
GlobalLogic

JavaScript

Basic to Intermediate

Agenda

01 Basics

02 Control Statements

03 Functions

04 Events

05 Page Redirect

06 Cookies

07 Implicit Objects

01 Basics

Basics

Brief Introduction

- JavaScript is a lightweight, interpreted programming language
  - ✓Designed for creating network-centric applications
  - ✓Complementary to and integrated with Java
  - ✓Complementary to and integrated with HTML
  - ✓Open and cross-platform

Basics

Advantage

❑**Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

❑**Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.

❑**Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

❑**Richer interfaces:** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Basics

Disadvantage

It lacks the following important features:

❑Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.

❑JavaScript can not be used for Networking applications because there is no such support available.

❑JavaScript doesn't have any multithreading or multiprocess capabilities.

Basics

Datatype and variables

- JavaScript allows you to work with three primitive data types:
  - Numbers eg. 123, 120.50 etc.
  - Strings of text e.g. "This text string" etc.
  - Boolean e.g. true or false.

- JavaScript also defines two trivial data types, null and undefined, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as object.

- Variables are declared with the **var** keyword as follows:

```
<script type="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

```
<script type="text/javascript">
<!--
var money, name;
//-->
</script>
```

```
<script type="text/javascript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

Basics

Datatype and variables

Scope

- **Global Variables:** A global variable has global scope which means it is defined everywhere in your JavaScript code.

- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
   var myVar = "local";  // Declare a local variable
   document.write(myVar);
}
//-->
</script>
```

This produces the following result:

```
local
```

02 Control Statements

Control Statements

Decision Control

- If statement
- If…else statement
- If …elseif statement
- Switch case

Loop Control

- For loop
- For In loop
- While loop

Control Statements

If statement

```
if (expression){
   Statement(s) to be executed if expression is true
}
```

```
<script type="text/javascript">
<!--
var age = 20;
if( age > 18 ){
   document.write("<b>Qualifies for driving</b>");
}
//-->
</script>
```

This will produce following result:

```
Qualifies for driving
```

Control Statements

If…else…. statement

Syntax:

```
if (expression){
   Statement(s) to be executed if expression is true
}else{
   Statement(s) to be executed if expression is false
}
```

Example:

```
<script type="text/javascript">
<!--
var age = 15;
if( age > 18 ){
   document.write("<b>Qualifies for driving</b>");
}else{
   document.write("<b>Does not qualify for driving</b>");
}
//-->
</script>
```

This will produce following result:

```
Does not qualify for driving
```

Control Statements

If…elseif…. statement

```
if (expression 1){
   Statement(s) to be executed if expression 1 is true
}else if (expression 2){
   Statement(s) to be executed if expression 2 is true
}else if (expression 3){
   Statement(s) to be executed if expression 3 is true
}else{
   Statement(s) to be executed if no expression is true
}
```

Control Statements

If…elseif…. statement

Example:

```
<script type="text/javascript">
<!--
var book = "maths";
if( book == "history" ){
   document.write("<b>History Book</b>");
}else if( book == "maths" ){
   document.write("<b>Maths Book</b>");
}else if( book == "economics" ){
   document.write("<b>Economics Book</b>");
}else{
  document.write("<b>Unknown Book</b>");
}
//-->
</script>
```

This will produce following result:

```
Maths Book
```

Control Statements

Switch case

Example:

Following example illustrates a basic while loop:

```
<script type="text/javascript">
<!--
var grade='A';
document.write("Entering switch block<br />");
switch (grade)
{
  case 'A': document.write("Good job<br />");
            break;
  case 'B': document.write("Pretty good<br />");
            break;
  case 'C': document.write("Passed<br />");
            break;
  case 'D': document.write("Not so good<br />");
            break;
  case 'F': document.write("Failed<br />");
            break;
  default:  document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
```

This will produce following result:

```
Entering switch block
Good job
Exiting switch block
```

Control Statements

Loop Control – While Loop

```
while (expression){
   Statement(s) to be executed if expression is true
}
```

```
<script type="text/javascript">
<!--
var count = 0;
document.write("Starting Loop" + "<br />");
while (count < 10){
  document.write("Current Count : " + count + "<br />");
  count++;
}
document.write("Loop stopped!");
//-->
</script>
```

Control Statements

Loop Control – While Loop

This will produce following result:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

Control Statements

Loop Control – Do ….While Loop

Syntax:

```
do{
   Statement(s) to be executed;
} while (expression);
```

```
<script type="text/javascript">
<!--
var count = 0;
document.write("Starting Loop" + "<br />");
do{
  document.write("Current Count : " + count + "<br />");
  count++;
}while (count < 0);
document.write("Loop stopped!");
//-->
</script>
```

This will produce following result:

```
Starting Loop
Current Count : 0
Loop stopped!
```

Control Statements

Loop Control – For Loop

Syntax:

```
for (initialization; test condition; iteration statement){
     Statement(s) to be executed if test condition is true
}
```

```
<script type="text/javascript">
<!--
var count;
document.write("Starting Loop" + "<br />");
for(count = 0; count < 10; count++){
  document.write("Current Count : " + count );
  document.write("<br />");
}
document.write("Loop stopped!");
//-->
</script>
```

Control Statements

Loop Control – For Loop

This will produce following result:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

Control Statements

Loop Control – For in loop

```
<script type="text/javascript">
<!--
var aProperty;
document.write("Navigator Object Properties<br /> ");
for (aProperty in navigator)
{
  document.write(aProperty);
  document.write("<br />");
}
document.write("Exiting from the loop!");
//-->
</script>
```

Control Statements

Loop Control – For in loop

```
<script type="text/javascript">
<!--
var aProperty;
document.write("Navigator Object Properties<br /> ");
for (aProperty in navigator)
{
  document.write(aProperty);
  document.write("<br />");
}
document.write("Exiting from the loop!");
//-->
</script>
```

This will produce following result:

```
Navigator Object Properties
appCodeName
appName
appMinorVersion
cpuClass
platform
plugins
opsProfile
userProfile
systemLanguage
userLanguage
appVersion
userAgent
onLine
cookieEnabled
mimeTypes
Exiting from the loop!
```

03 Functions

Functions

Functions

- Function Call

```
<script type="text/javascript">
<!--
sayHello();
//-->
</script>
```

```
<script type="text/javascript">
<!--
function functionname(parameter-list)
{
  statements
}
//-->
</script>
```

Example:

A simple function that takes no parameters called sayHello is defined here:

```
<script type="text/javascript">
<!--
function sayHello()
{
   alert("Hello there");
}
//-->
</script>
```

Functions

Functions with return

- Function Call

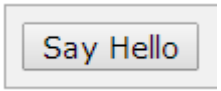
```
<script type="text/javascript">
<!--
function concatenate(first, last)
{
   var full;

   full = first + last;
   return  full;
}
//-->
</script>
```

```
<script type="text/javascript">
<!--
   var result;
   result = concatenate('Zara', 'Ali');
   alert(result );
//-->
</script>
```

04 Events

Events

- Events are a part of the Document Object Model (DOM) Level 3 and every HTML element have a certain set of events which can trigger JavaScript Code.



Example: onclick Event Type:

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
   alert("Hello World")
}
//-->
</script>
</head>
<body>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

Events

❑ Calling a *validate()* function before submitting a form data to the webserver. If *validate()* function returns true the form will be submitted otherwise it will not submit the data.

onsubmit event type:

Example:

```
<html>
<head>
<script type="text/javascript">
<!--
function validation() {
   all validation goes here
   .........
   return either true or false
}
//-->
</script>
</head>
<body>
<form method="POST" action="t.cgi" onsubmit="return validate()">
.......
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Events

❑ This example help you to create nice effects with images or even with text as well. The *onmouseover* event occurs when you bring your mouse over any element and the*onmouseout* occurs when you take your mouse out from that element.

onmouseover and onmouseout:

Example:

Following example shows how a division reacts when we bring our mouse in that division:

```
<html>
<head>
<script type="text/javascript">
<!--
function over() {
   alert("Mouse Over");
}
function out() {
   alert("Mouse Out");
}
//-->
</script>
</head>
<body>
<div onmouseover="over()" onmouseout="out()">
<h2> This is inside the division </h2>
</div>
</body>
</html>
```

Events

Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released
onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released

05 Page Redirect

Page Redirect

❑ When you click a URL to reach to a page X but internally you are directed to another page Y that simply happens because of page re-direction.

❑Reasons for page redirect:

- ❑ You did not like the name of your domain and you are moving to a new one. Same time you want to direct your all visitors to new site. In such case you can maintain your old domain but put a single page with a page re-direction so that your all old domain visitors can come to your new domain.

- ❑You have build-up various pages based on browser versions or their names or may be based on different countries, then instead of using your server side page redirection you can use client side page redirection to land your users on appropriate page.

- ❑The Search Engines may have already indexed your pages. But while moving to another domain then you would not like to lose your visitors coming through search engines. So you can use client side page redirection. But keep in mind this should not be done to make search engine a fool otherwise this could get your web site banned.

Page Redirect

Example 1:

```
<head>
<script type="text/javascript">
<!--
   window.location="http://www.newlocation.com";
//-->
</script>
</head>
```

Example 2:

```
<head>
<script type="text/javascript">
<!--
function Redirect()
{
    window.location="http://www.newlocation.com";
}

document.write("You will be redirected to main page in 10 sec.");
setTimeout('Redirect()', 10000);
//-->
</script>
</head>
```

06 Cookies

Cookies

❑ Cookies is the most efficient method of remembering and tracking preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

❑Cookies are a plain text data record of 5 variable-length fields:

- ✓**Expires** : The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- ✓**Domain** : The domain name of your site.
- ✓**Path** : The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- ✓**Secure** : If this field contains the word "secure" then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- ✓**Name=Value :** Cookies are set and retrieved in the form of key and value pairs.

Storing Cookies

Syntax:

```
document.cookie = "key1=value1;key2=value2;expires=date";
```

Storing Cookies

Example:

Following is the example to set a customer name in *input* cookie.

```
<html>
<head>
<script type="text/javascript">
<!--
function WriteCookie()
{
   if( document.myform.customer.value == "" ){
      alert("Enter some value!");
      return;
   }

   cookievalue= escape(document.myform.customer.value) + ";";
   document.cookie="name=" + cookievalue;
   alert("Setting Cookies : " + "name=" + cookievalue );
}
//-->
</script>
</head>
<body>
<form name="myform" action="">
Enter name: <input type="text" name="customer"/>
<input type="button" value="Set Cookie" onclick="WriteCookie();"/>
</form>
</body>
</html>
```

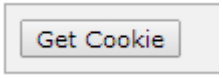
Storing Cookies

Example:

Following is the example to set a customer name in *input* cookie.

```
<html>
<head>
<script type="text/javascript">
<!--
function WriteCookie()
{
   if( document.myform.customer.value == "" ){
      alert("Enter some value!");
      return;
   }

   cookievalue= escape(document.myform.customer.value) + ";";
   document.cookie="name=" + cookievalue;
   alert("Setting Cookies : " + "name=" + cookievalue );
}
//-->
</script>
</head>
<body>
<form name="myform" action="">
Enter name: <input type="text" name="customer"/>
<input type="button" value="Set Cookie" onclick="WriteCookie();"/>
</form>
</body>
</html>
```

Reading Cookies



Example:

Following is the example to get the cookies set in previous section.

```
<html>
<head>
<script type="text/javascript">
<!--
function ReadCookie()
{
   var allcookies = document.cookie;
   alert("All Cookies : " + allcookies );

   // Get all the cookies pairs in an array
   cookiearray  = allcookies.split(';');

   // Now take key value pair out of this array
   for(var i=0; i<cookiearray.length; i++){
      name = cookiearray[i].split('=')[0];
      value = cookiearray[i].split('=')[1];
      alert("Key is : " + name + " and Value is : " + value);
   }
}
//-->
</script>
</head>
<body>
<form name="myform" action="">
<input type="button" value="Get Cookie" onclick="ReadCookie()"/>
</form>
</body>
</html>
```

Deleting Cookies

Example:

The following example illustrates how to delete cookie by setting expiration date one Month in past :

```
<html>
<head>
<script type="text/javascript">
<!--
function WriteCookie()
{
   var now = new Date();
   now.setMonth( now.getMonth() - 1 );
   cookievalue = escape(document.myform.customer.value) + ";"
   document.cookie="name=" + cookievalue;
   document.cookie = "expires=" + now.toUTCString() + ";"
   alert("Setting Cookies : " + "name=" + cookievalue );
}
//-->
</script>
</head>
<body>
<form name="formname" action="">
Enter name: <input type="text" name="customer"/>
<input type="button" value="Set Cookie" onclick="WriteCookie()"/>
</form>
</body>
</html>
```

07 Implicit Objects

Implicit Objects

- **String :** series of characters and wraps Javascript's string primitive data type with a number of helper methods.
- **Array:** let's you store multiple values in a single variable.
- **Date:** datatype built into the JavaScript language
- Objects: Follows OOPs principals

Implicit Objects

❑**String :**

Syntax:

Creating a **String** object:

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

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unicode value of the character at the given index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
localeCompare()	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

Implicit Objects

❑Array :

Creating a **Array** object:

```
var fruits = new Array( "apple", "orange", "mango" );
```

You can create array by simply assigning values as follows:

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows:

- fruits[0] is the first element
- fruits[1] is the second element
- fruits[2] is the third element

Implicit Objects

❑**Date :**

Syntax:

Here are different variant of Date() constructor:

```
new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond ])
```

Implicit Objects

❑**Objects :**

The syntax for adding a property to an object is:

```
objectName.objectProperty = propertyValue;
```

```
var str = document.title;
```

Example 1:

This example demonstrates how to create an object:

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
var book = new Object();   // Create the object
    book.subject = "Perl"; // Assign properties to the object
    book.author  = "Mohtashim";
</script>
</head>
<body>
<script type="text/javascript">
   document.write("Book name is : " + book.subject + "<br>");
   document.write("Book author is : " + book.author + "<br>");
</script>
</body>
</html>
```

Thank you

Please send your questions/ suggestions to subnesh.sharma@globallogic.com

Subnesh Sharma