

# **Web Engineering LAB**



**Lab # 05**  
**JavaScript: Basics, Functions and Loops**

**Instructor: Hurmat Hidayat**

**Email: [hurmathidayat@nu.edu.pk](mailto:hurmathidayat@nu.edu.pk)**

**Course Code: SL3003**

**Semester Spring 2022**

**Department of Computer Science,  
National University of Computer and Emerging Sciences FAST  
Peshawar Campus**

# Content

<b>JAVASCRIPT FOR WEB PAGE .....</b>	<b>3</b>
CREATE AND LINK JAVASCRIPT FILE .....	3
<b>BASIC JAVASCRIPT .....</b>	<b>4</b>
STATEMENTS .....	4
COMMENTS.....	4
VARIABLE .....	4
DATA TYPES.....	5
RULES FOR NAMING VARIABLES.....	6
ARRAYS.....	7
EXPRESSION.....	8
OPERATORS.....	8
BASIC JAVASCRIPT INSTRUCTION EXAMPLE.....	9
<b>FUNCTIONS OR METHODS .....</b>	<b>11</b>
A BASIC FUNCTION .....	12
DECLARING AND CALLING A FUNCTION .....	13
PARAMETERS .....	13
FUNCTION RETURN .....	14
ANONYMOUS FUNCTIONS .....	15
IMMEDIATELY INVOKED FUNCTION EXPRESSION.....	15
<b>LOOPS.....</b>	<b>16</b>
FOR LOOP .....	16
KEY LOOP CONCEPTS.....	16
USING FOR LOOP .....	17
USING WHILE LOOP .....	17
USING DO WHILE LOOP .....	18
REFERENCES .....	18
<b>LAB TASKS: .....</b>	<b>19</b>

## JavaScript for Web Page

Web developers usually talk about three languages that are used to create web pages: HTML, CSS, and JavaScript. Each language forms a separate layer with a different purpose.

- **Content Layer** : This is where the content of the page lives. The HTML gives the page structure and adds semantics.
- **Presentation Layer**: The CSS enhances the HTML page with rules that state how the HTML content is presented (backgrounds, borders, box dimensions, colors, fonts, etc.).
- **Behaviour Layer**: This is where we can change how the page behaves, adding interactivity. We aim to keep as much of our JavaScript as possible in separate files.

### Create and Link JavaScript file

A JavaScript file is just a text file (like HTML and CSS files are) but it has a .js file extension. When you want to use JavaScript with a web page, you use the HTML <script> element to tell the browser it is coming across a script. Its src attribute tells people where the JavaScript file is stored. You may see JavaScript in the HTML between opening <script> and closing </script> tags (but it is better to put scripts in their own files).

Example:

add-content.js

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening!';
} else if (hourNow > 12) {
    greeting = 'Good afternoon!';
} else if (hourNow > 0) {
    greeting = 'Good morning!';
} else {
    greeting = 'Welcome!';
}

document.write('<h3>' + greeting + '</h3>');
```

Add-content.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Constructive && Co.</title>
5      <link rel="stylesheet" href="css/c01.css" />
6  </head>
7  <body>
8      <h1>Constructive && Co.</h1>
9      <script src="js/add-content.js"></script>
10     <p>For all orders and inquiries please call <em>555-3344</em></p>
11  </body>
12  </html>
```

JavaScript runs where it is found in the HTML. When the browser comes across a <script> element, it stops to load the script and then checks to see if it needs to do anything.

If you view the source code of the page in the browser, the JavaScript will not have changed the HTML, because the script works with the model of the web page that the browser has created.

## Basic JavaScript

### Statements

A script is a series of instructions that a computer can follow one-by-one. Each individual instruction or step is known as a statement.

- Statements should end with a semicolon.
- JavaScript is case sensitive so hourNow means something different to HourNow or HOURNOW. A statement is an individual instruction , each one should start on a new line and end with a semicolon. This makes your code easier to read and follow.
- Some statements are surrounded by curly braces; these are known as code blocks. The closing curly brace is not followed by a semicolon.

### Comments

You should write comments to explain what your code does. They help make your code easier to read and understand. This can help you and others who read your code.

```
1  /* This script displays a greeting to the user based upon the current time.  
2   It is an example from JavaScript & jQuery book */  
3  
4   var today= new Date();           // Create a new date object  
5   var hour Now = today.getHours(); // Find the current hour  
6   var greeting;  
7  
8   //Display the appropriate greeting based on the current time  
9   if (hourNow > 18) {  
10     greeting= 'Good evening!';  
11   else if (hourNow > 12) {  
12     greeting = ' Good afternoon!';  
13   else if (hourNow > 0) {  
14     greeting = 'Good morni ng!';  
15   else {  
16     greeting = 'Welcome! ' ;  
17   }  
18  
19   document .write( '<h3>' +greeting + '</ h3>');  
20
```

**Multiline Comments:** To write a comment that stretches over more than one line, you use a multi-line comment, starting with the /\* characters and ending with the \*/ characters. Anything between these characters is not processed by the JavaScript interpreter.

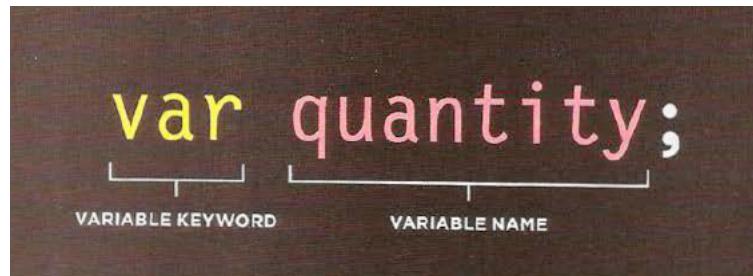
**Single Line Comments:** In a single-line comment, anything that follows the two forward slash characters // on that line will not be processed by the JavaScript interpreter. Single line comments are often used for short descriptions of what the code is doing.

### Variable

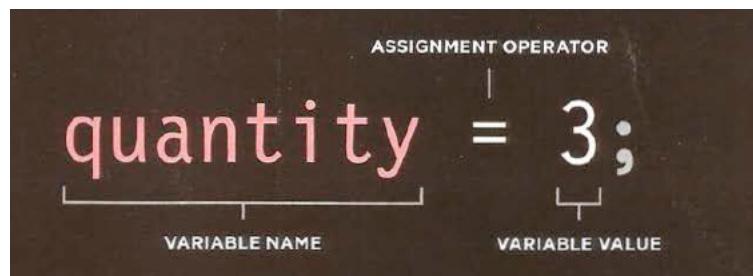
A script will have to temporarily store the bits of information it needs to do its job. It can store this data in variables.

## Variable Declaration

Before you can use a variable, you need to announce that you want to use it. This involves creating a variable and giving it a name. We can declare variable as follow:



## Assign Value to a Variable



## Data Types

JavaScript distinguishes between numbers, strings, and true or false values known as Booleans.

### NUMERIC DATA TYPE:

The numeric data type handles numbers. For example: 0.75

For tasks that involve counting or calculating sums, you will use numbers 0-9. For example, five thousand, two hundred and seventy-two would be written 5272 (note there is no comma between the thousands and the hundreds). You can also have negative numbers (such as -23678) and decimals (three quarters is written as 0.75).

### STRING DATA TYPE:

The strings data type consists of letters and other characters. For example: 'Hi, Alen!'

Note how the string data type is enclosed within a pair of quotes. These can be single or double quotes, but the opening quote must match the closing quote. Strings can be used when working with any kind of text. They are frequently used to add new content into a page and they can contain HTML markup.

```
1  var username;
2  var message;
3  username = 'Molly';
4  message = 'See our upcoming range';
5
```

Sometimes you will want to use a double or single quote mark within a string. Because strings can live in single or double quotes, if you just want to use double quotes in the string, you could surround the entire string in single quotes. If you just want to use single quotes in the string, you could surround the string in double quotes.

```

1  var title;
2  var message;
3
4  title= "Molly's Special Offers" ;
5  message = '<a href=\"sale .html\">25% off !</a>' ;
6
7  var elTitle = document.getElementById('title') ;
8  elTitle.innerHTML =title;
9
10 var elNote = document.getElementById ('note') ;
11 elNote.innerHTML =message;
12

```

### BOOLEAN DATA TYPE:

Boolean data types can have one of two values: true or false. For example: true  
Booleans are used when your code can take more than one path.

```

1  var inStock;
2  var shipping;
3  inStock = true;
4  shipping= false;
5

```

Note: Unlike some other programming languages, when declaring a variable in JavaScript, you do not need to specify what type of data it will hold.

### Shorthand For Creating Variables

Programmers sometimes use shorthand to create variables. Here are three variations of how to declare variables and assign them values:

The screenshot shows a browser's developer tools console with the tab labeled "JAVASCRIPT". The file path "c02/js/shorthand-variable.js" is shown at the top. Three numbered examples are displayed:

- ① `var price = 5;  
var quantity = 14;  
var total = price * quantity;`
- ② `var price, quantity, total;  
price = 5;  
quantity = 14;  
total = price * quantity;`
- ③ `var price = 5, quantity = 14;  
var total = price * quantity;`

### Rules For Naming Variables

Here are six rules you must always follow when giving a variable a name:

- The name must begin with a letter, dollar sign (\$), or an underscore (\_). It must not start with a number.

- The name can contain letters, numbers, dollar sign (\$), or an underscore (\_). Note that you must not use a dash(-) or a period(.) in a variable name.
- You cannot use keywords or reserved words.
- All variables are case sensitive, so score and Score would be different variable names, but it is bad practice to create two variables that have the same name using different cases.
- Use a name that describes the kind of information that the variable stores. For example, firstName might be used to store a person's first name, lastName for their last name, and age for their age.
- If your variable name is made up of more than one word, use a capital letter for the first letter of every word after the first word.

## Arrays

An array is a special type of variable. It doesn't just store one value; it stores a list of values. Arrays are especially helpful when you do not know how many items a list will contain because, when you create the array, you do not need to specify how many values it will hold.

For example, an array can be suited to storing the individual items on a shopping list because it is a list of related items.

```

1  var colors;
2  colors ['white', 'black', 'custom'];
3
4  var el = document.getElementById('colors');
5  el.textContent = colors[0];
6
|
```

The values are assigned to the array inside a pair of square brackets, and each value is separated by a comma. The values in the array do not need to be the same data type, so you can store a string, a number and a Boolean all in the same array. This technique for creating an array is known as an **array literal**.

### ACCESSING ITEMS IN AN ARRAY

To retrieve the third item on the list, the array name is specified along with the index number in square brackets.

```

1  var itemThree;
2  itemThree = colors [ 2 ];
3
|
```

Here you can see a variable called itemThree is declared. Its value is set to be the third color from the colors array.

### NUMBER OF ITEMS IN AN ARRAY

Each array has a property called length, which holds the number of items in the array. The name of the array is followed by a period symbol (or full stop) which is then followed by the length keyword.

```

1  var numColors ;
2  numColors = colors.length;
3
|
```

## Expression

An expression evaluates into (results in) a single value. Broadly speaking there are two types of expressions.

- Expression that just assign a value to a variable

```
var color = 'beige';
```

- Expression that use two or more values to return a single value

```
var area = 3 * 2;
```

## Operators

Expressions rely on things called operators; they allow programmers to create a single value from one or more values. Operators are of the following types:

- Assignment Operators
- Arithmetic Operators
- String Operators
- Comparison Operators
- Logical Operators

### Arithmetic Operators

JavaScript contains the following mathematical operators, which you can use with numbers.

NAME	OPERATOR	PURPOSE & NOTES	EXAMPLE	RESULT
ADDITION	+	Adds one value to another	10 + 5	15
SUBTRACTION	-	Subtracts one value from another	10 - 5	5
DIVISION	/	Divides two values	10 / 5	2
MULTIPLICATION	*	Multiplies two values using an asterisk (Note that this is not the letter x)	10 * 5	50
INCREMENT	++	Adds one to the current number	i = 10; i++;	11
DECREMENT	--	Subtracts one from the current number	i = 10; i--;	9
MODULUS	%	Divides two values and returns the remainder	10 % 3	1

## Using Arithmetic Operators

```
1 // Create a variable for the subtotal and make a calculation
2 var subtotal = (13 + 1) * 5; // Subtotal is 70
3
4 // Create a variable for the shipping and make a calculation
5 var shipping = 0.5 * (13 + 1); // Shipping is 7
6
7 // Create the total by combining the subtotal and shipping values
8 var total = subtotal + shipping; // Total is 77
9
10 // Write the results to the screen
11 var elSub = document.getElementById('subtotal');
12 elSub.textContent = subtotal;
13
14 var elShip = document.getElementById('shipping');
15 elShip.textContent = shipping;
16
17 var elTotal = document.getElementById('total');
18 elTotal.textContent = total;
19
20
```

## String Operator

There is just one string operator: the + symbol. It is used to join the strings on either side of it. There are many occasions where you may need to join two or more strings to create a single value. Programmers call the process of joining together two or more strings to create one new string concatenation.

## Example

```
var firstName = 'Ivy' ;
var lastName = 'Stone' ;
var full Name = firstName + lastName;
```

Output:

Ivy Stone

```
1 // Store the greeting in a variable
2 var greeting = 'Howdy ';
3
4 // Store the users name in a variable
5 var name = 'Molly';
6
7 /* Create the welcome message by concatenating the strings in the two variables */
8 var welcomeMessage = greeting + name + '!';
9
10 // Get the element that has an id of greeting
11 var el = document.getElementById('greeting');
12
13 // Replace the content of this element with the personal message
14 el.textContent = welcomeMessage;
15
```

## Basic JavaScript Instruction Example

HTML File:

```

1 <html>
2 <head>
3   <title> JavaScript Example</title>
4   <link rel="stylesheet" href="css/c02.css" />
5 </head>
6 <body>
7   <h1>Elderflower</h1>
8   <div id="content">
9     <div id="greeting" class="message">Hello!</div>
10    <table>
11      <tr>
12        <td>Custom sign:</td>
13        <td id="userSign"></td>
14      </tr>
15      <tr>
16        <td>Total tiles:</td>
17        <td id="tiles"></td>
18      </tr>
19      <tr>
20        <td>Subtotal:</td>
21        <td id="subTotal">$</td>
22      </tr>
23      <tr>
24        <td>Shipping:</td>
25        <td id="shipping">$</td>
26      </tr>
27      <tr>
28        <td>Grand total:</td>
29        <td id="grandTotal">$</td>
30      </tr>
31    </table>
32    <a href="#" class="action">Pay Now</a>
33  </div>
34  <script src="js/example.js"></script>
35 </body>
36 </html>

```

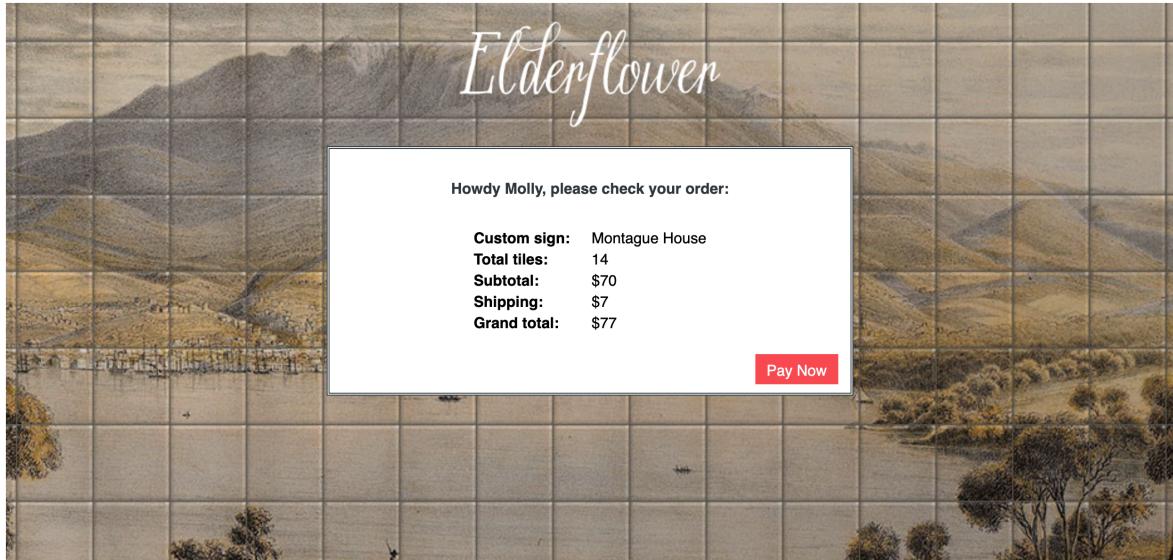
## JavaScript File:

```

1 // Create variables for the welcome message
2 var greeting = 'Howdy ';
3 var name = 'Molly';
4 var message = ', please check your order!';
5 // Concatenate the three variables above to create the welcome message
6 var welcome = greeting + name + message;
7
8 // Create variables to hold details about the sign
9 var sign = 'Montague House';
10 var tiles = sign.length;
11 var subTotal = tiles * 5;
12 var shipping = 7;
13 var grandTotal = subTotal + shipping;
14
15 // Get the element that has an id of greeting
16 var el = document.getElementById('greeting');
17 // Replace the content of that element with the personalized welcome message
18 el.textContent = welcome;
19
20 // Get the element that has an id of userSign then update its contents
21 var elSign = document.getElementById('userSign');
22 elSign.textContent = sign;
23
24 // Get the element that has an id of tiles then update its contents
25 var elTiles = document.getElementById('tiles');
26 elTiles.textContent = tiles;
27
28 // Get the element that has an id of subTotal then update its contents
29 var elSubTotal = document.getElementById('subTotal');
30 elSubTotal.textContent = '$' + subTotal;
31
32 // Get the element that has an id of shipping then update its contents
33 var elShipping = document.getElementById('shipping');
34 elShipping.textContent = '$' + shipping;
35
36 // Get the element that has an id of grandTotal then update its contents
37 var elGrandTotal = document.getElementById('grandTotal');
38 elGrandTotal.textContent = '$' + grandTotal;
39

```

Output:



## Functions or Methods

Functions let you group a series of statements together to perform a specific task. If different parts of a script repeat the same task, you can reuse the function (rather than repeating the same set of statements).

## A Basic Function

In this example, the user is shown a message at the top of the page. The message is held in an HTML element whose id attribute has a value of message. The message is going to be changed using JavaScript.

### HTML

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Basic Function</title>
5      <link rel="stylesheet" href="css/c03.css" />
6  </head>
7  <body>
8      <h1>TravelWorthy</h1>
9      <div id="message">Welcome to our site!</div>
10     <script src="js/basic-function.js"></script>
11   </body>
12 </html>
```

Before the closing </body> tag, you can see the link to the JavaScript file. The JavaScript file starts with a variable used to hold a new message, and is followed by a function called updateMessage().

### JavaScript

```
1  // Create a variable called msg to hold a new message
2  var msg = 'Sign up to receive our newsletter for 10% off!';
3
4  // Create a function to update the content of the element
5  // whose id attribute has a value of message
6
7  function updateMessage() {
8      var el = document.getElementById('message');
9      el.textContent = msg;
10 }
11
12 // Call the function
13 updateMessage();
```

### Output



## Declaring and Calling a Function

Function can store instructions for a specific task.

To create a function, you give it a name and then write the statements needed to achieve its task inside the curly braces.

This is known as a **function declaration**.

You declare a **function** using the **function** keyword.

You give the function a **name** (sometimes called an **identifier**) followed by parentheses.

The **statements** that perform the task sit in a **code block**. (They are inside curly braces.)

A diagram showing the structure of a JavaScript function declaration. It consists of three main parts: 'FUNCTION KEYWORD' (the word 'function'), 'FUNCTION NAME' (the identifier 'sayHello'), and 'CODE BLOCK (IN CURLY BRACES)' (the code block containing 'document.write('Hello!')'). Brackets above the first two parts point to them, and a bracket below the third part points to it.

```
function sayHello() {  
    document.write('Hello!');  
}
```

Having declared the function, you can then execute all of the statements between its curly braces with just one line of code.

This is known as **calling the function**.

To run the code in the function, you use the function name followed by parentheses.

In programmer-speak, you would say that this code **calls** a function.

You can call the same function as many times as you want within the same JavaScript file.

A diagram showing the structure of a function call. It consists of a single part labeled 'FUNCTION NAME' (the identifier 'sayHello') with a bracket pointing to it.

```
sayHello();
```

Sometimes you will see a function called before it has been declared. This still works because the interpreter runs through the script before executing each statement.

### Parameters

Sometimes a function needs specific information to perform its task. In such cases, when you declare the function you give it parameters. Inside the function, the parameters act like variable.

If a function needs information to work, you indicate what it needs to know in parentheses after the function name.

The items that appear inside these parentheses are known as the **parameters** of the function. Inside the function those words act like variable names.

```
function getArea(width, height) {  
    return width * height;  
}
```

THE PARAMETERS ARE USED LIKE  
VARIABLES WITHIN THE FUNCTION

This function will calculate area of a rectangle. To do this, it needs the rectangle width and height. Each time you call this function values could be different.

When you call a function that has parameters, you specify the values it should use in the parentheses that follow its name. The values are called **arguments**, and they can be provided as values or as variables.

#### ARGUMENTS AS VALUES

When the function below is called, the number 3 will be used for the width of the wall, and 5 will be used for its height.

```
getArea(3, 5);
```

#### ARGUMENTS AS VARIABLES

You do not have to specify actual values when calling a function - you can use variables in their place. So the following does the same thing.

```
wallWidth = 3;  
wallHeight = 5;  
getArea(wallWidth, wallHeight);
```

## Function Return

Some functions returns information to code that called them. For example when they perform calculation they return result. The calculateArea() function returns the area of rectangle to the code that called it.

```
function calculateArea(width, height) {  
    var area = width * height;  
    return area;  
}  
var wallOne = calculateArea(3, 5);  
var wallTwo = calculateArea(8, 5);
```

Functions can return more than one value using an array. For example the function calculate area and volume of a box and returns the result.

```
function getSize(width, height, depth) {  
    var area = width * height;  
    var volume = width * height * depth;  
    var sizes = [area, volume];  
    return sizes;  
}  
var areaOne = getSize(3, 2, 3)[0];  
var volumeOne = getSize(3, 2, 3)[1];
```

## Anonymous Functions

If you put a function where the interpreter would expect to see an expression, then it is treated as an expression, and it is known as a function expression. In function expressions, the name is usually omitted. A function with no name is called an anonymous function. Below, the function is stored in a variable called area. It can be called like any function created with a function declaration.

```
var area = function(width, height) {  
    return width * height;  
};  
  
var size = area(3, 4);
```

## Immediately Invoked Function Expression

Pronounced "iffy," these functions are not given a name. Instead, they are executed once as the interpreter comes across them. Below, the variable called area will hold the value returned from the function (rather than storing the function itself so that it can be called later).

```
var area = ((function() {  
    var width = 3;  
    var height = 2;  
    return width * height;  
})());
```

The final parentheses (shown on green) after the closing curly brace of the code block tell the interpreter to call the function immediately. The grouping operators (shown on pink) are parentheses there to ensure the interpreter treats this as an expression.

## Loops

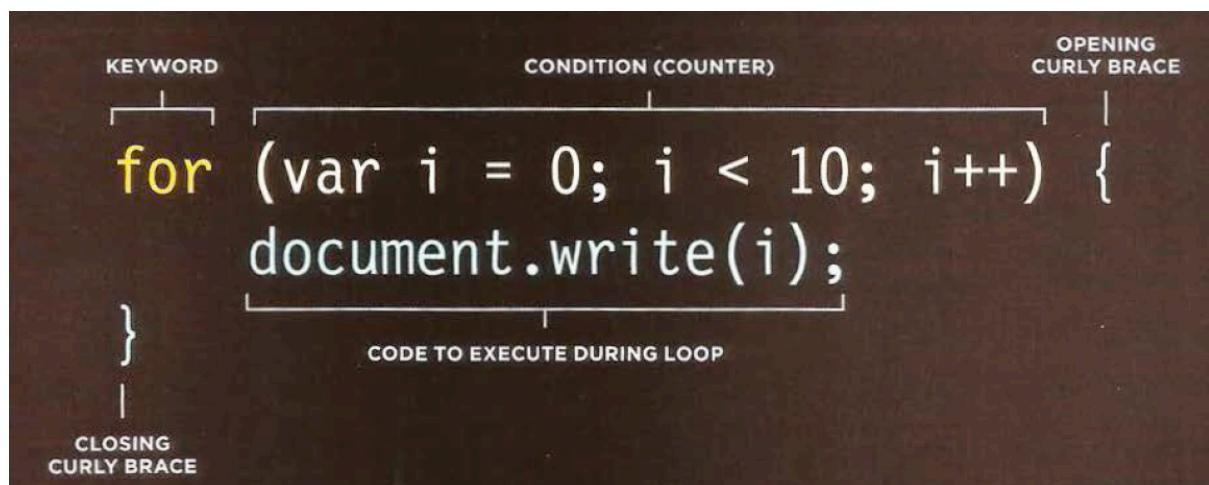
Loops checks a condition. If it returns true, a code block will run. Then the condition will be checked again and if it still returns true, the code block will run again. It repeats until the condition returns false. There are three common types of loops:

1. for loop
2. while loop
3. Do while loop

### For Loop

If you need to run code a specific number of times, use for loop. In a for loop, the condition is usually a counter which is used to tell how many times the loop should run. Here you can see the condition is made up of three statements:

```
Initialization : var i = 0 ;
Condition : i<10;
Update: i++
```



### Key Loop Concepts

You will commonly see these two keywords used with loops:

#### **break:**

This keyword causes the termination of the loop and tells the interpreter to go onto the next statement of code outside of the loop. (You may also see it used in functions.)

#### **continue:**

This keyword tells the interpreter to continue with the current iteration, and then check the condition again. (If it is true, the code runs again.)

#### **Note: PERFORMANCE ISSUES**

It is important to remember that when a browser comes across JavaScript, it will stop doing anything else until it has processed that script. If your loop is dealing with only a small number of items, this will not be an issue. If, however, your loop contains a lot of items, it can make the page slower to load.

## Using For loop

In this example, the scores for each round of a test are stored in an array called scores. The total number of items in the array is stored in a variable called arrayLength. This number is obtained using the length property of the array. There are three more variables: roundNumber holds the round of the test; msg holds the message to display.

```
1  var scores = [24, 32, 17];          // Array of scores
2  var arrayLength = scores.length; // Items in array
3  var roundNumber = 0;           // Current round
4  var msg = '';
5
6  // Loop through the items in the array
7  for (var i = 0; i < arrayLength; i++) {
8
9    // Arrays are zero based (so 0 is round 1)
10   // Add 1 to the current round
11   roundNumber = (i + 1);
12
13   // Write the current round to message
14   msg += 'Round ' + roundNumber + ': ';
15
16   // Get the score from the scores array
17   msg += scores[i] + '<br />';
18 }
19
20 document.getElementById('answer').innerHTML = msg;
21 |
```

## Using While loop

Here is an example of a while loop. It writes out the 5 times table. Each time the loop is run, another calculation is written into the variable called msg. This loop will continue to run for as long as the condition in the parentheses is true. That condition is a counter indicating that, as long as the variable i remains less than 10, the statements in the subsequent code block should run.

```
1  var i = 1;          // Set counter to 1
2  var msg = '';      // Message
3
4  // Store 5 times table in a variable
5  while (i < 10) {
6    msg += i + ' x 5 = ' + (i * 5) + '<br />';
7    i++;
8  }
9
10 document.getElementById('answer').innerHTML = msg;
11 |
```

## Using do while loop

The key difference between a while loop and a do while loop is that the statements in the code block come before the condition. This means that those statements are run once whether or not the condition is met. If you take a look at the condition, it is checking that the value of the variable called i is less than 1, but that variable has already been set to a value of 1.

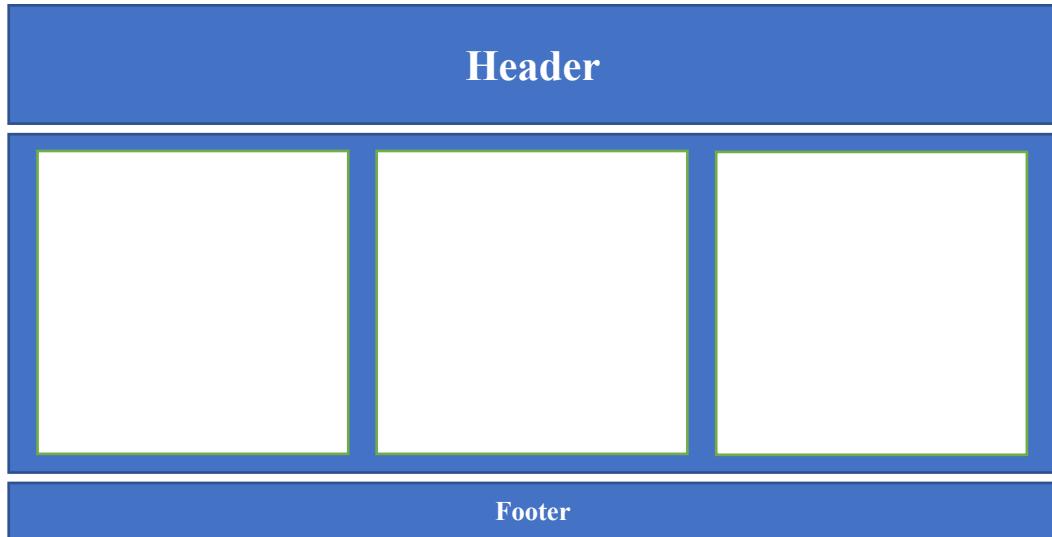
```
1  var i = 1;          // Set counter to 1
2  var msg = '';       // Message
3
4  // Store 5 times table in a variable
5  do {
6      msg += i + ' x 5 = ' + (i * 5) + '<br />';
7      i++;
8  } while (i < 1);
9  // Note how this is already 1 and it still runs
10
11 document.getElementById('answer').innerHTML = msg;
12
```

## References

1. Duckett J. JavaScript & jQuery. Wiley VCH; 2015.

**Lab Tasks:**

1. Create a 3- Column website layout using CSS and display it in html file with proper border, margin, background and padding.



2. Write a JavaScript function that will take 2 numbers as parameters and calculate first 20 multiples of each number. Store the multiple of each numbers as text content in two variables. Display the multiples of first number in 1<sup>st</sup> column of the above layout and display multiples of second number in 2<sup>nd</sup> Column of the above layout.
3. Write a JavaScript code that will display sum, subtraction, product and division of two numbers in the 3<sup>rd</sup> column of the above layout.