	Returns the CPU class of the browser's system	4	No	No
onLine	Returns a Boolean value that specifies whether the system is in offline	4	No	No
platform	Returns the operating system platform	4	1	9
systemLanguage	Returns the default language used by the OS	4	No	No
userAgent	Returns the value of the user-agent header sent by the client to the server	4	1	9
userLanguage	Returns the OS' natural language setting	4	No	9

Navigator Object Methods

Method	Description	IE	F	O
javaEnabled()	Specifies whether or not the browser has Java enabled	4	1	9
taintEnabled()	Specifies whether or not the browser has data tainting enabled	4	1	9

HTML DOM Screen Object

Screen Object

The Screen object is actually a JavaScript object, not an HTML DOM object..

The Screen object is automatically created by the JavaScript runtime engine and contains information about the client's display screen.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Screen Object Properties

Property	Description	IE	F	O
availHeight	Returns the height of the display screen (excluding the Windows	4	1	9
availWidth	Returns the width of the display screen (excluding the Windows	4	1	9
bufferDepth	Sets or returns the bit depth of the color palette in the off-screen	4	No	No
colorDepth	Returns the bit depth of the color palette on the destination device or	4	1	9
deviceXDPI	Returns the number of horizontal dots per inch of the display screen	6	No	No
deviceYDPI	Returns the number of vertical dots per inch of the display screen	6	No	No
fontSmoothingEnabled	Returns whether the user has enabled font smoothing in the display	4	No	No
height	The height of the display screen	4	1	9
logicalXDPI	Returns the normal number of horizontal dots per inch of the display	6	No	No
logicalYDPI	Returns the normal number of vertical dots per inch of the display	6	No	No
pixelDepth	Returns the color resolution (in bits per pixel) of the display screen	No	1	9
updateInterval	Sets or returns the update interval for the screen	4	No	No
width	Returns width of the display screen	4	1	9

HTML DOM History Object

History Object

The History object is actually a JavaScript object, not an HTML DOM object.

The History object is automatically created by the JavaScript runtime engine and consists of an array of URLs. These URLs are the URLs the user has visited within a browser window.

The History object is part of the Window object and is accessed through the window.history property.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

History Object Properties

Property	Description	IE	F	O
length	Returns the number of elements in the history list	4	1	9

History Object Methods

Method	Description	IE	F	O
back()	Loads the previous URL in the history list	4	1	9
forward()	Loads the next URL in the history list	4	1	9

go()	Loads a specific page in the history list	4	1	9

HTML DOM Location Object

Location Object

The Location object is actually a JavaScript object, not an HTML DOM object.

The Location object is automatically created by the JavaScript runtime engine and contains information about the current URL. Example: Send a user to a new location

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

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Location Object Properties

Property	Description	IE	F	O
hash	Sets or returns the URL from the hash sign (#)	4	1	9
host	Sets or returns the hostname and port number of the current URL	4	1	9
hostname	Sets or returns the hostname of the current URL	4	1	9
href	Sets or returns the entire URL	4	1	9
pathname	Sets or returns the path of the current URL	4	1	9
port	Sets or returns the port number of the current URL	4	1	9
protocol	Sets or returns the protocol of the current URL	4	1	9
search	Sets or returns the URL from the question mark (?)	4	1	9

Location Object Methods

Method	Description	IE	F	O
assign()	Loads a new document	4	1	9
reload()	Reloads the current document	4	1	9
replace()	Replaces the current document with a new one	4	1	9

HTML DOM Document Object

Document Object

The Document object represents the entire HTML document and can be used to access all elements in a page.

The Document object is part of the Window object and is accessed through the window.document property.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Document Object Collections

Collection	Description	IE	F	O	W3C

anchors[]	Returns a reference to all Anchor objects in the document	4	1	9	Yes
forms[]	Returns a reference to all Form objects in the document	4	1	9	Yes
images[]	Returns a reference to all Image objects in the document	4	1	9	Yes
links[]	Returns a reference to all Area and Link objects in the document	4	1	9	Yes

Document Object Properties

Property	Description	IE	F	O	W3C
body	Gives direct access to the <body> element				
cookie	Sets or returns all cookies associated with the current document	4	1	9	Yes
domain	Returns the domain name for the current document	4	1	9	Yes
lastModified	Returns the date and time a document was last modified	4	1	No	No
referrer	Returns the URL of the document that loaded the current document	4	1	9	Yes
title	Returns the title of the current document	4	1	9	Yes
URL	Returns the URL of the current document	4	1	9	Yes

Document Object Methods

Method	Description	IE	F	O	W3C
close()	Closes an output stream opened with the document.open() method, and displays the collected data	4	1	9	Yes
getElementById()	Returns a reference to the first object with the specified id	5	1	9	Yes
getElementsByName()	Returns a collection of objects with the specified name	5	1	9	Yes
getElementsByTagName()	Returns a collection of objects with the specified tagname	5	1	9	Yes
open()	Opens a stream to collect the output from any document.write() or document.writeln() methods	4	1	9	Yes
write()	Writes HTML expressions or JavaScript code to a document	4	1	9	Yes
writeln()	Identical to the write() method, with the addition of writing a new line character after each expression	4	1	9	Yes

HTML DOM Anchor Object

Anchor Object

The Anchor object represents an HTML hyperlink.

For each instance of an <a> tag in an HTML document, an Anchor object is created.

An anchor can be used to create a link to another document (with the href attribute) or to create a bookmark inside a document (with the name attribute).

You can access an anchor by searching through the anchors[] array in the Document object, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Anchor Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns a keyboard key to access a link	5	1	No	Yes
charset	Sets or returns the character-set of the linked resource	6	1	9	Yes
coords	Sets or returns a comma-separated list with coordinates of a link in an image-map	6	1	9	Yes
href	Sets or returns the URL of the linked resource	5	1	9	Yes
hreflang	Sets or returns the language code of the linked resource	6	1	9	Yes
id	Sets or returns the id of a link	4	1	9	Yes
innerHTML	Sets or returns the text of a link	4	1	9	No
name	Sets or returns the name of a link	4	1	9	Yes
rel	Sets or returns the relationship between the current document and the target URL	5	1	No	Yes
rev	Sets or returns the relationship between the target URL and the current document	5	1	No	Yes
shape	Sets or returns the shape of a link in an image-map	6	1	9	Yes
tabIndex	Sets or returns the tab order for a link	6	1	9	Yes
target	Sets or returns where to open a link	5	1	9	Yes
type	Sets or returns the MIME type of the linked resource	6	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Anchor Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from the link	5	1	9	Yes
focus()	Gives focus to the link	5	1	9	Yes

HTML DOM Area Object

Area Object

The Area object represents an area of an image-map (An image-map is an image with clickable regions).

For each instance of an <area> tag in an HTML document, an Area object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Area Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access an area	5	1	No	Yes
alt	Sets or returns an alternate text to be displayed, if a browser cannot show an area	5	1	9	Yes
coords	Sets or returns the coordinates of a clickable area in an image-map	5	1	9	Yes
hash	Sets or returns the anchor part of the URL in an area	4	1	No	No
host	Sets or returns the hostname and port of the URL in an area	4	1	No	No
href	Sets or returns the URL of a link in an image-map	4	1	9	Yes
id	Sets or returns the id of an area	4	1	9	Yes
noHref	Sets or returns whether an area should be active or inactive	5	1	9	Yes
pathname	Sets or returns the pathname of the URL in an area	4	1	9	No
protocol	Sets or returns the protocol of the URL in an area	4	1	9	No
search	Sets or returns the query string part of the URL in an area	4	1	9	No
shape	Sets or returns the shape of an area in an image-map	5	1	9	Yes
tabIndex	Sets or returns the tab order for an area	5	1	9	Yes
target	Sets or returns where to open the link-URL in an area	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Base Object

Base Object

The Base object represents an HTML base element.

For each instance of a <base> tag in an HTML document, a Base object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Base Object Properties

Property	Description	IE	F	O	W3C
href	Sets or returns a base URL for all the links in a page	5	1	9	Yes
id	Sets or returns the id of the <base> element	4	1	9	Yes
target	Sets or returns the default target frame for all the links in a page	5	1	9	Yes

HTML DOM Body Object

Body Object

The Body object represents the body of the document (the HTML body).

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Body Object Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of the element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
id	Sets or returns the id of the body	5	1	9	Yes
lang	Sets or returns the language code for the element	5	1	9	Yes
title	Sets or returns the element's advisory title	5	1	9	Yes

HTML DOM Button Object

Button Object

The Button object represents a push button.

For each instance of a <button> tag in an HTML document, a Button object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Button Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a button	6	1	9	Yes
disabled	Sets or returns whether a button should be disabled	6	1	9	Yes
form	Returns a reference to the form that contains the button	6	1	9	Yes
id	Sets or returns the id of a button	6	1	9	Yes
name	Sets or returns the name of a button	6	1	9	Yes
tabIndex	Sets or returns the tab order for a button	6	1	9	Yes
type	Returns the type of form element a button is	6	1	9	Yes

value	Sets or returns the text that is displayed on a button	6	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Event Object

The event object gives you information about an event that has occurred.

Examples

Which mouse button was clicked?

```
<html>
<head>
<script type="text/javascript">
	function whichButton(event) {
		if (event.button==2) {
			alert("You clicked the right mouse button!");
		} else {
			alert("You clicked the left mouse button!");
		}
	}
</script>
</head>

<body onmousedown="whichButton(event)">
	<p>Click in the document. An alert box will alert which mouse button you clicked.</p>
</body>
</html>
```

What are the coordinates of the cursor?

```
<html>
<head>
	<script type="text/javascript">
		function show_coords(event) {
			x=event.clientX;
			y=event.clientY;
			alert("X coords: " + x + ", Y coords: " + y);
		}
	</script>
</head>

<body onmousedown="show_coords(event)">
	<p>Click in the document. An alert box will alert the x and y coordinates of the mouse pointer.</p>
</body>
</html>
```

What is the unicode of the key pressed?

```
<html>
<head>
	<script type="text/javascript">
		function whichButton(event) {
			alert(event.keyCode);
		}
	</script>
</head>

<body onkeyup="whichButton(event)">
	<p><b>Note:</b> Make sure the right frame has focus when trying this example!</p>
	<p>Press a key on your keyboard. An alert box will alert the unicode of the key pressed.</p>
</body>
</html>
```

What are the coordinates of the cursor, relative to the screen?

```
<html>
<head>
	<script type="text/javascript">
		function coordinates(event) {
			x=event.screenX;
			y=event.screenY;
			alert("X=" + x + " Y=" + y);
		}
	</script>
</head>
<body onmousedown="coordinates(event)">
	<p>Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor, relative to
the screen.</p>
</body>
</html>
```

What are the coordinates of the cursor?

```
<html>
<head>
	<script type="text/javascript">
		function coordinates(event) {
			x=event.x;
			y=event.y;
			alert("X=" + x + " Y=" + y);
		}
	</script>
</head>
<body onmousedown="coordinates(event)">
	<p>Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor.</p>
</body>
</html>
```

Was the shift key pressed?

```
<html>
<head>
	<script type="text/javascript">
		function isKeyPressed(event) {
			if (event.shiftKey==1) {
				alert("The shift key was pressed!");
			} else {
				alert("The shift key was NOT pressed!");
			}
		}
	</script>
</head>
```

```
<body onmousedown="isKeyPressed(event)">
        <p>Click somewhere in the document. An alert box will tell you if you pressed the shift key or not.</p>
</body>
</html>
```

Which element was clicked?

```
<html>
<head>
<script type="text/javascript">
        function whichElement(e) {
                var targ;
                if (!e) {
                        var e=window.event;
                }
                if (e.target) {
                        targ=e.target;
                } else if (e.srcElement) {
                        targ=e.srcElement;
                }
                if (targ.nodeType==3)  {       // defeat Safari bug
                        targ = targ.parentNode;
                }
                var tname;
                tname=targ.tagName;
                alert("You clicked on a " + tname + " element.");
        }
</script>
</head>

<body onmousedown="whichElement(event)">
        <p>Click somewhere in the document. An alert box will alert the tag name of the element you clicked on.</p>

        <h3>This is a header</h3>
        <p>This is a paragraph</p>
        <img border="0" src="ball16.gif" width="29" height="28" alt="Ball">
</body>

</html>
```

Which eventype occured?

```
<html>
<head>
        <script type="text/javascript">
                function getEventType(event) {
                        alert(event.type);
                }
        </script>
</head>

<body onmousedown="getEventType(event)">
        <p>Click in the document.
        An alert box will tell what type of event
        that was triggered.</p>
</body>
</html>
```

Event Object

The Event object represents the state of an event, such as the element in which the event occurred, the state of the keyboard keys, the location of the mouse, and the state of the mouse buttons.

Events are normally used in combination with functions, and the function will not be executed before the event occurs!

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Milan Das, SARC

Event Handlers

New to HTML 4.0 was the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element. Below is a list of the attributes that can be inserted into HTML tags to define event actions.

Attribute	The event occurs when...	IE	F	O	W3C
onabort	Loading of an image is interrupted	4	1	9	Yes
onblur	An element loses focus	3	1	9	Yes
onchange	The content of a field changes	3	1	9	Yes
onclick	Mouse clicks an object	3	1	9	Yes
ondblclick	Mouse double-clicks an object	4	1	9	Yes
onerror	An error occurs when loading a document or an image	4	1	9	Yes
onfocus	An element gets focus	3	1	9	Yes
onkeydown	A keyboard key is pressed	3	1	No	Yes
onkeypress	A keyboard key is pressed or held down	3	1	9	Yes
onkeyup	A keyboard key is released	3	1	9	Yes
onload	A page or an image is finished loading	3	1	9	Yes
onmousedown	A mouse button is pressed	4	1	9	Yes
onmousemove	The mouse is moved	3	1	9	Yes
onmouseout	The mouse is moved off an element	4	1	9	Yes
onmouseover	The mouse is moved over an element	3	1	9	Yes
onmouseup	A mouse button is released	4	1	9	Yes
onreset	The reset button is clicked	4	1	9	Yes
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	3	1	9	Yes
onsubmit	The submit button is clicked	3	1	9	Yes
onunload	The user exits the page	3	1	9	Yes

Mouse / Keyboard Attributes

Property	Description	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	6	1	9	Yes
metaKey	Returns whether or not the "meta" key was pressed when an	6	1	9	Yes

	event was triggered				
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

Other Event Attributes

Property	Description	IE	F	O	W3C
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event	No	1	9	Yes
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented	No	1	9	Yes
currentTarget	Returns the element whose event listeners triggered the event	No	1	9	Yes
eventPhase	Returns which phase of the event flow is currently being evaluated				Yes
target	Returns the element that triggered the event	No	1	9	Yes
timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)	No	1	9	Yes
type	Returns the name of the event	6	1	9	Yes

HTML DOM Form Object

Form Object

The Form object represents an HTML form.

For each instance of a <form> tag in an HTML document, a Form object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Form Object Collections

Collection	Description	IE	F	O	W3C
elements[]	Returns an array containing each element in the form	5	1	9	Yes

Form Object Properties

Property	Description	IE	F	O	W3C
acceptCharset	Sets or returns a list of possible character-sets for the form data	No	No	No	Yes
action	Sets or returns the action attribute of a form	5	1	9	Yes

enctype	Sets or returns the MIME type used to encode the content of a form	6	1	9	Yes
id	Sets or returns the id of a form	5	1	9	Yes
length	Returns the number of elements in a form	5	1	9	Yes
method	Sets or returns the HTTP method for sending data to the server	5	1	9	Yes
name	Sets or returns the name of a form	5	1	9	Yes
target	Sets or returns where to open the action-URL in a form	5	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Form Object Methods

Method	Description	IE	F	O	W3C
reset()	Resets the values of all elements in a form	5	1	9	Yes
submit()	Submits a form	5	1	9	Yes

HTML DOM Frame Object

Frame Object

The Frame object represents an HTML frame.

For each instance of a <frame> tag in an HTML document, a Frame object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Frame Object Properties

Property	Description	IE	F	O	W3C
contentDocument	Returns the frame's document as an HTML object	No	1	9	Yes
frameBorder	Sets or returns whether or not to display borders around a frame	5	1	9	Yes
id	Sets or returns the id of a frame	4	1	9	Yes
longDesc	Sets or returns a URL to a document containing a description of the frame contents	6	1	9	Yes
marginHeight	Sets or returns the top and bottom margins of a frame	5	1	9	Yes
marginWidth	Sets or returns the left and right margins of a frame	5	1	9	Yes
name	Sets or returns the name of a frame	5	1	9	Yes
noResize	Sets or returns whether or not a frame can be resized	5	1	9	Yes
scrolling	Sets or returns whether or not a frame should have scrollbars	No	1	No	Yes

src	Sets or returns the URL of the document that should be loaded into a frame	5	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Frameset Object

Frameset Object

The Frameset object represents an HTML frameset.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Frameset Object Properties

Property	Description	IE	F	O	W3C
cols	Sets or returns the number of columns in a frameset	5	1	9	Yes
id	Sets or returns the id of a frameset	4	1	9	Yes
rows	Sets or returns the number of rows in a frameset	5	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM IFrame Object

IFrame Object

The IFrame object represents an HTML inline frame.

For each instance of an <iframe> tag in an HTML document, an IFrame object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Properties

Property	Description	IE	F	O	W3C
align	Sets or returns how to align an iframe according to the	6	1	9	Yes

	surrounding text				
contentDocument	Returns the iframe's document as an HTML object	No	1	9	Yes
frameBorder	Sets or returns whether or not to display a border around an iframe	No	1	9	Yes
height	Sets or returns the height of an iframe	5	1	9	Yes
id	Sets or returns the id of an iframe	4	1	9	Yes
longDesc	Sets or returns a URL to a document containing a description of the iframe contents	6	1	9	Yes
marginHeight	Sets or returns the top and bottom margins of an iframe	5	1	9	Yes
marginWidth	Sets or returns the left and right margins of an frame	5	1	9	Yes
name	Sets or returns the name of an iframe	5	1	9	Yes
scrolling	Sets or returns whether or not an iframe should have scrollbars	No	1	No	Yes
src	Sets or returns the URL of the document that should be loaded into an iframe	5	1	9	Yes
width	Sets or returns the width of an iframe	5	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Image Object

Image Object

The Image object represents an embedded image.

For each instance of an <img> tag in an HTML document, an Image object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Image Object Properties

Property	Description	IE	F	O	W3C
align	Sets or returns how to align an image according to the surrounding text	5	1	9	Yes
alt	Sets or returns an alternate text to be displayed, if a browser cannot show an image	5	1	9	Yes
border	Sets or returns the border around an image	4	1	9	Yes
complete	Returns whether or not the browser has finished loading the image	4	1	9	No
height	Sets or returns the height of an image	4	1	9	Yes
hspace	Sets or returns the white space on the left and right side of the image	4	1	9	Yes

id	Sets or returns the id of the image	4	1	9	Yes
isMap	Returns whether or not an image is a server-side image map	5	1	9	Yes
longDesc	Sets or returns a URL to a document containing a description of the image	6	1	9	Yes
lowsrc	Sets or returns a URL to a low-resolution version of an image	4	1	9	No
name	Sets or returns the name of an image	4	1	9	Yes
src	Sets or returns the URL of an image	4	1	9	Yes
useMap	Sets or returns the value of the usemap attribute of an client-side image map	5	1	9	Yes
vspace	Sets or returns the white space on the top and bottom of the image	4	1	9	Yes
width	Sets or returns the width of an image	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Button Object

Button Object

The Button object represents a button in an HTML form.

For each instance of an <input type="button"> tag in an HTML form, a Button object is created.

You can access a button by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Button Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a button	5	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support buttons	5	1	9	Yes
disabled	Sets or returns whether or not a button should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the button	4	1	9	Yes
id	Sets or returns the id of a button	4	1	9	Yes
name	Sets or returns the name of a button	4	1	9	Yes
tabIndex	Sets or returns the tab order for a button	5	1	9	Yes
type	Returns the type of form element a button is	4	1	9	Yes
value	Sets or returns the text that is displayed on the button	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Button Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a button	4	1	9	Yes
click()	Simulates a mouse-click on a button	4	1	9	Yes
focus()	Gives focus to a button	4	1	9	Yes

HTML DOM Button Object

Button Object

The Button object represents a button in an HTML form.

For each instance of an <input type="button"> tag in an HTML form, a Button object is created.

You can access a button by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Button Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a button	5	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support buttons	5	1	9	Yes
disabled	Sets or returns whether or not a button should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the button	4	1	9	Yes
id	Sets or returns the id of a button	4	1	9	Yes
name	Sets or returns the name of a button	4	1	9	Yes
tabIndex	Sets or returns the tab order for a button	5	1	9	Yes
type	Returns the type of form element a button is	4	1	9	Yes
value	Sets or returns the text that is displayed on the button	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes

lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Button Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a button	4	1	9	Yes
click()	Simulates a mouse-click on a button	4	1	9	Yes
focus()	Gives focus to a button	4	1	9	Yes

HTML DOM FileUpload Object

FileUpload Object

For each instance of an <input type="file"> tag in an HTML form, a FileUpload object is created.

You can access a FileUpload object by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

FileUpload Object Properties

Property	Description	IE	F	O	W3C
accept	Sets or returns a comma-separated list of MIME types that indicates the MIME type of the file transfer				Yes
accessKey	Sets or returns the keyboard key to access the FileUpload object	4			Yes
alt	Sets or returns an alternate text to display if the browser does not support <input type="file">				Yes
defaultValue	Sets or returns the initial value of the FileUpload object	4	1		Yes
disabled	Sets or returns whether or not the FileUpload object should be disabled	4			Yes
form	Returns a reference to the form that contains the FileUpload object	4	1		Yes
id	Sets or returns the id of the FileUpload object	4	1		Yes
name	Sets or returns the name of the FileUpload object	4	1		Yes
tabIndex	Sets or returns the index that defines the tab order for the FileUpload object	4			Yes
type	Returns the type of the form element. For a FileUpload object it will be "file"	4	1		Yes
value	Returns the file name of the FileUpload object after the text is set by user input	4	1		Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes

lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

FileUpload Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from the FileUpload object	4	1		Yes
focus()	Gives focus to the FileUpload object	4	1		Yes
select()	Selects the FileUpload object	4			Yes

HTML DOM Hidden Object

Hidden Object

The Hidden object represents a hidden input field in an HTML form.

For each instance of an <input type="hidden"> tag in an HTML form, a Hidden object is created.

You can access a hidden input field by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Hidden Object Properties

Property	Description	IE	F	O	W3C
alt	Sets or returns an alternate text to display if a browser does not support hidden fields	5	1	9	Yes
form	Returns a reference to the form that contains the hidden field	4	1	9	Yes
id	Sets or returns the id of a hidden field	4	1	9	Yes
name	Sets or returns the name of a hidden field	4	1	9	Yes
type	Returns the type of form element a hidden input field is	4	1	9	Yes
value	Sets or returns the value of the value attribute of the hidden field	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Milan Das, SARC

HTML DOM Password Object

Password Object

The Password object represents a password field in an HTML form.

For each instance of an HTML <input type="password"> tag on a form, a Password object is created.

You can access a password field by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Password Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a password field	4	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support password fields	5	1	9	Yes
defaultValue	Sets or returns the default value of a password field	4	1	9	Yes
disabled	Sets or returns whether or not a password field should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the password field	4	1	9	Yes
id	Sets or returns the id of a password field	4	1	9	Yes
maxLength	Sets or returns the maximum number of characters in a password field	4	1	9	Yes
name	Sets or returns the name of a password field	4	1	9	Yes
readOnly	Sets or returns whether or not a password field should be read-only	4	1	9	Yes
size	Sets or returns the size of a password field	4	1	9	Yes
tabIndex	Sets or returns the tab order for a password field	4	1	9	Yes
type	Returns the type of form element a password field is	4	1	9	Yes
value	Sets or returns the value of the value attribute of the password field	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Password Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a password field	4	1	9	Yes
focus()	Sets focus on a password field	4	1	9	Yes
select()	Selects the text in a password field	4	1	9	Yes

Milan Das, SARC

HTML DOM Radio Object

Radio Object

The Radio object represents a radio button in an HTML form.

For each instance of an <input type="radio"> tag in an HTML form, a Radio object is created.

You can access a radio object by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Radio Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a radio button	4	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support radio buttons	5	1	9	Yes
checked	Sets or returns the state of a radio button	4	1	9	Yes
defaultChecked	Returns the default state of a radio button	4	1	9	Yes
disabled	Sets or returns whether or not a radio button should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the radio button	4	1	9	Yes
id	Sets or returns the id of a radio button	4	1	9	Yes
name	Sets or returns the name of a radio button	4	1	9	Yes
tabIndex	Sets or returns the tab order for a radio button	4	1	9	Yes
type	Returns the type of form element a radio button is	4	1	9	Yes
value	Sets or returns the value of the value attribute of the radio button	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Radio Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a radio button	No	1	9	Yes
click()	Simulates a mouse-click on a radio button	No	2	9	Yes
focus()	Sets focus on a radio button	No	1	9	Yes

Milan Das, SARC

HTML DOM Reset Object

Reset Object

The Reset object represents a reset button in an HTML form.

For each instance of an <input type="reset"> tag in an HTML form, a Reset object is created.

You can access a reset button by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Reset Object Properties

Property	Description	IE	F	O	W3C
accesskey	Sets or returns the keyboard key to access a reset button	4	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support reset buttons	5	1	9	Yes
disabled	Sets or returns whether or not a reset button should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the reset button	4	1	9	Yes
id	Sets or returns the id of a reset button	4	1	9	Yes
name	Sets or returns the name of a reset button	4	1	9	Yes
tabIndex	Sets or returns the tab order for a reset button	4	1	9	Yes
type	Returns the type of form element a reset button is	4	1	9	Yes
value	Sets or returns the text that is displayed on a reset button	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Reset Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a reset button	4	1	9	Yes
click()	Simulates a mouse-click on a reset button	4	1	9	Yes
focus()	Sets focus on a reset button	4	1	9	Yes

Milan Das, SARC

HTML DOM Submit Object

Submit Object

The Submit object represents a submit button in an HTML form.

For each instance of an <input type="submit"> tag in an HTML form, a Submit object is created.

Example: Form validation

You can access a submit button by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Submit Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a submit button	4	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support submit buttons	5	1	9	Yes
disabled	Sets or returns whether or not a submit button should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the submit button	4	1	9	Yes
id	Sets or returns the id of a submit button	4	1	9	Yes
name	Sets or returns the name of a submit button	4	1	9	Yes
tabIndex	Sets or returns the tab order for a submit button	4	1	9	Yes
type	Returns the type of form element a submit button is	4	1	9	Yes
value	Sets or returns the text that is displayed on a submit button	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Submit Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a submit button	4	1	9	Yes
click()	Simulates a mouse-click on a submit button	4	1	9	Yes
focus()	Sets focus on a submit button	4	1	9	Yes

Milan Das, SARC

HTML DOM Text Object

Text Object

The Text object represents a text-input field in an HTML form.

For each instance of an <input type="text"> tag in an HTML form, a Text object is created.

You can access a text-input field by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Text Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a text field	4	1	9	Yes
alt	Sets or returns an alternate text to display if a browser does not support text fields	5	1	9	Yes
defaultValue	Sets or returns the default value of a text field	4	1	9	Yes
disabled	Sets or returns whether or not a text field should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the text field	4	1	9	Yes
id	Sets or returns the id of a text field	4	1	9	Yes
maxLength	Sets or returns the maximum number of characters in a text field	4	1	9	Yes
name	Sets or returns the name of a text field	4	1	9	Yes
readOnly	Sets or returns whether or not a text field should be read-only	4	1	9	Yes
size	Sets or returns the size of a text field	4	1	9	Yes
tabIndex	Sets or returns the tab order for a text field	4	1	9	Yes
type	Returns the type of form element a text field is	4	1	9	Yes
value	Sets or returns the value of the value attribute of a text field	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Text Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a text field	4	1	9	Yes
focus()	Sets focus on a text field	4	1	9	Yes
select()	Selects the content of a text field	4	1	9	Yes

Milan Das, SARC

HTML DOM Link Object

Link Object

The Link object represents an HTML <link> element. The <link> element defines the relationship between two linked documents

The <link> element is defined in the head section of an HTML document.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Link Object Properties

Property	Description	IE	F	O	W3C
charset	Sets or returns the character encoding of the target URL	4	1	9	Yes
disabled	Sets or returns whether or not the target URL should be disabled	4	1	9	Yes
href	Sets or returns the URL of a linked resource	4	1	9	Yes
hreflang	Sets or returns the base language of the target URL	4	1	9	Yes
id	Sets or returns the id of a <link> element	4	1	9	Yes
media	Sets or returns on what device the document will be displayed	6	1	9	Yes
name	Sets or returns the name of a <link> element	4	No	No	Yes
rel	Sets or returns the relationship between the current document and the target URL	4	1	9	Yes
rev	Sets or returns the relationship between the target URL and the current document	4	1	9	Yes
type	Sets or returns the MIME type of the target URL	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes

HTML DOM Meta Object

Meta Object

The Meta object represents an HTML <meta> element.

The <meta> element provides meta-information about a HTML document, such as descriptions and keywords for search engines and refresh rates.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Meta Object Properties

Property	Description	IE	F	O	W3C
content	Sets or returns the value of the content attribute of a <meta> element	5	1	9	Yes
httpEquiv	Connects the content attribute to an HTTP header	5	1	9	Yes

name	Connects the content attribute to a name	5	1	9	Yes
scheme	Sets or returns the format to be used to interpret the value of the content attribute	6	1	9	Yes

HTML DOM Option Object

Option Object

The Option object represents an option in a dropdown list in an HTML form.

For each instance of an <option> tag in an HTML form, an Option object is created.

You can access an Option object by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Option Object Properties

Property	Description	IE	F	O	W3C
defaultSelected	Returns the default value of the selected attribute	4	1	9	Yes
disabled	Sets or returns whether or not an option should be disabled	4	1	9	Yes
form	Returns a reference to the form that contains an option	4	1	9	Yes
id	Sets or returns the id of an option	4	1	9	Yes
index	Returns the index position of an option in a dropdown list	4	1	9	Yes
label	Sets or returns a label for an option (only for option-groups)	6			Yes
selected	Sets or returns the value of the selected attribute	4	1	9	Yes
text	Sets or returns the text value of an option	4	1	9	Yes
value	Sets or returns the value to be sent to the server	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Select Object

Select Object

The Select object represents a dropdown list in an HTML form.

For each instance of an HTML <select> tag in a form, a Select object is created.

You can access a Select object by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Select Object Collections

Collection	Description	IE	F	O	W3C
options[]	Returns an array of all the options in a dropdown list	4	1	9	Yes

Select Object Properties

Property	Description	IE	F	O	W3C
disabled	Sets or returns whether or not a dropdown list should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the dropdown list	4	1	9	Yes
id	Sets or returns the id of a dropdown list	4	1	9	Yes
length	Returns the number of options in a dropdown list	4	1	9	Yes
multiple	Sets or returns whether or not multiple items can be selected	4	1	9	Yes
name	Sets or returns the name of a dropdown list	4	1	9	Yes
selectedIndex	Sets or returns the index of the selected option in a dropdown list	4	1	9	Yes
size	Sets or returns the number of visible rows in a dropdown list	4	1	9	Yes
tabIndex	Sets or returns the tab order for a dropdown list	5	1	9	Yes
type	Returns the type of form element a dropdown list is	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Select Object Methods

Method	Description	IE	F	O	W3C
add()	Adds an option to a dropdown list	4	1	9	Yes
blur()	Removes focus from a dropdown list	4	1	9	Yes
focus()	Sets focus on a dropdown list	4	1	9	Yes
remove()	Removes an option from a dropdown list	4	1	9	Yes

HTML DOM Style Object

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.

Milan Das, SARC

Syntax for using the Style object properties:

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

The Style object property categories:

- Background
- Border and Margin
- Layout
- List
- Misc
- Positioning
- Printing
- Scrollbar
- Table
- Text
- Standard

IE: Internet Explorer, **M:** Mac IE only, **W:** Windows IE only, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Background properties

Property	Description	IE	F	O	W3C
background	Sets all background properties in one	4	1	9	Yes
backgroundAttachment	Sets whether a background-image is fixed or scrolls with the page	4	1	9	Yes
backgroundColor	Sets the background-color of an element	4	1	9	Yes
backgroundImage	Sets the background-image of an element	4	1	9	Yes
backgroundPosition	Sets the starting position of a background-image	4	No	No	Yes
backgroundPositionX	Sets the x-coordinates of the backgroundPosition property	4	No	No	No
backgroundPositionY	Sets the y-coordinates of the backgroundPosition property	4	No	No	No
backgroundRepeat	Sets if/how a background-image will be repeated	4	1	9	Yes

Border and Margin properties

Property	Description	IE	F	O	W3C
border	Sets all properties for the four borders in one	4	1	9	Yes
borderBottom	Sets all properties for the bottom border in one	4	1	9	Yes
borderBottomColor	Sets the color of the bottom border	4	1	9	Yes
borderBottomStyle	Sets the style of the bottom border	4	1	9	Yes
borderBottomWidth	Sets the width of the bottom border	4	1	9	Yes
borderColor	Sets the color of all four borders (can have up to four colors)	4	1	9	Yes
borderLeft	Sets all properties for the left border in one	4	1	9	Yes
borderLeftColor	Sets the color of the left border	4	1	9	Yes
borderLeftStyle	Sets the style of the left border	4	1	9	Yes
borderLeftWidth	Sets the width of the left border	4	1	9	Yes
borderRight	Sets all properties for the right border in one	4	1	9	Yes
borderRightColor	Sets the color of the right border	4	1	9	Yes
borderRightStyle	Sets the style of the right border	4	1	9	Yes

borderRightWidth	Sets the width of the right border	4	1	9	Yes
borderStyle	Sets the style of all four borders (can have up to four styles)	4	1	9	Yes
borderTop	Sets all properties for the top border in one	4	1	9	Yes
borderTopColor	Sets the color of the top border	4	1	9	Yes
borderTopStyle	Sets the style of the top border	4	1	9	Yes
borderTopWidth	Sets the width of the top border	4	1	9	Yes
borderWidth	Sets the width of all four borders (can have up to four widths)	4	1	9	Yes
margin	Sets the margins of an element (can have up to four values)	4	1	9	Yes
marginBottom	Sets the bottom margin of an element	4	1	9	Yes
marginLeft	Sets the left margin of an element	4	1	9	Yes
marginRight	Sets the right margin of an element	4	1	9	Yes
marginTop	Sets the top margin of an element	4	1	9	Yes
outline	Sets all outline properties in one	5M	1	9	Yes
outlineColor	Sets the color of the outline around an element	5M	1	9	Yes
outlineStyle	Sets the style of the outline around an element	5M	1	9	Yes
outlineWidth	Sets the width of the outline around an element	5M	1	9	Yes
padding	Sets the padding of an element (can have up to four values)	4	1	9	Yes
paddingBottom	Sets the bottom padding of an element	4	1	9	Yes
paddingLeft	Sets the left padding of an element	4	1	9	Yes
paddingRight	Sets the right padding of an element	4	1	9	Yes
paddingTop	Sets the top padding of an element	4	1	9	Yes

Layout properties

Property	Description	IE	F	O	W3C
clear	Sets on which sides of an element other floating elements are not allowed	4	1	9	Yes
clip	Sets the shape of an element	4	1	9	Yes
content	Sets meta-information	5M	1		Yes
counterIncrement	Sets a list of counter names, followed by an integer. The integer indicates by how much the counter is incremented for every occurrence of the element. The default is 1	5M	1		Yes
counterReset	Sets a list of counter names, followed by an integer. The integer gives the value that the counter is set to on each occurrence of the element. The default is 0	5M	1		Yes
cssFloat	Sets where an image or a text will appear (float) in another element	5M	1	9	Yes
cursor	Sets the type of cursor to be displayed	4	1	9	Yes
direction	Sets the text direction of an element	5	1	9	Yes

display	Sets how an element will be displayed	4	1	9	Yes
height	Sets the height of an element	4	1	9	Yes
markerOffset	Sets the distance between the nearest border edges of a marker box and its principal box	5M	1		Yes
marks	Sets whether cross marks or crop marks should be rendered just outside the page box edge	5M	1		Yes
maxHeight	Sets the maximum height of an element	5M	1	9	Yes
maxWidth	Sets the maximum width of an element	5M	1	9	Yes
minHeight	Sets the minimum height of an element	5M	1	9	Yes
minWidth	Sets the minimum width of an element	5M	1	9	Yes
overflow	Specifies what to do with content that does not fit in an element box	4	1	9	Yes
verticalAlign	Sets the vertical alignment of content in an element	4	1	No	Yes
visibility	Sets whether or not an element should be visible	4	1	9	Yes
width	Sets the width of an element	4	1	9	Yes

List properties

Property	Description	IE	F	O	W3C
listStyle	Sets all the properties for a list in one	4	1	9	Yes
listStyleImage	Sets an image as the list-item marker	4	1	No	Yes
listStylePosition	Positions the list-item marker	4	1	9	Yes
listStyleType	Sets the list-item marker type	4	1	9	Yes

Misc properties

Property	Description	IE	F	O	W3C
cssText		4	1		

Positioning properties

Property	Description	IE	F	O	W3C
bottom	Sets how far the bottom edge of an element is above/below the bottom edge of the parent element	5	1	9	Yes
left	Sets how far the left edge of an element is to the right/left of the left edge of the parent element	4	1	9	Yes
position	Places an element in a static, relative, absolute or fixed position	4	1	9	Yes
right	Sets how far the right edge of an element is to the left/right of the right edge of the parent element	5	1	9	Yes
top	Sets how far the top edge of an element is above/below the top edge of the parent element	4	1	9	Yes
zIndex	Sets the stack order of an element	4	1	9	Yes

Printing properties

Property	Description	IE	F	O	W3C
orphans	Sets the minimum number of lines for a paragraph that must be left at the bottom of a page	5M	1	9	Yes
page	Sets a page type to use when displaying an element	5M	1	9	Yes
pageBreakAfter	Sets the page-breaking behavior after an element	4	1	9	Yes
pageBreakBefore	Sets the page-breaking behavior before an element	4	1	9	Yes
pageBreakInside	Sets the page-breaking behavior inside an element	5M	1	9	Yes
size	Sets the orientation and size of a page		1	9	Yes
widows	Sets the minimum number of lines for a paragraph that must be left at the top of a page	5M	1	9	Yes

Scrollbar properties (IE-only)

Property	Description	IE	F	O	W3C
scrollbar3dLightColor	Sets the color of the left and top sides of the arrows and scroll boxes	5W	No	No	No
scrollbarArrowColor	Sets the color of the arrows on a scroll bar	5W	No	No	No
scrollbarBaseColor	Sets the base color of the scroll bar	5W	No	No	No
scrollbarDarkShadowColor	Sets the color of the right and bottom sides of the arrows and scroll boxes	5W	No	No	No
scrollbarFaceColor	Sets the front color of the scroll bar	5W	No	No	No
scrollbarHighlightColor	Sets the color of the left and top sides of the arrows and scroll boxes, and the background of a scroll bar	5W	No	No	No
scrollbarShadowColor	Sets the color of the right and bottom sides of the arrows and scroll boxes	5W	No	No	No
scrollbarTrackColor	Sets the background color of a scroll bar	5W	No	No	No

Table properties

Property	Description	IE	F	O	W3C
borderCollapse	Sets whether the table border are collapsed into a single border or detached as in standard HTML	5	1	9	Yes
borderSpacing	Sets the distance that separates cell borders	5M	1	9	Yes
captionSide	Sets the position of the table caption	5M	No	No	Yes
emptyCells	Sets whether or not to show empty cells in a table	5M	1	9	Yes
tableLayout	Sets the algorithm used to display the table cells, rows, and columns	5	No	No	Yes

Text properties

Property	Description	IE	F	O	W3C
color	Sets the color of the text	4	1	9	Yes
font	Sets all font properties in one	4	1	9	Yes

fontFamily	Sets the font of an element	4	1	9	Yes
fontSize	Sets the font-size of an element	4	1	9	Yes
fontSizeAdjust	Sets/adjusts the size of a text	5M	1	No	Yes
fontStretch	Sets how to condense or stretch a font	5M	No	No	Yes
fontStyle	Sets the font-style of an element	4	1	9	Yes
fontVariant	Displays text in a small-caps font	4	1	9	Yes
fontWeight	Sets the boldness of the font	4	1	9	Yes
letterSpacing	Sets the space between characters	4	1	9	Yes
lineHeight	Sets the distance between lines	4	1	9	Yes
quotes	Sets which quotation marks to use in a text	5M	1		Yes
textAlign	Aligns the text	4	1	9	Yes
textDecoration	Sets the decoration of a text	4	1	9	Yes
textIndent	Indents the first line of text	4	1	9	Yes
textShadow	Sets the shadow effect of a text	5M	1		Yes
textTransform	Sets capitalization effect on a text	4	1	9	Yes
unicodeBidi		5	1		Yes
whiteSpace	Sets how to handle line-breaks and white-space in a text	4	1	9	Yes
wordSpacing	Sets the space between words in a text	6	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Table Object

Table Object

The Table object represents an HTML table.

For each instance of a <table> tag in an HTML document, a Table object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Table Object Collections

Collection	Description	IE	F	O	W3C
cells[]	Returns an array containing each cell in a table	5	1	1	No
rows[]	Returns an array containing each row in a table	4	1	9	Yes
tBodies[]	Returns an array containing each tbody in a table	4			Yes

Table Object Properties

Property	Description	IE	F	O	W3C
border	Sets or returns the width of the table border	4	1	9	Yes
caption	Sets or returns the caption of a table	4	1	9	Yes
cellPadding	Sets or returns the amount of space between the cell border and cell content	4	1	9	Yes
cellSpacing	Sets or returns the amount of space between the cells in a table	4	1	9	Yes
frame	Sets or returns the outer-borders of a table	4	1	9	Yes
id	Sets or returns the id of a table	4	1	9	Yes
rules	Sets or returns the inner-borders of a table	4	1	9	Yes
summary	Sets or returns a description of a table	6	1	9	Yes
tFoot	Returns the TFoot object of a table	4	1	9	Yes
tHead	Returns the THead object of a table	4	1	9	Yes
width	Sets or returns the width of a table	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Table Object Methods

Method	Description	IE	F	O	W3C
createCaption()	Creates a caption element for a table	4	1	9	Yes
createTFoot()	Creates an empty tFoot element in a table	4	1	9	Yes
createTHead()	Creates an empty tHead element in a table	4	1	9	Yes
deleteCaption()	Deletes the caption element and its content from a table	4	1	9	Yes
deleteRow()	Deletes a row from a table	4	1	9	Yes
deleteTFoot()	Deletes the tFoot element and its content from a table	4	1	9	Yes
deleteTHead()	Deletes the tHead element and its content from a table	4	1	9	Yes
insertRow()	Inserts a new row in a table	4	1	9	Yes

HTML DOM TableCell Object

TableCell Object

The TableCell object represents an HTML table cell.

For each instance of a <td> tag in an HTML document, a TableCell object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

TableCell Object Properties

Property	Description	IE	F	O	W3C
abbr	Sets or returns an abbreviated version of the content in a table cell	6	1	9	Yes
align	Sets or returns the horizontal alignment of data within a table cell	4	1	9	Yes
axis	Sets or returns a comma-delimited list of related table cells	6	1	9	Yes
cellIndex	Returns the position of a cell in the cells collection of a row	4	1	9	Yes
ch	Sets or returns the alignment character for a table cell				Yes
chOff	Sets or returns the offset of alignment character for a table cell				Yes
colSpan	Sets or returns the number of columns a table cell should span	4	1	9	Yes
headers	Sets or returns a list of space-separated header-cell ids				Yes
id	Sets or returns the id of a table cell	4	1	9	Yes
innerHTML	Sets or returns the HTML between the start and end tags of a table cell	4	1	9	No
rowSpan	Sets or returns the number of rows a table cell should span	4	1	9	Yes
scope	Sets or returns if this cell provides header information				Yes
vAlign	Sets or returns the vertical alignment of data within a table cell	4	1	9	Yes
width	Sets or returns the width of a table cell	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM TableRow Object

TableRow Object

The TableRow object represents an HTML table row.

For each instance of a <tr> tag in an HTML document, a TableRow object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

TableRow Object Collections

Collection	Description	IE	F	O	W3C
cells[]	Returns an array containing each cell in the table row	4	1	9	Yes

TableRow Object Properties

Property	Description	IE	F	O	W3C

align	Sets or returns the horizontal alignment of data within a table row	4	1	9	Yes
ch	Sets or returns the alignment character for cells in a table row				Yes
chOff	Sets or returns the offset of alignment character for the cells in a table row				Yes
id	Sets or returns the id of a table row	4	1	9	Yes
innerHTML	Sets or returns the HTML between the start and end tags of a table row	5	1	9	No
rowIndex	Returns the position of a row in the table's rows collection	4	1	9	Yes
sectionRowIndex	Returns the position of a row in the tBody, tHead, or tFoot rows collection				Yes
vAlign	Sets or returns the vertically alignment of data within a table row	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

TableRow Object Methods

Method	Description	IE	F	O	W3C
deleteCell()	Deletes a cell in a table row	4	1	9	Yes
insertCell()	Inserts a cell in a table row	4	1	9	Yes

HTML DOM Textarea Object

Textarea Object

The Textarea object represents a text-area in an HTML form. For each instance of an HTML <textarea> tag in a form, a Textarea object is created.

You can access a Textarea object by indexing the elements array (by number or name) of the corresponding form or by using getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Textarea Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a textarea	4	1	9	Yes
cols	Sets or returns the width of a textarea	4	1	9	Yes
defaultValue	Sets or returns the default text in a textarea	4	1	9	Yes
disabled	Sets or returns whether or not a textarea should be disabled	5	1	9	Yes
form	Returns a reference to the form that contains the textarea	4	1	9	Yes
id	Sets or returns the id of a textarea	4	1	9	Yes
name	Sets or returns the name of a textarea	4	1	9	Yes
readOnly	Sets or returns whether or not a textarea should be read-only	4	1	9	Yes

rows	Sets or returns the height of a textarea	4	1	9	Yes
tabIndex	Sets or returns the tab order for the textarea	4	1	9	Yes
type	Returns the type of the form element	4	1	9	Yes
value	Sets or returns the text in a textarea	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Textarea Object Methods

Method	Description	IE	F	O	W3C
blur()	Removes focus from a textarea	4	1	9	Yes
focus()	Sets focus on a textarea	4	1	9	Yes
select()	Selects the text in a textarea	4	1	9	Yes

Milan Das, SARC

JavaScript Examples

Basic JavaScript Examples

Write text with JavaScript

```
<html>
	<body>
		<script type="text/javascript">
			document.write("Hello World!");
		</script>
	</body>
</html>
```

Write HTML tags with JavaScript

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<h1>This is a header</h1>");
		</script>
	</body>
</html>
```

JavaScript in the body section

```
<html>
	<head>
	</head>
	<body>
		<script type="text/javascript">
			document.write("This message is written by JavaScript");
		</script>
	</body>
</html>
```

JavaScript in the head section

```
<html>
	<head>
		<script type="text/javascript">
			function message() {
				alert("This alert box was called with the onload event");
			}
		</script>
	</head>
	<body onload="message()">
	</body>
</html>
```

An external JavaScript

```
<html>
	<head>
	</head>
	<body>
		<script src="xxx.js">
		</script>
		<p>The actual script is in an external script file called "xxx.js".</p>
	</body>
</html>
```

Examples explained

JavaScript Statements, Comments and Blocks

JavaScript statements.

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<h1>This is a header</h1>");
			document.write("<p>This is a paragraph</p>");
			document.write("<p>This is another paragraph</p>");
		</script>
	</body>
</html>
```

JavaScript blocks.

```
<html>
	<body>
		<script type="text/javascript">
			{
				document.write("<h1>This is a header</h1>");
				document.write("<p>This is a paragraph</p>");
				document.write("<p>This is another paragraph</p>");
			}
		</script>
	</body>
</html>
```

Single line comments.

```
<html>
	<body>
		<script type="text/javascript">
			// This will write a header:
			document.write("<h1>This is a header</h1>");
			// This will write two paragraphs:
			document.write("<p>This is a paragraph</p>");
			document.write("<p>This is another paragraph</p>");
		</script>
	</body>
</html>
```

Multiple lines comments.

```
<html>
	<body>
		<script type="text/javascript">
			/* The code below will write
			   one header and two paragraphs */
			document.write("<h1>This is a header</h1>");
			document.write("<p>This is a paragraph</p>");
			document.write("<p>This is another paragraph</p>");
		</script>
	</body>
</html>
```

Single line comment to prevent execution.

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<h1>This is a header</h1>");
			document.write("<p>This is a paragraph</p>");
			//document.write("<p>This is another paragraph</p>");
		</script>
```

```
	</body>
</html>
```

Multiple lines comment to prevent execution.

```
<html>
	<body>
		<script type="text/javascript">
			/*
			document.write("<h1>This is a header</h1>");
			document.write("<p>This is a paragraph</p>");
			document.write("<p>This is another paragraph</p>");
			*/
		</script>
	</body>
</html>
```

JavaScript Variables

Declare a variable, assign a value to it, and display it

```
<html>
	<body>
		<script type="text/javascript">
			var firstname;
			firstname="Hege";
			document.write(firstname);
			document.write("<br />");
			firstname="Tove";
			document.write(firstname);
		</script>
		<p>The script above declares a variable, assigns a value to it, displays the value, change the value,
		and displays the value again.</p>
	</body>
</html>
```

JavaScript Conditional If ... Else

If statement

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			var time = d.getHours();
			if (time < 10)  {
				document.write("<b>Good morning</b>");
			}
		</script>
		<p>This example demonstrates the If statement.</p>
		<p>If the time on your browser is less than 10, you will get a "Good morning"
		greeting.</p>
	</body>
</html>
```

If...else statement

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			var time = d.getHours();
			if (time < 10) {
				document.write("<b>Good morning</b>");
			} else {
				document.write("<b>Good day</b>");
			}
		</script>
		<p>This example demonstrates the If...Else statement.</p>
		<p>If the time on your browser is less than 10, you will get a "Good morning" greeting.
```

```
		Otherwise you will get a "Good day" greeting.</p>
	</body>
</html>
```

Random link

```
<html>
	<body>
		<script type="text/javascript">
			var r=Math.random();
			if (r>0.5) {
				document.write("<a href='http://www.w3schools.com'>Learn Web
				Development!</a>");
			} else {
				document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>");
			}
		</script>
	</body>
</html>
```

Switch statement

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			theDay=d.getDay();
			switch (theDay) {
				case 5: document.write("<b>Finally Friday</b>"); break;
				case 6: document.write("<b>Super Saturday</b>"); break;
				case 0: document.write("<b>Sleepy Sunday</b>"); break;
				default:
					document.write("<b>I'm really looking forward to this
					weekend!</b>");
			}
		</script>
		<p>This JavaScript will generate a different greeting based on what day it is.
		Note that Sunday=0, Monday=1, Tuesday=2, etc.</p>
	</body>
</html>
```

JavaScript Popup Boxes

Alert box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("I am an alert box!!");
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
</html>
```

Alert box with line breaks

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!");
			}
		</script>
	</head>
	<body>
```

```
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
</html>
```

Confirm box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_confirm() {
				var r=confirm("Press a button");
				if (r==true) {
					document.write("You pressed OK!");
				} else {
					document.write("You pressed Cancel!");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_confirm()" value="Display a confirm box" />
	</body>
</html>
```

Prompt box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_prompt() {
				var name=prompt("Please enter your name","Harry Potter");
				if (name!=null && name!="") {
					document.write("Hello " + name + "! How are you today?");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_prompt()" value="Display a prompt box" />
	</body>
</html>
```

JavaScript Functions

Call a function

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction() {
				alert("HELLO");
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" onclick="myfunction()" value="Call function">
		</form>
		<p>By pressing the button, a function will be called.
		The function will alert a message.</p>
	</body>
</html>
```

Function with an argument

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction() {
				alert("HELLO");
			}
```

```
		</script>
	</head>
	<body>
		<form>
			<input type="button" onclick="myfunction()" value="Call function">
		</form>
		<p>By pressing the button, a function will be called. The function will alert a message.</p>
	</body>
</html>
```

Function with an argument 2

```
<html>
	<head>
		<script type="text/javascript">
			function myfunction(txt)  {
				alert(txt);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" onclick="myfunction('Good Morning!')" value="In the Morning">
			<input type="button" onclick="myfunction('Good Evening!')" value="In the Evening">
		</form>
		<p>When you click on one of the buttons, a function will be called.
		The function will alert the argument that is passed to it.</p>
	</body>
</html>
```

Function that returns a value

```
<html>
	<head>
		<script type="text/javascript">
			function myFunction() {
				return ("Hello, have a nice day!");
			}
		</script>
	</head>
	<body>
		<script type="text/javascript">
			document.write(myFunction())
		</script>
		<p>The script in the body section calls a function.</p>
		<p>The function returns a text.</p>
	</body>
</html>
```

Function with arguments, that returns a value

```
<html>
	<head>
		<script type="text/javascript">
			function product(a,b) {
				return a*b;
			}
		</script>
	</head>
	<body>
		<script type="text/javascript">
			document.write(product(4,3));
		</script>
		<p>The script in the body section calls a function with two parameters (4 and 3).</p>
		<p>The function will return the product of these two parameters.</p>
	</body>
</html>
```

Milan Das, SARC

JavaScript Loops

For loop

```
<html>
	<body>
		<script type="text/javascript">
			for (i = 0; i <= 5; i++) {
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation:</p>
		<p>This for loop starts with i=0.</p>
		<p>As long as <b>i</b> is less than, or equal to 5, the loop will continue to run.</p>
		<p><b>i</b> will increase by 1 each time the loop runs.</p>
	</body>
</html>
```

Looping through HTML headers

```
<html>
	<body>
		<script type="text/javascript">
			for (i = 0; i <= 5; i++) {
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation:</p>
		<p>This for loop starts with i=0.</p>
		<p>As long as <b>i</b> is less than, or equal to 5, the loop will continue to run.</p>
		<p><b>i</b> will increase by 1 each time the loop runs.</p>
	</body>
</html>
```

While loop

```
<html>
	<body>
		<script type="text/javascript">
			i=0;
			while (i<=5) {
				document.write("The number is " + i);
				document.write("<br />");
				i++;
			}
		</script>
		<p>Explanation:</p>
		<p><b>i</b> is equal to 0.</p>
		<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>
		<p><b>i</b> will increase by 1 each time the loop runs.</p>
	</body>
</html>
```

Do While loop

```
<html>
	<body>
		<script type="text/javascript">
			i = 0;
			do {
				document.write("The number is " + i);
				document.write("<br />");
				i++;
			} while (i <= 5)
		</script>
		<p>Explanation:</p>
		<p><b>i</b>  equal to 0.</p>
		<p>The loop will run</p>
		<p><b>i</b> will increase by 1 each time the loop runs.</p>
```

```
		<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>
	</body>
</html>
```

Break a loop

```
<html>
	<body>
		<script type="text/javascript">
			var i=0;
			for (i=0;i<=10;i++) {
				if (i==3) {
					break;
				}
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation: The loop will break when i=3.</p>
	</body>
</html>
```

Break and continue a loop

```
<html>
	<body>
		<script type="text/javascript">
			var i=0;
			for (i=0;i<=10;i++) {
				if (i==3) {
					continue;
				}
				document.write("The number is " + i);
				document.write("<br />");
			}
		</script>
		<p>Explanation:
		The loop will break the current loop and continue with the next value when i=3.</p>
	</body>
</html>
```

Use a for...in statement to loop through the elements of an array

```
<html>
	<body>
		<script type="text/javascript">
			var x;
			var mycars = new Array();
			mycars[0] = "Saab";
			mycars[1] = "Volvo";
			mycars[2] = "BMW";
			for (x in mycars) {
				document.write(mycars[x] + "<br />");
			}
		</script>
	</body>
</html>
```

JavaScript Error Handling

The try...catch statement

```
<html>
	<head>
		<script type="text/javascript">
			var txt="";
			function message() {
				try {
					adddlert("Welcome guest!");
				} catch(err) {
```

```
                                        txt="There was an error on this page.\n\n";
                                        txt+="Error description: " + err.description + "\n\n";
                                        txt+="Click OK to continue.\n\n";
                                        alert(txt);
                                }
                        }
                </script>
        </head>
        <body>
                <input type="button" value="View message" onclick="message()" />
        </body>
</html>
```

The try...catch statement with a confirm box

```
<html>
        <head>
                <script type="text/javascript">
                        var txt=""
                        function message() {
                                try {
                                        adddlert("Welcome guest!");
                                } catch(err) {
                                        txt="There was an error on this page.\n\n";
                                        txt+="Click OK to continue viewing this page,\n";
                                        txt+="or Cancel to return to the home page.\n\n";
                                        if(!confirm(txt)) {
                                                document.location.href="http://www.w3schools.com/";
                                        }
                                }
                        }
                </script>
        </head>
        <body>
                <input type="button" value="View message" onclick="message()" />
        </body>
</html>
```

The onerror event

```
<html>
        <head>
                <script type="text/javascript">
                        onerror=handleErr;
                        var txt="";

                        function handleErr(msg,url,l) {
                                txt="There was an error on this page.\n\n";
                                txt+="Error: " + msg + "\n";
                                txt+="URL: " + url + "\n";
                                txt+="Line: " + l + "\n\n";
                                txt+="Click OK to continue.\n\n";
                                alert(txt);
                                return true;
                        }
                        function message() {
                                adddlert("Welcome guest!");
                        }
                </script>
        </head>
        <body>
                <input type="button" value="View message" onclick="message()" />
        </body>
</html>
```

Advanced JavaScript Examples

Detect the visitor's browser and browser version

```
<html>
	<body>
		<script type="text/javascript">
			var browser=navigator.appName;
			var b_version=navigator.appVersion;
			var version=parseFloat(b_version);
			document.write("Browser name: "+ browser);
			document.write("<br />");
			document.write("Browser version: "+ version);
		</script>
	</body>
</html>
```

More details about the visitor's browser

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<p>Browser: ");
			document.write(navigator.appName + "</p>");

			document.write("<p>Browserversion: ");
			document.write(navigator.appVersion + "</p>");

			document.write("<p>Code: ");
			document.write(navigator.appCodeName + "</p>");

			document.write("<p>Platform: ");
			document.write(navigator.platform + "</p>");

			document.write("<p>Cookies enabled: ");
			document.write(navigator.cookieEnabled + "</p>");

			document.write("<p>Browser's user agent header: ");
			document.write(navigator.userAgent + "</p>");
		</script>
	</body>
</html>
```

All details about the visitor's browser

```
<html>
	<body>
		<script type="text/javascript">
			var x = navigator;
			document.write("CodeName=" + x.appCodeName);
			document.write("<br />");
			document.write("MinorVersion=" + x.appMinorVersion);
			document.write("<br />");
			document.write("Name=" + x.appName);
			document.write("<br />");
			document.write("Version=" + x.appVersion);
			document.write("<br />");
			document.write("CookieEnabled=" + x.cookieEnabled);
			document.write("<br />");
			document.write("CPUClass=" + x.cpuClass);
			document.write("<br />");
			document.write("OnLine=" + x.onLine);
			document.write("<br />");
			document.write("Platform=" + x.platform);
			document.write("<br />");
			document.write("UA=" + x.userAgent);
			document.write("<br />");
			document.write("BrowserLanguage=" + x.browserLanguage);
			document.write("<br />");
			document.write("SystemLanguage=" + x.systemLanguage);
			document.write("<br />");
			document.write("UserLanguage=" + x.userLanguage);
```

```
		</script>
	</body>
</html>
```

Alert user, depending on browser

```
<html>
	<head>
		<script type="text/javascript">
			function detectBrowser() {
				var browser=navigator.appName;
				var b_version=navigator.appVersion;
				var version=parseFloat(b_version);
				if ((browser=="Netscape"||browser=="Microsoft Internet Explorer")
					&& (version>=4)) {
					alert("Your browser is good enough!");
				} else {
					alert("It's time to upgrade your browser!");
				}
			}
		</script>
	</head>
	<body onload="detectBrowser()">
	</body>
</html>
```

Create a welcome cookie

```
<html>
	<head>
		<script type="text/javascript">
			function getCookie(c_name) {
				if (document.cookie.length>0) {
					c_start=document.cookie.indexOf(c_name + "=");
					if (c_start!=-1) {
						c_start=c_start + c_name.length+1 ;
						c_end=document.cookie.indexOf(";",c_start);
						if (c_end==-1) c_end=document.cookie.length
						return
						unescape(document.cookie.substring(c_start,c_end));
					}
				}
				return ""
			}

			function setCookie(c_name,value,expiredays) {
				var exdate=new Date();
				exdate.setDate(exdate.getDate()+expiredays);
				document.cookie=c_name+ "=" +
				escape(value)+((expiredays==null) ? "" : "; expires="+exdate.toGMTString());
			}

			function checkCookie() {
				username=getCookie('username');
				if (username!=null && username!="") {
					alert('Welcome again '+username+'!');
				} else {
					username=prompt('Please enter your name:',"");
					if (username!=null && username!="") {
						setCookie('username',username,365);
					}
				}
			}
		</script>
	</head>
	<body onLoad="checkCookie()">
	</body>
</html>
```

Button animation

```
<html>
	<head>
		<script type="text/javascript">
			function mouseOver() {
				document.b1.src ="b_blue.gif";
			}
			function mouseOut() {
				document.b1.src ="b_pink.gif";
			}
		</script>
	</head>
	<body>
		<a href="http://www.w3schools.com" target="_blank">
		<img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1" width="26"
		height="26" onmouseover="mouseOver()" onmouseout="mouseOut()" /></a>
	</body>
</html>
```

Image map with added JavaScript

```
<html>
	<head>
		<script type="text/javascript">
			function writeText(txt) {
				document.getElementById("desc").innerHTML=txt;
			}
		</script>
	</head>
	<body>
		<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap" />
		<map id ="planetmap" name="planetmap">
			<area shape ="rect" coords ="0,0,82,126" onMouseOver="writeText('The Sun and the gas
			giant planets like Jupiter are by far the largest objects in our Solar System.')" href="sun.htm"
			target ="_blank" alt="Sun" />

			<area shape ="circle" coords ="90,58,3" onMouseOver="writeText('The planet
			Mercury is very difficult to study from the Earth because it is always so close to
			the Sun.')" href ="mercur.htm" target ="_blank" alt="Mercury" />

			<area shape ="circle" coords ="124,58,8" onMouseOver="writeText('Until the
			1960s, Venus was often considered a  twin sister to the Earth because Venus is
			the nearest planet to us, and because the two planets seem to share many
			characteristics.')" href ="venus.htm" target ="_blank" alt="Venus" />
		</map>
		<p id="desc"></p>
	</body>
</html>
```

Simple timing

```
<html>
	<head>
		<script type="text/javascript">
			function timedMsg() {
				var t=setTimeout("alert('5 seconds!')",5000);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Display timed alertbox!" onClick = "timedMsg()">
		</form>
		<p>Click on the button above. An alert box will be displayed after 5 seconds.</p>
	</body>
</html>
```

Another simple timing

```
<html>
```

```
	<head>
		<script type="text/javascript">
			function timedText() {
				var t1=setTimeout("document.getElementById('txt').value='2 seconds!'",2000);
				var t2=setTimeout("document.getElementById('txt').value='4 seconds!'",4000);
				var t3=setTimeout("document.getElementById('txt').value='6 seconds!'",6000);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Display timed text!" onClick="timedText()">
			<input type="text" id="txt"
		</form>
		<p>Click on the button above.
		The input field will tell you when two, four, and six seconds have passed.</p>
	</body>
</html>
```

Timing event in an infinite loop

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
		</form>
		<p>Click on the button above. The input field will count for ever, starting at 0.</p>
	</body>
</html>
```

Timing event in an infinite loop - with a Stop button

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}
			function stopCount() {
				clearTimeout(t);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
			<input type="button" value="Stop count!" onClick="stopCount()">
		</form>
		<p>Click on the "Start count!" button above to start the timer.
		The input field will count forever, starting at 0.
		Click on the "Stop count!" button to stop the counting.</p>
	</body>
</html>
```

A clock created with a timing event

```
<html>
	<head>
		<script type="text/javascript">
			function startTime() {
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				// add a zero in front of numbers<10
				m=checkTime(m);
				s=checkTime(s);
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}
			function checkTime(i) {
				if (i<10) {
					i="0" + i;
				}
				return i;
			}
		</script>
	</head>
	<body onload="startTime()">
		<div id="txt"></div>
	</body>
</html>
```

Create a direct instance of an object

```
<html>
	<head>
		<script type="text/javascript">
			function startTime() {
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				// add a zero in front of numbers<10
				m=checkTime(m);
				s=checkTime(s);
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}

			function checkTime(i) {
				if (i<10) {
					i="0" + i;
				}
				return i;
			}
		</script>
	</head>
	<body onload="startTime()">
		<div id="txt"></div>
	</body>
</html>
```

Create a template for an object

```
<html>
	<body>
		<script type="text/javascript">
			function person(firstname,lastname,age,eyecolor) {
				this.firstname=firstname;
				this.lastname=lastname;
				this.age=age;
				this.eyecolor=eyecolor;
			}
			myFather=new person("John","Doe",50,"blue");
			document.write(myFather.firstname + " is " + myFather.age + " years old.");
```

```
		</script>
	</body>
</html>
```

JavaScript Objects Examples

Examples of using the built-in JavaScript objects.

String Object

Return the length of a string

```
<html>
	<body>
		<script type="text/javascript">
			var txt="Hello World!";
			document.write(txt.length);
		</script>
	</body>
</html>
```

Style strings

```
<html>
	<body>
		<script type="text/javascript">
			var txt="Hello World!";
			document.write("<p>Big: " + txt.big() + "</p>");
			document.write("<p>Small: " + txt.small() + "</p>");

			document.write("<p>Bold: " + txt.bold() + "</p>");
			document.write("<p>Italic: " + txt.italics() + "</p>");

			document.write("<p>Blink: " + txt.blink() + " (does not work in IE)</p>");
			document.write("<p>Fixed: " + txt.fixed() + "</p>");
			document.write("<p>Strike: " + txt.strike() + "</p>");

			document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>");
			document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>");

			document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>");
			document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>");

			document.write("<p>Subscript: " + txt.sub() + "</p>");
			document.write("<p>Superscript: " + txt.sup() + "</p>");

			document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>");
		</script>
	</body>
</html>
```

Return the position of the first occurrence of a text in a string - indexOf()

```
<html>
	<body>
		<script type="text/javascript">
			var str="Hello world!";
			document.write(str.indexOf("Hello") + "<br />");
			document.write(str.indexOf("World") + "<br />");
			document.write(str.indexOf("world"));
		</script>
	</body>
</html>
```

Search for a text in a string and return the text if found - match()

```
<html>
	<body>
```

```
		<script type="text/javascript">
			var str="Hello world!";
			document.write(str.match("world") + "<br />");
			document.write(str.match("World") + "<br />");
			document.write(str.match("worlld") + "<br />");
			document.write(str.match("world!"));
		</script>
	</body>
</html>
```

Replace characters in a string - replace()

```
<html>
	<body>
		<script type="text/javascript">
			var str="Hello world!";
			document.write(str.match("world") + "<br />");
			document.write(str.match("World") + "<br />");
			document.write(str.match("worlld") + "<br />");
			document.write(str.match("world!"));
		</script>
	</body>
</html>
```

Date Object

Use Date() to return today's date and time

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Date());
		</script>
	</body>
</html>
```

Use getTime() to calculate the years since 1970

```
<html>
	<body>
		<script type="text/javascript">
			var minutes = 1000*60;
			var hours = minutes*60;
			var days = hours*24;
			var years = days*365;
			var d = new Date();
			var t = d.getTime();
			var y = t/years;
			document.write("It's been: " + y + " years since 1970/01/01!");
		</script>
	</body>
</html>
```

Use setFullYear() to set a specific date

```
<html>
	<body>
		<script type="text/javascript">
			var d = new Date();
			d.setFullYear(1992,10,3);
			document.write(d);
		</script>
	</body>
</html>
```

Use toUTCString() to convert today's date (according to UTC) to a string

```
<html>
	<body>
		<script type="text/javascript">
```

```
                var d = new Date();
                document.write (d.toUTCString());
        </script>
    </body>
</html>
```

Use getDay() and an array to write a weekday, and not just a number

```
<html>
    <body>
        <script type="text/javascript">
            var d=new Date();
            var weekday=new Array(7);
            weekday[0]="Sunday";
            weekday[1]="Monday";
            weekday[2]="Tuesday";
            weekday[3]="Wednesday";
            weekday[4]="Thursday";
            weekday[5]="Friday";
            weekday[6]="Saturday";
            document.write("Today it is " + weekday[d.getDay()]);
        </script>
    </body>
</html>
```

Display a clock

```
<html>
    <head>
        <script type="text/javascript">
            function startTime() {
                var today=new Date();
                var h=today.getHours();
                var m=today.getMinutes();
                var s=today.getSeconds();
                // add a zero in front of numbers<10
                m=checkTime(m);
                s=checkTime(s);
                document.getElementById('txt').innerHTML=h+":"+m+":"+s;
                t=setTimeout('startTime()',500);
            }

            function checkTime(i) {
                if (i<10) {
                    i="0" + i;
                }
                return i;
            }
        </script>
    </head>
    <body onload="startTime()">
        <div id="txt"></div>
    </body>
</html>
```

Array Object

Create an array

```
<html>
    <body>
        <script type="text/javascript">
            var mycars = new Array();
            mycars[0] = "Saab";
            mycars[1] = "Volvo";
            mycars[2] = "BMW";
            for (i=0;i<mycars.length;i++) {
                document.write(mycars[i] + "<br />");
            }
        </script>
```

```
	</body>
</html>
```

Use a for...in statement to loop through the elements of an array

```
<html>
	<body>
		<script type="text/javascript">
			var x;
			var mycars = new Array();
			mycars[0] = "Saab";
			mycars[1] = "Volvo";
			mycars[2] = "BMW";
			for (x in mycars) {
				document.write(mycars[x] + "<br />");
			}
		</script>
	</body>
</html>
```

Join two arrays - concat()

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(3);
			arr[0] = "Jani";
			arr[1] = "Tove";
			arr[2] = "Hege";

			var arr2 = new Array(3);
			arr2[0] = "John";
			arr2[1] = "Andy";
			arr2[2] = "Wendy";

			document.write(arr.concat(arr2));
		</script>
	</body>
</html>
```

Put array elements into a string - join()

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(3);
			arr[0] = "Jani";
			arr[1] = "Hege";
			arr[2] = "Stale";

			document.write(arr.join() + "<br />");
			document.write(arr.join("."));
		</script>
	</body>
</html>
```

Literal array - sort()

```
<html>
	<body>
		<script type="text/javascript">
			var arr = new Array(6);
			arr[0] = "Jani";
			arr[1] = "Hege";
			arr[2] = "Stale";
			arr[3] = "Kai Jim";
			arr[4] = "Borge";
			arr[5] = "Tove";

			document.write(arr + "<br />");
			document.write(arr.sort());
```

```
		</script>
	</body>
</html>
```

Numeric array - sort()

```
<html>
	<body>
		<script type="text/javascript">
			function sortNumber(a, b) {
				return a - b;
			}

			var arr = new Array(6);
			arr[0] = "10";
			arr[1] = "5";
			arr[2] = "40";
			arr[3] = "25";
			arr[4] = "1000";
			arr[5] = "1";

			document.write(arr + "<br />");
			document.write(arr.sort(sortNumber));
		</script>
	</body>
</html>
```

More Array object examples in our JavaScript reference.

Boolean Object

Check Boolean value

```
<html>
	<body>
		<script type="text/javascript">
			var b1=new Boolean( 0);
			var b2=new Boolean(1);
			var b3=new Boolean("");
			var b4=new Boolean(null);
			var b5=new Boolean(NaN);
			var b6=new Boolean("false");

			document.write("0 is boolean "+ b1 +"<br />");
			document.write("1 is boolean "+ b2 +"<br />");
			document.write("An empty string is boolean "+ b3 + "<br />");
			document.write("null is boolean "+ b4+ "<br />");
			document.write("NaN is boolean "+ b5 +"<br />");
			document.write("The string 'false' is boolean "+ b6 +"<br />");
		</script>
	</body>
</html>
```

More Boolean object examples in our JavaScript reference.

Math Object

Use round() to round a number

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.round(0.60) + "<br />");
			document.write(Math.round(0.50) + "<br />");
			document.write(Math.round(0.49) + "<br />");
			document.write(Math.round(-4.40) + "<br />");
			document.write(Math.round(-4.60));
```

```
		</script>
	</body>
</html>
```

Use random() to return a random number between 0 and 1

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.random());
		</script>
	</body>
</html>
```

Use max() to return the number with the highest value of two specified numbers

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.max(5,7) + "<br />");
			document.write(Math.max(-3,5) + "<br />");
			document.write(Math.max(-3,-5) + "<br />");
			document.write(Math.max(7.25,7.30));
		</script>
	</body>
</html>
```

Use min() to return the number with the lowest value of two specified numbers

```
<html>
	<body>
		<script type="text/javascript">
			document.write(Math.min(5,7) + "<br />");
			document.write(Math.min(-3,5) + "<br />");
			document.write(Math.min(-3,-5) + "<br />");
			document.write(Math.min(7.25,7.30));
		</script>
	</body>
</html>
```

Convert Celsius to Fahrenheit

```
<html>
	<head>
		<script type="text/javascript">
			function convert(degree) {
				if (degree=="C") {
					F=document.getElementById("c").value * 9 / 5 + 32;
					document.getElementById("f").value=Math.round(F);
				} else {
					C=(document.getElementById("f").value -32) * 5 / 9;
					document.getElementById("c").value=Math.round(C);
				}
			}
		</script>
	</head>
	<body>
		<p></p><b>Insert a number into one of the input fields below:</b></p>
		<form>
			<input id="c" name="c" onkeyup="convert('C')">
			degrees Celsius<br />
			equals<br />
			<input id="f" name="f" onkeyup="convert('F')"> degrees Fahrenheit
		</form>
		<p>Note that the <b>Math.round()</b>
		method is used, so that the result will be returned as an integer.</p>
	</body>
</html>
```

More Math object examples in our JavaScript reference.

Milan Das, SARC

HTML DOM Examples

Anchor Object

Change text, URL, and target attribute of a link

```
<html>
	<head>
		<script type="text/javascript">
			function changeLink() {
				document.getElementById('myAnchor').innerHTML="Visit W3Schools";
				document.getElementById('myAnchor').href="http://www.w3schools.com";
				document.getElementById('myAnchor').target="_blank";
			}
		</script>
	</head>
	<body>
		<a id="myAnchor" href="http://www.microsoft.com">Visit Microsoft</a>
		<input type="button" onclick="changeLink()" value="Change link">
		<p>In this example we change the text and the URL of a hyperlink.
		We also change the target attribute.
		The target attribute is by default set to "_self", which means that the link will open in the same window.
		By setting the target attribute to "_blank", the link will open in a new window.</p>
	</body>
</html>
```

Using focus() and blur()

```
<html>
	<head>
		<style type="text/css">
			a:active {
				color:green;
			}
		</style>
		<script type="text/javascript">
			function getfocus() {
				document.getElementById('myAnchor').focus();
			}
			function losefocus() {
				document.getElementById('myAnchor').blur();
			}
		</script>
	</head>
	<body>
		<a id="myAnchor" href="http://www.learnextglobal.com">Visit W3Schools.com</a>
		<br /><br/>
		<input type="button" onclick="getfocus()" value="Get focus">
		<input type="button" onclick="losefocus()" value="Lose focus">
	</body>
</html>
```

Add an accessKey to a link

```
<html>
	<head>
		<script type="text/javascript">
			function accesskey() {
				document.getElementById('w3').accessKey="w";
				document.getElementById('w3dom').accessKey="d";
			}
		</script>
	</head>
	<body onload="accesskey()">
		<p><a id="w3" href="http://www.w3schools.com/">W3Schools.com</a>
		(Use Alt + w to give focus to the link)</p>
		<p><a id="w3dom" href="http://www.w3schools.com/htmldom/">HTML DOM</a> (Use Alt + d to give
		focus to the link)</p>
	</body>
</html>
```

Milan Das, SARC

Document Object

Write text to the output

```
<html>
	<body>
		<script type="text/javascript">
			document.write("Hello World!");
		</script>
	</body>
</html>
```

Write text with formatting to the output

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<h1>This is a header</h1>");
		</script>
	</body>
</html>
```

Return the title of a document

```
<html>
	<head>
		<title>My title</title>
	</head>
	<body>
		The title of the document is:
		<script type="text/javascript">
			document.write(document.title);
		</script>
	</body>
</html>
```

Return the URL of a document

```
<html>
	<body>
		The URL of this document is:
		<script type="text/javascript">
			document.write(document.URL);
		</script>
	</body>
</html>
```

Return the referrer of a document

```
<html>
	<body>
		<p>The referrer property returns the URL of the document that loaded this document.</p>
		The referrer of this document is:
		<script type="text/javascript">
			document.write(document.referrer);
		</script>
	</body>
</html>
```

Return the domain name of the document's server

```
<html>
	<body>
		The domain name for this document is:
		<script type="text/javascript">
			document.write(document.domain);
		</script>
	</body>
</html>
```

Use getElementById()

```
<html>
	<head>
		<script type="text/javascript">
			function getValue() {
				var x=document.getElementById("myHeader");
				alert(x.innerHTML);
			}
		</script>
	</head>
	<body>
		<h1 id="myHeader" onclick="getValue()">This is a header</h1>
		<p>Click on the header to alert its value</p>
	</body>
</html>
```

Use getElementsByName()

```
<html>
	<head>
		<script type="text/javascript">
			function getElements() {
				var x=document.getElementsByName("myInput");
				alert(x.length);
			}
		</script>
	</head>
	<body>
		<input name="myInput" type="text" size="20" /><br />
		<input name="myInput" type="text" size="20" /><br />
		<input name="myInput" type="text" size="20" /><br />
		<br />
		<input type="button" onclick="getElements()" value="How many elements named 'myInput'?" />
	</body>
</html>
```

Open a new document, specify MIME type and add some text

```
<html>
	<head>
		<script type="text/javascript">
			function createNewDoc() {
				var newDoc=document.open("text/html","replace");
				var txt="<html><body>Learning about the DOM is FUN!</body></html>";
				newDoc.write(txt);
				newDoc.close();
			}
		</script>
	</head>
	<body>
		<input type="button" value="Open and write to a new document" onclick="createNewDoc()">
	</body>
</html>
```

Return the number of anchors in a document

```
<html>
	<body>
		<a name="first">First anchor</a><br />
		<a name="second">Second anchor</a><br />
		<a name="third">Third anchor</a><br />
		<br />
		Number of anchors in this document:
		<script type="text/javascript">
			document.write(document.anchors.length);
		</script>
	</body>
</html>
```

Return the innerHTML of the first anchor in a document

```
<html>
	<body>
		<a name="first">First anchor</a><br />
		<a name="second">Second anchor</a><br />
		<a name="third">Third anchor</a><br />
		<br />
		InnerHTML of the first anchor in this document:
		<script type="text/javascript">
			document.write(document.anchors[0].innerHTML);
		</script>
	</body>
</html>
```

Count the number of forms in a document

```
<html>
	<body>
		<form name="Form1"></form>
		<form name="Form2"></form>
		<form name="Form3"></form>

		<script type="text/javascript">
			document.write("This document contains: " + document.forms.length + " forms.");
		</script>
	</body>
</html>
```

Access an item in a collection

```
<html>
	<body>
		<form id="Form1" name="Form1">
			Your name: <input type="text">
		</form>
		<form id="Form2" name="Form2">
			Your car: <input type="text">
		</form>
		<p>To access an item in a collection you can either use the number or the name of the item:</p>

		<script type="text/javascript">
			document.write("<p>The first form's name is: " + document.forms[0].name + "</p>");
			document.write("<p>The first form's name is: " +
			document.getElementById("Form1").name + "</p>");
		</script>
	</body>
</html>
```

Count the number of images in a document

```
<html>
	<body>
		<img border="0" src="hackanm.gif" width="48" height="48">
		<br />
		<img border="0" src="compman.gif" width="107" height="98">
		<br /><br />

		<script type="text/javascript">
			document.write("This document contains: " + document.images.length + " images.");
		</script>
	</body>
</html>
```

Event Object

Which mouse button was clicked?

```
<html>
	<head>
```

```
		<script type="text/javascript">
			function whichButton(event) {
				if (event.button==2) {
					alert("You clicked the right mouse button!");
				} else {
					alert("You clicked the left mouse button!");
				}
			}
		</script>
	</head>
	<body onmousedown="whichButton(event)">
		<p>Click in the document. An alert box will alert which mouse button you clicked.</p>
	</body>
</html>
```

What are the coordinates of the cursor?

```
<html>
	<head>
		<script type="text/javascript">
			function show_coords(event) {
				x=event.clientX;
				y=event.clientY;
				alert("X coords: " + x + ", Y coords: " + y);
			}
		</script>
	</head>
	<body onmousedown="show_coords(event)">
		<p>Click in the document. An alert box will alert the x and y coordinates of the mouse pointer.</p>
	</body>
</html>
```

What is the unicode of the key pressed?

```
<html>
	<head>
		<script type="text/javascript">
			function whichButton(event) {
				alert(event.keyCode);
			}
		</script>
	</head>
	<body onkeyup="whichButton(event)">
		<p><b>Note:</b> Make sure the right frame has focus when trying this example!</p>
		<p>Press a key on your keyboard. An alert box will alert the unicode of the key pressed.</p>
	</body>
</html>
```

What are the coordinates of the cursor, relative to the screen?

```
<html>
	<head>
		<script type="text/javascript">
			function coordinates(event) {
				x=event.screenX;
				y=event.screenY;
				alert("X=" + x + " Y=" + y);
			}
		</script>
	</head>
	<body onmousedown="coordinates(event)">
		<p>Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor,
		relative to the screen.</p>
	</body>
</html>
```

What are the coordinates of the cursor?

```
<html>
	<head>
		<script type="text/javascript">
```

```
                function coordinates(event) {
                        x=event.x;
                        y=event.y;
                        alert("X=" + x + " Y=" + y);
                }
        </script>
</head>
<body onmousedown="coordinates(event)">
        <p>Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor.</p>
</body>
</html>
```

Was the shift key pressed?

```
<html>
<head>
        <script type="text/javascript">
                function isKeyPressed(event) {
                        if (event.shiftKey==1) {
                                alert("The shift key was pressed!");
                        } else {
                                alert("The shift key was NOT pressed!");
                        }
                }
        </script>
</head>
<body onmousedown="isKeyPressed(event)">
        <p>Click somewhere in the document. An alert box will tell you if you pressed the shift key or not.</p>
</body>
</html>
```

Which element was clicked?

```
<html>
<head>
        <script type="text/javascript">
                function whichElement(e) {
                        var targ;
                        if (!e) {
                                var e=window.event;
                        }
                        if (e.target) {
                                targ=e.target;
                        } else if (e.srcElement) {
                                targ=e.srcElement;
                        }
                        if (targ.nodeType==3) {           // defeat Safari bug
                                targ = targ.parentNode;
                        }
                        var tname;
                        tname=targ.tagName;
                        alert("You clicked on a " + tname + " element.");
                }
        </script>
</head>
<body onmousedown="whichElement(event)">
        <p>Click somewhere in the document.
        An alert box will alert the tag name of the element you clicked on.</p>

        <h3>This is a header</h3>
        <p>This is a paragraph</p>
        <img border="0" src="ball16.gif" width="29" height="28" alt="Ball">
</body>
</html>
```

Which eventype occured?

```
<html>
<head>
        <script type="text/javascript">
                function getEventType(event) {
                        alert(event.type);
```

```
                }
        </script>
</head>
<body onmousedown="getEventType(event)">
        <p>Click in the document.
        An alert box will tell what type of event that was triggered.</p>
</body>
</html>
```

Form and Form Input Objects

View and change the action URL of a form

```
<html>
        <head>
                <script type="text/javascript">
                        function changeAction() {
                                var x=document.getElementById("myForm");
                                alert("Original action: " + x.action);
                                x.action="default.asp";
                                alert("New action: " + x.action);
                                x.submit();
                        }
                </script>
        </head>
        <body>
                <form id="myForm" action="js_examples.asp">
                        Name: <input type="text" value="Mickey Mouse" />
                        <input type="button" onclick="changeAction()"
                        value="Change action attribute and submit form" />
                </form>
        </body>
</html>
```

View the method that is to be used when sending form data

```
<html>
        <head>
                <script type="text/javascript">
                        function showMethod() {
                                var x=document.getElementById("myForm");
                                alert(x.method);
                        }
                </script>
        </head>
        <body>
                <form id="myForm" method="post">
                        Name: <input type="text" size="20" value="Mickey Mouse" />
                        <input type="button" onclick="showMethod()" value="Show method" />
                </form>
        </body>
</html>
```

Alert id, type, and value of a Button object + disable button

```
<html>
        <head>
                <script type="text/javascript">
                        function alertId() {
                                var txt="Id: " + document.getElementById("myButton").id;
                                txt=txt + ", type: " + document.getElementById("myButton").type;
                                txt=txt + ", value: " + document.getElementById("myButton").value;
                                alert(txt);
                                document.getElementById("myButton").disabled=true;
                        }
                </script>
        </head>
        <body>
                <form>
                        <input type="button" value="Click me!" id="myButton" onclick="alertId()" />
```

```
		</form>
	</body>
</html>
```

Check and uncheck a checkbox

```
<html>
	<head>
		<script type="text/javascript">
			function check() {
				document.getElementById("myCheck").checked=true;
			}
			function uncheck() {
				document.getElementById("myCheck").checked=false;
			}
		</script>
	</head>
	<body>
		<form>
			<input type="checkbox" id="myCheck" />
			<input type="button" onclick="check()" value="Check Checkbox" />
			<input type="button" onclick="uncheck()" value="Uncheck Checkbox" />
		</form>
	</body>
</html>
```

Checkboxes in a form

```
<html>
	<head>
		<script type="text/javascript">
			function createOrder() {
				coffee=document.forms[0].coffee;
				txt="";
				for (i=0;i<coffee.length;++ i) {
					if (coffee[i].checked) {
						txt=txt + coffee[i].value + " ";
					}
				}
				document.getElementById("order").value="You ordered a coffee with " + txt;
			}
		</script>
	</head>
	<body>
		<p>How would you like your coffee?</p>
		<form>
			<input type="checkbox" name="coffee" value="cream">With cream<br />
			<input type="checkbox" name="coffee" value="sugar">With sugar<br />
			<br />
			<input type="button" onclick="createOrder()" value="Send order">
			<br /><br />
			<input type="text" id="order" size="50">
		</form>
	</body>
</html>
```

Checkbox - If the user clicks in a checkbox, the content of the text fields are converted to uppercase.

```
<html>
	<head>
		<script type="text/javascript">
			function convertToUcase() {
				document.getElementById("fname").value=
				document.getElementById("fname").value.toUpperCase();
				document.getElementById("lname").value=
				document.getElementById("lname").value.toUpperCase();
			}
		</script>
	</head>
	<body>
		<form name="form1">
```

```
                        First name: <input type="text" id="fname" size="20" />
                        <br /><br />
                        Last name: <input type="text" id="lname" size="20" />
                        <br /><br />
                        Convert to upper case
                        <input type="checkBox" onclick="if (this.checked) {convertToUcase()}">
                </form>
        </body>
</html>
```

Radio buttons

```
<html>
        <head>
                <script type="text/javascript">
                        function check(browser) {
                                document.getElementById("answer").value=browser;
                        }
                </script>
        </head>
        <body>
                <p>What's your favorite browser?</p>
                <form>
                <input type="radio" name="browser" onclick="check(this.value)" value="Internet Explorer">Internet
                Explorer<br />
                <input type="radio" name="browser" onclick="check(this.value)" value="Firefox">Firefox<br />
                <input type="radio" name="browser" onclick="check(this.value)" value="Netscape">Netscape<br />
                <input type="radio" name="browser" onclick="check(this.value)" value="Opera">Opera<br />
                <br />
                Your favorite browser is: <input type="text" id="answer" size="20">
                </form>
        </body>
</html>
```

Reset a form

```
<html>
        <head>
                <script type="text/javascript">
                        function formReset() {
                                document.getElementById("myForm").reset();
                        }
                </script>
        </head>
        <body>
                <p>Enter some text in the text fields below, and then press the "Reset" button to reset the form.</p>
                <form id="myForm">
                        Name: <input type="text" size="20"><br />
                        Age: <input type="text" size="20"><br />
                        <br />
                        <input type="button" onclick="formReset()" value="Reset">
                </form>
        </body>
</html>
```

Submit a form

```
<html>
        <head>
                <script type="text/javascript">
                        function formSubmit() {
                                document.getElementById("myForm").submit();
                        }
                </script>
        </head>
        <body>
                <p>Enter some text in the text fields below,
                and then press the "Submit" button to submit the form.</p>

                <form id="myForm" action="js_form_action.asp" method="get">
                        Firstname: <input type="text" name="firstname" size="20"><br />
```

```
                Lastname: <input type="text" name="lastname" size="20"><br />
                <br />
                <input type="button" onclick="formSubmit()" value="Submit">
        </form>
    </body>
</html>
```

Form validation

```
<html>
    <head>
        <script type="text/javascript">
            function validate() {
                var at=document.getElementById("email").value.indexOf("@");
                var age=document.getElementById("age").value;
                var fname=document.getElementById("fname").value;
                submitOK="true";

                if (fname.length>10) {
                    alert("The name must be less than 10 characters");
                    submitOK="false";
                }
                if (isNaN(age)||age<1||age>100) {
                    alert("The age must be a number between 1 and 100");
                    submitOK="false";
                }
                if (at==-1)  {
                    alert("Not a valid e-mail!");
                    submitOK="false";
                }
                if (submitOK=="false") {
                    return false;
                }
            }
        </script>
    </head>
    <body>
        <form action="tryjs_submitpage.htm" onsubmit="return validate()">
            Name (max 10 chararcters): <input type="text" id="fname" size="20"><br />
            Age (from 1 to 100): <input type="text" id="age" size="20"><br />
            E-mail: <input type="text" id="email" size="20"><br />
            <br />
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

Set focus to an input field when the page loads

```
<html>
    <head>
        <script type="text/javascript">
            function setFocus() {
                document.getElementById("fname").focus();
            }
        </script>
    </head>
    <body onload="setFocus()">
        <form>
            Name: <input type="text" id="fname" size="30"><br />
            Age: <input type="text" id="age" size="30">
        </form>
    </body>
</html>
```

Select the content of an input field

```
<html>
    <head>
        <script type="text/javascript">
            function selText() {
```

```
                document.getElementById("myText").select();
            }
        </script>
    </head>
    <body>
        <form>
            <input size="25" type="text" id="myText" value="A cat played with a ball">
            <input type="button" value="Select text" onclick="selText()">
        </form>
    </body>
</html>
```

Dropdown list in a form

```
<html>
    <head>
        <script type="text/javascript">
            function favBrowser() {
                var mylist=document.getElementById("myList");
                document.getElementById("favorite").value=mylist.options[mylist.selectedIndex].text;
            }
        </script>
    </head>
    <body>
        <form>
            Select your favorite browser:
            <select id="myList" onchange="favBrowser()">
                <option>Internet Explorer</option>
                <option>Netscape</option>
                <option>Opera</option>
            </select>
            <p>Your favorite browser is: <input type="text" id="favorite" size="20"></p>
        </form>
    </body>
</html>
```

Another dropdown list

```
<html>
    <head>
        <script type="text/javascript">
            function moveNumbers() {
                var no=document.getElementById("no");
                var option=no.options[no.selectedIndex].text;
                var txt=document.getElementById("result").value;
                txt=txt + option;
                document.getElementById("result").value=txt;
            }
        </script>
    </head>
    <body>
        <form>
            Select numbers:<br />
            <select id="no">
                <option>0</option>
                <option>1</option>
                <option>2</option>
                <option>3</option>
                <option>4</option>
                <option>5</option>
                <option>6</option>
                <option>7</option>
                <option>8</option>
                <option>9</option>
            </select>
            <input type="button" onclick="moveNumbers()" value="-->">
            <input type="text" id="result" size="20">
        </form>
    </body>
</html>
```

A dropdown menu

```
<html>
	<head>
		<script type="text/javascript">
			function go() {
				window.location=document.getElementById("menu").value;
			}
		</script>
	</head>
	<body>
		<form>
			<select id="menu" onchange="go()">
				<option>--Select a page--</option>
				<option value="http://www.w3schools.com">W3Schools</option>
				<option value="http://www.microsoft.com">Microsoft</option>
				<option value="http://www.altavista.com">AltaVista</option>
			</select>
		</form>
	</body>
</html>
```

Jump to the next field when the current field's maxlength has been reached

```
<html>
	<head>
		<script type="text/javascript">
			function checkLen(x,y) {
				if (y.length==x.maxLength) {
					var next=x.tabIndex;
					if (next<document.getElementById("myForm").length) {
						document.getElementById("myForm").elements[next].focus();
					}
				}
			}
		</script>
	</head>
	<body>
		<p>This script automatically jumps to the next field when a field's maxlength has been reached:</p>
		<form id="myForm">
			<input size="3" tabindex="1" maxlength="3" onkeyup="checkLen(this,this.value)">
			<input size="2" tabindex="2" maxlength="2" onkeyup="checkLen(this,this.value)">
			<input size="3" tabindex="3" maxlength="3" onkeyup="checkLen(this,this.value)">
		</form>
	</body>
</html>
```

Add accessKeys to form fields

```
<html>
	<head>
		<script type="text/javascript">
			function access() {
				document.getElementById('myName').accessKey="n";
				document.getElementById('myPwd').accessKey="p";
				document.getElementById('ie').accessKey="i";
				document.getElementById('fx').accessKey="f";
				document.getElementById('myButton').accessKey="b";
			}
		</script>
	</head>
	<body onload="access()">
		<form>
			Name: <input id="myName" type="text" />
			<br />
			Password: <input id="myPwd" type="password" />
			<br /><br />
			Select your favorite browser:
			<br />
			<input type="radio" name="browser" id="ie" value="Internet Explorer">Internet Explorer
			<br />
			<input type="radio" name="browser" id="fx" value="Firefox">Firefox
			<br /><br />
```

```
                <input type="button" value="Click me!" id="myButton" />
            </form>
            <p>(Use Alt + <i>accesskey</i> to give focus to the different form fields.)</p>
        </body>
</html>
```

Frame, Frameset, and IFrame Objects

Resizable and not resizable frames

```
<html>
        <frameset cols="50%,50%">
            <frame id="leftFrame" src="frame_noresize.htm">
            <frame id="rightFrame" src="frame_a.htm">
        </frameset>
</html>
```

Frames with and without scrollbars

```
<html>
        <frameset cols="50%,50%">
            <frame id="leftFrame" src="frame_scroll.htm">
            <frame id="rightFrame" src="frame_a.htm">
        </frameset>
</html>
```

Change the source / URL of two frames

```
<html>
        <frameset id="myFrameset" cols="50%,50%">
            <frame id="leftFrame" src="frame_src.htm">
            <frame id="rightFrame" src="frame_a.htm">
        </frameset>
</html>
```

Break out of a frame

```
<html>
        <head>
            <script type="text/javascript">
                function breakout() {
                    if (window.top!=window.self) {
                        window.top.location="tryjs_breakout.htm";
                    }
                }
            </script>
        </head>
        <body>
            <input type="button" onclick="breakout()" value="Break out of frame!">
        </body>
</html>
```

Update two iframes

```
<html>
        <head>
            <script type="text/javascript">
                function changeSource() {
                    document.getElementById("frame1").src="frame_c.htm";
                    document.getElementById("frame2").src="frame_d.htm";
                }
            </script>
        </head>
        <body>
            <iframe src="frame_a.htm" id="frame1"></iframe>
            <iframe src="frame_b.htm" id="frame2"></iframe>
            <br /><br />
            <input type="button" onclick="changeSource()" value="Change source of the two iframes">
        </body>
```

```
</html>
```

Image Object

Change the height and width of an image

```
<html>
	<head>
		<script type="text/javascript">
			function changeSize() {
				document.getElementById("compman").height="250";
				document.getElementById("compman").width="300";
			}
		</script>
	</head>
	<body>
		<img id="compman" src="compman.gif" width="107" height="98" />
		<br /><br />
		<input type="button" onclick="changeSize()" value="Change height and width of image">
	</body>
</html>
```

Change the src of an image

```
<html>
	<head>
		<script type="text/javascript">
			function changeSrc() {
				document.getElementById("myImage").src="hackanm.gif";
			}
		</script>
	</head>
	<body>
		<img id="myImage" src="compman.gif" width="107" height="98" />
		<br /><br />
		<input type="button" onclick="changeSrc()" value="Change image">
	</body>
</html>
```

Location Object

Send the client to a new location / URL

```
<html>
	<head>
		<script type="text/javascript">
			function currLocation() {
				alert(window.location);
			}
			function newLocation() {
				window.location="http://www.w3schools.com";
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="currLocation()" value="Show current URL">
		<input type="button" onclick="newLocation()" value="Change URL">
	</body>
</html>
```

Reload a page

```
<html>
	<head>
		<script type="text/javascript">
			function reloadPage() {
				window.location.reload();
			}
```

```
		</script>
	</head>
	<body>
		<input type="button" value="Reload page" onclick="reloadPage()" />
	</body>
</html>
```

Break out of a frame

```
<html>
	<head>
		<script type="text/javascript">
			function breakout() {
				if (window.top!=window.self) {
					window.top.location="tryjs_breakout.htm";
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="breakout()" value="Break out of frame!">
	</body>
</html>
```

Anchors array - This example opens two windows. The first window contains four buttons and the second window defines four anchors from 0 to 3. When a button is clicked in the first window, the onclick event handler goes to the specified anchor in the second window.

```
<html>
	<head>
		<script type="text/javascript">
			function linkTo(y) {
				var x=window.open("anchors.htm","","scrollbars=yes,width=250,height=200");
				x.location.hash=y;
			}
		</script>
	</head>
	<body>
		<h3>Links and Anchors</h3>
		<p>Click on a button to display that anchor in a second window!</p>
		<input type="button" value="0" onclick="linkTo(0)">
		<input type="button" value="1" onclick="linkTo(1)">
		<input type="button" value="2" onclick="linkTo(2)">
		<input type="button" value="3" onclick="linkTo(3)">
	</body>
</html>
```

Navigator Object

Detect the visitor's browser and browser version

```
<html>
	<body>
		<script type="text/javascript">
			var browser=navigator.appName;
			var b_version=navigator.appVersion;
			var version=parseFloat(b_version);
			document.write("Browser name: "+ browser);
			document.write("<br />");
			document.write("Browser version: "+ version);
		</script>
	</body>
</html>
```

More details about the visitor's browser

```
<html>
	<body>
		<script type="text/javascript">
			document.write("<p>Browser: ");
```

```
                document.write(navigator.appName + "</p>");

                document.write("<p>Browserversion: ");
                document.write(navigator.appVersion + "</p>");

                document.write("<p>Code: ");
                document.write(navigator.appCodeName + "</p>");

                document.write("<p>Platform: ");
                document.write(navigator.platform + "</p>");

                document.write("<p>Cookies enabled: ");
                document.write(navigator.cookieEnabled + "</p>");

                document.write("<p>Browser's user agent header: ");
                document.write(navigator.userAgent + "</p>");
            </script>
        </body>
</html>
```

All details about the visitor's browser

```
<html>
        <body>
            <script type="text/javascript">
                var x = navigator;
                document.write("CodeName=" + x.appCodeName);
                document.write("<br />");
                document.write("MinorVersion=" + x.appMinorVersion);
                document.write("<br />");
                document.write("Name=" + x.appName);
                document.write("<br />");
                document.write("Version=" + x.appVersion);
                document.write("<br />");
                document.write("CookieEnabled=" + x.cookieEnabled);
                document.write("<br />");
                document.write("CPUClass=" + x.cpuClass);
                document.write("<br />");
                document.write("OnLine=" + x.onLine);
                document.write("<br />");
                document.write("Platform=" + x.platform);
                document.write("<br />");
                document.write("UA=" + x.userAgent);
                document.write("<br />");
                document.write("BrowserLanguage=" + x.browserLanguage);
                document.write("<br />");
                document.write("SystemLanguage=" + x.systemLanguage);
                document.write("<br />");
                document.write("UserLanguage=" + x.userLanguage);
            </script>
        </body>
</html>
```

Alert user, depending on browser

```
<html>
        <head>
            <script type="text/javascript">
                function detectBrowser() {
                    var browser=navigator.appName;
                    var b_version=navigator.appVersion;
                    var version=parseFloat(b_version);
                    if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") && (version>=4)){
                        alert("Your browser is good enough!");
                    } else {
                        alert("It's time to upgrade your browser!");
                    }
                }
            </script>
        </head>
        <body onload="detectBrowser()">
        </body>
```

```
</html>
```

Option and Select Objects

Disable and enable a dropdown list

```
<html>
	<head>
		<script type="text/javascript">
			function disable() {
				document.getElementById("mySelect").disabled=true;
			}
			function enable() {
				document.getElementById("mySelect").disabled=false;
			}
		</script>
	</head>
	<body>
		<form>
			<select id="mySelect">
				<option>Apple</option>
				<option>Pear</option>
				<option>Banana</option>
				<option>Orange</option>
			</select>
			<br /><br />
			<input type="button" onclick="disable()" value="Disable list">
			<input type="button" onclick="enable()" value="Enable list">
		</form>
	</body>
</html>
```

Get the id of the form that contains the dropdown list

```
<html>
	<body>
		<form id="myForm">
			<select id="mySelect">
				<option>Apple</option>
				<option>Pear</option>
				<option>Banana</option>
				<option>Orange</option>
			</select>
		</form>
		<p>The id of the form is:
		<script type="text/javascript">
			document.write(document.getElementById("mySelect").form.id);
		</script>
		</p>
	</body>
</html>
```

Get the number of options in the dropdown list

```
<html>
	<head>
		<script type="text/javascript">
			function getLength() {
				alert(document.getElementById("mySelect").length);
			}
		</script>
	</head>
	<body>
		<form>
			<select id="mySelect">
				<option>Apple</option>
				<option>Pear</option>
				<option>Banana</option>
				<option>Orange</option>
```

```
                </select>
                <input type="button" onclick="getLength()" value="How many options in the list?">
        </form>
    </body>
</html>
```

Turn the dropdown list into a multiline list

```
<html>
    <head>
        <script type="text/javascript">
            function changeSize() {
                document.getElementById("mySelect").size=4;
            }
        </script>
    </head>
    <body>
        <form>
            <select id="mySelect">
                <option>Apple</option>
                <option>Banana</option>
                <option>Orange</option>
                <option>Melon</option>
            </select>
            <input type="button" onclick="changeSize()" value="Change size">
        </form>
    </body>
</html>
```

Select multiple options in a dropdown list

```
<html>
    <head>
        <script type="text/javascript">
            function selectMultiple() {
                document.getElementById("mySelect").multiple=true;
            }
        </script>
    </head>
    <body>
        <form>
            <select id="mySelect" size="4">
                <option>Apple</option>
                <option>Pear</option>
                <option>Banana</option>
                <option>Orange</option>
            </select>
            <input type="button" onclick="selectMultiple()" value="Select multiple">
        </form>
        <p>Before you click on the "Select multiple" button,
        try to select more than one option (by holding down the Shift or Ctrl key).
        Then click on the "Select multiple" button and try again.</p>
    </body>
</html>
```

Alert the selected option in a dropdown list

```
<html>
    <head>
        <script type="text/javascript">
            function getOptions() {
                var x=document.getElementById("mySelect");
                txt="All options: "
                for (i=0;i<x.length;i++) {
                    txt=txt + "\n" + x.options[i].text;
                }
                alert(txt);
            }
        </script>
    </head>
    <body>
```

```
		<form>
			Select your favorite fruit:
			<select id="mySelect">
				<option>Apple</option>
				<option>Orange</option>
				<option>Pineapple</option>
				<option>Banana</option>
			</select>
			<br /><br />
			<input type="button" onclick="getOptions()" value="Output all options">
		</form>
	</body>
</html>
```

Alert the index of the selected option in a dropdown list

```
<html>
	<head>
		<script type="text/javascript">
			function getIndex() {
				var x=document.getElementById("mySelect");
				alert(x.selectedIndex);
			}
		</script>
	</head>
	<body>
		<form>
			Select your favorite fruit:
			<select id="mySelect">
				<option>Apple</option>
				<option>Orange</option>
				<option>Pineapple</option>
				<option>Banana</option>
			</select>
			<br /><br />
			<input type="button" onclick="getIndex()" value="Alert index of selected option">
		</form>
	</body>
</html>
```

Change the text of the selected option

```
<html>
	<head>
		<script type="text/javascript">
			function changeText() {
				var x=document.getElementById("mySelect");
				x.options[x.selectedIndex].text="Melon";
			}
		</script>
	</head>
	<body>
		<form>
			Select your favorite fruit:
			<select id="mySelect">
				<option>Apple</option>
				<option>Orange</option>
				<option>Pineapple</option>
				<option>Banana</option>
			</select>
			<br /><br />
			<input type="button" onclick="changeText()" value="Set text of selected option">
		</form>
	</body>
</html>
```

Remove options from a dropdown list

```
<html>
	<head>
		<script type="text/javascript">
			function removeOption() {
```

```
                        var x=document.getElementById("mySelect");
                        x.remove(x.selectedIndex);
                }
        </script>
</head>
<body>
        <form>
                <select id="mySelect">
                        <option>Apple</option>
                        <option>Pear</option>
                        <option>Banana</option>
                        <option>Orange</option>
                </select>
                <input type="button" onclick="removeOption()" value="Remove selected option">
        </form>
</body>
</html>
```

Screen Object

Detect details about the client's screen

```
<html>
        <body>
                <script type="text/javascript">
                        document.write("Screen resolution: ");
                        document.write(screen.width + "*" + screen.height);
                        document.write("<br />");
                        document.write("Available view area: ");
                        document.write(screen.availWidth + "*" + screen.availHeight);
                        document.write("<br />");
                        document.write("Color depth: ");
                        document.write(screen.colorDepth);
                        document.write("<br />");
                        document.write("Buffer depth: ");
                        document.write(screen.bufferDepth);
                        document.write("<br />");
                        document.write("DeviceXDPI: ");
                        document.write(screen.deviceXDPI);
                        document.write("<br />");
                        document.write("DeviceYDPI: ");
                        document.write(screen.deviceYDPI);
                        document.write("<br />");
                        document.write("LogicalXDPI: ");
                        document.write(screen.logicalXDPI);
                        document.write("<br />");
                        document.write("LogicalYDPI: ");
                        document.write(screen.logicalYDPI);
                        document.write("<br />");
                        document.write("FontSmoothingEnabled: ");
                        document.write(screen.fontSmoothingEnabled);
                        document.write("<br />");
                        document.write("PixelDepth: ");
                        document.write(screen.pixelDepth);
                        document.write("<br />");
                        document.write("UpdateInterval: ");
                        document.write(screen.updateInterval);
                        document.write("<br />");
                </script>
        </body>
</html>
```

Table, TableHeader, TableRow, TableData Objects

Change the width of a table border

```
<html>
        <head>
```

```
		<script type="text/javascript">
			function changeBorder() {
				document.getElementById('myTable').border="10";
			}
		</script>
	</head>
	<body>
		<table border="1" id="myTable">
			<tr>
				<td>100</td>
				<td>200</td>
			</tr>
			<tr>
				<td>300</td>
				<td>400</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="changeBorder()" value="Change Border">
	</body>
</html>
```

Change the cellPadding and cellSpacing of a table

```
<html>
	<head>
		<script type="text/javascript">
			function padding() {
				document.getElementById('myTable').cellPadding="15";
			}
			function spacing() {
				document.getElementById('myTable').cellSpacing="15";
			}
		</script>
	</head>
	<body>
		<table id="myTable" border="1">
			<tr>
				<td>100</td>
				<td>200</td>
			</tr>
			<tr>
				<td>300</td>
				<td>400</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="padding()" value="Change Cellpadding">
		<input type="button" onclick="spacing()" value="Change Cellspacing">
	</body>
</html>
```

Specify frames of a table

```
<html>
	<head>
		<script type="text/javascript">
			function aboveFrames() {
				document.getElementById('myTable').frame="above";
			}
			function belowFrames() {
				document.getElementById('myTable').frame="below";
			}
		</script>
	</head>
	<body>
		<table id="myTable">
			<tr>
				<td>100</td>
				<td>200</td>
			</tr>
			<tr>
```

```
                        <td>300</td>
                        <td>400</td>
                    </tr>
                </table>
                <br />
                <input type="button" onclick="aboveFrames()" value="Show above frames">
                <input type="button" onclick="belowFrames()" value="Show below frames">
        </body>
</html>
```

Specify rules for a table

```
<html>
        <head>
                <script type="text/javascript">
                        function rowRules() {
                                document.getElementById('myTable').rules="rows";
                        }
                        function colRules() {
                                document.getElementById('myTable').rules="cols";
                        }
                </script>
        </head>
        <body>
                <table id="myTable" border="1">
                        <tr>
                                <td>Row1 cell1</td>
                                <td>Row1 cell2</td>
                        </tr>
                        <tr>
                                <td>Row2 cell1</td>
                                <td>Row2 cell2</td>
                        </tr>
                        <tr>
                                <td>Row3 cell1</td>
                                <td>Row3 cell2</td>
                        </tr>
                </table>
                <br />
                <input type="button" onclick="rowRules()" value="Show only row borders">
                <input type="button" onclick="colRules()" value="Show only col borders">
        </body>
</html>
```

InnerHTML of a row

```
<html>
        <head>
                <script type="text/javascript">
                        function showRow() {
                                alert(document.getElementById('myTable').rows[0].innerHTML);
                        }
                </script>
        </head>
        <body>
                <table id="myTable" border="1">
                        <tr>
                                <td>Row1 cell1</td>
                                <td>Row1 cell2</td>
                        </tr>
                        <tr>
                                <td>Row2 cell1</td>
                                <td>Row2 cell2</td>
                        </tr>
                        <tr>
                                <td>Row3 cell1</td>
                                <td>Row3 cell2</td>
                        </tr>
                </table>
                <br />
                <input type="button" onclick="showRow()" value="Show innerHTML of first row">
        </body>
```

```
</html>
```

InnerHTML of a cell

```
<html>
	<head>
		<script type="text/javascript">
			function cell() {
				var x=document.getElementById('myTable').rows[0].cells;
				alert(x[0].innerHTML);
			}
		</script>
	</head>
	<body>
		<table id="myTable" border="1">
			<tr>
				<td>cell 1</td>
				<td>cell 2</td>
			</tr>
			<tr>
				<td>cell 3</td>
				<td>cell 4</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="cell()" value="Alert first cell">
	</body>
</html>
```

Create a caption for a table

```
<html>
	<head>
		<script type="text/javascript">
			function createCaption() {
				var x=document.getElementById('myTable').createCaption();
				x.innerHTML="My table caption";
			}
		</script>
	</head>
	<body>
		<table id="myTable" border="1">
			<tr>
				<td>Row1 cell1</td>
				<td>Row1 cell2</td>
			</tr>
			<tr>
				<td>Row2 cell1</td>
				<td>Row2 cell2</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="createCaption()" value="Create caption">
	</body>
</html>
```

Delete rows in a table

```
<html>
	<head>
		<script type="text/javascript">
			function deleteRow(r) {
				var i=r.parentNode.parentNode.rowIndex;
				document.getElementById('myTable').deleteRow(i);
			}
		</script>
	</head>
	<body>
		<table id="myTable" border="1">
			<tr>
				<td>Row 1</td>
```

```
                        <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
                    </tr>
                    <tr>
                        <td>Row 2</td>
                        <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
                    </tr>
                    <tr>
                        <td>Row 3</td>
                        <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
                    </tr>
                </table>
        </body>
</html>
```

Add rows to a table

```
<html>
        <head>
                <script type="text/javascript">
                        function insRow() {
                                var x=document.getElementById('myTable').insertRow(0);
                                var y=x.insertCell(0);
                                var z=x.insertCell(1);
                                y.innerHTML="NEW CELL1";
                                z.innerHTML="NEW CELL2";
                        }
                </script>
        </head>
        <body>
                <table id="myTable" border="1">
                        <tr>
                                <td>Row1 cell1</td>
                                <td>Row1 cell2</td>
                        </tr>
                        <tr>
                                <td>Row2 cell1</td>
                                <td>Row2 cell2</td>
                        </tr>
                        <tr>
                                <td>Row3 cell1</td>
                                <td>Row3 cell2</td>
                        </tr>
                </table>
                <br />
                <input type="button" onclick="insRow()" value="Insert row">
        </body>
</html>
```

Add cells to a table row

```
<html>
        <head>
                <script type="text/javascript">
                        function insCell() {
                                var x=document.getElementById('tr2').insertCell(0);
                                x.innerHTML="John";
                        }
                </script>
        </head>
        <body>
                <table border="1">
                        <tr id="tr1">
                                <th>Firstname</th>
                                <th>Lastname</th>
                        </tr>
                        <tr id="tr2">
                                <td>Peter</td>
                                <td>Griffin</td>
                        </tr>
                </table>
                <br />
                <input type="button" onclick="insCell()" value="Insert cell">
```

```
	</body>
</html>
```

Align the cell content in a table row

```
<html>
	<head>
		<script type="text/javascript">
			function leftAlign() {
				document.getElementById('header').align="left";
			}
		</script>
	</head>
	<body>
		<table width="100%" border="1">
			<tr id="header">
				<th>Firstname</th>
				<th>Lastname</th>
			</tr>
			<tr>
				<td>Peter</td>
				<td>Griffin</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="leftAlign()" value="Left-align table row" />
	</body>
</html>
```

Vertical align the cell content in a table row

```
<html>
	<head>
		<script type="text/javascript">
			function topAlign() {
				document.getElementById('tr2').vAlign="top";
			}
		</script>
	</head>
	<body>

		<table width="50%" border="1">
			<tr id="tr1">
				<th>Firstname</th>
				<th>Lastname</th>
				<th>Text</th>
			</tr>
			<tr id="tr2">
				<td>Peter</td>
				<td>Griffin</td>
				<td>Hello my name is Peter Griffin.
					I need a long text for this example.
					I need a long text for this example.</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="topAlign()" value="Top-align table row" />
	</body>
</html>
```

Align the cell content in a single cell

```
<html>
	<head>
		<script type="text/javascript">
			function alignCell() {
				document.getElementById('td1').align="right";
			}
		</script>
	</head>
	<body>
```

```
		<table border="1">
			<tr>
				<th>Firstname</th>
				<th>Lastname</th>
			</tr>
			<tr>
				<td id="td1">Peter</td>
				<td id="td2">Griffin</td>
			</tr>
		</table>
		<br />
		<input type="button" onclick="alignCell()" value="Align cell" />
	</body>
</html>
```

Vertical align the cell content in a single cell

```
<html>
	<head>
		<script type="text/javascript">
			function valignCell() {
				var x=document.getElementById('myTable').rows[0].cells;
				x[0].vAlign="bottom";
			}
		</script>
	</head>
	<body>
		<table id="myTable" border="1" height="70%">
			<tr>
				<td>First cell</td>
				<td>Second cell</td>
			</tr>
			<tr>
				<td>Third cell</td>
				<td>Fourth cell</td>
			</tr>
		</table>
		<form>
			<input type="button" onclick="valignCell()" value="Vertical align cell content">
		</form>
	</body>
</html>
```

Change the content of a table cell

```
<html>
	<head>
		<script type="text/javascript">
		function changeContent()
		{
		var x=document.getElementById('myTable').rows[0].cells;
		x[0].innerHTML="NEW CONTENT";
		}
		</script>
	</head>
	<body>
		<table id="myTable" border="1">
			<tr>
				<td>Row1 cell1</td>
				<td>Row1 cell2</td>
			</tr>
			<tr>
				<td>Row2 cell1</td>
				<td>Row2 cell2</td>
			</tr>
			<tr>
				<td>Row3 cell1</td>
				<td>Row3 cell2</td>
			</tr>
		</table>
		<form>
			<input type="button" onclick="changeContent()" value="Change content">
```

```
		</form>
	</body>
</html>
```

Change the colspan of a table row

```
<html>
	<head>
		<script type="text/javascript">
			function setColSpan() {
				var x=document.getElementById('myTable').rows[0].cells;
				x[0].colSpan="2";
				x[1].colSpan="6";
			}
		</script>
	</head>

	<body>
		<table id="myTable" border="1">
			<tr>
				<td colspan="4">cell 1</td>
				<td colspan="4">cell 2</td>
			</tr>
			<tr>
				<td>cell 3</td>
				<td>cell 4</td>
				<td>cell 5</td>
				<td>cell 6</td>
				<td>cell 7</td>
				<td>cell 8</td>
				<td>cell 9</td>
				<td>cell 10</td>
			</tr>
		</table>
		<form>
			<input type="button" onclick="setColSpan()" value="Change colspan">
		</form>
	</body>
</html>
```

Window Object

Display an alert box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("I am an alert box!!");
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
</html>
```

Alert box with line-breaks

```
<html>
	<head>
		<script type="text/javascript">
			function disp_alert() {
				alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!");
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_alert()" value="Display alert box" />
	</body>
```

```
</html>
```

Display a confirm box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_confirm() {
				var r=confirm("Press a button");
				if (r==true) {
					document.write("You pressed OK!");
				} else {
					document.write("You pressed Cancel!");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_confirm()" value="Display a confirm box" />
	</body>
</html>
```

Display a prompt box

```
<html>
	<head>
		<script type="text/javascript">
			function disp_prompt() {
				var name=prompt("Please enter your name","Harry Potter");
				if (name!=null && name!="") {
					document.write("Hello " + name + "! How are you today?");
				}
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="disp_prompt()" value="Display a prompt box" />
	</body>
</html>
```

Open a new window when clicking on a button

```
<html>
	<head>
		<script type="text/javascript">
			function open_win() {
				window.open("http://www.w3schools.com");
			}
		</script>
	</head>
	<body>
		<form>
			<input type=button value="Open Window" onclick="open_win()">
		</form>
	</body>
</html>
```

Open a new window and control its appearance

```
<html>
	<head>
		<script type="text/javascript">
			function open_win() {
				window.open("http://www.w3schools.com","_blank","toolbar=yes, location=yes,
				directories=no, status=no, menubar=yes, scrollbars=yes, resizable=no,
				copyhistory=yes, width=400, height=400");
			}
		</script>
	</head>
	<body>
		<form>
```

```
                <input type="button" value="Open Window" onclick="open_win()">
            </form>
        </body>
</html>
```

Open multiple windows with one click

```
<html>
        <head>
            <script type="text/javascript">
                function open_win() {
                    window.open("http://www.microsoft.com/");
                    window.open("http://www.w3schools.com/");
                }
            </script>
        </head>
        <body>
            <form>
                <input type=button value="Open Windows" onclick="open_win()">
            </form>
        </body>
</html>
```

Send the client to a new location / URL

```
<html>
        <head>
            <script type="text/javascript">
                function currLocation() {
                    alert(window.location);
                }
                function newLocation() {
                    window.location="http://www.w3schools.com";
                }
            </script>
        </head>
        <body>
            <input type="button" onclick="currLocation()" value="Show current URL">
            <input type="button" onclick="newLocation()" value="Change URL">
        </body>
</html>
```

Reload a page

```
<html>
        <head>
            <script type="text/javascript">
                function reloadPage() {
                    window.location.reload();
                }
            </script>
        </head>
        <body>
            <input type="button" value="Reload page" onclick="reloadPage()" />
        </body>
</html>
```

Write some text in the windows status bar

```
<html>
        <body>
            <script type="text/javascript">
                window.status="Some text in the status bar!!";
            </script>
            <p>Look at the text in the statusbar.</p>
        </body>
</html>
```

Print a page

```
<html>
```

```
<head>
	<script type="text/javascript">
		function printpage() {
			window.print();
		}
	</script>
</head>
<body>
	<input type="button" value="Print this page" onclick="printpage()" />
</body>
</html>
```

Break out of a frame

```
<html>
<head>
	<script type="text/javascript">
		function breakout() {
			if (window.top!=window.self) {
				window.top.location="tryjs_breakout.htm";
			}
		}
	</script>
</head>
<body>
	<input type="button" onclick="breakout()" value="Break out of frame!">
</body>
</html>
```

Resize a window

```
<html>
<head>
	<script type="text/javascript">
		function resizeWindow() {
			top.resizeBy(-100,-100);
		}
	</script>
</head>
<body>
	<form>
		<input type="button" onclick="resizeWindow()" value="Resize window">
	</form>
	<p><b>Note:</b> We have used the <b>top</b> element instead of the <b>window</b>
	element, to represent the top frame. If you do not use frames, use the <b>window</b>
	element instead.</p>
</body>
</html>
```

Resize a window to a specified size

```
<html>
<head>
	<script type="text/javascript">
		function resizeWindow() {
			top.resizeTo(500,300);
		}
	</script>
</head>
<body>
	<form>
		<input type="button" onclick="resizeWindow()" value="Resize window">
	</form>
	<p><b>Note:</b> We have used the <b>top</b> element instead of the <b>window</b>
	element, to represent the top frame. If you do not use frames, use the <b>window</b>
	element instead.</p>
</body>
</html>
```

Scroll the window

```
<html>
	<head>
		<script type="text/javascript">
			function scrollWindow() {
				window.scrollBy(100,100);
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="scrollWindow()" value="Scroll" />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
```

```
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
	</body>
</html>
```

Scroll the window to a specified position

```
<html>
	<head>
		<script type="text/javascript">
			function scrollWindow() {
				window.scrollTo(100,500);
			}
		</script>
	</head>
	<body>
		<input type="button" onclick="scrollWindow()" value="Scroll" />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
```

```
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
		<br />
	</body>
</html>
```

Simple timing

```
<html>
	<head>
		<script type="text/javascript">
			function timedMsg() {
				var t=setTimeout("alert('5 seconds!')",5000);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Display timed alertbox!" onClick = "timedMsg()">
		</form>
		<p>Click on the button above. An alert box will be displayed after 5 seconds.</p>
	</body>
</html>
```

Another simple timing

```
<html>
	<head>
		<script type="text/javascript">
			function timedText() {
				var t1=setTimeout("document.getElementById('txt').value='2 seconds!'",2000);
				var t2=setTimeout("document.getElementById('txt').value='4 seconds!'",4000);
				var t3=setTimeout("document.getElementById('txt').value='6 seconds!'",6000);
			}
		</script>
	</head>

	<body>
		<form>
			<input type="button" value="Display timed text!" onClick="timedText()">
			<input type="text" id="txt">
		</form>
		<p>Click on the button above. The input field will tell you when two, four,
		and six seconds have passed.</p>
	</body>
</html>
```

Timing event in an infinite loop

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
		</form>
		<p>Click on the button above. The input field will count for ever, starting at 0.</p>
	</body>
</html>
```

Timing event in an infinite loop - with a Stop button

```
<html>
	<head>
		<script type="text/javascript">
			var c=0;
			var t;
			function timedCount() {
				document.getElementById('txt').value=c;
				c=c+1;
				t=setTimeout("timedCount()",1000);
			}
			function stopCount() {
				clearTimeout(t);
			}
		</script>
	</head>
	<body>
		<form>
			<input type="button" value="Start count!" onClick="timedCount()">
			<input type="text" id="txt">
			<input type="button" value="Stop count!" onClick="stopCount()">
		</form>
		<p>Click on the "Start count!" button above to start the timer. The input field will count forever,
		starting at 0. Click on the "Stop count!" button to stop the counting.</p>
	</body>
</html>
```

A clock created with a timing event

```
<html>
	<head>
		<script type="text/javascript">
			function startTime() {
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				// add a zero in front of numbers<10
				m=checkTime(m);
				s=checkTime(s);
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}
			function checkTime(i) {
				if (i<10) {
					i="0" + i;
				}
```

```
                        return i;
                }
        </script>
</head>
<body onload="startTime()">
        <div id="txt"></div>
</body>
</html>
```

Create a pop-up

```
<html>
<head>
        <script type="text/javascript">
                function show_popup() {
                        var p=window.createPopup();
                        var pbody=p.document.body;
                        pbody.style.backgroundColor="lime";
                        pbody.style.border="solid black 1px";
                        pbody.innerHTML="This is a pop-up! Click outside the pop-up to close.";
                        p.show(150,150,200,50,document.body);
                }
        </script>
</head>
<body>
        <button onclick="show_popup()">Show pop-up!</button>
</body>
</html>
```