**The return Statement**

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product (a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

</body>
</html>
```

**Loops execute a block of code a specified number of times, or while a specified condition is true.**

**JavaScript Loops**

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kinds of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

## The for Loop

The for loop is used when you know in advance how many times the script should run.

### Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

### Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

## The while Loop

The while loop loops through a block of code while a specified condition is true.

### Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

**Note:** The <= could be any comparing statement.

**Example**

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
</html>
```

**The do...while Loop**

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

**Syntax**

```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

**Example**

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
```

```
 document.write("The number is " + i);
 document.write("<br />");
 i++;
 }
while (i<=5);
</script>
</body>
</html>
```

**The break Statement**

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  break;
  }
 document.write("The number is " + i);
 document.write("<br />");
 }
</script>
</body>
</html>
```

**The continue Statement**

The continue statement will break the current loop and continue with the next value.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
 {
 if (i==3)
```

```
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

**JavaScript For...In Statement**

The for...in statement loops through the elements of an array or through the properties of an object.

**Syntax**

```
for (variable in object)
  {
  code to be executed
  }
```

**Note:** The code in the body of the for...in loop is executed once for each element/property.

**Note:** The variable argument can be a named variable, an array element, or a property of an object.

**Example**

Use the for...in statement to loop through an array:

```
<html>
<body>

<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars = "Volvo";
mycars = "BMW";

for (x in mycars)
  {
  document.write(mycars[x] + "<br />");
  }
```

```
</script>

</body>
</html>
```

**JavaScript is an Object Oriented Programming (OOP) language.**

## Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Note that an object is just a special kind of data. An object has properties and methods.

### Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

### Methods

Methods are the actions that can be performed on objects.

In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```