JavaScript Tutorial

Table of Contents

Introduction

Introduction

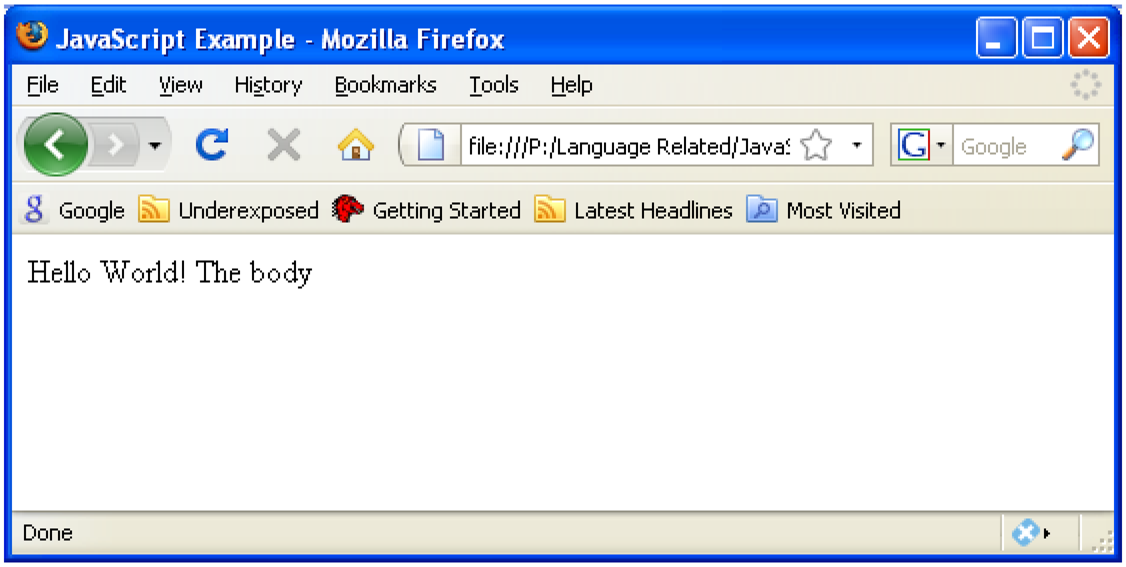
JavaScript is a dynamic language that executes within a browser. JavaScript code is embedded within an HTML page using the JavaScript tag. The <script> tag is used to embed JavaScript code. JavaScript code can be embedded in:

- An external file
- The header of the page
- The body of the page

In this example, JavaScript is embedded within the header. As soon as the page is loaded this code is executed.

```
<html>
<head>
<title>JavaScript  Example</title>
<script language="JavaScript 1.2">
<!--
document.write("Hello World!");
//-->
</script>
</head>
<body>The body</body>
</html>
```

The Document *write* method displays the text.


JavaScript Example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Language Related/Java:
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

Hello World! The body

Notice that the JavaScript code is enclosed in HTML comment tags:

```
<!--

//-->
```

These are often used to surround JavaScript code. In older browsers JavaScript was not recognized or handled. To avoid the display of this code in a page, the browser would ignore the contents of the comment. However, in a browser that supports JavaScript the comments tags are ignored and the code is executed.

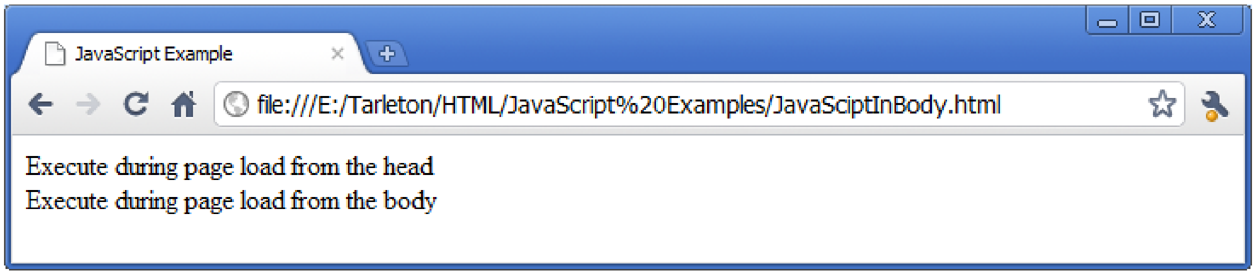
Internal JavaScript Code

JavaScript code that is not found in a function is executed as the page containing it is loaded. To illustrate this, JavaScript code is placed in the head and body section of an HTML page.

```
<html>
<head>
<title>JavaScript Example</title>
<script type="text/javascript">
        document.write("Execute during page load from the head<br>");
</script>
</head>

<body>
<script type="text/javascript">
        document.write("Execute during page load from the body<br>");
</script>
</body>

</html>
```


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
Execute during page load from the head
Execute during page load from the body
```

JavaScript code found in a function is not executed until the function is called. If we modify the previous example by adding a function to return a string, the function is not loaded when the page is loaded.

```
<html>
<head>
<title>JavaScript Example</title>
<script type="text/javascript">
function displayString() {
        return "<h1>Main Heading<h1>"
}
        document.write("Execute during page load from the head<br>");
</script>
```

```
</head>

<body>
<script type="text/javascript">
        document.write("Execute during page load from the body<br>");
</script>
</body>

</html>
```

The output will be the same.

Functions

A function consists of the function keyword followed by the name of the function , a set of open and close parentheses enclosing an optional parameter list and a body enclosed in a set of curly braces.

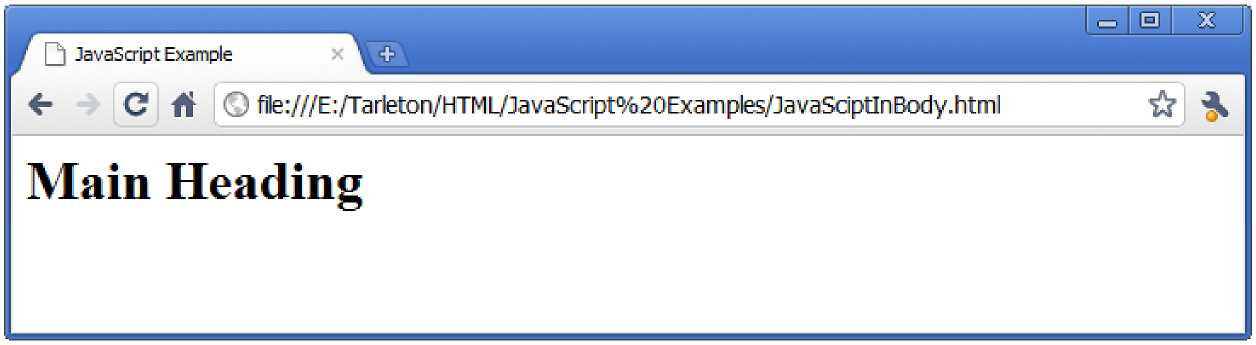
```
        function functionName(parameterList) {
                // body
        }
```

A function uses the return keyword to return a value from a function.

```
<html>
<head>
<title>JavaScript Example</title>
<script type="text/javascript">
function getHeader() {
        return "<h1>Main Heading</h1>"
}
</script>
</head>

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

</html>
```


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

Main Heading

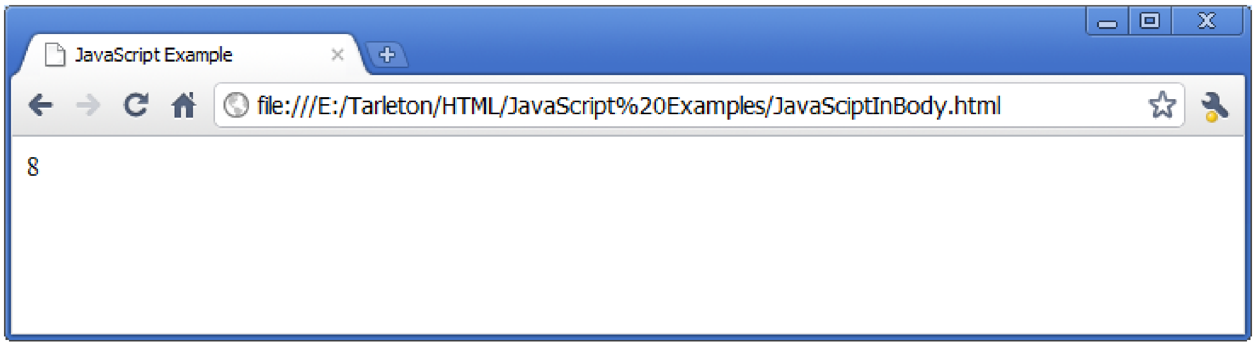
Parameters are separated by commas in the function declaration.

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

```
<title>JavaScript Example</title>
<script type="text/javascript">
function multiply(num1, num2) {
	return num1*num2;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(multiply(2,4));
</script>
</body>

</html>
```


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
8
```

External JavaScript Code

It is advantageous to group common functions in an external JavaScript file. This permits the reuse of the functions in the file in multiple HTML pages.

JavaScript functions are stored in a file using the .js extension. If we placed the following functions in a file named scripts.js we can reference and subsequently use the functions from an HTML page.

```
// functions.js
function getHeader() {
        return "<h1>Main Heading</h1>"
}

function multiply(num1, num2) {
        return num1*num2;
}
```

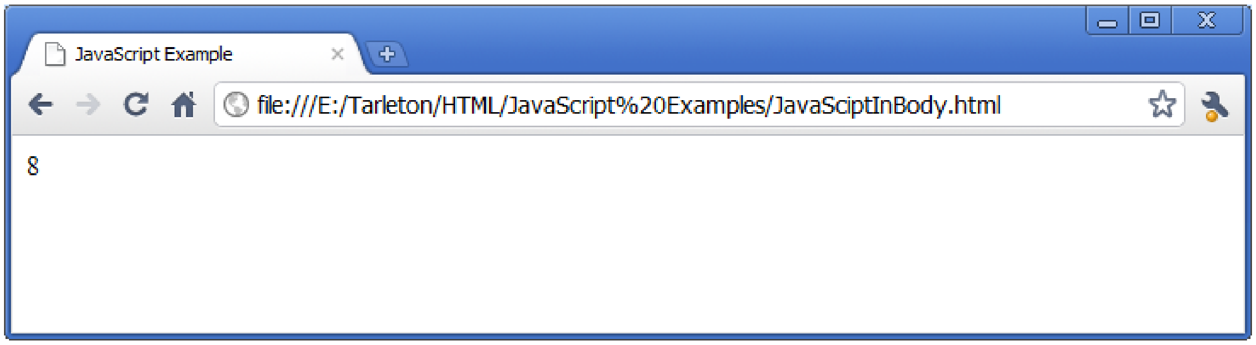
Notice that the C++ style comment can be used in JavaScript. Also notice that the <script> tag is not and should not be used in a JavaScript file.

In the HTML file, the <script> tag can also be used to indicate the location of a JavaScript file. The *src* attribute is assigned the path and filename of the file.

```
<html>
<head>
<title>JavaScript Example</title>
<script type="text/javascript" src="functions.js">
</script>
</head>

<body>
<script type="text/javascript">
document.write(multiply(2,4));
</script>
</body>

</html>
```


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
8
```

<script> Attributes

There are two attributes of the <script> tag that are of immediate interest:

- type – The value assigned to this attribute specifies the scripting language
- src – The location of an external scripting file

The *src* attribute specifies that the code is actually found in a file which should be loaded and then executed. The .js extension is normally used for JavaScript code files. The following example illustrates the use of these attributes.

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

JavaScript Language Elements

It is useful to discuss JavaScript in terms of language elements including:

- Variables
- Operators
- Expressions
- Statements
- Objects
- Functions and methods

Variables

Variables are used to hold data. A JavaScript identifier:

- Starts with a letter or underscore, and
- Is followed by letters, underscore or digits

JavaScript is a case-sensitive language

Scope

The scope of an identifier is either

- Global – An identifier that is accessible anywhere on the page
- Local – Is accessible only within the function it is declared within

A global variable is typically declared simply by assigning a value to it.

```
globalVariable = 100;
```

A local variable is declared within a function using the var keyword.

```
function someFunction() {
        var counter = 0;
        globalVariable = 100;
        ...
}
```

The identifier, *counter*, is local to the function and can only be used in that function. However, the identifier, *globalVariable*, is not preceded by the *var* keyword and is thus a global variable that can be used anywhere on the page, inside or outside of the function.

Data Types

There are six data types in JavaScript :

- Numbers – Integer or floating point numbers
- Booleans – Either true/false or a number (0 being false) can be used for boolean values
- Strings – Sequence of characters enclosed in a set of single or double quotes
- Objects – Entities that typically represents elements of a HTML page
- Null – No value assigned which is different from a 0
- Undefined – Is a special value assigned to an identifier after it has been declared but before a value has been assigned to it

JavaScript is a dynamically typed language. The data type of the identifier is not assigned when the identifier is declared. When a value is assigned to the identifier the identifier takes on that type. The data type of the variable is not important until an operator is applied to the variable. The behavior of the operator is dependent of the data type being acted upon.

For example:

```
var name = “Sally”
name = 34
```

The string, Sally, is first assigned to the variable. Next, the integer 34 is assigned to the variable. Both are legal but usage of the identifier is inconsistent. It is better if we are consistent when assigning a data type to a variable. This leads to less confusing code.

Literals

Literals are simple constants such as:

```
34
3.14159
“frog beaks”
‘/nTitle/n’
true
```

For string, escape sequence can be used to embed special values. An escape sequence consists of the back slash character followed by a character that has special meaning. Escape sequences recognized by JavaScript include:

Character	Meaning
\b	backspace
\f	form feed
\n	new line
\r	carriage return
\t	tab
\\	backslash character
\"	double quote
\’	Single quote
\ddd	Octal number
\xdd	Tow digit hexadecimal number
\xdddd	Four digit hexadecimal number

Operators

The JavaScript operators include:

Precedence	Operator	Associativity	Meaning
1	member	Left-to-right	.
			[]
	new	Right-to-left	new
2	function call	Left-to-right	()
3	++	n/a	Increment by 1
	--		Decrement by 1
4	!	Right-to-left	logical not
	~		bitwise not
	+		unary plus
	-		unary minus
	typeof		type of
	void		void
	delete		delete
5	*	Left-to-right	Multiplication
	/		Division
	%		Modulo division
6	+	Left-to-right	addition
	-		subtraction
7	<<	Left-to-right	shift left
	>>		shift right
	>>>		arithmetic shift right
8	>	Left-to-right	Greater than
	>=		Greater than or equal
	<		Less than
	<=		Less than or equal

Precedence	Operator	Associativity	Meaning
9	= =	Left-to-right	equality
	!=		not equal
	= = =		strict equality
	!= =		strict inequality
10	&	Left-to-right	bitwise and
11	^	Left-to-right	bitwise xor
12	|	Left-to-right	bitwise or
13	&&	Left-to-right	logical and
14	||	Left-to-right	logical or
15	(condition)?value1:value2	Right-to-left	tertiary operator
16	=	Right-to-left	assignment
	+=		
	-=		
	*=		
	/=		
	%=		
	<<=		
	>>=		
	>>>=		
	&=		
	^=		
	|=		
17	,	Left-to-right	comma operator

Arrays

Arrays are allocated using the *new* keyword.

```
names = new Array(10);
numbers = new Array(5);
```

Array indexes start at 0 and extend to the size of the array minus 1. To assign a value to an element of an array open and close brackets are used.

```
names[0] = "Rabbit";
names[1] = "Happy";
names[9] = "Dover";
```

The size of an array can be increased dynamically by assigning a value to an element pass the end of the array. Array can be created that initially has no elements at all. In addition, they are not of a fixed size but can grow dynamically.

```
pictures = new Array();
pictures[35] = "Mona Lisa";
```

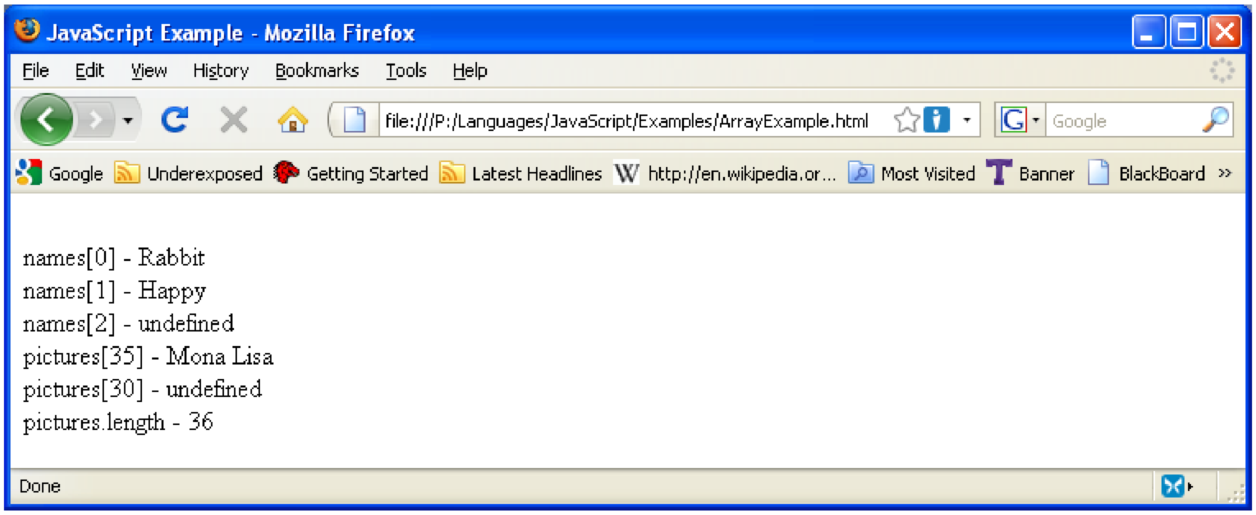
The array, pictures, initially has no elements. After "Mona Lisa" has been assigned the array has 36 elements. The unassigned elements are set to Undefined.

The *length* property of arrays returns the number of elements in the array.

```
<html>
<head>
<title>JavaScript Example</title>
<script language="JavaScript1.2">
<!--
names = new Array(10);
names[0] = "Rabbit";
names[1] = "Happy";
names[9] = "Dover";

document.write("<br>names[0]  - " + names[0] );
document.write("<br>names[1]  - " + names[1] );
document.write("<br>names[2]  - " + names[2] );
```

```
pictures = new Array();
pictures[35] = "Mona Lisa";
document.write("<br>pictures[35]  - " + pictures[35] );
document.write("<br>pictures[30]  - " + pictures[30] );
document.write("<br>pictures.length  - " +pictures.length);
//-->
</script>
</head>
<body>
</body>
</html>
```


JavaScript Example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScript/Examples/ArrayExample.html
Google
Google Underexposed Getting Started Latest Headlines http://en.wikipedia.or... Most Visited Banner BlackBoard
Done

```
names[0] - Rabbit
names[1] - Happy
names[2] - undefined
pictures[35] - Mona Lisa
pictures[30] - undefined
pictures.length - 36
```

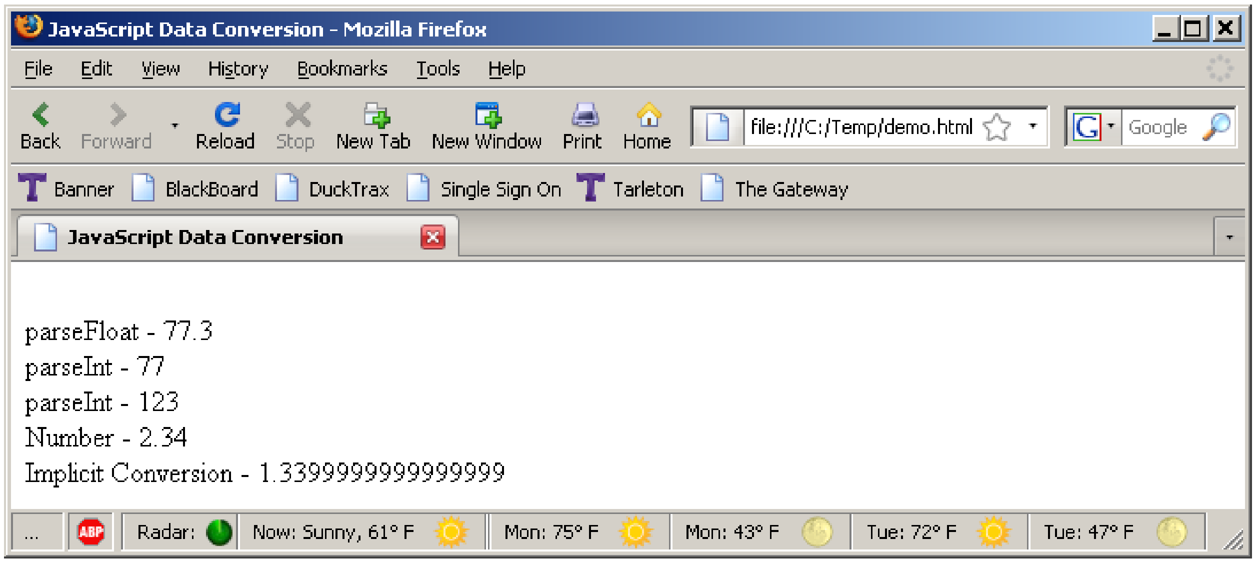
Converting Between Data Types

There are a number of techniques for converting between data types. To convert from a string several parse and other functions are available.

- parseFloat – Converts a string to a float
- parseInt – Converts a string to an integer
- Number – Converts a string to a number

The last example below uses an arithmetic expression to implicitly convert the string to a number.

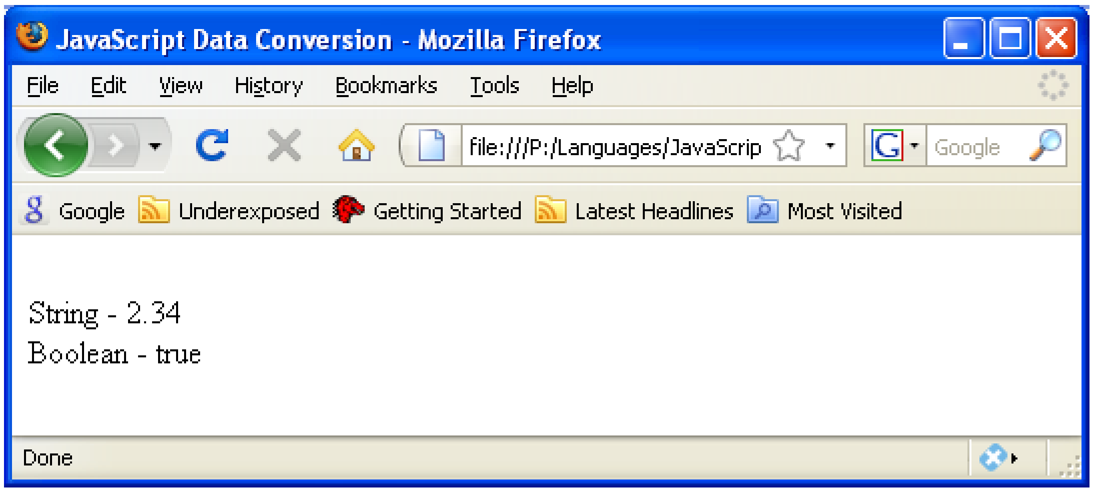
```
<html>
<head>
<title>JavaScript  Data Conversion</title>
<script language="JavaScript 1.2">
<!--
document.write("<br>parseFloat - " + parseFloat('77.3'));
document.write("<br>parseInt - " + parseInt('77'));
document.write("<br>parseInt - " + parseInt('123.45'));
document.write("<br>Number - " + Number("2.34"));
document.write("<br>Implicit Conversion - " + ("2.34"-1));
//-->
</script>
</head>
<body>
</body>
</html>
```


JavaScript Data Conversion - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop New Tab New Window Print Home
file:///C:/Temp/demo.html
Google
Banner BlackBoard DuckTrax Single Sign On Tarleton The Gateway
JavaScript Data Conversion
Radar: Now: Sunny, 61° F Mon: 75° F Mon: 43° F Tue: 72° F Tue: 47° F

```
parseFloat - 77.3
parseInt - 77
parseInt - 123
Number - 2.34
Implicit Conversion - 1.3399999999999999
```

A number can be converted to a string or Boolean using the String and Boolean functions.

```
<html>
<head>
<title> JavaScript  Data Conversion </title>
<script language="JavaScript 1.2">
document.write("<br> String  - " + String(2.34));
document.write("<br> Boolean  - " + Boolean(2.34));
</script>
</head>
<body>
</body>
</html>
```


JavaScript Data Conversion - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScrip
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

```
String - 2.34
Boolean - true
```

Regular Expressions

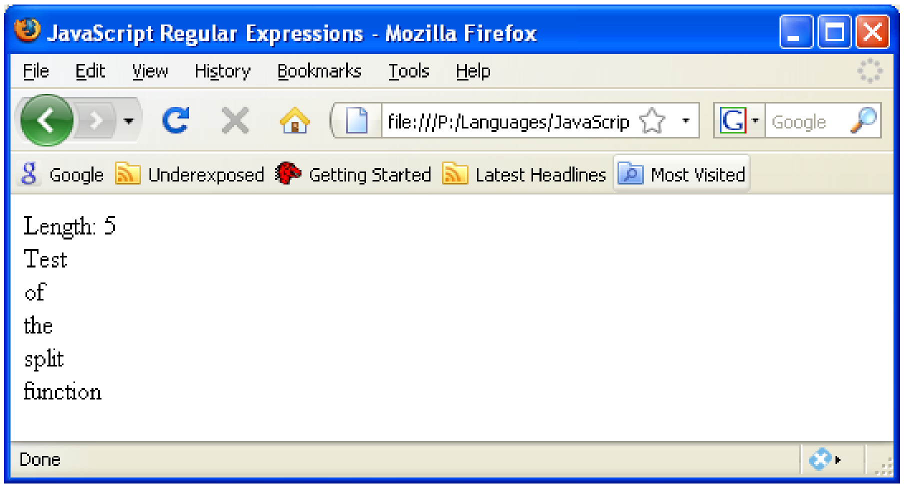
A regular expression is a way of performing pattern matching. A pattern is defined and then applied to a target string. The form of a regular expression and how they are applied to a target string varies somewhat between languages.

In JavaScript, a regular expression is defined using a series of characters that define the pattern enclosed in a pair of forward slashes. For example to match white spaces the \s is used.

re = /\s/g;

The *\s* means that all white spaces are to be matched and the *g* means that this needs to be applied to the entire target string. The split function can be used to illustrate this pattern. The split function is executed against a target string and will break the target up into individual string based on the split functions regular expression argument. The split function returns an array of strings.

```
<html>
<head>
<title>JavaScript  Data Conversion</title>
<script language="JavaScript 1.2">
re=/\s/g;
target="Test of the split function";
result = target.split(re);
document.write("Length: " + result.length + "<br>");
for(i=0;i<result.length;i++) {
        document.write(result[i]+"<br>");
}
</script>
</head>
<body>
</body>
</html>
```

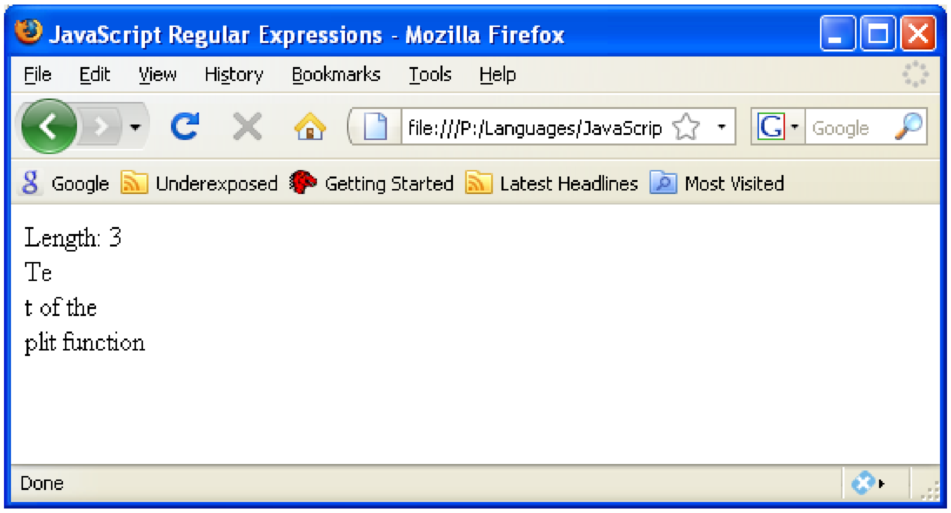

JavaScript Regular Expressions - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScrip
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

```
Length: 5
Test
of
the
split
function
```

There are several character sequences that have special meaning in a regular expression. The tutorial found at http://www.zytrax.com/tech/web/regex.htm provides an overview of regular expressions. Here we will look at only a few.

The \ is an escape sequence character which means do not treat the following character as a literal. Consider the following example:

```
…
re=/s/g;
target="Test of the split function";
result = target.split(re);
…
```


JavaScript Regular Expressions - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScrip
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

```
Length: 3
Te
t of the
plit function
```

The split function split the target based on the presence of the letter s. The \s in the previous example treated the s as a special character which represented white spaces. Other escape sequences include:

Escape Sequence	Meaning
\d	Any digit in the range 0-9
\s	White space
\w	Any character in the range 0-9, A-Z and a-z
\b	Match any character at the beginning of a word

These escape sequences are case sensitive. An upper case letter for these escape sequences generally means NOT. That is for \D match any character not in the range 0-9.

Metacharacters also convey special meaning in a regular expression.

Metacharacter	Meaning
[ ]	Match any character within the brackets
-	Is used within brackets to indicate a range [a-d]
^	When used within braces it means negation
^	When used outside of a set of brackets it means to match only at the beginning of a target ^First
$	Means to only match at the end of a target [word$]
.	Match any character at that position [ton.]

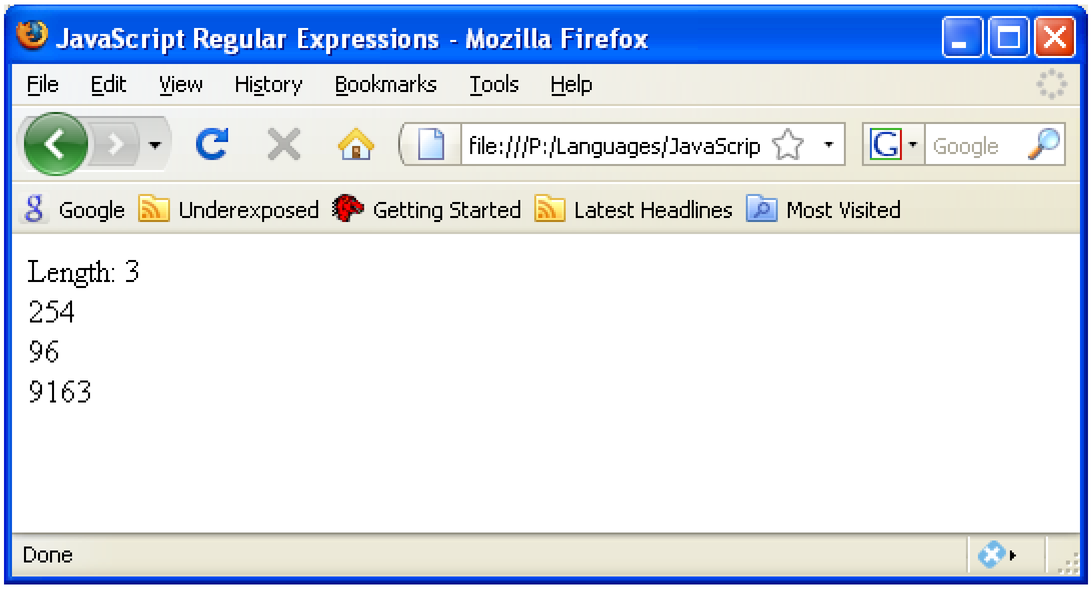
Using the regular expression:

```
re=/[ ]/;
target="Test of the split function";
result = target.split(re);
```

Results in the same output for /\s/ for this example.

The brackets and the dash is illustrated for a SSN.

```
re=/[-]/;
target="254-96-9163";
result = target.split(re);
```


JavaScript Regular Expressions - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScrip
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

```
Length: 3
254
96
9163
```

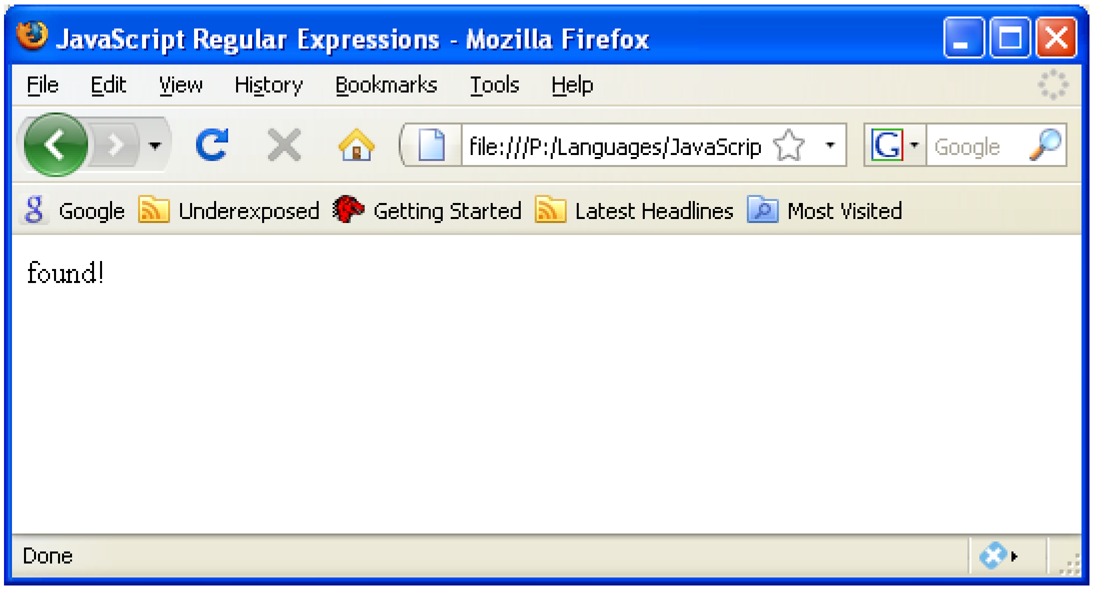
Regular Expression Functions

There are other JavaScript functions that use regular expressions other than the split function including:

- test – Will return true/false depending if a match occurs
- match – Returns a match if found
- search – Returns the index of the first match
- replace – Replaces matches with a given string

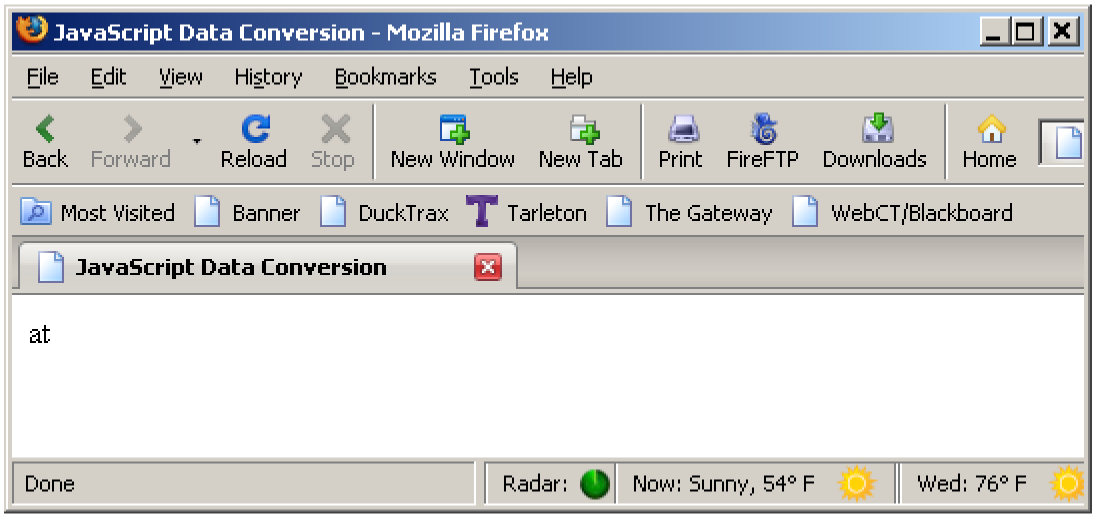
The test function will return a true or a false.

```
rexp = /at/
if(rexp.test("catalog")) {
        document.write("found!<br>");
} else {
        document.write("not found!<br>");
}
```


JavaScript Regular Expressions - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScrip
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Done

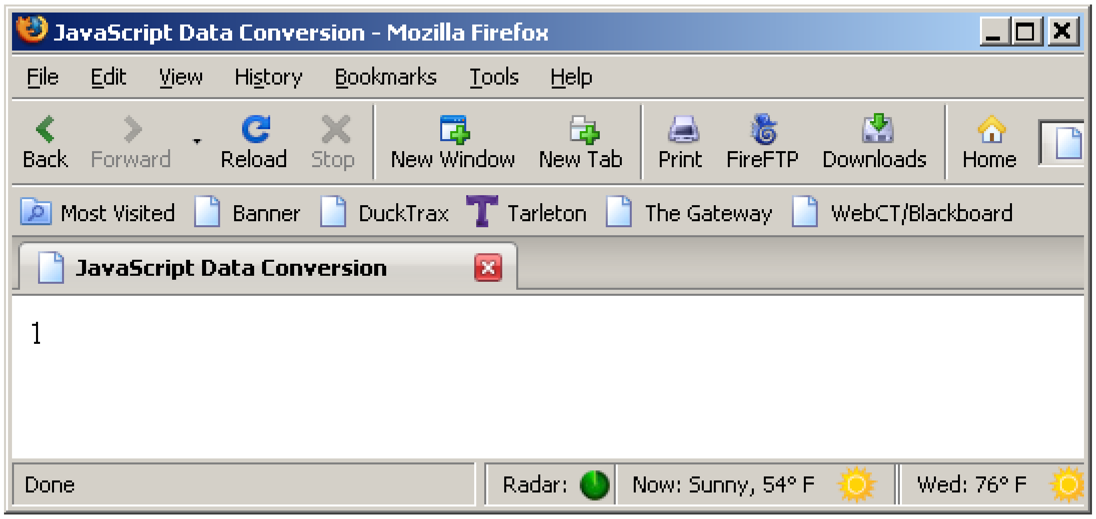
found!

```
<html>
<head>
<title>JavaScript  Data Conversion</title>
<script language="JavaScript 1.2">
rexp = /at/
document.write("catalog".match(rexp));
</script>
</head>
<body>
</body>
</html>
```


JavaScript Data Conversion - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop New Window New Tab Print FireFTP Downloads Home
Most Visited Banner DuckTrax Tarleton The Gateway WebCT/Blackboard
JavaScript Data Conversion
Done
Radar:
Now: Sunny, 54° F
Wed: 76° F

```
at
```

```
rexp = /at/
document.write("catalog".search(rexp));
```


JavaScript Data Conversion - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop New Window New Tab Print FireFTP Downloads Home
Most Visited Banner DuckTrax Tarleton The Gateway WebCT/Blackboard
JavaScript Data Conversion
Done Radar: Now: Sunny, 54° F Wed: 76° F

```
1
```

Math Object

The JavaScript Math object provides several properties and methods that can be useful.

Property	Description
E	Euler's number (~ 2.718)
LN2	the natural logarithm of 2
LN10	the natural logarithm of 10
LOG2E	the base-2 logarithm of E
LOG10E	the base-10 logarithm of E
PI	PI
SQRT1_2	the square root of 1/2
SQRT2	the square root of 2

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x (radians)
asin(x)	Returns the arcsine of x, in (radians)
atan(x)	Returns the arctangent of x as a value
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 (radians)
exp(x)	Returns the value of Ex
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 (radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

For example, to compute the area of a circle use the function:

```
function areaOfACircle(radius) {
        return Math.PI*radius*radius;
}
```

JavaScript Objects

There exist a number of predefined objects associated with the web browser and the HTML document loaded. Each of these objects has certain properties associated with them.

Document	Input Password
Events	Input Radio
Elements	Input Reset
Anchor	Input Submit
Area	Input Text
Base	Link
Body	Meta
Button	Object
Form	Option
Frame/IFrame	Select
Frameset	Style
Image	Table
Input Button	TableCell
nput Checkbox	TableRow
Input File	Textarea
Input Hidden	

An object frequently consists of sub elements which are separated by periods.

```
document.myform.text1.value
```

Objects also can have methods which are distinguished from properties by the use of the open and close parentheses. Here the values associated with the first form are reset.

```
document.forms[0].reset();
```

Window

The window object can be used to create new windows and dialog boxes and includes these method:

- Open – Opens a new window
- Close – Closes the window
- alert – Displays an alert message box
- confirm – Displays a confirms dialog box
- prompt – Displays a prompt dialog box

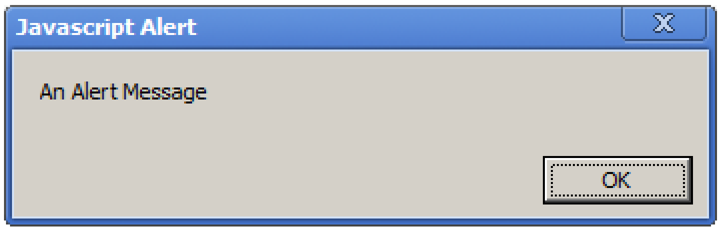
It also possesses several properties including:

- document – Returns the Document object
- innerHeight – The height of the content area of the window
- innerWidth – The width of the content area of the window
- outerHeight – The height or the window including toolbars
- innerWidth – The width of the window

Alert Message Box

The alert message box displays a simple message.

alert('An Alert Message');

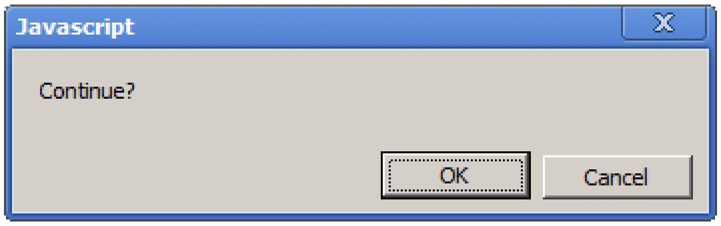


An Alert Message

Confirm Dialog Box

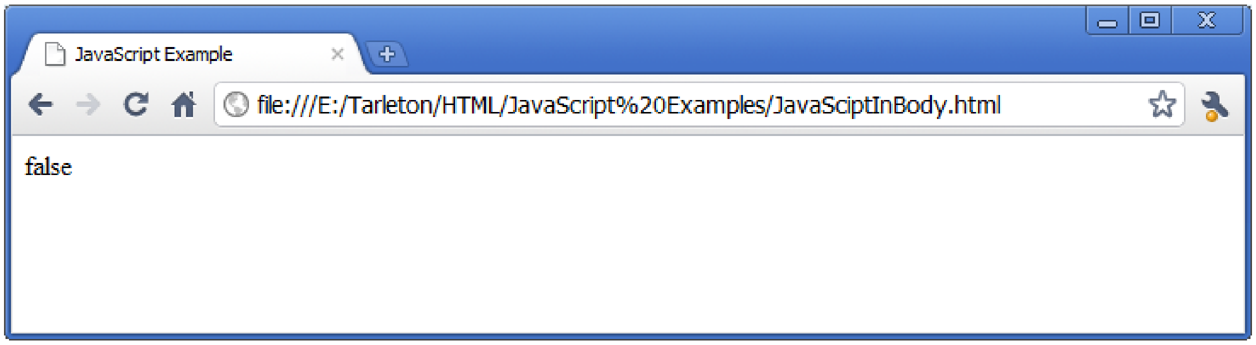
The confirm dialog box displays a confirm type message and then either returns a true or false value depending on which button is pressed.

var result = confirm("Continue?");
document.write(result);



Continue?

If Cancel is selected, false is returned.

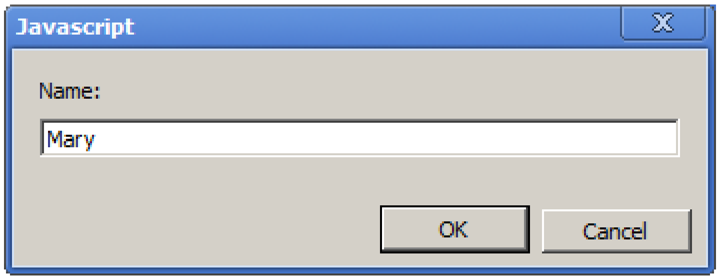

JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
false
```

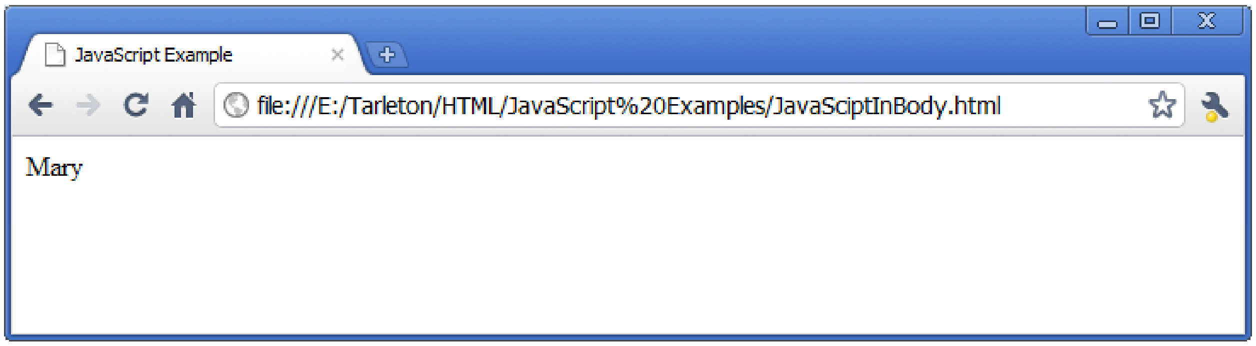
Prompt Dialog Box

The prompt dialog box provides a way of getting input from the user. The prompt function has two arguments. The first is the prompt message and the second is a default value if any.

```
var result = prompt("Name:","");
document.write(result);
```



The value returned is the value entered by the user.


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
Mary
```

Document

The Document object provides access to all of the HTML elements of the current page. Useful properties include:

- cookie – Will return name/value pairs of the cookies used by the document
- domain – Returns the domain name of the server
- title – Returns or set the title
- URL – Returns the URL of the document.

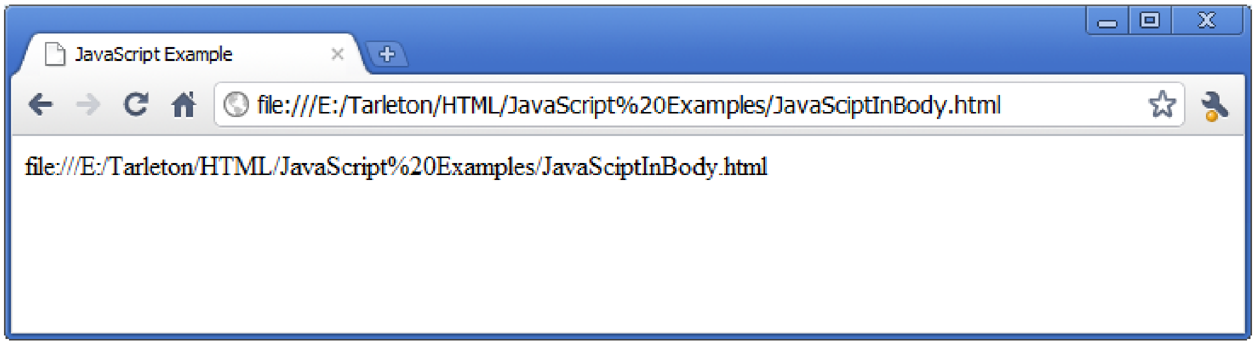
In addition, it consists of a series of array that hold the contents of the page. These objects can be accessed and modified. For example, the forms array contains a list of all of the forms that make up a page. Here the first form is selected. The value of the third element of the form is returned.

```
document.forms[0].elements[2].value
```

URL Property

The URL property is easy to use.

```
document.write(document.URL);
```


JavaScript Example
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html

```
file:///E:/Tarleton/HTML/JavaScript%20Examples/JavaSciptInBody.html
```

Frame

The Frame object refers to a frame of the web page. The Frames array is a list of the frames that make up a web page.

Properties of a frame include:

- frames – An array listing the frames that make up the page. Indexes start at 0
- length – The number of elements in the frames array
- self – Designates the current frame
- name – The name of the frame
- parent – The parent frame of the current frame

Methods of the frame object that are of interest include:

- blur – Removes the focus from the frame
- focus – Gives the frame focus
- setInterval -
- clearInterval
- setTimeout
- clearTimeout

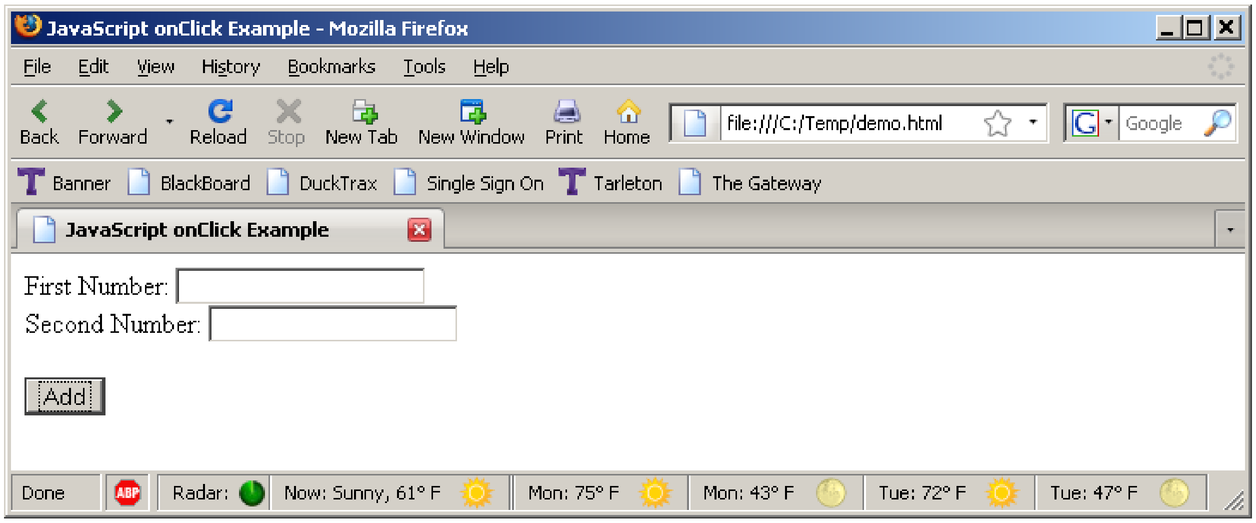
JavaScript Events

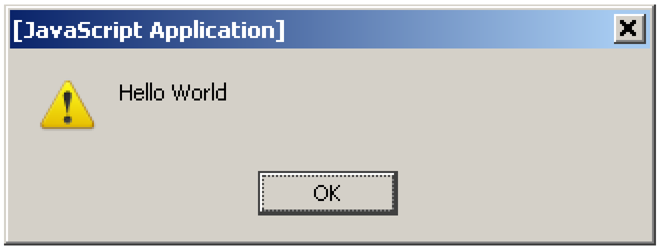
Many elements of DOM support events. These events are normally the result of some user actions.

Event	Meaning
onload	Occurs when a window or frame has loaded
onunload	Occurs when a document is removed from a window or frame
onclick	The mouse is clicked on an element
ondblclick	The double click event
onmousedown	Mouse down event
onmouseup	Mouse up event
onmouseover	Mouse moves onto an element
onmousemove	Mouse moves over an element
onmouseout	Mouse leaves an element
onfocus	Element receives focus
onblur	Element loses focus
onkeypress	Key press event
onkeydown	Key is pressed down
onkeyup	Key is released
onsubmit	Submit button is pressed
onreset	Form reset event occurs
onselect	Some text in an element is selected
onchange	Element loses focus and its value changes

onClick Example

```
<html>
<head>
<title>JavaScript  onClick Example</title>
<script language="JavaScript">
<!--
function popup() {
          alert("Hello World")
}
//-->
</script>
</head>
<body>
<form action="SampleServlet" method="POST">
      First Number: <input type="text" name="num1" size="20"><br>
      Second Number: <input type="text" name="num2" size="20">
      <br><br>
      <input type="submit" onclick="popup()"value="Add">
</form>
</body>
</html>
```


JavaScript onClick Example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop New Tab New Window Print Home
file:///C:/Temp/demo.html
Google
Banner BlackBoard DuckTrax Single Sign On Tarleton The Gateway
JavaScript onClick Example
First Number:
Second Number:
Add
Done
Radar:
Now: Sunny, 61° F
Mon: 75° F
Mon: 43° F
Tue: 72° F
Tue: 47° F



Hello World

Animation

JavaScript does not have a function such as Java's sleep method that pauses a task for a specified period of time. However, JavaScript has two functions that can be used to delay the execution of a function.

- **setTimeout** – Will execute a function a specific number of milliseconds in the future
- **setInterval** – Will execute a function every milliseconds

Both functions take on two arguments:

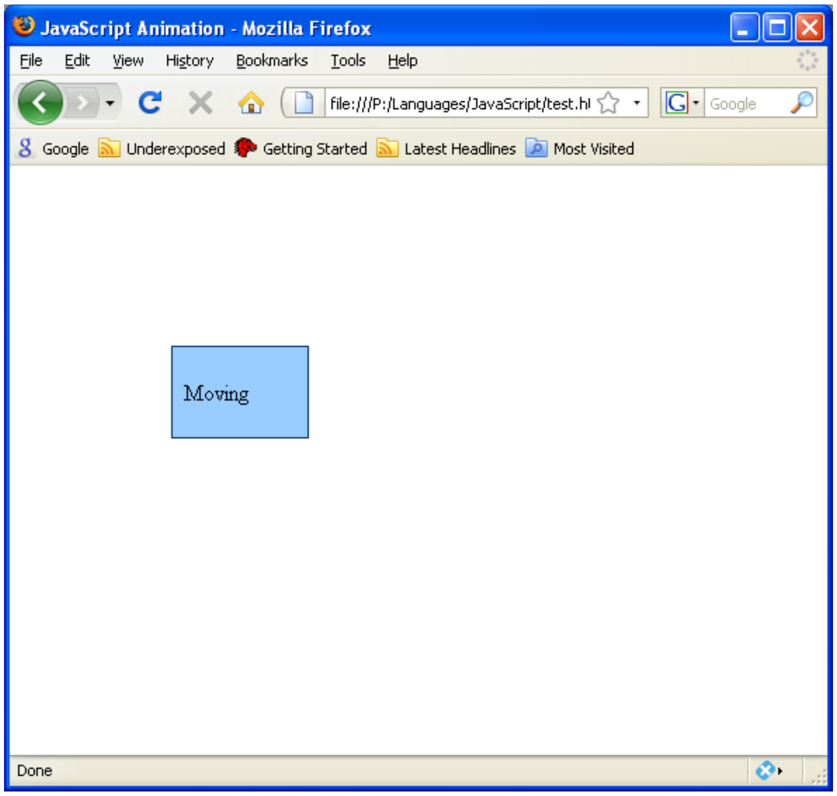
- Function – The first argument identifies the function to execute
- Time – The number of milliseconds

```
setTimeout(someFunction,500);        // The function will be executed 500 milliseconds
                                     // in the future
setInterval(someFunction,500);       // The function will be executed 500 every milliseconds
```

The use of the setTimeout is illustrated here by moving a <div> tag across the screen. The int function setups the animation by retrieving a reference to the tag and calling the move function. The function move modifies the position of the tag and recursively schedules itself for future invocation.

```
function move() {
        square.style.left = parseInt(square.style.left)+1+'px';
        setTimeout(move,20);
}

function init() {
        square = document.getElementById('Square');
        square.style.left = '0px';
        move();
}
```


JavaScript Animation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///P:/Languages/JavaScript/test.ht
Google
Google Underexposed Getting Started Latest Headlines Most Visited
Moving
Done

The complete page follows:

```
<html>
<head>
<title>JavaScript  Animation</title>


<script language="JavaScript 1.2">
var square = null;

function move() {
        square.style.left = parseInt(square.style.left)+1+'px';
        setTimeout(move,20);
}

function init() {
        square = document.getElementById('Square');
        square.style.left = '0px';
        move();
}

window.onload = init;
```

```
</script>
</head>
<body>
<br>
<div id="Square" style="position:absolute;
        left:0px;
        top:8em;
        width:5em;
        line-height:3em;
        background:#99ccff;
        border:1px solid #003366;
        white-space:nowrap;
        padding:0.5em;"
>
Moving
</div>

</body>
</html>
```

The same effect can be created using the setInterval function.

```
function move() {
        square.style.left = parseInt(square.style.left)+1+'px';
}

function init() {
        square = document.getElementById('Square');
        square.style.left = '0px';
        setInterval(move,20);
}
```