CO456
Web

- most materials adapted from ***Moseley (2007)***, Chapter 5 – supplemented with extracts from Bates (2006) and w3schools.com

Week 6

JavaScript ***branches, loops*** & ***functions***

Module schedule

Wk.	Lecture/subject area(s)	Practical	Reading (Moseley, 2007)
1	Introduction How the Web works	Internet/Web definitions and HTML report	Ch 1 (The way the Web works)
2	HTML 1 (Introductory - inc. lists and hyperlinks)	HTML	Ch 2 pp 24-36 (HTML)
3	HTML 2 (inc. tables, images and forms)	HTML	Ch 2 pp 36-48 (HTML) Ch 3 (XHTML and frames)
4	CSS 1 (Introduction and core CSS principles)	CSS – introductory styles, embedded styles.	Ch 4 pp 76-96.
5	CSS 2 (Positioning elements).	CSS– using IDs, classes and layout control.	Ch 4 pp 97-103.
6	CSS 3 (Advanced layout & navigation)	CSS – using CSS to produce button-like navigation from HTML list elements. (CW2a to be demonstrated).	Specialised articles.
7	JavaScript 1 (Fundamentals, variables)	JS – foundation constructs.	Ch 5 pp 108-116
8	Guided Learning Week	Consolidate Internet & W3 knowledge and HTML & CSS skills.	Review Ch 1 to Ch 4.
9	JavaScript 2 (Functions, branches, loops).	JS – calling functions.	Ch 5 pp 117-124.
10	JavaScript 3 (Objects and the DOM).	JS – manipulating the DOM.	Ch 6 126-139.
11	JavaScript 4 (Forms and validation). And DHTML	JS– validating user completed forms.	Ch 6 139-145, Ch 7.
12	HTML***5***, CSS***3***, - media, forms, gradients, SVG ('Edge') and other enhancements	Web frameworks taster session 1	See practical sheets for information sources
	Vacation		
13	Advanced HTML5, CSS3 & JS frameworks (e.g. jQuery, jQuery Mobile, Box2DWeb)	Web frameworks taster session 2	See practical sheets for information sources
14	Assignment workshop 1	Assignment workshop 1	N/A
15	Assignment workshop 2	Assignment workshop 2	N/A

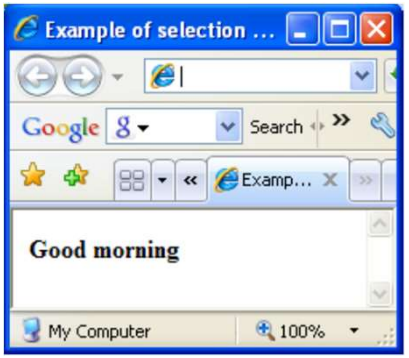
JavaScript - Branches

- Three fundamental programming structures – *sequence*, *selection* (branches) and *iteration* loops
- In JavaScript there are the following branching - selection statements:
  - if statement - use to execute some code only when one specified condition is true
  - if...else statement - use to execute code if the condition is true and to execute another code if the condition is false
  - if...else if....else “ladder” statement - use to execute different blocks of code if there are more than two conditions
  - switch statement – an alternative and more ‘condensed’ means of achieving “if...else if....else”
  - the “conditional operator” … var=(condition) ?value1:value2 (NOTE: this links to a short script on W3Schools that also demonstrates much about the DOM – definitely worth a closer look!!!)

JavaScript - Branches

Example of an "*if - else if - else*" ladder with the Date class

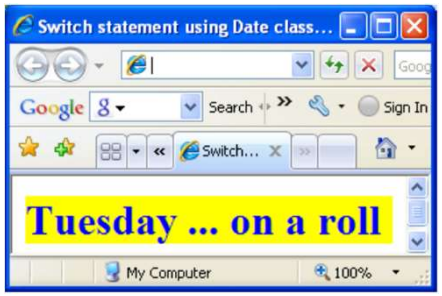
```
<script type="text/javascript">
        var d = new Date()
        var time = d.getHours()
        if (time<12)
        {
                document.write("<b>Good morning</b>")
        }
        else if (time>=12 && time<18)
        {
                document.write("<b>Good afternoon</b>")
        }
        else
        {
                document.write("<b>Good evening</b>")
        }
</script>
```



JavaScript - Branches

Example of a "*switch*" statement with the Date class

```
<script type="text/javascript"> //Greeting depends on what day it is - Note: Sunday=0,
    var d=new Date(); theDay=d.getDay();
    switch (theDay)
    {
     case 1:  document.write("<h1><p style=\"color:blue; background:yellow; font-weight=bold\">Oh!  ...
     Monday</p></h1>")
       break
     case 2:  document.write("<h1><p style=\"color:blue; background:yellow; font-weight=bold\">Tuesday ... on a
     roll</p></h1>")
       break
     case 3:  document.write("<h1><p style=\"color:blue; background:yellow; font-weight=bold\">Wednesday ...
     midweek already</p></h1>")
       break
     case 4:  document.write("<h1><p style=\"color:blue; background:yellow; font-weight=bold\"> Thursday ... Nearly
     the w/e</p></h1>")
       break
     default:
       document.write("<h1><p style=\"color:blue; background:yellow; font-weight=bold\">relax - the weekend!
     </p></h1>")
    }
</script>
```



JavaScript - Branches

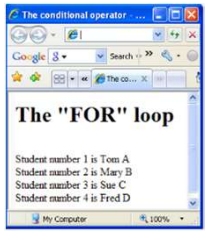
Example of the "*conditional operator*" with the Date class

Note: *don't forget to escape quotations in string variables!*

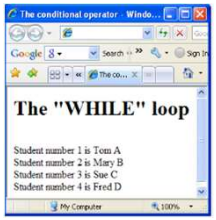
```
<script type="text/javascript"> //Rule ... var=(condition) ?value1:value2
    var d=new Date();
    theMonth=d.getMonth(); //Months are numbered 0 to 11

    season=(theMonth>=9 || theMonth<=2) ?
   "<p style=\"color:white; background:blue; font-weight=bold\">winter</p>"
   :
   "<p style=\"color:red; background:orange; font-weight=bold\">summer</p>"

    document.write("It must be " + season);
</script>
```

JavaScript – Iteration (loops)

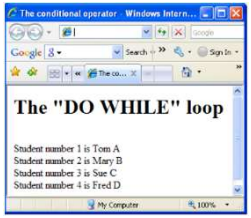
- Allow iteration/repetition of sections of code
- Three types of loops – “for”, “while” and “do while”
- For loop:
  - rule: for(start value; limiting condition; increment)
  - counter (e.g. “i”) may also be used to process contents of an array
  - e.g. **for(i=0; i<=3; i++) {** *document.write(myArray[i]);* **}**



- While loop:
  - must declare and initialise counter first (e.g. “count”)
  - e.g. **count = 0; while (i<=3) {** *document.write(myArray[i])* ; i++; **}**



- Do while loop
  - again, must declare and initialise counter first (e.g. “count”)
  - tests condition ***after*** loop so will always execute at least once
  - e.g. **count = 0; {** *document.write(myArray[i])* ; i++; **} while (i<=3)**



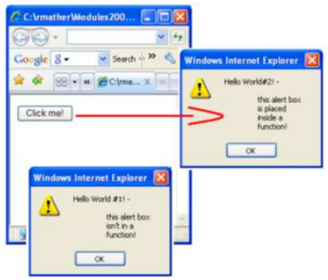
JavaScript – Functions

- Separate from the main program
- Principle of “write once use many”
- Advantages to breaking a program up into discrete subroutines include:
  - reducing the duplication of code in a program
  - enabling reuse of code across multiple programs,
  - decomposing complex problems into simpler pieces (improves maintainability and ease of upgrade)
  - improving readability of a program
  - hiding or regulating part of the program (see “Information hiding”)
- The components of a subroutine include:
  - always - a body of code to be executed when the subroutine [function] is called
  - sometimes - parameters that are passed to the subroutine from the point where it is called
  - sometimes - a value that is returned to the point where the call occurs [
  - remember: returned values may be capture by a variable. For example, to capture a value returned by JavaScript confirm/prompt box … **var c=confirm("Select one ...")**

JavaScript – Functions

- Functions contain code that is executed either by an event or by a call to the function.
- Functions are often defined in the <head> section.

```
<html>
<head>
	<script type="text/javascript">
		function displaymessage()	//This is only executed by the “onclick” event of the form below
		{	alert("Hello World#2! – \n\n\tthis alert box\n\tis placed \n\tinside a \n\tfunction!")	}
	</script>
</head>

<body>
<form><input type="button" value="Click me!" onclick="displaymessage()" ></form>

<script type="text/javascript">
	alert("Hello World #1! – \n\n\tthis alert box\n\tisn’t in a \n\tfunction!")	//Not a function - executed on page load
</script>
</body>
</html>
```

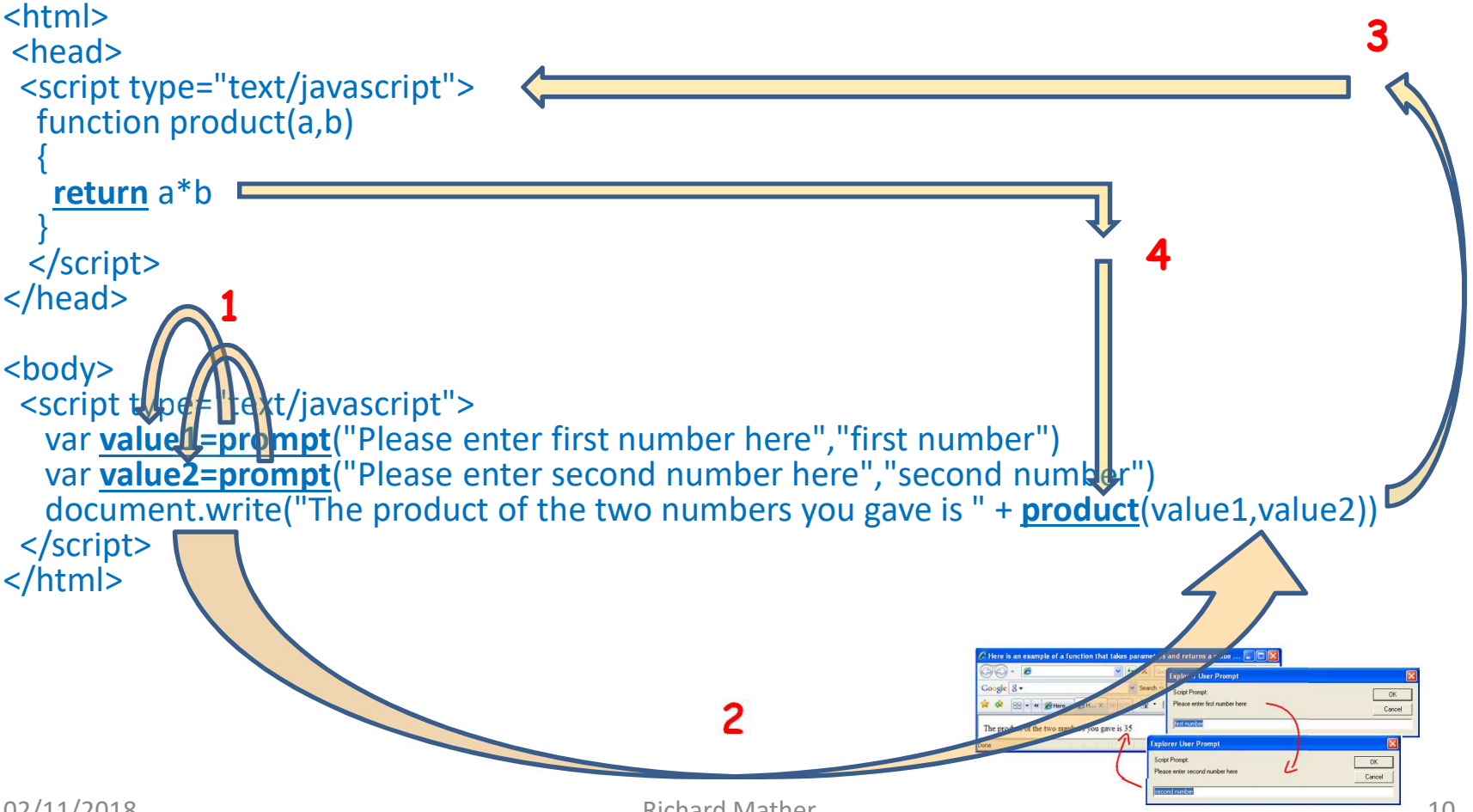


- [1] alert("Hello World #1!") is executed as soon as the page is loaded.
- [2] alert("Hello World #1!") is only executed by the onClick event which calls the function displaymessage() when the button is clicked by a user.

JavaScript – Functions

Here is an example of a function that takes ***parameters*** and ***returns*** a value

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

<body>
  <script type="text/javascript">
     var value1=prompt("Please enter first number here","first number")
     var value2=prompt("Please enter second number here","second number")
     document.write("The product of the two numbers you gave is " + product(value1,value2))
  </script>
</html>
```



JavaScript summary

- JavaScript provides “normal” programming facilities for implementing sequence, selection and iteration using “familiar” C-language syntax
- JavaScript provides many internal functions (including readymade pop up ones - alert, prompt, confirm) and allows developers to write their own
- Functions allow repeatable, reusable code to be separated so that it may be executed by events/calls
- Placing functions in the <head> element ensures that they are read and loaded by browser before being called by some event
- The basic “C-like” syntax for ***defining*** a function is …
  - function *functionName*(*parameter1, ...,parameterX*) { *some code; return someVariable (optional); }*

- NEXT WEEK – JavaScript objects and the Document Object Model

Practical 6

- Work towards assignment 2a
- Rewrite last week's JS solution using loops, functions and the Array.length property
- Produce a simple popup driven calculator with interface similar to the one below