PROGRAMMING IN A SCRIPTING LANGUAGE

When building a web site, you work through three layers:

1. First you start by producing the **content** in HTML format. This is the base layer which any user can view using any browser.

2. Secondly you make the site look better by using CSS (**presentation**).

3. Finally you can add a scripting language to add functionality to a web page. For example you can use JavaScript to add interaction with a user or for **dynamic behavior**.

We will focus on this last layer (JavaScript).

JavaScript is a scripting language most often used for client-side web development. It was originally developed by Brendan Eich of Netscape under the name Mocha, later LiveScript, and finally renamed to JavaScript. The change of name from LiveScript to JavaScript roughly coincided with Netscape adding support for Java technology in its Netscape Navigator web browser. JavaScript was first introduced and deployed in the Netscape browser version 2.0B3 in December of 1995. The naming has caused confusion, giving the impression that the language is a spinoff of Java and has been characterized by many as a marketing ploy by Netscape to give JavaScript the cachet of what was then the hot new web-programming language.

There have been efforts to standardise JavaScript. In fact, there's a JavaScript standard now. ECMAScript is the international standard name and specification for the JavaScript and Jscript. For more information you can visit their website at http://www.ecma-international.org/

ECMA stands for European Computer Manufacturers Association. Ecma International is an industry association founded in 1961 and dedicated to the standardization of Information and Communication Technology (ICT) and Consumer Electronics (CE).

1. Using JavaScript

You can add JavaScript to your web pages in a number of ways:

1. Embed it directly into your HTML document (inline). This **should not be used today**, but you may come across it on some old web pages.
```
<a href="somedoc.html" onClick="JavaScript code here">
```

2. Included in an HTML document in <script> tag. This **should also not be used now days**.
```
<script type = "text/JavaScript">
      script code goes here
</script>
```

3. In an external file (*.js) then link to the HTML document using 'src' attribute
```
<script type="text/JavaScript" src="scriptFileName.js">
</script>
```
In HTML5 it's `<script src="scriptFileName.js"></script>` 'type' is omitted

External script files are the way to go because:
1. they enhance usability without cluttering the HTML document
2. they do not cause problems for users that have JavaScript disabled
3. more than one HTML page can use the external file
4. they do not interfere with other JavaScript code that might be applied to the HTML page

External scripting is called **unobtrusive scripting**.

2. Display a Message

We will look at two ways to display a message.

i) First write to a screen page using the *write()* method.

Exercise 1

1. You will create your first Web page as follows and save it as **ex1-1.html**. This script will write a line to the Web page.

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>A JavaScript Example</title>
</head>

<body>
  <script>
      document.write("Hello and welcome to JavaScript!");
  </script>
</body>
</html>
```

The typical syntax is:	**object.method()**
for example	**document.write()**

2. Now we shall improve on this (ex1-1.html) by separating the JavaScript from the HTML by creating an external script file. Type the following HTML and save as **ex1-2.html**

```
<!doctype html>

<html lang="en">

<head>
  <meta charset="utf-8">
  <title>A JavaScript Example</title>

</head>

<body>
  <script src="ex1-2.js"></script>
</body>
</html>
```

Now type in a new notepad file:

```
document.write("Hello and welcome to JavaScript!");
```

and save it as **ex1-2.js**

ii) You can display a message in a window using the *alert()* method.

3. Now you will create a page where an alert window will pop up, with a message.
Open **ex1-2.html** and save as **ex1-3.html**. In **ex1-3.html** change the 'src' attribute to **src="ex1-3.js"** as shown below:

```
<script src="ex1-3.js"></script>
```

Now type in a new notepad file:

```
window.alert("Hello and welcome to JavaScript!");
```

and save it as **ex1-3.js**

The typical syntax is:	**object.method()**
for example	**document.write()**
	window.alert()

The 'window' object is the most basic object and it actually goes in front of all objects, that is,

```
window.document.write();
window.alert();
```

Because the 'window' object is so basic, we can omit it, that is,

```
document.write();
alert();
```

A line of code is called a **statement**.

You can write two statements in three different ways:

i) a new line will separate two different statements

```
document.write("Hello Mary")
document.write("Hello John")
```

ii) a semicolon **(;)** will separate two statements

```
document.write("Hello Mary"); document.write("Hello John");
```

iii) do both

```
document.write("Hello Mary");
document.write("Hello John");
```

The third (iii) way is the best way to go because it is easiest to read and will avoid errors if two statements accidentally come together. Also PHP, the server-side language you will learn, must have **';'** at end of each statement.

Exercise 1 continued

4. Now your script will write two statements to the web page.
   Open **ex1-3.html** and save as **ex1-4.html**. In **ex1-4.html**, change the src attribute **to src="ex1-4.js"**

   Now type in a new notepad file:

```
document.write("Hello and welcome to JavaScript! ");
document.write("This is my first lesson.");
```

   and save it as **ex1-4.js**

3. New Line

You can create a new line of text by using the "**
**" tag as shown in exercise below.

Exercise 1 continued

5. Open **ex1-4.html** and save as **ex1-5.html**. In **ex1-5.html**, change the src attribute **to src="ex1-5.js"**

   Now type in a new notepad file:

```
document.write("Hello and welcome to JavaScript!<br>");
document.write("This is my first lesson.");
```

   and save it as **ex1-5.js**

4. Inserting JavaScript Comments

A browser will ignore a comment. Comments are there to be read by you or other programmers who may need to work on your program. Comments explain the code so that updating at a later time becomes an easier task.

There are two types of comments:

i) Single line comments

Use "//", for a single line comment. (The same convention used in Java and C++). For example:

```
document.write("This is very good");    // writes out my opinion
```

ii) Multiple line comments

Use "**/***" to denote the beginning of a comment and "***/**" to denote end of a comment. For example:

```
/*  the following script will greet Mary and John
    and it will write it on two separate lines  */
```

Exercise 1 cont.

6. Use **ex1-2.js** and modify the line to look like:

```
document.write("Hello and welcome to JavaScript!"); // a simple greeting
```

7. Use **ex1-3.js** and add the following comment above the code

```
/* Alert windows are very handy window to use when alerting
to an error.  In this exercise we are using it to message
a greeting.  */
```

5. Variables - Storing Data in JavaScript

Variables are used to store data. You can then at a later stage recall this data. Another scenario may be that the user is to enter data, so you don't know what it is in advance. Again variables are used to store the data which can then be tested by the program at a later stage.

A **variables** is a named storage location in a program that can contain a value. A variable needs to be **declared** (or set up) in the program using the "***var***" keyword. For example

i)
```
var number;
number = 15;
```

ii)
```
var username;
userName = "Mary";
```

or you can create a variable and assign a value to it in one line:

```
var number = 15;
```

```
var userName = "Mary";
```

The above has declared **number** as a variable and has placed a value of 15 inside this storage.
The above has also declared **userName** as a variable and placed "Mary" inside this storage. Note that text or string data must have quotes, single (**'**) or double (**"**) surrounding it.

You can declare more than one variable on a single line, but you cannot then assign values on that line. For example

```
var num1, num2;
num1 = 20;  //an integer or whole number
num2 =  3.205;  //a floating point number or decimal or fraction
```

There are rules to be followed when **naming a variable**:

- names can include letters, digits and underscore "_"
- **no spaces** or any other punctuation characters
- first character must begin with a letter or underscore
- **names are case sensitive,** so userName is a different variable to username
- no official limit on length of name, but must fit within one line

Use user-friendly names, for example, totalProfit or total_profit. Don't use reserved words as names, for example, "window" or "document" or "function", etc.

Exercise 2

1. Open **ex1-5.html** and save as **ex2-1.html**. In **ex2-1.html**, change the src attribute **to src="ex2-1.js"**

   Now type in a new notepad file:

```
var firstName = "John";
document.write(firstName);
```

   and save it as **ex2-1.js**

6. User Input

When you may wish the user to enter data, you use the ***prompt()*** method.

Exercise 2 continued

2. Open **ex2-1.html** and save as **ex2-2.html**. In **ex2-2.html**, change the src attribute **to src="ex2-2.js"**

   Now type in a new notepad file:

```
var firstName = window.prompt("Please enter your first name");
document.write(firstName);
```

   and save it as **ex2-2.js**

   don't forget you can omit the 'window' object, so your code could look like:

```
var firstName = prompt("Please enter your first name");
document.write(firstName);
```

By now you should have worked out the system of naming files in this subject, so please continue to save with this name methodology.

7. Operators

You can apply operators to numbers. For example:

```
var total = 15 + 1;    // total will store 16
```

You can also add variables together. For example

```
var num1, num2, total;
num1 = 5;
num2 = 7.6;
total = num1 + num2;    //total will store 12.6
```

Some JavaScript operators:

Symbol	Explanation
	Mathematical:
+	addition
-	subtraction
*	multiplication
/	division
%	modulus operator (remainder division)
++	increment by 1
--	decrement by 1
	Logical:
!	not
&&	and
\| \| | or |

Symbol	Explanation
	Relational:
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
= =	equality
!=	inequality

Note 1
The increment (++) operator and the decrement (--) operator can be used as prefix or postfix operators. For example

var y=4
x = ++y the value in y is incremented and then assigned to x, so x=5

var y=4
x = y ++ x is assigned the original value in y and then y is incremented, so x=4

We will be using ++ as follows:
var x=4
x++ the result of this is that x=5 now

Note 2
A very common procedure in programming that we will be using goes as follows:

var cost = 50
cost = cost + 1 this statement says
new cost = current cost + 1 so cost=51 now

there is a short cut for this expression
cost += 1

So an expression like age = age +10 could be written as age +=10

8. Strings

A string is a series of characters composed of text or numbers or symbols or a combination of all. You use quotes around a string. For example:

```
var userName = "Mary Smith";
var address = '50 Happy Street';
```

You can add or **concatenate** strings together. For example:

```
var firstName = "Tom";
var surname = "Cruise";
var userName = firstName + surname;    // userName will store Tom Cruise
```

You can **concatenate different objects** together.

```
document.write("The user name is: " + userName + "<br>");
```

Here you are concatenating a literal string – "The username is:"
a variable – userName
and a tag -

Exercise 3

1. Create a web page that will ask the user for their first name then their surname, and then it will display an alert window with the full name.

   The **ex3-1.js** file will hold the following code:

```
var firstName = prompt("Please enter your first name");
var lastName = prompt("Please enter your surname");
var fullName = firstName + lastName;
alert("Your name is " + fullName);
```

2. Create a web page that will ask the user to enter two numbers, one at a time. It will then multiply the numbers and display "the product is " as well as the answer after multiplication.

Sometimes you may want to display quotes or brackets or parentheses, but you mustn't confuse the browser with the quotes and brackets/parentheses that are part of the syntax. You need to use the backslash (\) before your character. Try the following code.

3. Create a web page that displays the following message: He shouted "Happy New Year" to every one

   The **ex3-3.js** file will hold the following code:

```
document.write("He shouted \"Happy New Year\" to every one");
```

SELECTION IN PROGRAMMING

9. The 'if' Statement

The ***if*** statement allows you to examine data and take actions based on its value. The structure is as follows:

```
if(condition)
{
      code block;
}
```

If the condition is true then the code in the code block that follows will be executed, otherwise it will be skkipped.
It is standard practice to indent the code that is inside the curly braces.

Exercise 4

1. In this example we will allow the user to enter a number via the "prompt" method and this number will be stored in a variable called "number". We will use a series of "if" statements to test the value in the number variable.

   The **ex4-1.js** file will hold the following code (use copy and paste where possible):

```
var number=prompt("Enter a number between 10 and 14");

if(number<10 || number>14)
{
  alert("You didn't enter a number between 10 and 14");
}

if(number == 10)
{
  alert("the number is 10");
}

if(number == 11)
{
  alert("the number is 11");
}

if(number == 12)
{
  alert("the number is 12");
}

if(number == 13)
{
  alert("the number is 13");
}

if(number == 14)
{
  alert("the number is 14");
}
```

2. Create a web page that will ask the user for their age. If age entered is >=18 then display message "You can vote", other wise display the message "You cannot vote".

3. In this exercise a value will be assigned to a variable called 'mark'. If mark is:
   0 - 49 the document will write that it is a Fail
   50 - 69 the document will write that it is a C Pass
   70 - 82 the document will write that it is a B Pass
   83 - 100 the document will write that it is an A Pass
   Note if mark is <0 or >100 then an alert window should indicate that it is incorrect.

10 The 'else' Statement

It is customary to use the ***else*** statement so that its code block is executed if the condition is false.
The structure is as follows:

```
if(condition)
{
	code block
}
else
{
	alternate code block
}
```

Exercise 4 cont.

4. We will change the code of exercise 4 no.2 so that it incorporates the *else* statement.

   The **ex4-4.js** file will hold the following code:

```
var age=prompt("Please enter your age");

if(age>=18)
{
	alert("You can vote");
}

else
{
	alert("You cannot vote");
}
```

11. The 'switch' Statement (case)

The JavaScript ***switch*** statement works much like the *if* statement. It is useful when multiple choice is required. The structure is as follows:

```
switch(variable name)
{
	case value1:
		statements;
		break;

	case value2:
		statements;
		break;

	case value3:
		statements;
		break;

	default:
		statements;
		break;
}
```

Note - if no case matches the value then the ***default*** statement is executed.

Note - if the ***break*** statement is omitted then execution will continue to the next case value statement, which is not the intention.

Exercise 4 cont.

5. The following example is the ***switch*** version of Exercise 4 No. 1

   The **ex4-5.js** file will hold the following code (use copy and paste where possible):

```
var number;
number=window.prompt("Please enter a number between 10 and 14");

switch(number)
{
      case "10":
            document.write("The number is 10");
            break;
      case "11":
            document.write("The number is 11");
            break;
      case "12":
            document.write("The number is 12");
            break;
      case "13":
            document.write("The number is 13");
            break;
      case "14":
            document.write("The number is 14");
            break;
      default:
            alert("The number is not in the range 10 to 14");
            break;
}
```

6. Create a Web page similar to the one above but instead of the "number" variable use a variable called "firstName" and allow the user to enter a name which will be assigned to this variable. Then test to see if it is Jane, Sue, Elizabeth or Mary. Use the alert window if it is not one of those names. **Use the switch statement**.

7. Create a Web page similar to the one above but instead of the "firstName" variable use a variable called "year". Ask the user to enter a year between 1990 and 1993 and assign this value to the variable "year". It will then be checked if it is a leap year. The document will then display a message informing that it is a leap year or not. (Note- only 1992 was a leap year in this range)

8. Create a web page that allows the user to enter a basic colour, and then the web page is converted to that background colour. Use the following code to add background colour to a page.

```
document.bgColor = "red"
```

Note – you will need to put the <script> tags inside the body area for this to work. The reason is that the browser has to load the body and therefore the body has to exist before JavaScript can refer to the 'document' object. The document object represents all of the HTML document including the head and body areas.

The syntax now is:	**object.property = a value**
for example	**document.bgColor = "red"**

REPETITION IN PROGRAMMING

We use **loops** to repeat the same task, or, use loops to automate repetitive processes. We will be looking at the ***'for'*** and ***'while'*** loops.

12. The 'for' Statement

for is a loop statement and its structure looks like:

```
for(initialisation;  test;  increment)
{
   statement
}
```

where:

initialisation	usually starts a loop index	
test	is a conditional statement involving the loop index which permits the loop to continue while true	
increment	a statement executed after each loop (usually used to increment or decrement the loop index)	
statement	can be	i) one statement or ii) compound statements which is a code block surrounded by { } and is made up of many lines of code

Exercise 5

1. The following code in **ex5-1.js** file will display all numbers from 1 to 10, in a column, using the ***for*** loop statement.

```
var i;

for (i=1;  i<=10;  i++)       // i is the loop index
{
      document.write("The number is  " + i + "<br>");
}
```

2. Start a new Web page that will display all odd numbers from 1 to 15. Create script so that you use the ***for*** statement.

3. This Web page will display the squares of numbers from 1 to 5 using the ***for*** loop.

13. Using 'break'

There are times when a user wishes to end a repetitive task due to real life interruptions. You can use a ***break*** statement to jump out of a loop. The following example tests for a 'mark' value of 999, which signifies the last mark (mark values should be between 0 and 100, so a value of 999 is a trigger to stop)

```
for (var i = 1;  i <1000000;  i++)
{
   mark=prompt("Please enter a mark.  Enter 999 to stop.");
   if (mark == 999) {
         break;
   }
         document.write("mark is " + mark);
}
```

In JavaScript 1.2 and onwards, you can use **labeled breaks**, which lets you label the loop you want to break. The following example has two loops, one nested inside the other. If you label the outer loop and break the inner loop, then both loops will end. The following code will enter marks for 1000 students. Each of the students had done 5 tests, so there are 5 marks to enter for each student. If a user wishes to stop at any time they can enter a mark of 999 which will end the program (exit both loops).

```
labelX:
  for(student=1; student<1000; student++)
  {
        for(test=1; test<=5; test++)
        {
              mark=prompt("Please enter a mark.  Enter 999 to stop.");
              if (mark == 999)
              {
                    break labelX;
              }
        }
  }
```

Exercise 5 cont.

4. Create a Web page that will prompt the user to enter 10 marks, and write the mark to the page. It will break out of the loop if the mark has a value of 999, which signifies the end. Use the following code in file **ex5-4.js**:

```
var i, mark;

for (i=1;  i<=10;  i++)
{
      mark = window.prompt("enter mark.  Enter 999 if you wish to stop");
      if(mark==999)
      {
            break;
      }

      document.write("The mark is  " + mark +"<br>");
}
document.write("We have come to the last mark");
```

14. The 'while' Statement

The ***'while'*** is a loop statement and its structure looks like:

```
while (expression)
{
      statement;
}
```

The loop keeps executing its statement while the expression is true. Unlike the *for* loop, the *while* loop does not require a loop index and you can use any expression that evaluates to a true or false.

Exercise 5 cont.

5. This exercise is the same as exercise 5 no. 1. It will display all numbers from 1 to 10,.in a column, but using the ***while*** loop (not the for loop). The **ex5-5.js** file will contain the following code:

```
var i=1;

while(i<11)
{
      document.write("The number is  " + i + "<br>");
      i++;
}
```

6. Start a new Web page that will display all odd numbers from 1 to 15. Create script so that you use the ***while*** statement.

7. This Web page will display the squares of numbers from 1 to 5 using the ***while*** loop.

8. Create a Web page that will prompt the user to enter marks, and write the mark to the page. The loop will stop when a mark of 999 is entered. Use the ***while*** statement

15. The do while' Statement

The ***do while*** statement is similar to the *while*, except that it tests the expression at the end of the loop. It has the following structure;

```
do
{
   statement;
}while (expression)
```

This ensures that the loop's statement is executed at least once.

Exercise 5 cont.

9. We will change the code of exercise 5 no. 8 so that it incorporates the ***do while*** statement. The **ex5-9.js** file will contain the following code:

```
var mark;
mark = window.prompt("enter a mark");

do
{
   document.write("The mark is  " + mark +"<br>");
   mark = window.prompt("enter a mark");
}while(mark!=999)

document.write("We have come to the last mark");
```

16. JavaScript Functions

Quite often you will want different parts of your program to be executed at different times, usually as a response due to user interaction. To do this you need to put your code into ***functions***. Its purpose is to perform one or more tasks. Also a function can be used more than once, and so saves code space by not being repeatedly typed. A function will sit stationary, it's code will not be executed until it is called by its name.

The syntax of a function is as follows:

```
function  functionName (parameter1,  parameter2,……)
{
statements
}
```

where ***parameters or arguments*** are variables that the function works on. Parameters are optional.

17. Calling a Function

The code in a function is not executed until the function is called by its name followed by brackets. This usually happens at a later time, however, at this stage we will call the function straight away.

Exercise 6

1. You will create a function that will send a greeting via an alert window. The **ex6-1.js** file will contain the following code:

```
function fnGreeting()
{
   window.alert("Welcome!");
}

/* now we will call the function
and the code inside the function block will be executed*/

fnGreeting();
```

2. You will modify the above exercise by using arguments / parameters in the function. The **ex6-2.js** file will contain
the following code:

```
function fnGreeting(aName)
{
   window.alert("Welcome! " + aName);
}

// call the function and pass the value "Mary" to the function
fnGreeting("Mary");

// call the function and pass the value "James" to the function
fnGreeting("James");
```

3. Modify the code above by using a prompt window to allow the user to enter their name, and then that name is passed to the function.

18. Returning a Value

A function can also return a value (silently) to the script that called it, by using the ***return*** key word.

Exercise 6 cont.

4. In this exercise we will create a function that returns a value 12. The **ex6-4.js** file will contain the following code:

```
function fnNumber()
{
   return 12;
}

// call the function inside the write() statement

document.write("My favourite number is " + fnNumber() );
```

5. Create a Web page that uses a function to return a line of text.

6. In this exercise we will pass two values to a function and the function will return their sum. The **ex6-6.js** file will contain the following code:

```
function fnAdd(a,b)
{
   return (a+b);
}

// call the function inside the write() method
document.write("The sum is " + fnAdd(4,5) );
```

7. Create a Web page that uses a function. You can send four values to the function and the function will return the average.

19. Inbuilt Functions

The functions you have worked with above are functions that you have created. Now we will look at inbuilt functions, also called "top-level functions" that come with predefined jobs. Here are some inbuilt functions

Function	Description	Example
eval	Evaluates a string.	X = eval("2+10") X becomes 12 X = eval(Number("14 Happy Street")) X becomes 14
isFinite	Evaluates an argument and returns 'true' if finite, otherwise 'false'	X = isFinite(147) X stores 'true' X = isFinite("Hello") X stores 'false'
isNaN	Evaluates an argument and returns 'true' if it **is** **N**ot **a** **N**umber	X = isNaN("Hello") X stores 'true' X = isNaN(45) X stores 'false'
Number	Converts an object to a number	X = Number("7") X becomes 7
parseFloat	Extracts a floating point number from a string	X = parseFloat("62.5 kg") X becomes 62.5
parseInt	Extracts an integer from a string	X = parseInt("14 Happy Street") X becomes 14
String	Converts an object to a string	X = String(25) X becomes "25"

Exercise 6 cont.

8. Create a Web site that will ask the user to enter "something" through the prompt window. It will then test this "something" and produce a message that will state if it was a number or not. (Hint – use ***isNaN*** function)

9. Create a Web page that will ask the user to enter two numbers via the prompt window. It will then proceed to add them and display the sum.

10. Add to the above problem, that is, add more code below the existing code of Q9, so that the two numbers entered by the user are displayed as a string(ie. one next to the other).

11. Create a Web page that asks the user to enter a date as follows (for example) – 30th May, 2003, where the day number is first. Then use the ***parseInt*** function to extract the day number from the date.