Quick JavaScript Learning

In Just 3 Days

Fast-Track Learning Course

Vijay K. R.

Quick
JavaScript Learning
In
Just 3 Days

(Fast-Track Learning Course)

By

Vijay K. R.

Disclaimer

The publisher and the author of this publication have utilized their best efforts to maintain the accuracy of all the contents. The content in this eBook is intended to be used only for educational purposes. No part of this eBook can be copied, reconstructed or rewritten by any means, electronic, Photostat or by any mode, without the written consent of the author and the publisher.

The readers are completely responsible for all the actions taken by reading the content of this book. The author and the publisher hold no responsibility for any damage, loss or disruption because of omissions or errors, to any party.

The author and the publisher disclaim any liability regarding the effectiveness, performance or applicability of anything in this eBook.

Table of Content

Introduction

JavaScript is one the language which helps you make browser to do what you want to do. This is easy language to learn and I am going to teach you now. My language will be user friendly and I will try to tell difficult stuffs in easy language with examples. Let's move on one by one with JavaScript lessons. This is 3 days basic JavaScript Learning Course.

Before starting JavaScript, I am assuming that you have at least basic knowledge of HTML & CSS. If you don't have knowledge of HTML & CSS, then you should go and study that first via online tutorials. You may be able to learn basic HTML & CSS in 2-3 days.

Day 1 = Learn Chapter 1 & 2

Day 2 = Learn Chapter 3, 4 & 5

Day 3 = Learn Chapter 6, 7 & 8

Chapter 1
JavaScript Basics

JavaScript can change HTML Contents. If you are PHP developer and if you want to get done some work without page loads or you want HTML & CSS to perform work then you will need JavaScript.

<script> Tag

JavaScript can be placed inside body and head tag. The JavaScript codes should be inside <script> and </script> tags.

```
<script>
document.getElementById(“demo”).innerHTML = “My First JavaScript Code”;
</script>
```

JavaScript Events

JavaScript codes are triggered on events. Events are performed when a person perform an event. There are many events provided by browser like onclick, onmouseover, onkeyup, onkeydown, ondoubleclick and many more. When any of events performs then JS Codes triggered and do something according to codes.

```
<button onclick=“myFunction()”>Press Me!</button>
```

Explanation – here, onclick is event and myFunction() is JS function.

JavaScript Function

JavaScript block of codes which is inside <script> and </script> tags which are used to

perform some work is called function.

```
<script>
function myFunction(){
document.getElementById(“demo”).innerHTML = “My First JavaScript Code”;
}
</script>
```

Explanation – Here **myFunction**() is the JS function inside <script> tags.

Best Place for JS Codes

It is best practice to use all JS codes inside <head> tags.

```
<html>
<head>
<script>
function myFunction() {
    document.getElementById(“demo”).innerHTML = “Paragraph has changed.”;
}
</script>
</head>

<body>

<h1>New Page</h1>

<p id=“demo”>This is a Paragraph</p>

<button type=“button” onclick=“myFunction()”>Press Me!</button>
```

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

Explanation – We used all script inside head tags.

External JS Files

You can shore all codes in one another file and save file with .js extension. You must include the file inside <head> tags by this syntax.

```
<head>
<script src="newScript.js"></script>
</head>
```

Explanation – Here, we used newScript.js file, where we stored all JS codes, when any function is invoked the codes will be triggered directly from the newScript.js file.

Chapter 2
JavaScript Display & Output Codes

JavaScript uses four ways to display output.

1st Way: document.write()

It is used to write or output anything in the webpage.

```
<html>
<head>
<script>
function add(){
document.write(6 + 6);
}
</script>
</head>
<body>

<h1>My First Code</h1>
<button onclick="add()">Press Me!</button>

</body>
</html>
```

Explanation – When you click on Press me button, document.write shows output in the webpage.

2nd Way: Window.alert()

It is used to display message in alert box.

```
<html>
<body>

<h1>My Alert Web Page</h1>

<script>
window.alert(“Warning”);
</script>

</body>
</html>
```

Explanation – When you open the webpage, you will see an alert box containing Warning.

3rd Way: innerHTML

This method is used for writing something inside HTML element like inside paragraph, textbox, textarea and etc.

```
<html>
<head>
<script>
function something(){
document.getElementById(“demo”).innerHTML = “This paragraph is filled with text Now”;
}

</script>
</head>
<body>
```

```
<h1>Write Inside HTML Element Paragraph </h1>
<p id="demo">P is blank now</p>
<button onclick="something()">Press Me!</button>

</body>
</html>
```

Explanation – when you click Press Me, **document.getElementById** takes id **demo** and writes inside **demo** id through **innerHTML** keyword.

4th Way: console.log()

It is used to write inside console, click on F12 function key in your firefox, chrome, IE browser. You will see one console tag. Click on tab, you will get the message which you write in console.log(). See example.

```
<html>
<head>
<script>
console.log(2+2);

</script>
</head>
<body>

<h1>Click F12 function key and go to console tag to see results </h1>


</body>
</html>
```

Chapter 3
JavaScript Variables, Statements, Comments, Keywords

JavaScript Variables

Variables are used to stored data in any language. In JavaScript, variables do the same like C and other high languages. Variables are stored in the form of data types like int, str and etc. however at JS, you can use **var** keyword for all types of data types.

Writing String: For writing string, you need to use single or double quote.

Example:

```
var x = “This is a Boy.”;
var y = ‘This is a Girl.’;
```

Note: semi colon “;” is used to end the line.

Writing Numeric: For writing numeric, you need not to use double or single quotes. If you use it then JS will take it as string.

Example:

```
var x = 8;
var y = 6;
var z = x+y;
```

JavaScript Statements

JavaScript statements are set of instructions which are executed by browser.

Example1: Change HTML Content

Think of the scenario; if you want to add some content on anywhere in HTML by clicking one button.

You can do this work easily through PHP but the page will be reloaded. With the help of JavaScript you can do this work without page reloads. Here, JavaScript come at existence when we need to perform any work by browser.

```
<html>
<body>

<h1>Change Content of Box<h1>

<p id="demo">JavaScript will change HTML content.</p>

<button type="button"
onclick="document.getElementById('demo').innerHTML = 'I learnt first example!'">
Press Me!</button>

</body>
</html>
```

Explanation – Here, I made on paragraph with html **p tag** and gave **id name** "demo". Secondly I made one button with **onclick** event. I wrote one JavaScript code "document.getElementById('demo').innerHTML = 'I learnt first example!'. You will learn all these in coming chapters but let's start now. **document** is JavaScript object and **getElementById** is document method. It clearly shows that get element by id (demo). Here, it will get element by id demo which is the id of **p tag** and then **innerHTML** tells that inside the p HTML tag make changes. And you have to perform the change which is written after equal to sign. Now when you press the button <p id="demo"> inner html tag is changed to "I learnt first example!". I hope you will get this, if not read it again and

even not. Not a problem, we will see many examples ahead to know all these.

Example2: Change CSS Style

If you want to change the color of some text in HTML page by clicking one button then you can do this work easily through JavaScript. I am going to show you color changing in example.

```
<html>
<body>

<h1>Change Color of paragraph< h1>

<p id="demo">JavaScript will change the color.</p>

<script>
function colorChange() {
    var x = document.getElementById("demo");
    x.style.color = "red";
}
</script>

<button type="button" onclick="colorChange()">Press Me!</button>

</body>
</html>
```

Explanation – If you read the first example, you can understand the paragraph p tag id is demo and button onclick event. In previous example, we have written the code directly on event name onclick. As the code was small so we wrote inline, but now code is big so we used <script> </script> tag to write JavaScript code. Inside the code, we made one function named colorChange() and open the function with curly bracket '{ 'and close the function with '}' bracket. I will teach you regarding function in later chapters. For

understanding it, just assume that function is the place where we write code which we can use at many places. Function always has one name, and function is called by its name. Now come to example, here at onclick event we called the function by its name **colorChange()**. And when this event triggered than it will work according the code which is written inside the function open curly bracket { and close curly bracket } .

Here, we have taken one variable **x**. Variable is defined by keyword **var** in JavaScript. We assigned demo tag to variable x and in next line **x.style.color** says that x style color will be changed to red when it is called. And when you press “Press Me!” button the color of text will be changed. Just use it and you will start learning it automatically. If not then close your eyes for a couple of minutes and think of this code. You will start gaining knowledge.

Example3: Change HTML Attributes

Think, if you want to change image to different image with one click. Let’s look on this example for now. If you just keep little attention on previous two examples, you will gain many things which will help you go ahead. If you are studying in hurry, wait and watch the previous two examples. If you understand the basics, you won’t feel any difficulty in coming codes. Now come on the example.

```
<html>
<body>

<h1>JavaScript will Change Images</h1>

<img id=“myImage”
src=“https://upload.wikimedia.org/wikipedia/en/c/c2/Single_red_rose.jpg” width=“100”
height=“180”>

<script>
function changeImage() {
    var image = document.getElementById(‘myImage’);
```

```
    if (image.src.match(“red_rose”)) {
    image.src =
“https://upload.wikimedia.org/wikipedia/commons/c/c3/Extracted_pink_rose.png”;
    } else {
    image.src = “https://upload.wikimedia.org/wikipedia/en/c/c2/Single_red_rose.jpg”;
    }
}
</script>

<button onclick=“changeImage()”>Press Me!</button>
</body>
</html>
```

Explanation – Here, I took two images from Wikimedia first image is red rose and second is pink rose. You can use your images.

My red rose image –

https://upload.wikimedia.org/wikipedia/en/c/c2/Single_red_rose.jpg

My pink rose image - https://upload.wikimedia.org/wikipedia/en/c/c2/Single_red_rose.jpg

When we click on button, the image gets changed to different image. Now move to code inside function **changeImage**(). Here we have used if and else conditions, I will teach it in later chapters. For here, think that if **image.src.match** means (image source matches to red_rose word which is written in the url) then it will show red rose, and if it will not match then it will show else part which is **image.src** (means image source of pink rose).

JavaScript Comments

JavaScript comments are used to comment on code sections for future references. After writing many codes, we forget the idea of writing codes and with comments, we can easily remember what the code is and why we wrote any specific code.

There are two types of commenting in JS.

1. **Single Line Comments:** // is used for single line comments.

Example:

```
var x= 5;                    //this is the value of x
var y=6;                     //this is the value of y
```

2. Multi Line Comments: Multiline comments are used to comment more than one line. It starts with /* and ends with */.

Example:

```
var x= 5;
var y=6;
/*
Here, value of x is 5 and
Value of y is 6
*/
```

JavaScript Keywords

Keywords are words which are reserved by JS to perform certain tasks. You cannot use keyword as variable. Keyword list

Keywords	**Description**
Var	For declaring a variable
Break	Terminates a switch loop or another loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript
do … while	Execute the block of statements

for	Block of statements to be executed until condition is true
function	Declares a function
if … else	Block of statements to be executed, depending on a condition
return	Exits a function
switch	Block of statements to be executed which depends on different cases
try … catch	Implements error handling for statement blocks

Chapter 4
JavaScript Data Types & Operators

JavaScript Data Types

JavaScript Data Types can be string, number, Boolean, object, array and many more. It is very important to have data types, without data types computers involve in risk.

1. **JS String Data Types** - String data types always inside single or double quotes.

   Example:

   var name = “John”;

   var name = ‘John’;

2. **JS Number Data Types -** It is numeric data types.

   Example:

   var x = 5;

   var y = 6;

3. **JS Boolean Data Types** – It is conditional data type, it is either true or false.

   Example:

```
var x = true;
var y = false;
```

4. **JS Arrays Data Types** – It is written in bracket and separated by comma.

   Example:

```
var language = [“php”, “javascript”, “java”];
```

5. **JS Object Data Types** – It is written in curly braces & separated by comma, and its syntax is - (Property name: value)

   Example:

```
var man = {name:”Bonnie”, age:50, surname:”smith” };
```

Type of Operator

We can know data type of any operator with type of keyword.

Example:

```
typeof “bonnie”                    //it will return string
typeof 50                          //it will return number
typeof true                        //it will return Boolean
typeof [“man”, “women”]            //it will return array
typeof {name:”Jonnie”, age:50}     // it will return object
```

Undefined, Empty and Null

1. **Undefined Data Types** – Data types which are not defined with any value called undefined data types.

   Example:

```
var x ;                          // x is undefined
var Bonnie;                      //Bonnie is undefined
```

2. **Empty**

   Example:

```
var x = " " ;                    // x is empty value
```

3. **Null** – Null is nothing but type is an object.

   Example:

```
var x = null;                    // x is null
```

JavaScript Operators

JavaScript has many operators like arithmetic operators, assignment operators, string operators, logical operators and comparison operators.

1. **Arithmetic Operators** – Operators which are used to perform certain mathematic operations on numbers called arithmetic operators.

   Example:

Arithmetic Operator	**Descriptions**

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
—	Decrement

2. **Assignment Operators** – For assigning value, assignment operators are used.

Example:

Operator
=
+=
-=
*=
/=
%=

3. **String Operators** – You can add strings with + operator.

Example:

```
var name = “Jonnie”;
var surname = “Smith”;
var full name = “Jonnie” + “Smith”;
```

4. Comparison & Logical Operators

Example:

Operators	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than

Chapter 5
JavaScript Functions

Function is a block of code which is used to perform any task and can be reused in many places. Function is invoked when it is called. Syntax is simple, here it is:

Syntax:

```
function myFun(parameter1, parameter2, parameter3, …)
{
//block of code
}
```

Here, myFun is function name and parameter1, parameter2 and parameter 3 are parameters, which are optional, then a curly bracket starts and ends. In between curly bracket, there will be code which will be executed when the function will be called.

Example:

```
<html>
<head>
    <script>
    function myFun(x,y){
    var x;
    var y;
    var z=x+y;
    document.getElementById(“demo”).innerHTML = z;
```

```
    }
    </script>
</head>
<body>
    <p id="demo"></p>
    <button onclick="myFun(5,10)">Press IT!</button>
</body>
</html>
```

Here, when we click the button "Press IT", an onclick event is generated and myFun(5,10) function has executed. When the function executed, we pass two parameters to function with values and those values becomes x and y value of function, and finally results stored in variable and z variable data goes inside the p tag.

Function return

Function is stop executing when it reaches to return statement. Return is JavaScript inbuilt statement to stopping the execution and sending the return value to function. You can think as the "overall value of function" can be denoted by return statement.

Example

```
<html>
<head>
</head>
<body>
    <p id="demo"></p>
    <script>
    function myFun(x,y){
    var x;
    var y;
    var z=x+y;
    return z;
```

```
    }
    document.getElementById(“demo”).innerHTML = myFun(150,10);
    </script>
</body>
</html>
```

Here, myFun(150,10) value will be assigned to demo id. Here, you have seen that I have written script inside the body tag, because head tags are executed when it is called but body tags are automatically executed when body is loaded on browser. Here, I have not given any event like onclick, therefore I have written entire script inside the body tag.

Chapter 6
JavaScript Objects

Think objects as the real object, for example car is an object. Which can perform any task; it has its properties and methods. Properties like its color, model, weight and method likes start, stop and etc.

Object Properties Example

```
var car = {color:”red”, model=“5520”, weight=“900kgs”};
```

Properties are invoked by two types of syntaxes.

1st syntax = Object.property ;
2nd Syntax = Object[“property”];

Example:

```
<html>
<head>
</head>
<body>
    <p id=“demo”></p>
    <script>
    var car = {color:“red”, model:552, weight:“900kgs”};
    document.getElementById(“demo”).innerHTML = car.color;
    </script>
</body>
</html>
```

Object Method

Object method is a function inside the object which has to perform certain task. Its syntax to invoked is = object.method();

Example:

```
<html>
<head>
</head>
<body>
    <p id=“demo”></p>
    <script>
    var car = {
    color:“red”,
    model:552,
    weight:“900kgs”,
    fullDetails: function () {
    return “Car color is “+ this.color + “. Car model is “+ this.model + ” and Car weight is “+this.weight; }
    };
    document.getElementById(“demo”).innerHTML = car.fullDetails();
    </script>
</body>
</html>
```

Here, a method fullDetails is made as a function and it is invoked by car.fullDetails(). Here, you have seen new keyword “this”. “this” is used to use own properties of any object. You cannot directly write color, model and weight. You need to use this.color, this.model and this.weight to use inside the function.

Object Constructor

It is best way to use constructor for using objects with parameters.

```
function person(name, age, weight, height){
    this.name = name;
    this.age = age;
    this.weight = weight;
    this.height = height;
        }
```

Here person is the constructor object

Create an object

For creating an object, new keyword is used in JavaScript as it is used in PHP, C and in many languages.

```
var myFriend = new person(Jonnie, Smith, 50kg, 6ft);
```

Here, myFriend is new object of person Object. Now, myFriend can access whole properties and methods of object person.

```
myFriend.name                    //accessing property
myFriend.age                    //accessing property
```

JavaScript Prototypes

All JavaScript objects are inherited with their object prototypes which are denoted like object.prototype. Example: new Date() is used to create new Date object and inherit Date.prototype.

Creating a Prototype

```
function person(firstname, surname, age) {
    this.firstname = firstname;
    this.surname = surname;
    this.age = age;
    }
```

Creating new objects with new keyword

```
var student = new person(“John”, “Smith”, 12, “blue”);
```

Adding Properties, Methods to Objects

Sometime you want to add some properties, methods to objects, all existing objects or to a prototype object.

Adding property to an object

```
student.haircolor = “Gray”;
```

Adding property to a prototype

You can not directly add property to a prototype as similar to object. You need to do by this way.

```
person.haircolor = “Gray”;
```

Adding method to an object

```
student.fullname = function(){
        return this.firstname + " " + this.surname;
        };
```

Adding method to a prototype

```
function person(firstname, surname, age) {
    this.firstname = firstname;
    this.surname = surname;
    this.age = age;
    this.fullname = function(){
    return this.firstname + " " + this.surname; };
    }
```

Chapter 7
JavaScript Arrays

JavaScript Arrays are used to store multiple values in one variable.

Example:

var cars = Array(“BMW”, “Volvo”, “Jeep”);
var cars = new Array(“BMW”, “Volvo”, “Jeep”);

Array are made with or without new keyword, here three values are stored in cars variable.

cars[0]=“BMW”
cars[1]=“Volvo”
cars[2]=“Jeep”

It means all values are actually index of one variable “cars”. It means array stores multiple values in the form of indexes.

If you would like to access any array then you can simply access by writing its index.
Example:

```
<html>
<head>
</head>
<body>
    <p id=“demo”></p>
<script>
    var cars = Array(“BMW”, “Volvo”, “Jeep”);
    document.getElementById(“demo”).innerHTML = cars[1];
```

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

Accessing an array

You can access any array with their index values directly and indirectly.

```
var carname = cars[0];
or
cars[0];
```

Arrays as Objects

Objects uses name indexes whereas array uses numbered indexes. Means, array uses numbers to access elements whereas object uses names to access elements.

Example

```
var person = [“John”, “Smith”, 12];                                    //array
var person = {firstname:“John”,  lastname:“Smith”, age:12};          //object
```

Chapter 8

JavaScript Conditions (if, else, else if & Switch) and Loops (for & while)

JavaScript Conditions (if, else, else if & switch)

In JavaScript, many times you have to choose one condition or many conditions for performing any task.

Syntax

```
If(condition){
   Block of code will execute, when if condition becomes true.
} else {
   Block of code will execute, when if condition becomes false.
}
```

Using if statement

When you have to perform one task on any condition, and “if” condition becomes true then “if statement block” will execute.

Using else statement

When “if” condition becomes false, then “else” block will be executed.

Example: if age is above 18 then the person can vote and if the person below 18 age then cannot vote.

```
<html>
```

```
<head>
<title>Vote Eligibility</title>

</head>
<body>
<div>Vote Eligibility</div>
<script>
function myfunction(){
var age = document.getElementById(“age”).value;
if(age>=18){
var eligibility = “this person can vote”;
document.getElementById(“demo”).innerHTML = eligibility;
}else{
var eligibility = “this person cannot vote”;
document.getElementById(“demo”).innerHTML = eligibility;
}
}

</script>
Enter Your Age: <input type=“text” id=“age” name=“age” /><br>
<p id=“demo”></p>
<input type=“button” onclick=“myfunction()” value=“Check Eligibility”>
</body>
</html>
```

Using else if statement

When we have more than one condition then we use “else if” statement.

Syntax:

```
If(condition 1){
    Block of code will execute, when if condition 1 becomes true.
} else if(condition 2){
    Block of code will execute, when if condition2 becomes true.
} else {
    Block of code will execute when both conditions becomes false.
}
```

Example: if a person meets before 12 hours, we wish him good morning, if he meets after 16 hours, we wish him good afternoon and if both the conditions become false then we wish good evening.

```
<html>
    <head>
    <title>Example</title>

    </head>
    <body>
    <div>Greet Message (Enter Time in 24 hours format) </div>
    <script>
    function myfunction(){
    var time = document.getElementById("time").value;
    if(time<12){
    var greet = "Good Morning";
    document.getElementById("demo").innerHTML = greet;
    }else if(time<16){
    var greet = "Good Afternoon";
    document.getElementById("demo").innerHTML = greet;
    }else{
    var greet = "Good Evening";
```

```
    document.getElementById(“demo”).innerHTML = greet;
    }

    }

    </script>
    Enter Time: <input type=“text” id=“time” name=“time” /><br>
    <p id=“demo”> Greet Message Here :</p>
    <input type=“button” onclick=“myfunction()” value=“Check Greet Message”>
    </body>
</html>
```

By this we can increase as many as else if conditions in the code. For many conditions we can also use switch statement.

Switch Statement

Switch statement is used when you have many conditions to check. You can do the same work with else if but switch is better than else if statement. You need to put “break” command after each condition and one condition will “default” if no condition will be executed.

Syntax:

```
switch(expression) {
    case x:
    this code block will execute.
    break;
    case y:
    this code block will execute.
    break;
    default:
```

```
    this code block will execute.
}
```

Example:

```
<html>
    <head>
    <title>Example</title>

    </head>
    <body>
    <script>
    function myfunction(){
    var today = new Date().getDay();
    switch (today) {
    case 0:
    day = “Today is Sunday”;
    break;
    case 1:
    day = “Today is Monday”;
    break;
    case 2:
    day = “Today is Tuesday”;
    break;
    case 3:
    day = “Today is Wednesday”;
    break;
    case 4:
    day = “Today is Thursday”;
    break;
    case 5:
```

```
day = “Today is Friday”;
break;
case 6:
day = “Today is Saturday”;
break;
}
document.getElementById(“demo”).innerHTML = day;
}

</script>
What Day is Today:
<p id=“demo”></p>
<input type=“button” onclick=“myfunction()” value=“Check To See Today’s Day”>
</body>
</html>
```

JavaScript Loops

JavaScript Loops are used to execute one code many times by changing values. For example, you want to write 1 to 1000 in one page in each line. By doing manually, it will take huge time but with loops, it will be done in few seconds.

Example:

```
<html>
<head>
<title>Example</title>

</head>
<body>
<script>
function myfunction(){
var star=””;
for(var i=1; i<1000; i++){
star += i + “<Br>”;
```

```
    }
    document.getElementById(“demo”).innerHTML = star ;
    }

    </script>

    <p id=“demo”></p>
    <input type=“button” onclick=“myfunction()” value=“Check Loop”>
    </body>
</html>
```

For Loop

In the previous example, you saw that loop needs three statements.

First statement – to initialize the value

Second statement – to check the condition

Third statement – to increase or decrease the counter

Syntax:

```
for (statement 1; statement 2; statement 3) {
                This code block to be executed
    }
```

In the previous example, loop begins with statement 1 which is var i=1, here we are initializing the value of “i” and then statement 2 checks the condition, if condition is fulfilled then block of code executes. Here statement 3 works after block of code execution, then statement 3 works to increase or decrease the value of “i”.

While Loop

When we need to execute the block of code when any condition fulfills then we

use while loop.

Syntax:

```
while (condition) {
    This code block to be executed
}
```

Example:

```
<html>
    <head>
    <title>Example</title>

    </head>
    <body>
    <script>
    function myfunction(){
    var num="";
    var i=1;
    while (i < 20) {
    num += "Number is " + i +"<Br>";
    i++;
    }
    document.getElementById("demo").innerHTML = num ;
    }
```

```
    </script>

    <p id="demo"></p>
    <input type="button" onclick="myfunction()" value="Check Loop">
    </body>
</html>
```

Here, while checks the statement, and execute the code and then increase the value of "i" and again checks the statement and execute the code and so on…

Do while Loop

do while loop is used when you need to execute the block of code atleast one time, first do statement works and then while statement checks the condition.

Syntax:

```
do{
    This block of code will execute.
}while(condition)
```

Example:

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

```
<title>Example</title>

</head>
<body>
<script>
function myfunction(){
var num="";
var i=1;
do {
num += "Number is " + i +"<Br>";
i++;
}
while (i < 10);
document.getElementById("demo").innerHTML = num ;
}

</script>

<p id="demo"></p>
<input type="button" onclick="myfunction()" value="Check Loop">
</body>
</html>
```

Here, firstly do statements works and then checks the condition, if condition becomes true then again do statement code executes.

Summary

You have successfully learnt JavaScript, now make your own JS codes and work in real working scenario. I hope you enjoyed this three days course.