

JavaScript Tutorial

Table of Contents

Introduction	3
Introduction	4
JavaScript Attributes	6
JavaScript Language Elements	7
Variables.....	7
Scope.....	7
Data Types.....	9
Literals.....	10
Operators	11
Arrays	12
Converting Between Data Types.....	14
Regular Expressions	16
Regular Expression Functions	20
JavaScript Objects	23
Window	24
Document.....	25
Frame	26
JavaScript Events.....	27
Animation.....	28

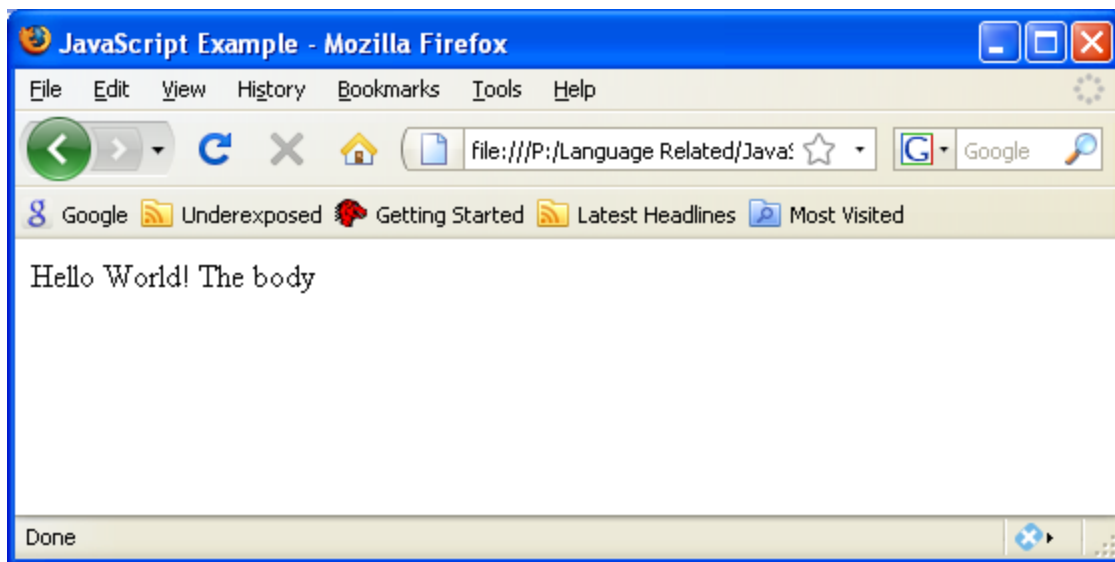
Introduction

Introduction

JavaScript is code that is embedded within a browser. The code is executed within the browser and is referred to as client side execution. JavaScript code is embedded within an HTML page using the JavaScript tag. The code is normally placed within the HTML Header tag.

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

Notice that the JavaScript code is placed within the head section of the file. This code is executed when the page is loaded. The write method displays the text and then the body of the page is displayed.



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.

JavaScript Attributes

There are several tags supported in the JavaScript attributes including:

- language
- src

The language attribute of the JavaScript tag is optional but allows the browser to determine which version of JavaScript is being used.

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 language="JavaScript 1.2" 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 any number/0 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”
true
```

There value is as specified. 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.

Character	Meaning
<code>\b</code>	backspace
<code>\f</code>	form feed
<code>\n</code>	new line
<code>\r</code>	carriage return
<code>\t</code>	tab
<code>\\</code>	backslash character
<code>\"</code>	double quote

Operators

The JavaScript operators include:

Operator	Meaning
+	Addition
*	Multiplication
/	Division
%	Modulo division
++	Increment by 1
--	Decrement by 1
-	Subtraction
==	Equality
!=	Not equal
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
&&	And
	Or
!	Not
+	String concatenation
=	Assignment
+= -= ...	Compound assignment
(condition)?value1:value2	Tertiary operator
typeof	Returns the type of the data
Regular Expressions	

Arrays

Arrays are declared using the new Array sequence:

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

The new keyword is used to create a new object. Thus arrays are objects which have properties as will be discussed shortly.

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 by assigning a value to an element past the end of the array. Array can be created that initially has no elements at all.

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

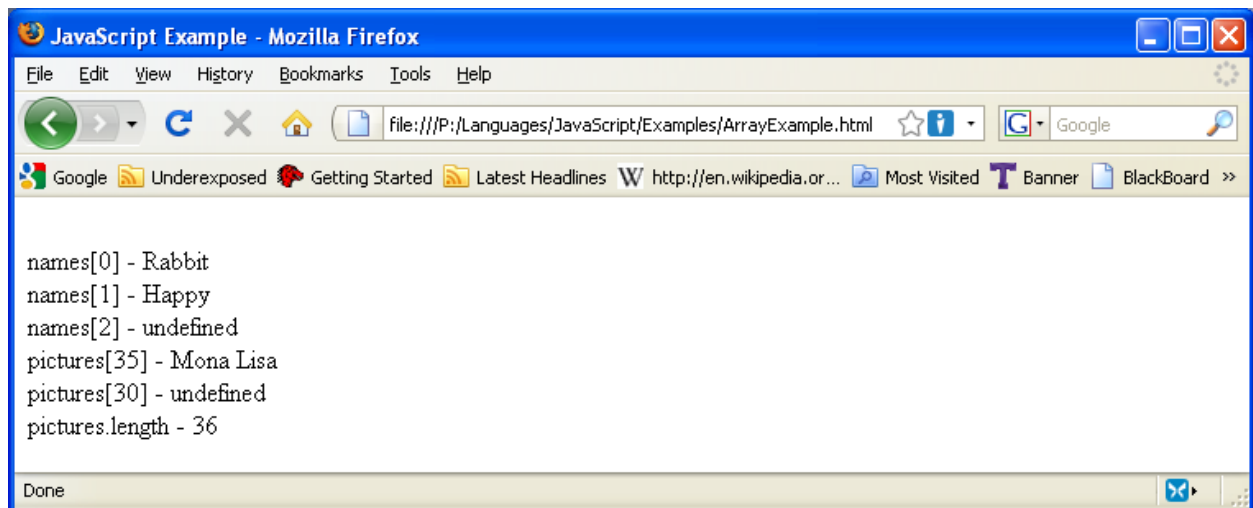
The array, pictures, initially has no elements. After "Mona Lisa" has been assigned the array has 36 elements. The unassigned elements are set to null.

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

```
<html>  
<head>  
<title>JavaScript Example</title>  
<script language="JavaScript 1.2">  
<!--  
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>
```



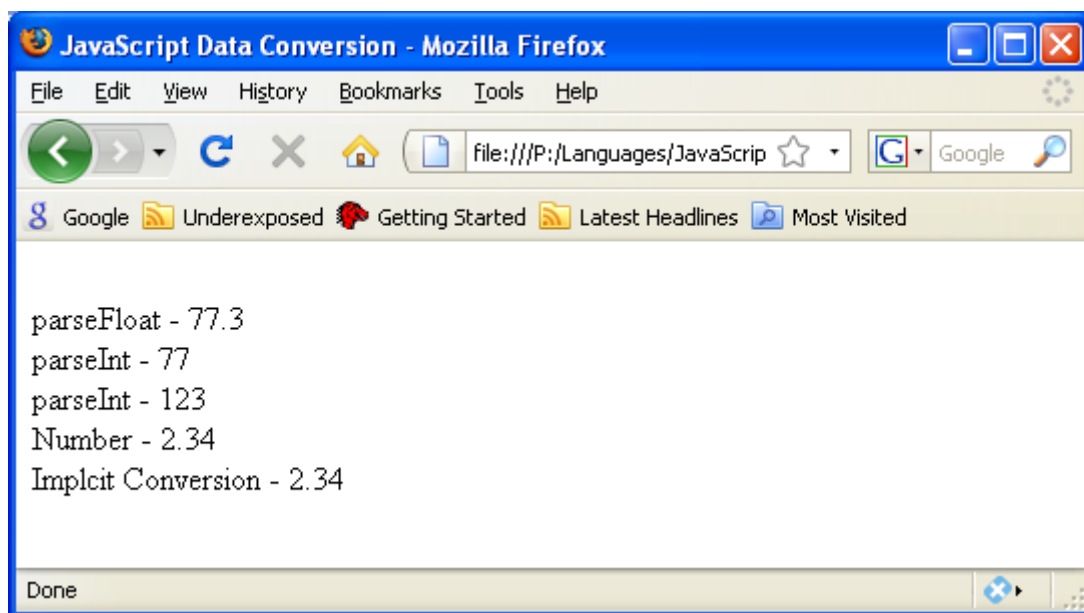
Converting Between Data Types

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

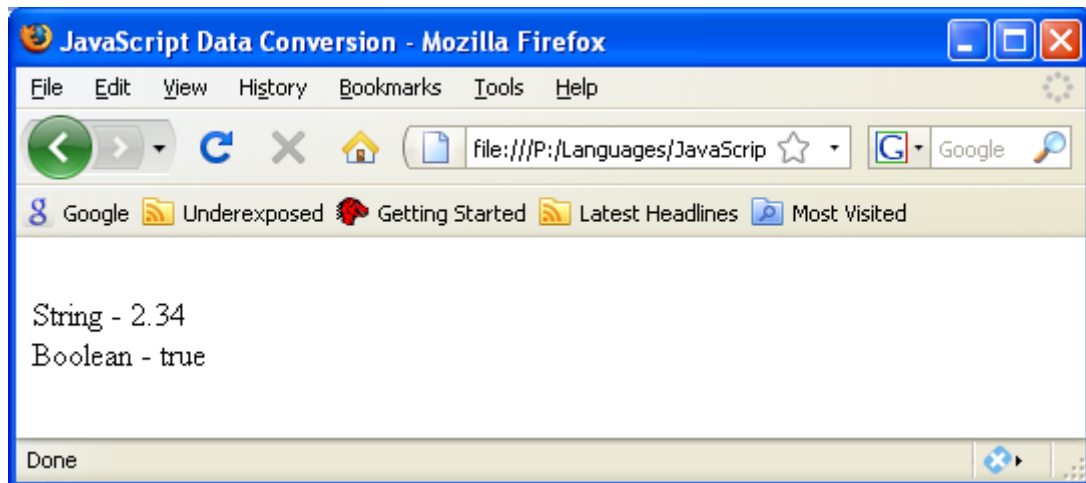
The last example below uses an arithmetic expression to implicitly convert the string to a number.

```
<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"-0));
//-->
</script>
</head>
<body>
</body>
</html>
```



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>
```



Regular Expressions

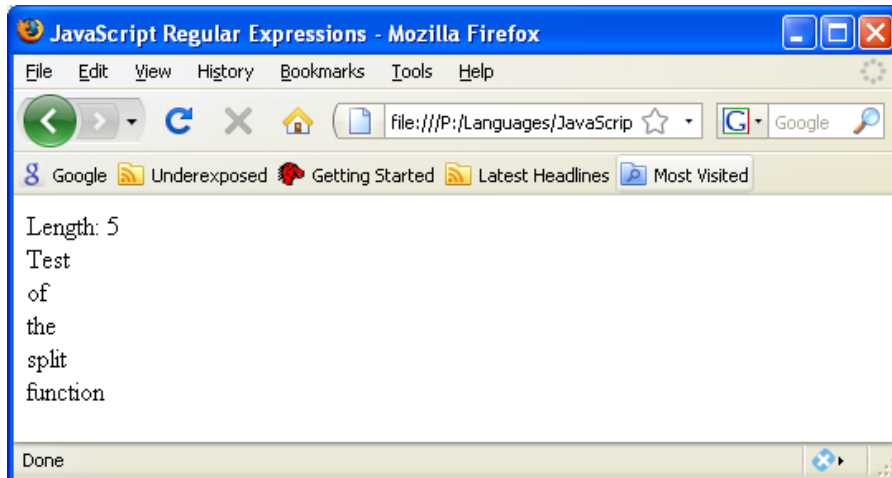
A regular expression is a way of performing pattern matching. A pattern is defined and then applied to a target string. The form of a regular expression and how they are applied to a target string varies somewhat between languages.

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.

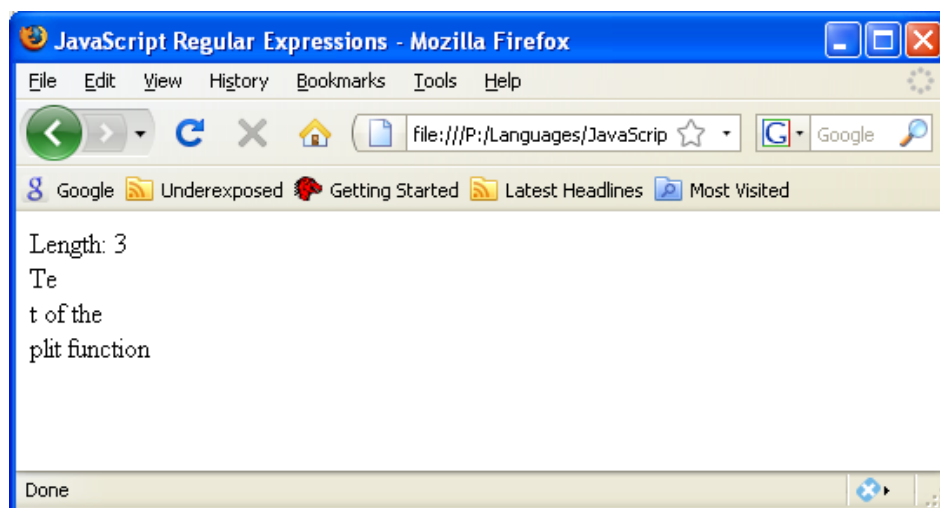
```
<html>
<head>
<title>JavaScript Data Conversion</title>
<script language="JavaScript 1.2">
re=/\s/g;
target="Test of the split function";
result = target.split(re);
document.write("Length: " + result.length + "<br>");
for(i=0;i<result.length;i++) {
    document.write(result[i]+"<br>");
}
</script>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



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);
...
```



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.]

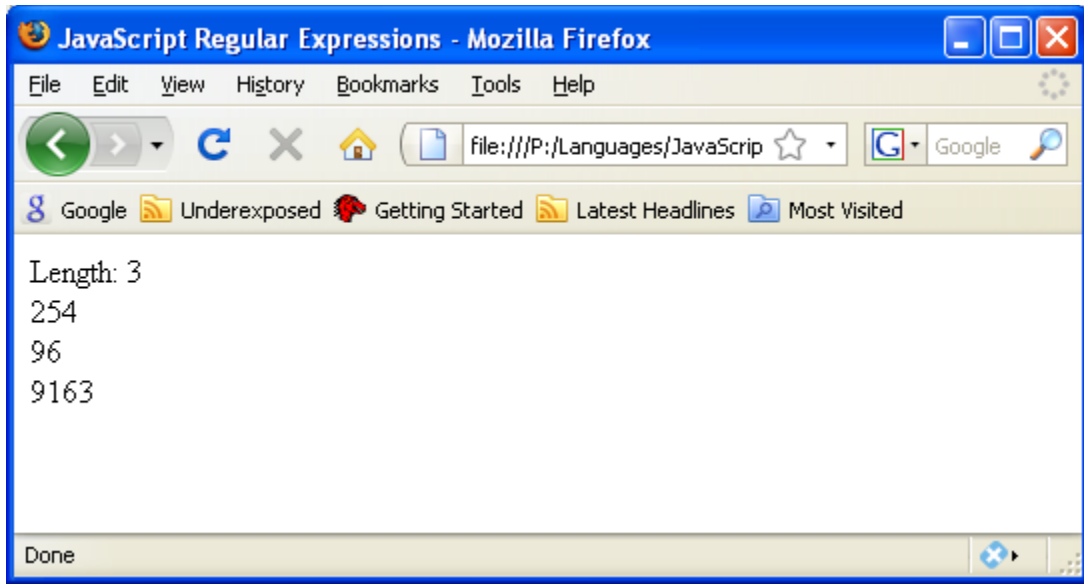
Using the regular expression:

```
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);
```



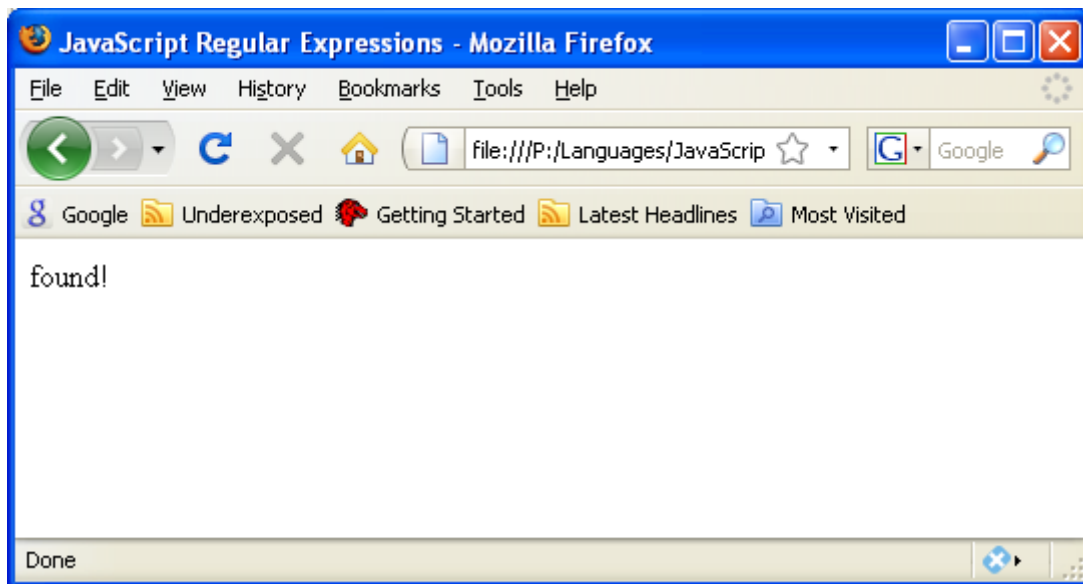
Regular Expression Functions

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>");  
}
```



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



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



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.

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

Document

The document object contains the structure of the HTML page. 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
```

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

```
<script type="text/JavaScript ">
<!--
function popup() {
    alert("Hello World")
}
//-->
</script>

...

<input type="button" value="Click Me!" onclick="popup()"></input>
```

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:

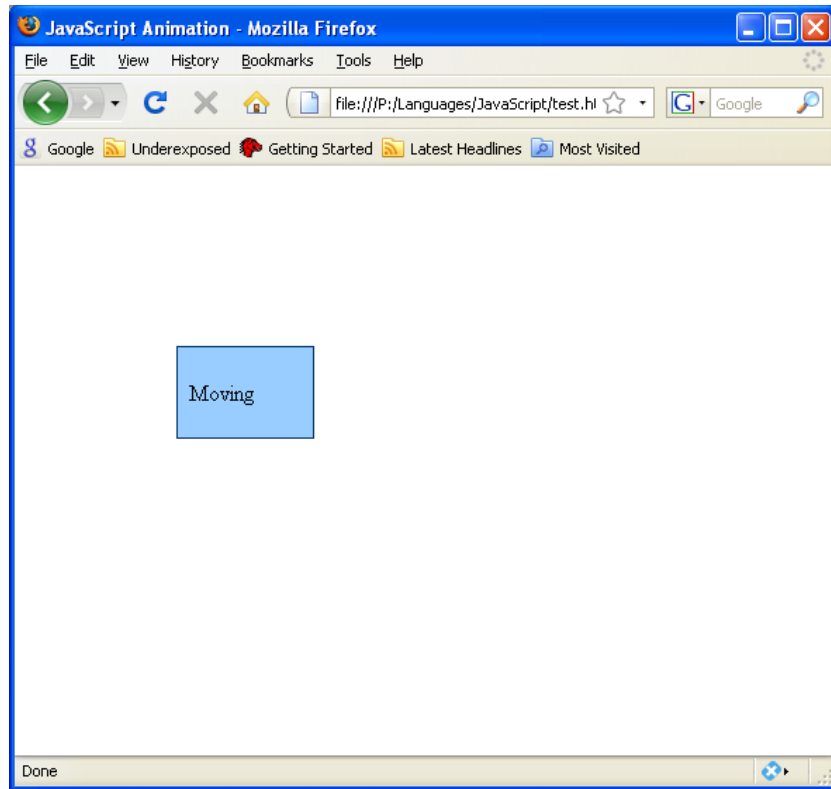
- `Function` – The first argument identifies the function to execute
- `Time` – The number of milliseconds

```
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 `init` 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 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();
}
```



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);
}
```