JavaScript If-else

The JavaScript if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

- If Statement
- If else statement
- if else if statement

JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
}
```

Code:

```
<html>
<head>
<title>  If else in javascript</title>
</head>
  <body>
 <script type = "text/javascript">
        var a=30;
        if(a>10)
        {
        document.write("value of a is greater than 10");
        }
</script>
 </body>
</html>
```

Output:

value of a is greater than 10

JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Code:

```
<html>
<head>
<title>  statments in javascript</title>
</head>
  <body>
 <script type = "text/javascript">
        var a=30;
        if(a>40)
        {
        document.write("value of a is greater than 40");
        }
     else
```

```
    {
     document.write("value of a is smaller than 40");
    }
</script>
 </body>
</html>
```

Output:

value of a is smaller than 40

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Code:

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

```
<title>  statments in javascript</title>
</head>
  <body>
 <script type = "text/javascript">
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
 </body>
</html>
```

Output:

a is equal to 20

JavaScript Switch

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if..else..if because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

```
switch(expression){
case value1:  code to be executed;
                  break;
case value2:  code to be executed;
                 break;
......

default:    code to be executed if above values are not matched;
}
```

Code:

```
<html>
<head>
<title>  Switch in javascript</title>
</head>
  <body>
<script type = "text/javascript">
        var grade='B';
        var result;
        switch(grade)
{
```

```
case 'A':  result="A Grade";
                    break;
case 'B':  result="B Grade";
                break;
 case 'C':  result="C Grade";
                break;
default:  result="No Grade";
}
document.write(result);
</script>
  </body>
</html>
```

Output:

B Grade

JavaScript Loops

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

- for loop
- while loop
- do-while loop
- for-in loop (related to object)

JavaScript For loop

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
   code to be executed
}
```

Code:

```
<html>
<head>
<title>  for loop in javascript</title>
</head>
  <body>
<script type = "text/javascript">
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
  </body>
</html>
```

Output:

1
2
3

4
5

JavaScript while loop

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

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

Code:

```
<html>
<head>
<title>  while loop in javascript</title>
</head>
  <body>
<script type = "text/javascript">
var i=10;
while (i<=14)
{
document.write(i + "<br/>");
i++;
}
</script>
  </body>
</html>
```

Output:

10
11
12
13
14

JavaScript do while loop

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

```
do{
   code to be executed
}while (condition);
```

Code:

```
<html>
<head>
<title>  do while loop in javascript</title>
</head>
  <body>
<script type = "text/javascript">
var i=1;
do{
document.write(i + "<br/>");
i++;
}while (i<=5);
```

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

Output:

1

2

3

4

5

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

Code reusability: We can call a function several times so it save coding.

Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

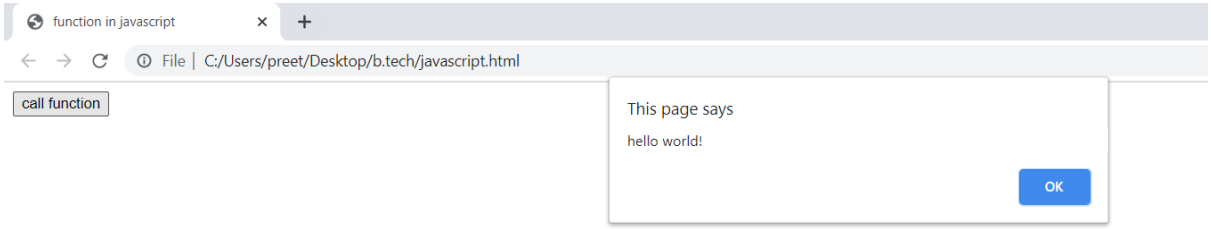
The syntax of declaring function is given below.

```
    function functionName([arg1, arg2, ...argN]) {
 //code to be executed
}
```

Code:

```
<html>
<head>
<title>  function in javascript</title>
</head>
  <body>
<script type = "text/javascript">
function myfunction(){
alert("hello world!");
}
</script>
</script>
<input type="button" onclick="  myfunction ()" value="call function"/>
  </body>
</html>
```

Output:



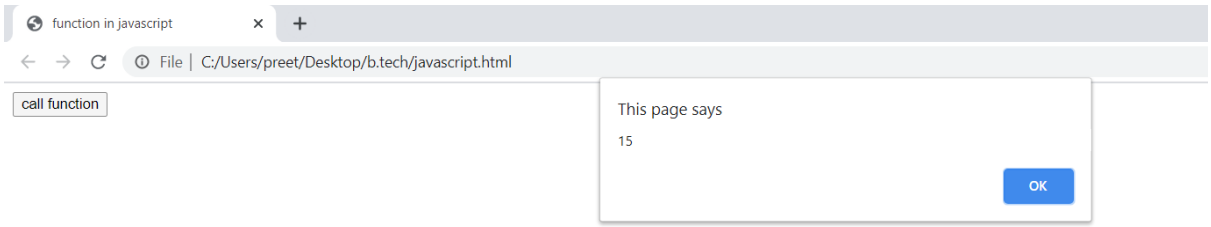
JavaScript Function Arguments

Call function by passing arguments.

Code:

```
<html>
<head>
<title>  function in javascript</title>
</head>
  <body>
<script type = "text/javascript">
function myfunction(num){
alert(num+num+num);
}
</script>
</script>
<input type="button" onclick="  myfunction (5)" value="call function"/>
  </body>
</html>
```

Output:



Function with Return Value

We can call function that returns a value and use it in our program

Code:

```
<html>
<head>
<title>  function in javascript</title>
</head>
  <body>
<script type = "text/javascript">
function myfunction(){
return 5+5+5;
}
</script>
<script>
document.write(myfunction());
</script>
</body>
</html>
```

Output:

15

JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

- By object literal
- By creating instance of Object directly (using new keyword)
- By using an object constructor (using new keyword)

JavaScript Object by object literal

The syntax of creating object using object literal is given below:

object={property1:value1,property2:value2.....propertyN:valueN}

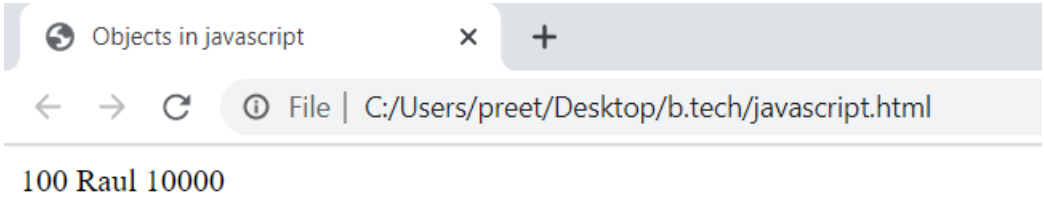
Code:

```
<html>
<head>
<title>  Objects in javascript</title>
</head>
  <body>
<script type = "text/javascript">

emp={id:102,name:"Shyam Kumar",salary:40000}
```

```
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

Output:



By creating instance of Object

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

Here, new keyword is used to create object.

Code:

```
<html>
<head>
<title>  Objects in javascript</title>
</head>
  <body>
<script type = "text/javascript">
 var emp=new Object();
emp.id=100;
```

```
emp.name="Rahul";
emp.salary=10000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

Output: