## What is JavaScript?

When the web was conceived, browsers were limited to text and image - later tables and frames. The extent to which they provided interactivity with the user was very limited. JavaScript was developed by Netscape as a simple programming language (often referred to as a scripting language). It is easy to learn and small sections of JavaScript can be added to a web page rather than needing to develop complicated programs. It was specially designed for web page interaction and manipulating the web browser and page elements. It is often used to respond to user actions such as mouse clicks. Although developed by Netscape, and other variants exist, such as JScript from Microsoft, a standard has been developed by the European Computer Manufacturers Association. It is known as ECMAScript, using the standard ECMA262, which can be found fully documented at the address on the slide.

- JavaScript is called a scripting language
- JavaScript is interpreted, each of code is translated to machine code and run a line at a time.
- The interpreter is built into the web browser. So no need to compile.
- A browser reads a web page with a JavaScript program in it, the web browser translates the JavaScript in order to let the computer understands.
- A scripting language operates slowly than a compiled language, so every time it runs JavaScript, the program must be translated for the computer.
- Javacript is an object-based client-side scripting language
- Browsers have limited functionality
    - Text, images, tables etc
- JavaScript allows for interactivity
- Browser/page manipulation
    - Reacting to user actions
- A type of programming language
    - Easy to learn

## What is scripting language?

- Scripting language is good for Web developers Scripts are less complex and smaller than other desktop languages such as java, c++ etc.
- Using Scripting language is fast process.
- All Web Browsers are built to understand HTML and CSS and execute these languages, in order to display on the computer screen.
- Browsers are built in JavaScript interpreter in order to execute JavaScript.
- So, if you want to add JavaScript in your webpage, you need to tell the browser by using the <script> tag.
- When a browser meet the closing </script> tag, it knows it is end of JavaScript, so it will get back to its normal behaviour.

## A Limited-Featured Programming Language

You can do many things using JavaScript that you can’t do by simply using HTML. Here are a few of them:

- Build dynamic web pages
- Display alert boxes
- Control features of the browser
- Open new browser windows
- Customize reactions to mouse actions and keystrokes
- Perform calculations
- Validate information in forms
- Create interactive forms

Although JavaScript is more powerful than HTML, JavaScript can’t do everything. Here are some common things that JavaScript **can’t do**:

- Write files to your hard disk
- Read files from your hard disk (except for cookies)
- Close windows other than those the JavaScript application opened
- Read information from a web page that resides on a domain different from the domain where the JavaScript resides

## How Does It Work?

JavaScript is embedded/included within HTML. You can often see JavaScript in the source of a web page or it is provided for information on the page as with the calculator example.

JavaScript is mainly used as a client-side language - it downloads with the web page. Once the page has downloaded and is on the users' machine, it is actually the web browser which then interprets the JavaScript instructions. JavaScript pages run quickly, you are not relying on an internet connection to a web server. Short pieces of JavaScript can be combined with HTML without the need to develop a fully blown program.

There are two types of computer language, compiled and interpreted. To write or edit a compiled language requires a special piece of software called a compiler. JavaScript belongs to the other category, called interpreted. In the case of JavaScript, this interpretation is done by the browser software at run-time. Because JavaScript is interpreted, this means that no special tools are required to write or edit JavaScript, just a normal text editor. JavaScript web pages can be platform independent i.e. they will run on different browsers and computers (as long as the browser is JavaScript enabled). If you see a JavaScript web page that you like, you may be able to take that JavaScript and use it for your own purposes. (Remember to acknowledge the original author!)

- Embedded within HTML page
  - View source
- Executes on client
  - Fast, no connection needed once loaded

- Simple programming statements combined with HTML tags
- Interpreted (not compiled)
    - No special tools required

## Interactivity

HTML pages are static .......

.... Using JavaScript - now they become more interactive :

- The page “responds” to user activity,
- It changes somehow in response to user input.
- User input needs to be detected.
- The browser needs to know what to do in response to which activity

## The <script> Element

- The <script> Element is part of HTML.
- Is used to place script code, e.g. JavaScript statements and data, into the HMTL page.
- You cannot place HTML code in a <script> section.
- Is often placed into the <head> section
- Can also place scripts in external files.
- The script is interpreted when the page is loaded unless specified differently.

## <Script> tag

You can add the <script> tag in the <head> section like this:

```
<head>
<script type="text/JavaScript">
 ...... script code goes here.........
</script>
</head>
For Example:
<head>
<title> My first JavaScript webpage</title>
<script type="text/JavaScript">
document.write("Hello World!");
</script>
</head>
```

Syntax: Element name: *script*, Required attributes: *type*, Type values: *text/javascript*

## When to use Comments in JavaScript

When you need to remind yourself what you have done your code successfully and how you did it and what you have achieved such as

- What is that variable for? and
- why would you program it like this?
- What is going on in this section of the program?
- Why is this new feature? Etc.

Using comments are for as follows:

- Add the comments to help you understand the overall logic of the program
- Explain any particularly confusing or complex codes.

Adding a lot of comments to a script make the script slower to download but also make it easier to remember the logic behind the code. There is a trade off between machine efficiency and programmer efficiency.

## Comments in JavaScript

JavaScript supports two types of comments:

Double-slashes (//)

- JavaScript will ignore everything to the end of the line.
- used most often to describe what is happening on a particular line
- // …………………..

Block quotes ( /* …. */)

- JavaScript will ignore everything from the start of the comment block (/*) until it encounters an asterisk-slash (*/).
- Block quotes are useful for detailed comments across many lines or for temporally disabling large areas of code,
  /* ……
  …….
  …….. */

## Scripts can be found in three places:

Scripts can be found in three places:

Head Section:

- Scripts placed in the head section are loaded before anyone can use it and are only executed when they are called or when triggered by an event.

Body Section:

- Scripts placed in the body section are executed when the page is loaded and can be used to generate page content.

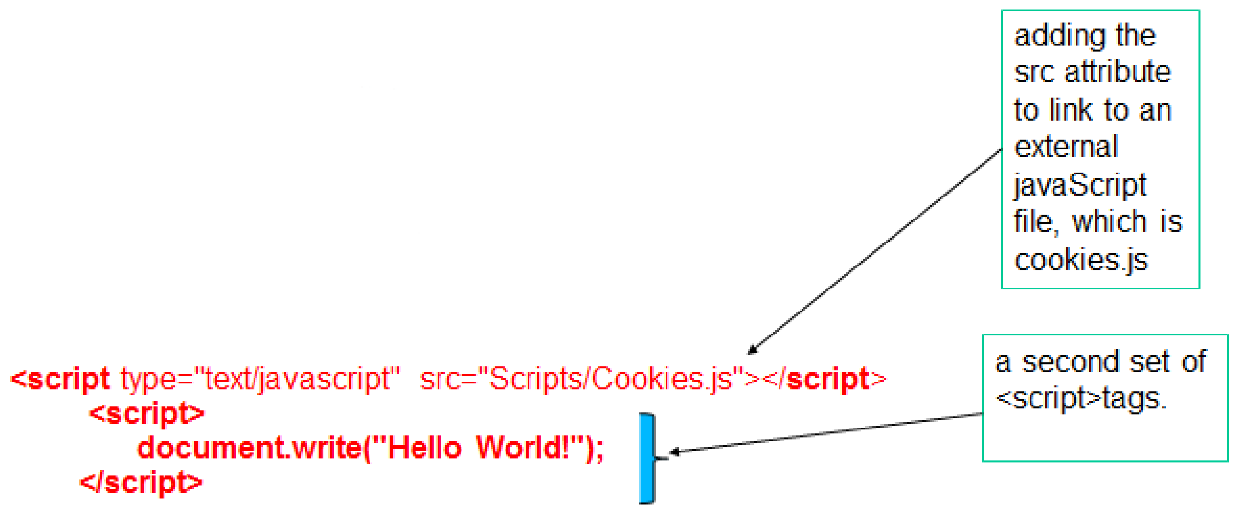
External File:
This allows us to write scripts that can be used across several pages. The script is written once and saved in a file with a .js extension. The external script file does not contain the <script> tag. The script tag will be used in the HTML document to point to the external script file.

## Place JavaScript in External file

External File:

- This allows us to write scripts that can be used across several pages. The script is written once and saved in a file with a .js extension. The external script file does not contain the <script> tag. The script tag will be used in the HTML document to point to the external script file.
- The src attribute of the <script> tag works same as the src iattribute of an <img> tag, or an <a> tag's href attribute.
- When adding the src attribute to link to an external javaScript file, no need to add any JavaScript code between the opening and closing <script></script> tags.
- But if you want to add some JavaScript code to a page for extra functions, use a second set of opening and closing <script></script> tags.

For Example:



```
<script type="text/javascript"  src="Scripts/Cookies.js"></script>
      <script>
          document.write("Hello World!");
      </script>
```

adding the src attribute to link to an external javaScript file, which is cookies.js
a second set of <script>tags.

## Hello World Example

```
<!DOCTYPE HTML>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <title>Hello World</title>
</head>
<body>
<script type="text/javascript">
//Script within Body Section (Not very common – Usually Head Section)

// how to use document.write(..) to write a text on a web page
// Comment

document.write("Hello World");
//Display “Hello World” Text on Web Page
</script>
</body>
</html>
```

## Writing a text on a webpage

When you want to show a message directly onto your web page using JavaScript. You can use alert () or document.write() function, they are JavaScript commands which allow you to write out whatever you want to your webpage. Remember you need to place these functions between opening and closing <script> tags.

## For example: using alert() function

```
<script>
   alert(“Have a good day!");
</script>
```

## For example: using document.write() function

```
<script>
   document.write("Hello World!");
</script>
```

## Make your webpage more interactive with user by using events

JavaScript allows for interactivity, which means JavaScript allows your webpages react to something a user does, such as when you move your mouse over a navigation button which contain a link or set of links. You can also create a radio button which allow user to select a set of form options; When you clicking a image makes the image larger or change colour, etc.

## What are Events?

Events are when each page loading or moving a mouse or typing a keyword or resizing the browser window or select the options from a radio button, etc.

- Often trigger the execution of scripts.
- Occur when the page is loaded, rendered or being browsed by the user.

To make the webpage more interactive, we write programs that respond to events, e.g. click the submit button when you have completed your booking form online, etc.

## Handling Events

Events go unnoticed unless an event handler for this event has been implemented. Event handlers can be attached to HTML elements, e.g.

```
<body onload="alert('Hiya!');">
```

We will come back to Events/Event Handlers – but they may appear in certain examples

## "If" statement

checks an age – pay senior fare if 60 or above ...

```
function TestAge()
{
 var age = parseInt (document.getElementById("in_Age").value) ;

 if (isNaN(age))
  {
   alert("The input is NOT valid");
  }
  else
  {
    if( age >= 60 )
     {
      alert("You pay the senior fare! ");
     }
    else
     {
      alert("You pay the regular adult fare. ");
     }
  }
}
```

## Invalid Data

- Must always check for INVALID data
- Software must deal with all scenarios that may cause errors
- parseInt() function: it takes a value and tries to convert it to an integer (whole number, e.g. 1, 4,8, 10 etc).
- If user types in text, the parseInt() command won't able to convert text to number. That is why, we need isNan() function to check if user enter text, then the webpage will pup up an alert box and tell the user, it is an invalid number.
- The isNan() function in JavaScript.
- NaN stands for "not a number", you can use this information to pop up another prompt dialog box, if a number is not entered the first time.

## Boolean Logical Operators

Boolean data type is either true or false.

And

condition_a && condition_b

Is true if condition_a and condition_b are both true.

OR

condition_a || condition_b

Is true if condition_a or condition_b are both true.

NOT

! condition

Is true : if condition is false

or ….

is false : if condition is true

## If statement

(assume ‘age’ has been validated using parseInt)

```
if( age > 0 && age <= 12 )
 {
  alert("You pay the child's fare. ");
 }
 else if( age > 12 && age < 60 )
  {
   alert("You pay the regular adult fare. ");
  }
  else if ( age >= 60 )
   {
    alert("You pay the senior fare! ");
   }
  else
   {
    alert("Invalid Age ");
    // catch error age conditions //
   }
```

## Boolean

The userID MUST be "brian" AND the password MUST be “gcu"

```
function Logon() {
var userID =  document.getElementById("in_Id").value;
var password  = document.getElementById("in_Pass").value;
if (userID == "brian" && password == "gcu")
 {
   alert ('Logon valid -  Welcome !!!')
 }
else
 {
   alert ('Logon invalid')
 }
}
<body>
 Enter User ID : <input type="text" id="in_Id" size="10">
<br/>
 Enter Password : <input type="text" id="in_Pass" size="5">
<br/>
 <input type="button" value="Logon" id="B1" onClick="Logon()">
</body>
```

## Examples:

- http://www.w3schools.com/js/js_if_else.asp

Note that these examples using some Javascript methods

- E.g Date(), get_hours() etc

These are all defined in the JS References Section at w3schools :

- JS References -> JavaScript Objects -> Date Object

## Loops

Loop is part of conditional statement.
Use Loops when you want to execute code potentially several times.

Examples:

- Add up the numbers from 1 to 100.
- A countdown.
- Add all the correct answers marks for online class test

## The for Loop

Syntax:

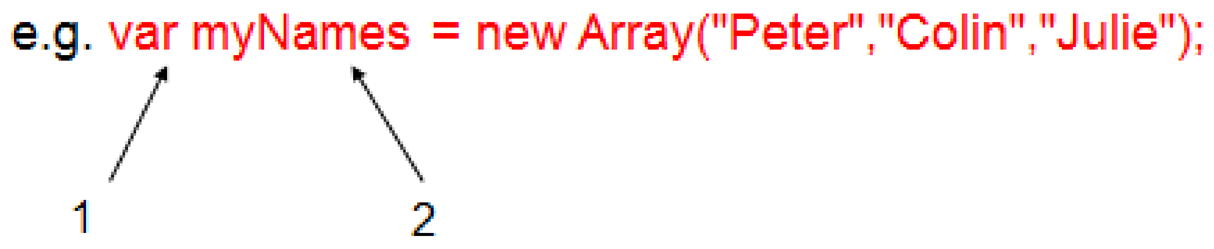
```
for ( <init>;   <test>;   <increment>)
{
//...code to execute...
}
Example:
for (var i = 0 ;    i<5;    i++)
{
document.write("i is .... " + i);
document.write("<br/>");
}
```

**Output**

```
i is .... 0
i is .... 1
i is .... 2
i is .... 3
i is .... 4
```

## Arrays

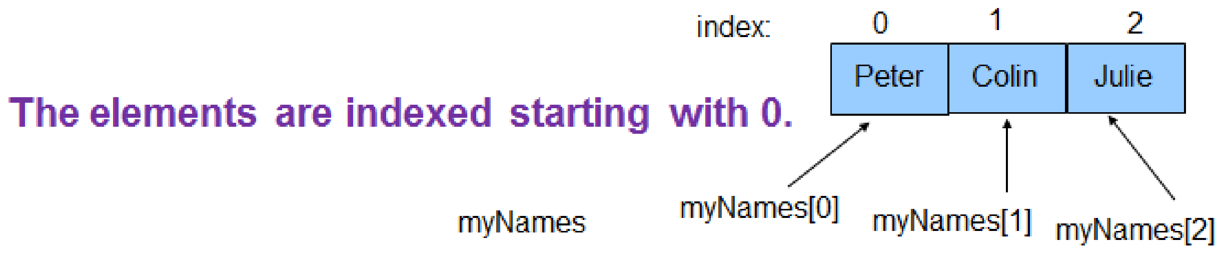
Array is a variable, it perfect to keep track of a group of related items or images- e.g. name of days in a week e.g. Monday, Tuesday, Wednesday…Sunday. Or the list of images on a web page
An Array is used to store multiple values in a single variable.
It can store one or more elements
It must have a Unique name
**A variable** is a way to store information so that you can later use and manipulate it.
It is a two-step process to create a variable.
1. declaring the variable
2. name the variable.



```
e.g. var myNames = new Array("Peter","Colin","Julie");
```

Defining an Array :

```
var myNames = new Array("Peter","Colin","Julie");
OR :
var myNames = new Array;
myNames[0] = "Peter"
myNames[1] = "Colin"
myNames[2] = "Julie"
```



The elements are indexed starting with 0.

To iterate over all elements of the array:

```
for (var i = 0; i < myNames.length; i++) {
//statements go here;
}
```

- i is initialised to 0 -> start of array
- The array length controls how many times the For loop is used - i.e. number of elements in the array ( from 0 to length – 1)
- will exit loop when i = length
- i++ - ensures we index each item in the array

```
<script type="text/JavaScript">
function showArray()
{
  // initialise array
  var myNames=new Array("Peter","Colin","Julie");
  // get array length
  var len = myNames.length

  document.write ( " The Length (total number of elements) of myNames is " , len );
  document.write("<br/>");

   // use Array length to access each element
   for (var i = 0; i < len; i++)
   {
     document.write ( "Element " + i + " is   " + myNames[i])
     document.write("<br/>");
   }
}
</script>
</head>
<body>
<h1>Array Example</h1>
<script type="text/JavaScript">
<!--
showArray()  .........    USES a Function
// -->
</script>
```

## Sort()

sort() - method used to sort the Array
Use : myNames.sort();

```
// initialise array
var myNames=new Array("Peter","Colin","Julie");
myNames.sort();

   // use Array length to access each element
   for (var i = 0; i < myNames.length; i++)
   {
     document.write ( "Element " + i + " is   " + myNames[i])
     document.write("<br/>");
   }
```

- Element 0 is Colin
- Element 1 is Julie
- Element 2 is Peter

## The String Object

String is used any name, text, sentence or any letters, words, or paragraph.
A string is a series of letters and other symbols.
Syntax:

```
var myStr  =  new String(“Hello”);
```

Once a string’s opening quote mark signals to the JavaScript interpreter that the follows is string. So interpreter will accept what it is without trying to interpret the string like other commands
or – more simply …

```
var myStr  = “Hello”;
```

Properties:

- length: the total number of characters in a String

```
var strlength = myStr.length
```

return: 5

## How to Joining strings?

Joining strings : Simple just use “+”

```
var str1 = “hello”
var str2 = “world”
var str3 = str1 + " " + str2
document.write (str3)
-> “hello world”
```

The String Object

String object reference:
http://www.w3schools.com/jsref/jsref_obj_string.asp
MANY of the following examples are from the above website.
You should refer to this website to view AND execute the code
You can also modify the code

## charAt()

charAt()

- Returns the character at the specified index
- First character is at position : ‘0’
- Last Character is at position : length – 1

| Parameter | Description |
|---|---|
| index | Required. An integer between 0 and string.length-1 |

```
var str = "Hello World!";
str.charAt(0)
//-> H     (output: the first letter of Hello World!)

str.charAt(6)
//-> W     (output: the six letter of Hello World!)
```

## indexOf() / lastIndexOf()

indexOf() - returns the position of the FIRST occurrence of a specified value in a string.
Useful link: http://www.w3schools.com/jsref/jsref_indexof.asp lastIndexOf() - returns the position of the LAST occurrence of a specified value in a string.
returns -1 if the value to search for NEVER occurs.
Useful link: http://www.w3schools.com/jsref/jsref_lastindexof.asp

| Parameter | Description |
|---|---|
| searchstring | Required. The string to search for |
| start | Optional. The position where to start the search. If omitted, the default value is the length of the string |

## Example of str.indocOf() and str.lastIndexOf()

```
var str="Hello world!";
```

indexOf :

```
str.indexOf("d")
// return  "10"

str.indexOf("world")
// return "6"

str.indexOf("WORLD")
// return "-1"
```

lastIndexOf :

```
str.lastIndexOf("o")
// return  "7"
```

## slice()

extracts a part of a string and returns the extracted part in a new string.
Otherwise it returns -1.
Useful link: http://www.w3schools.com/jsref/jsref_slice_array.asp

| Parameter | Description |
|---|---|
| begin | Required. The index where to begin the extraction. First character is at index 0 |
| end | Optional. Where to end the extraction. If omitted, slice() selects all characters from the begin position to the end of the string |

```
var str="Hello happy world!";
// extract all characters, start at position 6:
str.slice(6) //return:  happy world!

// extract only the first character:
str.slice(0,1) //return: H

// extract the characters from position 6 to position 11:
str.slice(6,11) //return: happy
```

## substring()

extracts the characters from a string, between two specified indices, and returns the new sub string.
extracts the characters in a string between "from" and "to", not including "to" itself.
Useful link: http://www.w3schools.com/jsref/jsref_substring.asp

| Parameter | Description |
|---|---|
| from | Required. The index where to start the extraction. First character is at index 0 |
| to | Optional. The index where to stop the extraction. If omitted, it extracts the rest of the string |

```
var str="Hello world!";
str.substring(3)  //return: lo world!
str.substring(3,7)  //return: lo w
```

## toUpperCase / toLowerCase

toUpperCase() - converts a string to uppercase letters.
Useful link: http://www.w3schools.com/jsref/jsref_touppercase.asp
toLowerCase() - converts a string to lowercase letters.
Useful link: http://www.w3schools.com/jsref/jsref_tolowercase.asp

```
var txt="Hello World!";
txt.toLowerCase() // return: hello world!
txt.toUpperCase() // return: HELLO WORLD!
```

## split()

used to split a string into an ARRAY of substrings, and returns the new array.
Useful link: http://www.w3schools.com/jsref/jsref_split.asp

| Parameter | Description |
|---|---|
| separator | Optional. Specifies the character to use for splitting the string. If omitted, the entire string will be returned |
| limit | Optional. An integer that specifies the number of splits |

```
  var str = "How are you doing today?";
  NewArray = str.split(" ")
  // -> "How" "are" "you" "doing" "today?"
NewArray :  "How"   "are"   "you"   "doing"   "today?"


              0       1       2       3         4
```

```
  NewArray = str2.split("=")
  //return: "name" "Brian"
NewArray :    "name"   "Brian"
                0        1
```

## Examples -1 string and array

Can divide a string into various elements - and put them into an array

- Use split(" ") to get each element – separated by a " " character
- Elements are then assigned to an Array
- NewArray.length will now contain number of elements in string AND array

```
  var DataString = "Tom Fred Joe Ian"
  document.write("The String is : ", DataString);

  var NewArray = DataString.split(" ")

  document.write("The Array elements are: ");
   for (i=0; i<NewArray.length; i++)
    {
      document.write(NewArray[i])
      document.write('<br>')
    }
```

```
The String is : Tom Fred Joe Ian
The Array elements are : Tom,Fred,Joe,Ian
```

This example gets the email address from the user
It breaks it down into the username and website - uses split("@")
Full website is updated by appending “www”

```
var emailAddress = “b.shields@gcu.ac.uk”

var NewArray = emailAddress.split("@")

var username = NewArray[0];
document.write("Your user name is ... ", username);

var website = NewArray[1];
document.write("The website is ... ", website);

var fullwebsite = "www." + website
document.write("The full website is  .... ", fullwebsite);
Your user name is ... b.shields
The website is .... gcu.ac.uk
The full website is ...www.gcu.ac.uk
```