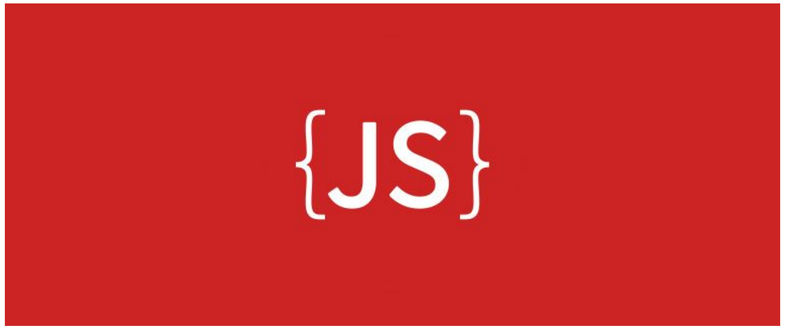

Javascript Fundamentals

Introduction

What is Javascript?

JavaScript (JS for short) is the programming language that enables web pages to respond to user interaction beyond the basic level. It was created in 1995, and is today one of the most famous and used programming languages.

What is Javascript?

JavaScript can be implemented using JavaScript statements that are placed within the HTML tags in a web page.

You can place the <script> tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the <head> tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
    JavaScript code
</script>
```

Your First JavaScriptCode

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function document.write which writes a string into our HTML document.

This function can be used to write text, HTML, or both. Take a look at the following code.

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

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">
    <!--
     var1 = 10
     var2 = 20
    //-->
</script>
```

But when formatted in a single line as follows, you must use semicolons:

```
<script language="javascript" type="text/javascript">
    <!--
    var1 = 10; var2 = 20;
    //-->
</script>
```

Note: It is a good programming practice to use semicolons

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords,variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers Time and TIME will convey different meanings in JavaScript.

NOTE: Care should be taken while writing variable and function names in JavaScript.

Comments in JavaScript

JavaScript supports both C-style and C++-style comments. Thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--.
- JavaScript treats this as a single-line comment, just as it does the // comment.

The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

The following example shows how to use comments in JavaScript.

```
<script language="javascript" type="text/javascript">
    <!--
        // This is a comment. It is similar to comments in C++
        /*
        * This is a multiline comment in JavaScript
        * It is very similar to comments in C Programming
        */
    //-->
</script>
```

Javascript – Placement A

There is a flexibility given to include JavaScript code anywhere in an HTML document.

However the most preferred ways to include JavaScript in an HTML file are as follows:

1. Script in <head>...</head> section.
2. Script in <body>...</body> section.
3. Script in <body>...</body> and <head>...</head> sections.
4. Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can place JavaScript in an HTML file in different ways.

JavaScript in <head>...</head> Section

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows.

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

JavaScript in <body>...</body> Section

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

```
<html>
    <head>
    </head>
    <body>
        <script type="text/javascript">
            <!--
            document.write("Hello World")
            //-->
        </script>
        <p>This is web page body </p>
    </body>
</html>
```

JavaScript in <body> and <head> Sections

You can put your JavaScript code in <head> and <body> section altogether as follows.

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

JavaScript in External File

As you begin to work more extensively with JavaScript, you will be likely to find that there are cases where you are reusing identical JavaScript code on multiple pages of a site.

You are not restricted to be maintaining identical code in multiple HTML files.

The script tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.

Here is an example to show how you can include an external JavaScript file in your HTML code using script tag and its src attribute.

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

To use JavaScript from an external file source, you need to write all your JavaScript source code in a simple text file with the extension ".js" and then include that file as shown above.

For example, you can keep the following content in filename.js file and then you can use sayHello function in your HTML file after including the filename.js file.

```
function sayHello() {
    alert("Hello World")
}
```

JAVASCRIPT – VARIABLES

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows.

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

You can also declare multiple variables with the same var keyword as follows:

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

JAVASCRIPT – VARIABLES

Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

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

Note: Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

Global Variables: A global variable has global scope which means it can be defined anywhere 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.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

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

JavaScript Reserved Words

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

A list of all the reserved words in JavaScript are given in the following table.

They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

JAVASCRIPT – OPERATORS

What is an Operator?

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

JavaScript Arithmetic Operators

Arithmetic operators perform arithmetic on numbers (literals or variables).

Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

Javascript Assignment operators

The Assignment operator is equal (=) which assigns the value of right-hand operand to its left-hand operand. That is if ***a = b*** assigns the value of b to a.

The simple assignment operator is used to assigning a value to a variable. The assignment operation evaluates to the assigned value. Chaining the assignment operator is possible in order to assign a single value to multiple variables. See the example.

Syntax:

data=value

NAME	SHORTHAND OPERATOR	MEANING
Addition Assignment	a+=b	a=a+b
Subtraction Assignment	a-=b	a=a-b
Multiplication Assignment	a*=b	a=a*b
Division Assignment	a/=b	a=a/b
Remainder Assignment	a%=b	a=a%b
Exponentiation Assignment	a**=b	a=a**b
Left Shift Assignment	a<<=b	a=a<<b
Right Shift Assignment	a>>=b	a=a>>b
Bitwise AND Assignment	a&=b	a=a&b
Bitwise OR Assignment	a|=b	a=a | b
Bitwise XOR Assignment	a^=b	a=a^b

JavaScript Comparison Operators

The Comparison operators are mainly used to perform the logical operations that determine the equality or difference between the values.

Operators are used to performing specific mathematical and logical computations on operands. Comparison operators are used in logical expressions to determine their equality or differences in variables or values.

Operator	**Description**
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

There are various Logical Operators in JavaScript –

- **&& (Logical AND):**

  It checks whether two operands are non-zero (0, false, undefined, null or “” are considered as zero), if yes then return 1 otherwise 0.

  Example :

```
Y = 5 and X = 6
 Y && X is true.
```

- **|| (Logical OR) :**

  It checks whether any one of the two operands is non-zero (0, false, undefined, null, or “” is considered as zero). Thus || returns true if either operand is true and if both are false it returns false.

  Example :

```
Y = 5 and X = 0
 Y || X is true.
```

- **! (Logical NOT):**

  It reverses the boolean result of the operand (or condition).

  Example :

```
Y = 5 and X = 0
 !(Y || X) is false.
```

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript – Loops In Javascript

Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true. For example, suppose we want to print "Hello World" 10 times. This can be done in two ways as shown below

There are mainly two types of loops:

- **Entry Controlled loops:** In these types of loops, the test condition is tested before entering the loop body. ***For Loops and While Loops*** are entry-controlled loops.
- **Exit Controlled loops:** In these types of loops the test condition is tested or evaluated at the end of the loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. The ***do-while*** loop is exit controlled loop.

```
<script type = "text/javascript">
   var i;

   for (i = 0; i < 10; i++)
   {
         document.write("Hello
      World!\n");
   }
</script>
```

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

    while (i < 10)
    {
          document.write("Hello World!\n");
          i++;
    }

</script>
```

JavaScript –Dialogue Boxes

JavaScript uses 3 kinds of dialog boxes: ALERT, PROMPT and CONFIRM. These dialog boxes can be of very much help for making our website look more attractive.

Alert Box:

An alert box is used on the website to show a warning message to the user that they have entered the wrong value other than what is required to fill in that position. Nonetheless, an alert box can still be used for friendlier messages. The alert box gives only one button "OK" to select and proceed.

```
<!DOCTYPE html>
<html>
<head>

    <script type="text/javascript">
            function Warning() {
            alert ("Warning danger you have not filled everything");
            document.write ("Warning danger you have not filled everything");
            }
    </script>

</head>
<body>

<p> Click me </p>


    <form>
        <input type="button" value="Click Me" onclick="Warning();" />
    </form>

</body>

</html>
```

JavaScript –Dialogue Boxes

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 on the OK button, the window method confirm() will return true. If the user clicks on the Cancel button, then confirm() returns false and will show null.

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

        function Confirmation(){
            var Val = confirm("Do you want to continue ?");
            if( Val == true ){
                document.write (" CONTINUED!");
                return true;
            }
            else{
                document.write ("NOT CONTINUED!");
                return false;
            }
        }
    </script>

</head>
<body>
<p>Click me: </p>
    <form>
        <input type="button" value="Click Me" onclick="Confirmation();" />
    </form>
</body>
</html>
```

JavaScript –Dialogue Boxes

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 the OK button, the window method prompt() will return the entered value from the text box. If the user clicks the Cancel button, the window method prompt() returns null.

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
            function Value(){
            var Val = prompt("Enter your name : ", "name");
            document.write("You entered : " + Val);
            }
    </script>
</head>

<body>


<p>Click me: </p>


    <form>
        <input type="button" value="Click Me" onclick="Value();" />
    </form>

</body>
</html>
```

THANK YOU!