Part 1

JavaScript

History

JavaScript was designed to 'plug a gap' in the techniques available for creating web-pages.

HTML is relatively easy to learn, but it is static. It allows the use of links to load new pages, images, sounds, etc., but it provides very little support for any other type of interactivity.

To create dynamic material it was necessary to use either:

- CGI (Common Gateway Interface) programs
    - Can be used to provide a wide range of interactive features, but...
    - Run on the server, i.e.:
        - A user-action causes a request to be sent over the internet from the client machine to the server.
        - The server runs a CGI program that generates a new page, based on the information supplied by the client.
        - The new page is sent back to the client machine and is loaded in place of the previous page.

      Thus every change requires communication back and forth across the internet.
    - Written in languages such as Perl, which are relatively difficult to learn.
- Java applets
    - Run on the client, so there is no need to send information back and forth over the internet for every change, but...
    - Written in Java, which is relatively difficult to learn.

Netscape Corporation set out to develop a language that:

- Provides dynamic facilities similar to those available using CGI programs and Java applets.
- Runs on the Client.

- Is relatively easy to learn and use.

They came up with ***LiveScript***.

Netscape subsequently teamed-up with Sun Microsystems (the company that developed Java) and produced ***JavaScript***.

Javascript only runs on Netscape browsers (e.g., Netscape Navigator). However, Microsoft soon developed a version of JavaScript for their Internet Explorer browser. It is called ***JScript***. The two languages are almost identical, although there are some minor differences.

Internet browsers such as Internet Explorer and Netscape Navigator provide a range of features that can be controlled using a suitable program. For example, windows can be opened and closed, items can be moved around the page, colours can be changed, information can be read or modified, etc..

However, in order to do this you need to know what items the browser contains, what operations can be carried out on each item, and the format of the necessary commands.

Therefore, in order to program internet browsers, you need to know:

- How to program in a suitable language (e.g., Javascript/JScript)
- The internal structure of the browser.

In this course we will be using JavaScript/JScript to program browsers. However, there are several other languages we could use should we wish to. Therefore, we shall try to distinguish clearly between those aspects of internet programming which are specific to JavaScript/JScript and those which remain the same regardles of which language we choose to use.

We'll start by looking at some of the basic features of the JavaScript language.

Variables & Literals

A variable is a *container* which has a name. We use variables to hold information that may change from one moment to the next while a program is running.

For example, a shopping website might use a variable called *total* to hold the total cost of the goods the customer has selected. The amount stored in this variable may change as the customer adds more goods or discards earlier choices, but the name *total* stays the same. Therefore we can find out the current total cost at any time by asking the program to tell us what is currently stored in *total*.

A literal, by contrast, doesn't have a name - it only has a value.

For example, we might use a literal to store the VAT rate, since this doesn't change very often. The literal would have a value of (e.g.) 0.21. We could then obtain the final cost to the customer in the following way:

VAT is equal to *total* x 0.21

final total is equal to *total* + *VAT*

JavaScript accepts the following types of variables:

Numeric	Any numeric value, whether a whole number (an *integer*) or a number that includes a fractional part (a *real*), e.g., 12 3.14159 etc.
String	A group of text characters, e.g., Ian Macintosh G4 etc.
Boolean	A value which can only be either True or False, e.g. completed married etc.

We create variables and assign values to them in the following way:

```
var christianName = "Fred"                    (string)
var surname = "Jones"                         (string)
var age = 37                                  (numeric)
var married = false                           (Boolean)
```

Note that:

- When a new variable is created (or *declared*) its name must be preceded by the word `var`
- The type of the variable is determined by the way it is declared:
  - if it is enclosed within quotes, it's a string
  - if it is set to true or false (without quotes) it's a boolean
  - if it is a number (without quotes) it's numeric
- We refer to the equals sign as the *assignment operator* because we use it to assign values to variables;
- Variable names must begin with a letter or an underscore
- Variable names must not include spaces
- JavaScript is case-sensitive
- Reserved words (i.e., words which indicate an action or operation in JavaScript) cannot be used as variable names.

Operators

Operators are a type of command. They perform operations on variables and/or literals and produce a result.

JavaScript understands the following operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

(If you're not sure what a modulus operator does, here are some notes and an example)

These are known as *binary* operators because they require *two* values as input, i.e.:

4 + 3

7 / 2

15 % 4

In addition, JavaScript understands the following operators:

+ +	Increment	Increase value by 1
- -	Decrement	Decrease value by 1
-	Negation	Convert positive to negative, or vice versa

These are known as *unary* operators because they require only *one* value as input, i.e.:

4++	increase 4 by 1 so it becomes 5
7--	decrease 7 by 1 so it becomes 6
-5	negate 5 so it becomes -5

JavaScript operators are used in the following way:

```
var totalStudents = 60
var examPasses = 56
var resits = totalStudents - examPasses
```

Note that by carrying out this operation we have created a new variable - `resits.`

There is no need to declare this variable in advance.

It will be a numeric value because it has been created as a result of an operation performed on two numeric values.

We can also combine these operators (and the assignment operator, =) in certain ways.

For example:

```
total += price
```

performs the same task as:

```
total = total + price
```

Similarly,

```
total *= quantity
```

performs the same task as:

```
total = total * quantity
```

Below is a full list of these 'combined' operators.

+ =	'becomes equal to itself plus'
- =	'becomes equal to itself minus'
* =	'becomes equal to itself multiplied by'
/ =	'becomes equal to itself divided by'
% =	'becomes equal to the amount which is left when it is divided by'

You may find the descriptions helpful when trying to remember what each operator does. For example:

5 * = 3

can be thought of as meaning:

5 becomes equal to itself multiplied by 3