Part 2

Functions in JavaScript

In JavaScript, as in other languages, we can create *functions*. A function is a kind of mini-program that forms part of a larger program.

Functions:

- consist of one or more *statements* (i.e., lines of program code that perform some operation)
- are separated in some way from the rest of the program, for example, by being enclosed in curly brackets, {.....}
- are given a unique name, so that they can be *called* from elsewhere in the program.

Functions are used:

- Where the same operation has to be performed many times within a program.

  Rather than writing the same code again and again, it can be written once as a function and used repeatedly. For example:

  ```
  request_confirmation_from_user
  ```

- To make it easier for someone else to understand your program.

  Rather than writing long, rambling programs in which every single operation is listed in turn, it is usually better to divide programs up into small groups of related operations. For example:

  ```
  set_variables_to_initial_values

  welcome_user

  obtain_user_input

  perform_calculations

  display_results
  ```

In JavaScript, functions are created in the following way:

```
function name()
{
        statement;
```

```
        statement;
        statement
}
```

Note that all the statements except the last statement must be followed by semi-colons. The last statement doesn't need one, but if you do put a semi-colon after the last statement it won't cause any problems.

Here is an example of a simple function:

```
function initialiseVariables()
{
        itemsSold = 0;
        nettPrice = 0;
        priceAfterTax = 0
}
```

When called, this function will set the three variables `itemsSold, nettPrice` and `priceAfterTax` to zero.

To run this function from somewhere else in a program, we would simply call it by name, e.g.:

```
initialiseVariables();
```

Note that the name must be followed by a pair of brackets. The purpose of these will become clear later.

Functions can be called from within other functions.

For example:

```
function sayGoodbye()
{
        alert("Goodbye!")
}

function sayHello()
{
        alert("Hi, there!");
        sayGoodbye()
}
```

When the function `sayHello()` is called, it first displays an *alert* on the screen. An alert is simply box containing some text and an 'OK' button that the user can press to make the box disappear when the text has been read. In this case the box will contain the words "Hi, there!".

The `sayHello()` function then calls the function `sayGoodbye()`, which posts another alert saying

"Goodbye".

Say Hello Click here to see this example working.

Note that the function `sayGoodbye()` is written first. Browsers interpet JavaScript code line-by-line, starting at the top, and some browsers will report an error if they find a reference to a function before they find the function itself. Therefore functions should be declared before the point in the program where they are used.

Passing Parameters to Functions

Some functions perform a simple task for which no extra information is needed.

However, it is often necessary to supply information to a function so that it can carry out its task.

For example, if we want to create a function which adds VAT to a price, we would have to tell the function what the price is.

To do this we would pass the price into the function as a *parameter*. Parameters are listed in between the brackets that follow the function name. For example:

```
function name(parameter_1, parameter_2)
{
        statement(s);
}
```

In this case two parameters are used, but it's possible to use more than this if necessary. The additional parameter names would simply be added on to the list of parameters inside the brackets, separated from one another by commas. It's also possible to use just one prameter if that's all that is needed.

Here's an example of a simple function that accepts a single parameter:

```
function addVAT(price)
{
        price *= 1.21;
        alert(price)
}
```

This function accepts a parameter called `price`, multiplies it by 1.21 (i.e., adds an extra 21% to it), and then displays the new value in an alert box.

We would call this function in the following way:

```
addVAT(nettPrice)
```

The parameter `nettPrice` could be either:

- a literal - for example:

```
addVAT(5)
```

- a variable - for example:

```
var nettPrice = 5;
addVAT(nettPrice)
```

Returning values from Functions

Sometimes we also need to get some information back from a function.

For example, we might want to add VAT to a price and then, instead of just displaying the result, pass it back to the user or display it in a table.

To get information back from a function we do the following:

```
function addVAT(price)
{
        price *= 1.21;
        return price
}
```

To call this function we would do the following:

```
var newPrice = addVAT(nettPrice)
```

The value returned by the funtion will be stored in the variable `newPrice`. Therefore this funtion will have the effect of making `newPrice` equal to `nettPrice` multiplied by 1.21.