JavaScript

A. Introduction

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari. JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

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

Java vs. JavaScript

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++.

Uses of JS

1. **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

2. **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.

3. **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

4. **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element

5. **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

6. **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

7. **JavaScript can be used to create cookies** - A JavaScript can be used to store andretrieve information on the visitor's computer

B. How to use JavaScript

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

The example below shows how to use JavaScript to write text on a web page:

Example

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

The example below shows how to add HTML tags to the JavaScript:

Example

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello World!</h1>");
</script>
</body>
</html>
```

Example Explained

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we usethe 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 **document.write** command is a standard JavaScript command for writing output to a page.By entering the document.write command between the <script> and </script> tags, the browserwill recognize it as a JavaScript command and execute the code line. In this case the browser willwrite Hello World! to the page:

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

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.

How 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 should 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, like this:

```
<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. Thi prevents JavaScript from executing the --> tag.

C. Where to Place JavaScript

JavaScripts can be put in the body and in the head sections of an HTML page. JavaScripts in a page will be executed immediately while the page loads into the browser. This isnot always what we want. Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button. When this is the case we put the script inside a function.

Scripts in <head>

Scripts to be executed when they are called, or when an event is triggered, are placed in functions.

Put your functions in the head section, this way they are all in one place, and they do not interfere with page content.

Example

```
<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>
```

Scripts in <body>
If you don't want your script to be placed inside a function, or if your script should write page content, it should be placed in the body section.

Example

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

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

Example

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

Using an External JavaScript
If you want to run the same JavaScript on several pages, without having to write the same script one very page, 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></script> tags! To use the external script, point to the .js file in the "src" attribute of the <script> tag:

Example

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

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

D. JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser.
JavaScript is Case SensitiveUnlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when youwrite JavaScript statements, create or call variables, objects and functions.

A JavaScript statement is a command to a browser. The purpose of the command is to tell thebrowser 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 agood 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 tointerpret the end of the line as the end of the statement. Because of this you will often seeexamples without the semicolon at the end.

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

JavaScript CodeJavaScript 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 heading and two paragraphs to a web page:

Example

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

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 heading and two paragraphs to a web page:

Example

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

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).

Comments
JavaScript comments can be used to make the code more readable. Comments can be added to explain the JavaScript, or to make the code more readable.

Single line comments start with //.
The following example uses single line comments to explain the code:

Example

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

Multi line comments start with /* and end with */.
The following example uses a multi line comment to explain the code:

Example

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

Using Comments to Prevent Execution
In the following example the comment is used to prevent the execution of a single code line (can be suitable for debugging):

Example

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

In the following example the comment is used to prevent the execution of a code block (can besuitable for debugging):

Example

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

Using Comments at the End of a Line
In the following example the comment is placed at the end of a code line:

Example

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

E. Variables

Variables are "containers" for storing information. 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

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** willhold 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 bedeclared. 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.

F. Operators

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

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 very nice 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

The rule is: **If you add a number and a string, the result will be a string!**

Example

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

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference betweenvariables 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 x==="5" is false
!=	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

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".

If...Else Statements
Conditional statements are used to perform different actions based on different conditions.

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 to execute some code • only if a specified condition is true
- **if...else statement** - use this statement 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 to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement
Use the if statement 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 JavaScripterror!

Example

```
<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>
```

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

If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if thecondition is not true.

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

Use the if....else if...else statement to select one of several blocks of code to be executed.

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>
```

Switch Statement

Conditional statements are used to perform different actions based on differentconditions.Use the switch statement 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 evaluatedonce. The value of the expression is then compared with the values for each case in the structure. Ifthere is a match, the block of code associated with that case is executed. Use **break** to prevent thecode 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>
```

G. Popup Boxes

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

Alert Box

An alert box is often used if you want to make sure information comes through to the user.
When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

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

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>
```

```
<input type="button" onclick="show_alert()" value="Show 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");
```

Example

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

Example

```
<html>
<head>
<script type="text/javascript">
function show_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="show_prompt()" value="Show prompt box" />
>
</body>
</html>
```

H. Functions

A function will be executed by an event or by a call to the function. To keep the browser from executing a script when the page loads, you can put your script into afunction.

A function contains code that will be executed by an event or by a call to the function. You may call a function from anywhere within a 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 a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

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

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

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 wouldhave been executed as soon as the line was loaded. Now, the script is not executed before a userhits the input button. The function displaymessage() will be executed if the input button is clicked.

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.The example below returns the product of two numbers (a and b):

Example

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

The Lifetime of JavaScript Variables

If 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. Youcan have local variables with the same name in different functions, because each is recognized onlyby 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 lifetimeof these variables starts when they are declared, and ends when the page is closed.

I. Loops

Loops execute a block of code a specified number of times, or while a specifiedcondition is true.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 throu • gh 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

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** isless than, or equal to 5. **i** will increase by 1 each time the loop runs.

> **Note:** The increment parameter could also be negative, and the <= could be any comparingstatement.

Example

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

The while loop

The while loop loops through a block of code while a specified condition is true.

Syntax

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

Note: The <= could be any comparing statement.

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** isless than, or equal to 5. **i** will increase by 1 each time the loop runs:

Example

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

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, andthen it will repeat the loop as long as the specified condition is true.

Syntax

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

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

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

The break Statement

The break statement 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>
```

The continue Statement

The continue statement 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>
```

For...In Statement

The for...in statement loops through the elements of an array or through the properties of anobject.

Syntax

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

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

Note: The variable argument can be a named variable, an array element, or a property of anobject.

Use the for...in statement to loop through an array:

Example

```
<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>
```

J. Events

Events are actions that can be detected by JavaScript.
Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clickson 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 field 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!

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 loadthe 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 setwhen a user enters or leaves a page. For example, you could have a popup asking for the user'sname upon his first arrival to your page. The name is then stored in a cookie. Next time the visitorarrives 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 formfields.Below is an example of how to use the onChange event. The checkEmail() function will be calledwhenever 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 calledwhen the user clicks the submit button in the form.

If the field values are not accepted, the submitshould be cancelled. The function checkForm() returns either true or false. If it returns true theform 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 eventis detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver
event');return false"><img src="w3s.gif" alt="W3Schools" /></a>
```

Try...Catch Statement

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

Catching Errors

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us thereis a runtime error and asking "Do you wish to debug?". Error message like this may be useful fordevelopers but not for users. When users see errors, they often leave the Web page.This chapter will teach you how to catch and handle JavaScript error messages, so you don't loseyour audience.

The 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 aJavaScript error!

The example below is supposed to alert "Welcome guest!" when the button is clicked. However,there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript erroroccurs. The catch block catches the error and executes a custom code to handle it. The codedisplays a custom error message informing the user what happened:

Example

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

Example

```
<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 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 JavaScripterror!

The example below determines the value of a variable called x. If the value of x is higher than 10,lower than 0, or not a number, we are going to throw an error. The error is then caught by thecatch argument and the proper error message is displayed:

Example

```
<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>
```

K. Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters intoa 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 backslashsign:

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

L. Objects

Object Oriented Programming

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

However, creating your own objects will be explained later, in the Advanced JavaScript section. Wewill start by looking at the built-in JavaScript objects, and how they are used. The next pages willexplain 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 numberof 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 atext 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!
```

Number Object

The Number object is an object wrapper for primitive numeric values.Number objects are created with new Number().

Syntax

```
var num = new Number(value);
```

Note: If the value parameter cannot be converted into a number, it returns NaN (Not-a-Number).
Number Object Properties

Number Object Properties

Property	Description
constructor	Returns the function that created the Number object's prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object

Number Object Methods

Method	Description
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a Number object to a string
valueOf()	Returns the primitive value of a Number object

Global

The JavaScript global properties and functions can be used with all the built-inJavaScript objects.

JavaScript Global Functions

Function	Description
decodeURI()	Decodes a URI
decodeURIComponent()	Decodes a URI component
encodeURI()	Encodes a URI
encodeURIComponent()	Encodes a URI component
escape()	Encodes a string
eval()	Evaluates a string and executes it as if it was script code
isFinite()	Determines whether a value is a finite, legal number
isNaN()	Determines whether a value is an illegal number
Number()	Converts an object's value to a number
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer
String()	Converts an object's value to a string
unescape()	Decodes an encoded string

String Object

The String object is used to manipulate a stored piece of text.
String objects are created with new String().

Syntax

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

or more simply:

```
var txt = string;
```

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 touppercase letters:

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

The code above will result in the following output:

```
HELLO WORLD!
```

String Object Properties

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

String Object Properties

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

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
valueOf()	Returns the primitive value of a String object

String HTML Wrapper Methods

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

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

Date Object
The Date object is used to work with dates and times.
Date objects are created with the Date() constructor.

There are four ways of instantiating a date:

```
new Date() // current date and time
new Date(milliseconds) //milliseconds since 1970/01/01
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

Most parameters above are optional. Not specifying, causes 0 to be passed in.Once a Date object is created, a number of methods allow you to operate on it. Most methods allowyou to get and set the year, month, day, hour, minute, second, and milliseconds of the object,using either local time or UTC (universal, or GMT) time.

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) witha day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00")
d2 = new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

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 automaticallyby 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");
}
```

Date Object Properties

Property	Description
constructor	Returns the function that created the Date object's prototype
prototype	Allows you to add properties and methods to an object

Date Object Methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the time difference between GMT and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month (from 1-31)
setFullYear()	Sets the year (four digits)
setHours()	Sets the hour (from 0-23)
setMilliseconds()	Sets the milliseconds (from 0-999)
setMinutes()	Set the minutes (from 0-59)
setMonth()	Sets the month (from 0-11)
setSeconds()	Sets the seconds (from 0-59)
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month, according to universal time (from 1-31)
setUTCFullYear()	Sets the year, according to universal time (four digits)
setUTCHours()	Sets the hour, according to universal time (from 0-23)
setUTCMilliseconds()	Sets the milliseconds, according to universal time (from 0-999)

setUTCMinutes()	Set the minutes, according to universal time (from 0-59)
setUTCMonth()	Sets the month, according to universal time (from 0-11)
setUTCSeconds()	Set the seconds, according to universal time (from 0-59)
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
valueOf()	Returns the primitive value of a Date object

Array Object

An array is a special variable, which can hold more than one value, at a time.If you have a list of items (a list of car names, for example), storing the cars in single variablescould look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not3 cars, but 300?The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values byreferring to the array name.Each element in the array has its own ID so that it can be easily accessed.

Create an Array

An array can be defined in three ways.

The following code creates an Array object called myCars:

1:

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab"; // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

2:

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

3:

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

Note: If you specify numbers or true/false values inside the array then the variable type will beNumber 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 indexnumber. 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 indexnumber:

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

Now, the following code line:

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

will result in the following output:

```
Opel
```

Array Object Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an object

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

Boolean Object
The Boolean object is used to convert a non-Boolean value to a Boolean value (trueor false).

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");
```

Boolean Object Properties

Property	Description
constructor	Returns the function that created the Boolean object's prototype
prototype	Allows you to add properties and methods to an object

Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object

Math Object
The Math object allows you to perform mathematical tasks.The Math object includes several mathematical constants and methods.

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

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 asan object without creating it.

Mathematical Constants
JavaScript provides eight mathematical constants that can be accessed from the Math object. Theseare: 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 alsoseveral methods available.

The following example uses the round() method of the Math object to round a number to thenearest 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 randomnumberbetween 0 and 1:

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

The code above can result in the following output:

```
0.7179836678443205
```

The following example uses the floor() and random() methods of the Math object toreturn arandom number between 0 and 10:

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

The code above can result in the following output:

```
7
```

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14159)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,....,n)	Returns the number with the highest value
min(x,y,z,....,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

RegExp Object
RegExp, is short for regular expression.A regular expression is an object that describes a pattern of characters.

When you search in a text, you can use a pattern to describe what you are searching for.A simple pattern can be one single character.A more complicated pattern can consist of more characters, and can be used for parsing, formatchecking, substitution and more.

Regular expressions are used to perform powerful pattern-matching and "search-and-replace"functions on text.

Syntax

```
var txt=new RegExp(pattern,modifiers);
or more simply:
var txt=/pattern/modifiers;
```

pattern: - specifies the pattern of an expression
modifiers: - specify if a search should be global, case-sensitive, etc.

RegExp Modifiers
Modifiers are used to perform case-insensitive and global searches.The i modifier is used to perform case-insensitive matching.The g modifier is used to perform a global match (find all matches rather than stopping after thefirst match).

Example 1

Do a case-insensitive search for "w3schools" in a string:

```
var str="Visit W3Schools";
var patt1=/w3schools/i;
```

The marked text below shows where the expression gets a match:

```
Visit W3Schools
```

Example 2

Do a global search for "is":

```
var str="Is this all there is?";
var patt1=/is/g;
```

The marked text below shows where the expression gets a match:

```
Is this all there is?
```

Example 3

Do a global, case-insensitive search for "is":

```
var str="Is this all there is?";
var patt1=/is/gi;
```

The marked text below shows where the expression gets a match:

```
Is this all there is?
```

test()

The test() method searches a string for a specified value, and returns true or false, depending onthe result.

The following example searches a string for the character "e":

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
```

exec()

The exec() method searches a string for a specified value, and returns the text of the found value.If no match is found, it returns *null*.The following example searches a string for the character "e":

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
```

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[a-z]	Find any character from lowercase a to lowercase z
[A-Z]	Find any character from uppercase A to uppercase Z
[a-Z]	Find any character from lowercase a to uppercase Z
[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
[red|blue|green]	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

Quantifier	Description

n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X or Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
?=n	Matches any string that is followed by a specific string n
?!n	Matches any string that is not followed by a specific string n

RegExp Object Properties

Property	Description
global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set
lastIndex	The index at which to start the next match
multiline	Specifies if the "m" modifier is set
source	The text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false

Navigator Object

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

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. Sometimes it can be useful to detect the visitor's browser, and then serve 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.

> The Navigator object contains information about the visitor's browser name, version, and more.
> **Note:** There is no public standard that applies to the navigator object, but all major browsers support it.

Example

```
<html>
```

```
<body>
<script type="text/javascript">
document.write("Browser CodeName: " + navigator.appCodeName);
document.write("<br /><br />");
document.write("Browser Name: " + navigator.appName);
document.write("<br /><br />");
document.write("Browser Version: " + navigator.appVersion);
document.write("<br /><br />");
document.write("Cookies Enabled: " + navigator.cookieEnabled);
document.write("<br /><br />");
document.write("Platform: " + navigator.platform);
document.write("<br /><br />");
document.write("User-agent header: " + navigator.userAgent);
</script>
</body>
</html>
```

M. Cookies

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 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

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.toUTCString());
}
```

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 ourcookie 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:

Example

```
<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.toUTCString());
}
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.

N. Form Validation

JavaScript can be used to validate 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="submit.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="submit.htm" onsubmit="return validate_form(this);"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

O. 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" id="b1"
onmouseOver="mouseOver()" onmouseOut="mouseOut()" /></a>
```

Note that we have given the image an id, to make it possible for a 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.getElementById("b1").src ="b_blue.gif";
}
function mouseOut()
{
document.getElementById("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
Example

```
<html>
```

```
<head>
<script type="text/javascript">
function mouseOver()
{
document.getElementById("b1").src ="b_blue.gif";
}
function mouseOut()
{
document.getElementById("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" id="b1"
onmouseover="mouseOver()" onmouseout="mouseOut()" /></a>
</body>
</html>
```

P. 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.

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 an HTML image-map example, with some JavaScript added:

Example

```
<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 name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onMouseOver="writeText('The Sun and the gas giant planets like
Jupiterare 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,
andbecause the two planets seem to share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>
<p id="desc"></p>
</body>
</html>
```

Q. Timing Events

With JavaScript, it is possible to execute some code after a specified time-interval. This is calledtiming 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.

The setTimeout() Method

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 a button is clicked, the input field will start to count (for ever), starting at 0.

Notice that we also have a function that checks if the timer is already running, to avoid timers, if the button is pressed more than once:

Example

```
<html>
<head>
<script type="text/javascript">
var c=0;
var t;
var timer_is_on=0;
function timedCount()
{
document.getElementById('txt').value=c;
1
2
3
4
7 6 5
8
9
10
11 12
c=c+1;
t=setTimeout("timedCount()",1000);
}
function doTimer()
{
if (!timer_is_on)
{
timer_is_on=1;
timedCount();
}
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!" onClick="doTimer()">
<input type="text" id="txt" />
</form>
```

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

The clearTimeout() Method

Syntax

```
clearTimeout(setTimeout_variable)
```

Example

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

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

Create Your Own 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 eye color 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();
```

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 tothis.propertyName. The reason for all the "this" stuff is that you're going to have more than oneperson 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").
```

R. Window Object

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

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

Window Object Properties

Window Object Properties

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

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
blur()	Removes focus from the current window
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
createPopup()	Creates a pop-up window
focus()	Sets focus to the current window
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	
scrollBy()	Scrolls the content by the specified number of pixels
scrollTo()	Scrolls the content to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds

Navigator Object

The navigator object contains information about the browser.

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

Navigator Object Properties

Navigator Object Properties

Property	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
platform	Returns for which platform the browser is compiled
userAgent	Returns the user-agent header sent by the browser to the server

Navigator Object Methods

Method	Description
javaEnabled()	Specifies whether or not the browser has Java enabled
taintEnabled()	Specifies whether or not the browser has data tainting enabled

Screen Object

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

> **Note:** There is no public standard that applies to the screen object, but all major browserssupport it.

Screen Object Properties

Property	Description
availHeight	Returns the height of the screen (excluding the Windows Taskbar)
availWidth	Returns the width of the screen (excluding the Windows Taskbar)
colorDepth	Returns the bit depth of the color palette for displaying images
height	Returns the total height of the screen
pixelDepth	Returns the color resolution (in bits per pixel) of the screen
width	Returns the total width of the screen

History Object
The history object contains the URLs visited by the user (within a browser window).
The history object is part of the window object and is accessed through the window.historyproperty.
Note: There is no public standard that applies to the history object, but all major browserssupport it.

History Object Properties

Property	Description
length	Returns the number of URLs in the history list

History Object Methods

Method	Description
back()	Loads the previous URL in the history list
forward()	Loads the next URL in the history list
go()	Loads a specific URL from the history list

Location Object
The location object contains information about the current URL.The location object is part of the window object and is accessed through the window.locationproperty.

> **Note:** There is no public standard that applies to the location object, but all major browserssupport it.

Location Object Properties

Property	Description
hash	Returns the anchor portion of a URL
host	Returns the hostname and port of a URL
hostname	Returns the hostname of a URL
href	Returns the entire URL
pathname	Returns the path name of a URL
port	Returns the port number the server uses for a URL
protocol	Returns the protocol of a URL
search	Returns the query portion of a URL

Location Object Methods

Method	Description
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

S. HTML DOM Document Object

Document Object
Each HTML document loaded into a browser window becomes a Document object. The Document object provides access to all HTML elements in a page, from within a script.

Tip: The Document object is also part of the Window object, and can be accessed through the window.document property.

Document Object Collections

W3C: W3C Standard.

Collection	Description	W3C
anchors[]	Returns an array of all the anchors in the document	Yes
forms[]	Returns an array of all the forms in the document	Yes
images[]	Returns an array of all the images in the document	Yes
links[]	Returns an array of all the links in the document	Yes

Document Object Properties

Property	Description	W3C
cookie	Returns all name/value pairs of cookies in the document	Yes
documentMode	Returns the mode used by the browser to render the document	No
domain	Returns the domain name of the server that loaded the document	Yes
lastModified	Returns the date and time the document was last modified	No
readyState	Returns the (loading) status of the document	No
referrer	Returns the URL of the document that loaded the current document	Yes
title	Sets or returns the title of the document	Yes
URL	Returns the full URL of the document	Yes

Document Object Methods

Method	Description	W3C
close()	Closes the output stream previously opened with document.open()	Yes
getElementById()	Accesses the first element with the specified id	Yes
getElementsByName()	Accesses all elements with a specified name	Yes
getElementsByTagName()	Accesses all elements with a specified tagname	Yes
open()	Opens an output stream to collect the output from document.write() or document.writeln()	Yes
write()	Writes HTML expressions or JavaScript code to a document	Yes
writeln()	Same as write(), but adds a newline character after each statement	Yes

Event Object

The event object gives you information about an event that has occurred. 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!

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 beinserted into HTML tags to define event actions.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C Standard.

Attribute	The event occurs when...	IE	F	O	W3C
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
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
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	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 event was triggered	6	1	9	Yes
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

T. HTML DOM Object

The collections, properties, methods, and events below can be used on all HTML elements.

HTML Element Object Collections

HTMLElement Object Collections

W3C: W3C Standard.

Collection	Description	W3C
attributes[]	Returns an array of the attributes of an element	Yes
childNodes[]	Returns an array of child nodes for an element	Yes

HTMLElement Object Properties

Property	Description	W3C
accessKey	Sets or returns an accesskey for an element	Yes
className	Sets or returns the class attribute of an element	Yes
clientHeight	Returns the viewable height of the content on a page (not including borders, margins, or scrollbars)	Yes
clientWidth	Returns the viewable width of the content on a page (not including borders, margins, or scrollbars)	Yes
dir	Sets or returns the text direction of an element	Yes
disabled	Sets or returns the disabled attribute of an element	Yes
firstChild	Returns the first child of an element	Yes
height	Sets or returns the height attribute of an element	Yes
id	Sets or returns the id of an element	Yes
innerHTML	Sets or returns the HTML contents (+text) of an element	Yes
lang	Sets or returns the language code for an element	Yes
lastChild	Returns the last child of an element	Yes
length		Yes
nextSibling	Returns the element immediately following an element	Yes
nodeName	Returns the tagname of an element (in uppercase)	Yes
nodeType	Returns the type of the element	Yes
nodeValue	Returns the value of the element	Yes
offsetHeight	Returns the height of an element, including borders and padding if any, but not margins	No
offsetLeft	Returns the horizontal offset position of the current element relative to its offset container	Yes
offsetParent	Returns the offset container of an element	Yes
offsetTop	Returns the vertical offset position of the current element relative to its offset container	Yes
offsetWidth	Returns the width of an element, including borders and padding if any, but not margins	No
ownerDocument	Returns the root element (document object) for an element	Yes

Anchor Object

The Anchor object represents an HTML hyperlink. For each <a> tag in an HTML document, an Anchor object is created. An anchor allows you to create a link to another document (with the href attribute), or to a different point in the same document (with the name attribute). You can access

an anchor by using `getElementById()`, or by searching through the anchors[] array of the Document object.

Anchor Object Properties

W3C: W3C Standard.

Property	Description	W3C
charset	Sets or returns the value of the charset attribute of a link	Yes
href	Sets or returns the value of the href attribute of a link	Yes
hreflang	Sets or returns the value of the hreflang attribute of a link	Yes
name	Sets or returns the value of the name attribute of a link	Yes
rel	Sets or returns the value of the rel attribute of a link	Yes
rev	Sets or returns the value of the rev attribute of a link	Yes
target	Sets or returns the value of the target attribute of a link	Yes
type	Sets or returns the value of the type attribute of a link	Yes

Standard Properties, Methods, and Events
The Anchor object also supports the *standard properties, methods, and events.*

Area Object
The Area object represents an area inside an HTML image-map (an image-map is an image with clickable areas). For each <area> tag in an HTML document, an Area object is created.

Area Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns the value of the alt attribute of an area	Yes
coords	Sets or returns the value of the coords attribute of an area	Yes
hash	Sets or returns the anchor part of the href attribute value	Yes
host	Sets or returns the hostname:port part of the href attribute value	Yes
hostname	Sets or returns the hostname part of the href attribute value	Yes
href	Sets or returns the value of the href attribute of an area	Yes
noHref	Sets or returns the value of the nohref attribute of an area	Yes
pathname	Sets or returns the pathname part of the href attribute value	Yes
port	Sets or returns the port part of the href attribute value	Yes
protocol	Sets or returns the protocol part of the href attribute value	Yes
search	Sets or returns the querystring part of the href attribute value	Yes
shape	Sets or returns the value of the shape attribute of an area	Yes
target	Sets or returns the value of the target attribute of an area	Yes

Base Object
The Base object represents an HTML base element. The base element is used to specify a default address or a default target for all links on a page. For each <base> tag in an HTML document, a Base object is created.

Base Object Properties

W3C: W3C Standard.

Property	Description	W3C
href	Sets or returns the value of the href attribute in a base element	Yes
target	Sets or returns the value of the target attribute in a base element	Yes

Body Object
The Body object represents the HTML body element. The body element defines a document's body.
The body element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

Body Object Properties

W3C: W3C Standard.

Property	Description	W3C
aLink	Sets or returns the value of the alink attribute of the body element	Yes
background	Sets or returns the value of the background attribute of the body element	Yes
bgColor	Sets or returns the value of the bgcolor attribute of the body element	Yes
link	Sets or returns the value of the link attribute of the body element	Yes
text	Sets or returns the value of the text attribute of the body element	Yes
vLink	Sets or returns the value of the vlink attribute of the body element	Yes

Body Object Events

Event	Description	W3C
onload	Script to be run immediately after a page is loaded	Yes

Button Object
The Button object represents a push button. For each <button> tag in an HTML document, a Button object is created. Inside an HTML button element you can put content, like text or images. This is the difference between this element and buttons created with the input element.

Button Object Properties

W3C: W3C Standard.

Property	Description	W3C
form	Returns a reference to the form that contains a button	Yes
name	Sets or returns the value of the name attribute of a button	Yes
type	Sets or returns the type of a button	Yes
value	Sets or returns the value of the value attribute of a button	Yes

Form Object
The Form object represents an HTML form. For each <form> tag in an HTML document, a Form object is created.

Forms are used to collect user input, and contain input elements like text fields, checkboxes, radio buttons, submit buttons and more. A form can also contain select menus, textarea, fieldset, legend, and label elements. Forms are used to pass data to a server.

Form Object Collections

W3C: W3C Standard.

Collection	Description	W3C
elements[]	Returns an array of all elements in a form	Yes

Form Object Properties

Property	Description	W3C
acceptCharset	Sets or returns the value of the accept-charset attribute in a form	Yes
action	Sets or returns the value of the action attribute in a form	Yes
enctype	Sets or returns the value of the enctype attribute in a form	Yes
length	Returns the number of elements in a form	Yes
method	Sets or returns the value of the method attribute in a form	Yes
name	Sets or returns the value of the name attribute in a form	Yes
target	Sets or returns the value of the target attribute in a form	Yes

Form Object Methods

Method	Description	W3C
reset()	Resets a form	Yes
submit()	Submits a form	Yes

Form Object Events

Event	The event occurs when...	W3C
onreset	The reset button is clicked	Yes
onsubmit	The submit button is clicked	Yes

Frame Object
The Frame object represents an HTML frame. The <frame> tag defines one particular window (frame) within a frameset. For each <frame> tag in an HTML document, a Frame object is created.

IFrame Object
The IFrame object represents an HTML inline frame. The <iframe> tag defines an inline frame that contains another document. For each <iframe> tag in an HTML document, an IFrame object is created. Frame/IFrame Object Properties

Frame/IFrame Object Properties

W3C: W3C Standard.

Property	Description	W3C
align	Sets or returns the value of the align attribute in an iframe	Yes
contentDocument	Returns the document object generated by a frame/iframe	Yes
contentWindow	Returns the window object generated by a frame/iframe	No
frameBorder	Sets or returns the value of the frameborder attribute in a frame/iframe	Yes
height	Sets or returns the value of the height attribute in an iframe	Yes
longDesc	Sets or returns the value of the longdesc attribute in a frame/iframe	Yes
marginHeight	Sets or returns the value of the marginheight attribute in a frame/iframe	Yes
marginWidth	Sets or returns the value of the marginwidth attribute in a frame/iframe	Yes
name	Sets or returns the value of the name attribute in a frame/iframe	Yes
noResize	Sets or returns the value of the noresize attribute in a frame	Yes
scrolling	Sets or returns the value of the scrolling attribute in a frame/iframe	Yes
src	Sets or returns the value of the src attribute in a frame/iframe	Yes
width	Sets or returns the value of the width attribute in an iframe	Yes

Frame/IFrame Object Events

Event	Description	W3C
onload	Script to be run immediately after a frame/iframe is loaded	Yes

Frameset Object
The Frameset object represents an HTML frameset. The HTML frameset element holds two or more frame elements. Each frame element holds a separate document. The HTML frameset element states only how many columns or rows there will be in the frameset.

Frameset Object Properties

W3C: W3C Standard.

Property	Description	W3C
cols	Sets or returns the value of the cols attribute in a frameset	Yes
rows	Sets or returns the value of the rows attribute in a frameset	Yes

Frameset Object Events

Event	Description	W3C
onload	Script to be run immediately after a page is loaded	Yes

Image Object

The Image object represents an embedded image. For each <img> tag in an HTML document, an Image object is created. Notice that images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a holding space for the referenced image.

Image Object Properties

W3C: W3C Standard.

Property	Description	W3C
align	Sets or returns the value of the align attribute of an image	Yes
alt	Sets or returns the value of the alt attribute of an image	Yes
border	Sets or returns the value of the border attribute of an image	Yes
complete	Returns whether or not the browser is finished loading an image	No
height	Sets or returns the value of the height attribute of an image	Yes
hspace	Sets or returns the value of the hspace attribute of an image	Yes
longDesc	Sets or returns the value of the longdesc attribute of an image	Yes
lowsrc	Sets or returns a URL to a low-resolution version of an image	No
name	Sets or returns the name of an image	Yes
src	Sets or returns the value of the src attribute of an image	Yes
useMap	Sets or returns the value of the usemap attribute of an image	Yes
vspace	Sets or returns the value of the vspace attribute of an image	Yes
width	Sets or returns the value of the width attribute of an image	Yes

Image Object Events

Event	The event occurs when...	W3C
onabort	Loading of an image is interrupted	Yes
onerror	An error occurs when loading an image	Yes
onload	An image is finished loading	Yes

<input type="button"> Object

The <input type="button"> object represents a clickable button in an HTML form. The button type is most often used to activate a JavaScript when a user clicks on the button. For each instance of an <input type="button">

tag in an HTML form, a Button object is created. You can access a button object by searching through the elements[] array of a form, or by using document.getElementById().

Button Object Properties

W3C: W3C Standard.

Property	Description	W3C
form	Returns a reference to the form that contains the input button	Yes
name	Sets or returns the value of the name attribute of an input button	Yes
type	Returns the type of form element the button is	Yes
value	Sets or returns the value of the value attribute of a button	Yes

Checkbox Object

The Checkbox object represents a checkbox in an HTML form. Checkboxes let a user select one or more options of a limited number of choices. For each <input type="checkbox"> tag in an HTML form, a Checkbox object is created. You can access a checkbox object by searching through the elements[] array of a form, or by using document.getElementById().

Checkbox Object Properties

W3C: W3C Standard.

Property	Description	W3C
checked	Sets or returns whether or not a checkbox should be checked	Yes
defaultChecked	Returns the default value of the checked attribute	Yes
form	Returns a reference to the form that contains the checkbox	Yes
name	Sets or returns the name of a checkbox	Yes
type	Returns the type of form element a checkbox is	Yes
value	Sets or returns the value of the value attribute of a checkbox	Yes

FileUpload Object

For each <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().

FileUpload Object Properties

W3C: W3C Standard.

Property	Description	W3C
accept	Sets or returns a comma-separated list of MIME types that indicates the MIME type of the file transfer	Yes
defaultValue	Sets or returns the initial value of the FileUpload object	Yes
form	Returns a reference to the form that contains the FileUpload object	Yes
name	Sets or returns the name of the FileUpload object	Yes
type	Returns the type of the form element. For a FileUpload object it will be "file"	Yes
value	Returns the file name of the FileUpload object after the text is set by user input	Yes

Hidden Object

The Hidden object represents a hidden input field in an HTML form. For each <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().

Hidden Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support hidden fields	Yes
form	Returns a reference to the form that contains the hidden field	Yes
name	Sets or returns the name of a hidden field	Yes
type	Returns the type of form element a hidden input field is	Yes
value	Sets or returns the value of the value attribute of the hidden field	Yes

Password Object

The Password object represents a password field in an HTML form. For each <input type="password"> tag in an HTML 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().

Password Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support password fields	Yes
defaultValue	Sets or returns the default value of a password field	Yes
disabled	Sets or returns whether or not a password field should be disabled	Yes
form	Returns a reference to the form that contains the password field	Yes
maxLength	Sets or returns the maximum number of characters in a password field	Yes
name	Sets or returns the name of a password field	Yes
readOnly	Sets or returns whether or not a password field should be read-only	Yes
size	Sets or returns the size of a password field	Yes
type	Returns the type of form element a password field is	Yes
value	Sets or returns the value of the value attribute of the password field	Yes

Radio Object

The Radio object represents a radio button in an HTML form. For each <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().

Radio Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support radio buttons	Yes
checked	Sets or returns the state of a radio button	Yes
defaultChecked	Returns the default state of a radio button	Yes
disabled	Sets or returns whether or not a radio button should be disabled	Yes
form	Returns a reference to the form that contains the radio button	Yes
name	Sets or returns the name of a radio button	Yes
type	Returns the type of form element a radio button is	Yes
value	Sets or returns the value of the value attribute of the radio button	Yes

Reset Object

The Reset object represents a reset button in an HTML form. For each <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().

Reset Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support reset buttons	Yes
disabled	Sets or returns whether or not a reset button should be disabled	Yes
form	Returns a reference to the form that contains the reset button	Yes
name	Sets or returns the name of a reset button	Yes
type	Returns the type of form element a reset button is	Yes
value	Sets or returns the text that is displayed on a reset button	Yes

Submit Object
The Submit object represents a submit button in an HTML form. For each <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().

Submit Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support submit buttons	Yes
disabled	Sets or returns whether or not a submit button should be disabled	Yes
form	Returns a reference to the form that contains the submit button	Yes
name	Sets or returns the name of a submit button	Yes
type	Returns the type of form element a submit button is	Yes
value	Sets or returns the text that is displayed on a submit button	Yes

Text Object
The Text object represents a text-input field in an HTML form. For each <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().

Text Object Properties

W3C: W3C Standard.

Property	Description	W3C
alt	Sets or returns an alternate text to display if a browser does not support text fields	Yes
defaultValue	Sets or returns the default value of a text field	Yes
disabled	Sets or returns whether or not a text field should be disabled	Yes
form	Returns a reference to the form that contains the text field	Yes
maxLength	Sets or returns the maximum number of characters in a text field	Yes
name	Sets or returns the name of a text field	Yes
readOnly	Sets or returns whether or not a text field should be read-only	Yes
size	Sets or returns the size of a text field	Yes
type	Returns the type of form element a text field is	Yes
value	Sets or returns the value of the value attribute of a text field	Yes

Text Object Methods

Method	Description	W3C
select()	Selects the content of a text field	Yes

Link Object

The Link object represents an HTML link element. A link element defines the relationship between two linked documents. The link element is defined in the head section of an HTML document.

Link Object Properties

W3C: W3C Standard.

Property	Description	W3C
charset	Sets or returns the character encoding of the target URL	Yes
disabled	Sets or returns whether or not the target URL should be disabled	Yes
href	Sets or returns the URL of a linked resource	Yes
hreflang	Sets or returns the base language of the target URL	Yes
media	Sets or returns on what device the document will be displayed	Yes
name	Sets or returns the name of a <link> element	Yes
rel	Sets or returns the relationship between the current document and the target URL	Yes
rev	Sets or returns the relationship between the target URL and the current document	Yes
type	Sets or returns the MIME type of the target URL	Yes

Meta Object

The Meta object represents an HTML meta element. Metadata is information about data.

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata. The <meta> tag always goes inside the head element.

Meta Object Properties

W3C: W3C Standard.

Property	Description	W3C
content	Sets or returns the value of the content attribute of a <meta> element	Yes
httpEquiv	Connects the content attribute to an HTTP header	Yes
name	Connects the content attribute to a name	Yes
scheme	Sets or returns the format to be used to interpret the value of the content attribute	Yes

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

Object Properties

W3C: W3C Standard.

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

Option Object
The Option object represents an option in a dropdown list in an HTML form. For each <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().

Option Object Properties

W3C: W3C Standard.

Property	Description	W3C
defaultSelected	Returns the default value of the selected attribute	Yes
disabled	Sets or returns whether or not an option should be disabled	Yes
form	Returns a reference to the form that contains an option	Yes
index	Returns the index position of an option in a dropdown list	Yes
label	Sets or returns a label for an option (only for option-groups)	Yes
selected	Sets or returns the value of the selected attribute	Yes
text	Sets or returns the text value of an option	Yes
value	Sets or returns the value to be sent to the server	Yes

Select Object
The Select object represents a dropdown list in an HTML form. For each <select> tag in an HTML 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().

Select Object Collections

W3C: W3C Standard.

Collection	Description	W3C
options[]	Returns an array of all the options in a dropdown list	Yes

Select Object Properties

Property	Description	W3C
disabled	Sets or returns whether or not a dropdown list should be disabled	Yes
form	Returns a reference to the form that contains the dropdown list	Yes
length	Returns the number of options in a dropdown list	Yes
multiple	Sets or returns whether or not multiple items can be selected	Yes
name	Sets or returns the name of a dropdown list	Yes
selectedIndex	Sets or returns the index of the selected option in a dropdown list	Yes
size	Sets or returns the number of visible rows in a dropdown list	Yes
type	Returns the type of form element a dropdown list is	Yes

Select Object Methods

Method	Description	W3C
add()	Adds an option to a dropdown list	Yes
remove()	Removes an option from a dropdown list	Yes

Style Object
The Style object represents an individual style statement. The Style object can be accessed from the document or from the elements to which that style is applied.

Syntax for using the Style object properties:
`document.getElementById("id").style.property="value"`

The Style object property categories:

- Background
- Border and Margin
- Layout
- List
- Misc
- Positioning
- Printing
- Table
- Text

Background properties

W3C: W3C Standard.

Property	Description	W3C
background	Sets all background properties in one	Yes
backgroundAttachment	Sets whether a background-image is fixed or scrolls with the page	Yes
backgroundColor	Sets the background-color of an element	Yes
backgroundImage	Sets the background-image of an element	Yes
backgroundPosition	Sets the starting position of a background-image	Yes
backgroundPositionX	Sets the x-coordinates of the backgroundPosition property	No
backgroundPositionY	Sets the y-coordinates of the backgroundPosition property	No
backgroundRepeat	Sets if/how a background-image will be repeated	Yes

Border and Margin properties

Property	Description	W3C
border	Sets all properties for the four borders in one	Yes
borderBottom	Sets all properties for the bottom border in one	Yes
borderBottomColor	Sets the color of the bottom border	Yes
borderBottomStyle	Sets the style of the bottom border	Yes
borderBottomWidth	Sets the width of the bottom border	Yes
borderColor	Sets the color of all four borders (can have up to four colors)	Yes
borderLeft	Sets all properties for the left border in one	Yes
borderLeftColor	Sets the color of the left border	Yes
borderLeftStyle	Sets the style of the left border	Yes
borderLeftWidth	Sets the width of the left border	Yes
borderRight	Sets all properties for the right border in one	Yes
borderRightColor	Sets the color of the right border	Yes
borderRightStyle	Sets the style of the right border	Yes
borderRightWidth	Sets the width of the right border	Yes
borderStyle	Sets the style of all four borders (can have up to four styles)	Yes
borderTop	Sets all properties for the top border in one	Yes
borderTopColor	Sets the color of the top border	Yes
borderTopStyle	Sets the style of the top border	Yes
borderTopWidth	Sets the width of the top border	Yes
borderWidth	Sets the width of all four borders (can have up to four widths)	Yes
margin	Sets the margins of an element (can have up to four values)	Yes

marginBottom	Sets the bottom margin of an element	Yes
marginLeft	Sets the left margin of an element	Yes
marginRight	Sets the right margin of an element	Yes
marginTop	Sets the top margin of an element	Yes
outline	Sets all outline properties in one	Yes
outlineColor	Sets the color of the outline around a element	Yes
outlineStyle	Sets the style of the outline around an element	Yes
outlineWidth	Sets the width of the outline around an element	Yes
padding	Sets the padding of an element (can have up to four values)	Yes
paddingBottom	Sets the bottom padding of an element	Yes
paddingLeft	Sets the left padding of an element	Yes
paddingRight	Sets the right padding of an element	Yes
paddingTop	Sets the top padding of an element	Yes

Layout properties

Property	Description	W3C
clear	Sets on which sides of an element other floating elements are not allowed	Yes
clip	Sets the shape of an element	Yes
content	Sets meta-information	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	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	Yes
cssFloat	Sets where an image or a text will appear (float) in another element	Yes
cursor	Sets the type of cursor to be displayed	Yes
direction	Sets the text direction of an element	Yes
display	Sets how an element will be displayed	Yes
height	Sets the height of an element	Yes
markerOffset	Sets the distance between the nearest border edges of a marker box and its principal box	Yes
marks	Sets whether cross marks or crop marks should be rendered just outside the page box edge	Yes
maxHeight	Sets the maximum height of an element	Yes
maxWidth	Sets the maximum width of an element	Yes
minHeight	Sets the minimum height of an element	Yes
minWidth	Sets the minimum width of an element	Yes
overflow	Specifies what to do with content that does not fit in an element box	Yes
verticalAlign	Sets the vertical alignment of content in an element	Yes
visibility	Sets whether or not an element should be visible	Yes
width	Sets the width of an element	Yes

List properties

Property	Description	W3C
listStyle	Sets all the properties for a list in one	Yes
listStyleImage	Sets an image as the list-item marker	Yes
listStylePosition	Positions the list-item marker	Yes
listStyleType	Sets the list-item marker type	Yes

Misc properties

Property	Description	W3C
cssText		

Positioning properties

Property	Description	W3C
bottom	Sets how far the bottom edge of an element is above/below the bottom edge of the parent element	Yes
left	Sets how far the left edge of an element is to the right/left of the left edge of the parent element	Yes
position	Places an element in a static, relative, absolute or fixed position	Yes
right	Sets how far the right edge of an element is to the left/right of the right edge of the parent element	Yes
top	Sets how far the top edge of an element is above/below the top edge of the parent element	Yes
zIndex	Sets the stack order of an element	Yes

Printing properties

Property	Description	W3C
orphans	Sets the minimum number of lines for a paragraph that must be left at the bottom of a page	Yes
page	Sets a page type to use when displaying an element	Yes
pageBreakAfter	Sets the page-breaking behavior after an element	Yes
pageBreakBefore	Sets the page-breaking behavior before an element	Yes
pageBreakInside	Sets the page-breaking behavior inside an element	Yes
size	Sets the orientation and size of a page	Yes
widows	Sets the minimum number of lines for a paragraph that must be left at the top of a page	Yes

Table properties

Property	Description	W3C
borderCollapse	Sets whether the table border are collapsed into a single border or detached as in standard HTML	Yes
borderSpacing	Sets the distance that separates cell borders	Yes
captionSide	Sets the position of the table caption	Yes
emptyCells	Sets whether or not to show empty cells in a table	Yes
tableLayout	Sets the algorithm used to display the table cells, rows, and columns	Yes

Text properties

Property	Description	W3C
color	Sets the color of the text	Yes
font	Sets all font properties in one	Yes
fontFamily	Sets the font of an element	Yes
fontSize	Sets the font-size of an element	Yes
fontSizeAdjust	Sets/adjusts the size of a text	Yes
fontStretch	Sets how to condense or stretch a font	Yes
fontStyle	Sets the font-style of an element	Yes
fontVariant	Displays text in a small-caps font	Yes
fontWeight	Sets the boldness of the font	Yes
letterSpacing	Sets the space between characters	Yes
lineHeight	Sets the distance between lines	Yes
quotes	Sets which quotation marks to use in a text	Yes
textAlign	Aligns the text	Yes
textDecoration	Sets the decoration of a text	Yes
textIndent	Indents the first line of text	Yes
textShadow	Sets the shadow effect of a text	Yes
textTransform	Sets capitalization effect on a text	Yes
unicodeBidi		Yes
whiteSpace	Sets how to handle line-breaks and white-space in a text	Yes
wordSpacing	Sets the space between words in a text	Yes

Table Object

The Table object represents an HTML table. For each <table> tag in an HTML document, a Table object is created.

Table Object Collections

W3C: W3C Standard.

Collection	Description	W3C
cells[]	Returns an array containing each cell in a table	No
rows[]	Returns an array containing each row in a table	Yes
tBodies[]	Returns an array containing each tbody in a table	Yes

Table Object Properties

Property	Description	W3C
border	Sets or returns the width of the table border	Yes
caption	Sets or returns the caption of a table	Yes
cellPadding	Sets or returns the amount of space between the cell border and cell content	Yes
cellSpacing	Sets or returns the amount of space between the cells in a table	Yes
frame	Sets or returns the outer-borders of a table	Yes
rules	Sets or returns the inner-borders of a table	Yes
summary	Sets or returns a description of a table	Yes
tFoot	Returns the TFoot object of a table	Yes
tHead	Returns the THead object of a table	Yes
width	Sets or returns the width of a table	Yes

Table Object Methods

Method	Description	W3C
createCaption()	Creates a caption element for a table	Yes
createTFoot()	Creates an empty tFoot element in a table	Yes
createTHead()	Creates an empty tHead element in a table	Yes
deleteCaption()	Deletes the caption element and its content from a table	Yes
deleteRow()	Deletes a row from a table	Yes
deleteTFoot()	Deletes the tFoot element and its content from a table	Yes
deleteTHead()	Deletes the tHead element and its content from a table	Yes
insertRow()	Inserts a new row in a table	Yes

TableCell Object

The TableCell object represents an HTML table cell. For each <td> tag in an HTML document, a TableCell object is created.

TableCell Object Properties

W3C: W3C Standard.

Property	Description	W3C
abbr	Sets or returns an abbreviated version of the content in a table cell	Yes
align	Sets or returns the horizontal alignment of data within a table cell	Yes
axis	Sets or returns a comma-delimited list of related table cells	Yes
cellIndex	Returns the position of a cell in the cells collection of a row	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	Yes
headers	Sets or returns a list of space-separated header-cell ids	Yes
rowSpan	Sets or returns the number of rows a table cell should span	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	Yes
width	Sets or returns the width of a table cell	Yes

TableRow Object

The TableRow object represents an HTML table row. For each <tr> tag in an HTML document, a TableRow object is created.

TableRow Object Collections

W3C: W3C Standard.

Collection	Description	W3C
cells[]	Returns an array containing each cell in the table row	Yes

TableRow Object Properties

Property	Description	W3C
align	Sets or returns the horizontal alignment of data within a table row	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
rowIndex	Returns the position of a row in the table's rows collection	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	Yes

TableRow Object Methods

Method	Description	W3C
deleteCell()	Deletes a cell in a table row	Yes
insertCell()	Inserts a cell in a table row	Yes

Textarea Object

The Textarea object represents a text-area in an HTML form. For each <textarea> tag in an HTML form, a Textarea object is created. You can access a Textarea object by indexing the elements array (by number or name) of the form or by using getElementById().

Textarea Object Properties

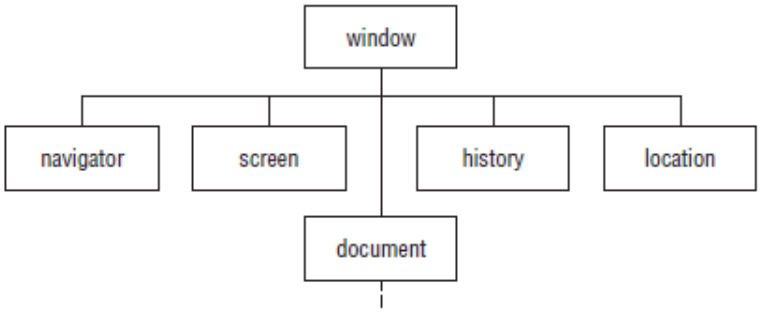
W3C: W3C Standard.

Property	Description	W3C
cols	Sets or returns the width of a textarea	Yes
defaultValue	Sets or returns the default text in a textarea	Yes
disabled	Sets or returns whether or not a textarea should be disabled	Yes
form	Returns a reference to the form that contains the textarea	Yes
name	Sets or returns the name of a textarea	Yes
readOnly	Sets or returns whether or not a textarea should be read-only	Yes
rows	Sets or returns the height of a textarea	Yes
type	Returns the type of the form element	Yes
value	Sets or returns the text in a textarea	Yes

Textarea Object Methods

Method	Description	W3C
select()	Selects the text in a textarea	Yes

Basic object model for all modern browsers.



Element hierarchy of an empty HTML 4.01 document.