Javascript Basics - Module 2

In the previous module we started to take a more in-depth look at JavaScript and in this module we'll cover some more JavaScript topics including some basic event handling and interaction with web pages.

We'll also cover some additional coding structures and look at how we can use arrays to store multiple items of related data.

As usual, if you have any questions then fire away!

Review of Previous Module

In the previous module we started to use JavaScript, the main scripting language for the web.

We looked at the basic syntax of statements and expressions and also variables and data.

We saw how to use JavaScript to add content to a web page and the use of conditional expressions for making decisions and controlling the operation of loops.

Topics for this module

In this module we'll look at some more JavaScript concepts and techniques, including the following topics

- Functions
- Basic event handling
- While loops
- Switch structure
- Arrays
- DOM interaction

Functions Basics

A function is a block of code that is given a name and can be executed (called) multiple times in a program or script.

A very simple function is shown below:

```
<script>
  function helloWorld() {
    window.alert('Hello World!');
  }

  helloWorld();
</script>
```

The function is first defined, and then called later by using its name. It could also be called when a user clicks a button on the page.

Functions Basics

A function is a block of code that is given a name and can be executed (called) multiple times in a program or script.

A very simple function is shown below:

```
<script>
  function helloWorld() {
    window.alert('Hello World!');
  }

  helloWorld();
</script>
```

The function is first defined, and then called later by using its name. It could also be called when a user clicks a button on the page.

Functions Basics

Example using a button:

```
<script>
  function helloWorld() {
    window.alert('Hello World!');
  }
</script>

<button onclick="helloWorld();">Say Hi</button>
```

In this example the **function** is defined in a script element and then it is **called** when a **button is clicked** by assigning a snippet of JavaScript code to the **onclick attribute** of the button.

This is know as **event handling**.

Exercise 1

Create a new file HTML file named javascript-functions-1.html

Add the <script> element from the previous slide in the <head> section of your document.

Then add the following code to the <body> element

```
<h1>JavaScript Functions 1</h1>
<p>Please click the button below to say Hi</p>

<button onclick="helloWorld();">Say Hi</button>
```

Save and load the page and click the button.

Exercise

Create a new file HTML file named javascript-functions-2.html by copying javascript-functions-1.html

In your external script, add a function named helloWorld()

This function should include the statement that displays the ‘Hello World’ message.

In the new HTML file, modify the event handler so that it calls the new function.

Save and test the page – it should still display the message when the button is clicked.

Interacting With the DOM

JavaScript has access to the elements on a page via the Document Object Model.

This means that we can use JavaScript to modify elements in some way or to get some information about them.

One thing that we might want to do is to get the value from a field and use it in some way.

In order to get the value from an element we first need to identify it and then we can retrieve it’s value.

One way that we can select an element in JavaScript is with the following expression:

```
document.getElementById(id)
```

Interacting With the DOM

The expression that we looked at on the previous slide was as follows:

```
document.getElementById(id);
```

This is actually an example of Object Oriented Programming (OOP) which we'll cover in more detail on the bootcamps. For now we can just think of this as meaning:

Start with the document (the HTML document)
Then get/find the element with a given id

That will create a special data type, known as an object, that is like a copy of the element that the JavaScript code can work with.

We can then add use that to retrieve some information about the element, or to modify it in some way.

Interacting With the DOM

To get the value of a form field that has an id of ‘field1’) we could use the following statement:

```
var fieldVal = document.getElementById(‘field1’).value;
```

Reading that statement from left to right we have:

- Create a variable named fieldVal
- Assign a value to the variable using the expression
- Start with the document (the HTML document)
- Then get/find the element with a given id
- Then get its value

The end result is that the value of the field, which is whatever has been entered into it, will be stored in the fieldVal variable.

Variable Scope

In the context of programming, a variable’s scope means where and when a variable exists.

We’ll look at scope in more detail in the next module, but for now you should be aware that if you declare a variable in an inline or external script then, as long as it wasn’t declared inside a function, it’s value will remain available to any other script elements that follow in the document.

Let’s look at an example on the next slide...

Variable Scope

In the example code below, both the first and second script elements are able to access the myVar variable. You can think of all individual script elements as being part of one global script.

```
<script>
  var myVal = 10;
  document.write('myVal is ' + myVal + '<br>');
</script>

<p>Paragraph of text...</p>

<script>
  document.write('myVal is ' + myVal + '<br>');
</script>
```

DOM Events

One of the features of the Document Object Model (DOM) specification is that it allows event handling.

What this means is that languages like JavaScript can respond to events as the occur on a document (page).

Examples of events that can be used are:

- Opening a page
- Clicking a button or some part of the page
- Hovering the mouse pointer over something
- Resizing the browser window
- Entering something into a field

There are many events that can be detected with JavaScript

DOM Events

In a previous exercise we used the following code:

```
<button onclick="helloWorld();">Say Hi</button>
```

This is an example of creating an **event handler.**

An event handler does two things:

- The name of the event to responded to/**handle**
- The JavaScript code to be executed when the event occurs

The JavaScript code is enclosed in quotes and acts like a mini-script , or **snippet**, that is executed when the event occurs.

The JavaScript snippet can call **functions** that have been defined previously in **internal or external scripts**.

DOM Events

Example events

Event	When the event occurs
onclick	The element is clicked on
onchange	The user changes the value of a form element
onmouseover	The mouse pointer is hovered over the element
onmouseout	The mouse pointer is moved away from the element
onload	The document has loaded
ondblclick	The mouse is double-clicked on the element
onkeydown	The user presses down on a key on the keyboard
onfocus	The element is given focus (clicking into a field for example)
onsubmit	A form is submitted

Exercise

Create a new file HTML file named javascript-events-1.html

Add the following to the body section:

```
<form>
  <input type="text" id="username" placeholder="Enter your name...">
</form>
<button onclick="greetUser();">Click me</button>
```

Now add a function named greetUser() to your external script that will display a greeting in the form:

‘Hi Nick’ (if Nick had been entered in the field)

While Loops

In the previous module we looked at for loops, now we'll look at the other commonly used type of loop – a while loop.

In JavaScript the pseudo-code for a while loop is as follows:

```
while (conditional expression) {
  statements to be executed
}
```

Yup, it's pretty much exactly the same as the example that we looked at in the previous module.

The next slide shows an example...

While Loops

Example while loop

```
var limit = 10, i = 1;

while (i < limit) {
  document.write(‘The value of i is ‘ + i)’
}
```

What would the example code do?

While Loops

When using a while loop it is important to ensure that the conditional expression will return false at some point, otherwise the loop will never terminate.

In the example on the previous slide the value of i is never changed, so it will always be 1 and therefore the loop will continue for ever – this is often referred to as an infinite loop.

```
var limit = 10, i = 1;

while (i < limit) {
  document.write(‘The value of i is ‘ + i)’
}
```

Let’s fix that...

While Loops

Now there is a statement inside the loop code block that increases the value of i by 1 each time the loop is executed.

Therefore the loop will terminate one the value of i is no longer less than the value in the limit variable, which in this case is 10

```
var limit = 10, i = 1;

while (i < limit) {
  document.write('The value of i is ' + i)'
  i++;
}
```

Switch structure

In the previous module we looked at one of the most common conditional structures – the if structure.

Another conditional structure is the switch structure, the pseudo code for which looks like this:

```
switch (expression) {
  case value1:
    statement block
    break;
  case value2:
    statement block
    break;
  default:
    statement block
}
```

Switch structure

A switch structure compares the value in an expression, usually a variable, with one or more possible case values.

If a match is found any code following the case value is executed.

A break statement is used to terminate the switch..

```
switch (expression) {
  case value1:
    statement block
    break;
  case value2:
    statement block
    break;
  default:
    statement block
}
```

Switch structure

Example switch

```
var parcelSize = ‘medium’;
var deliveryCharge;

switch (parcelSize) {
  case ‘small’:
    deliveryCharge = 5.00;
    break;
  case ‘medium’:
    deliveryCharge = 7.50;
    break;
  default:
    deliveryCharge = 10.00;
}

// deliveryCharge is 7.50
```

Match found here

break statement terminates execution of switch

Switch structure

Once a match is found, ALL remaining statements in the switch will be executed until either a break statement is executed or the end of the switch is reached

```
var parcelSize = ‘small’;
var deliveryCharge;

switch (parcelSize) {
  case ‘small’:
    deliveryCharge = 5.00;
  case ‘medium’:
    deliveryCharge = 7.50;
  default:
    deliveryCharge = 10.00;
}

// deliveryCharge is 10.00
```

Match found here but no break statement so code continues to execute.

Arrays

An array is a special kind of variable that can be used to store multiple items. This can be useful when you want to store a number of values that are related in some way – perhaps the names of employees in your organization.

If you used individual variables you would have to do something like this:

```
var employee1 = ‘Mike’,
employee2 = ‘Mary’,
employee3 = ‘Joe’,
employee4 = ‘Sue’,
etc
```

Now suppose we want to write a loop that retrieves the name of each person in turn. That would not be impossible.

Arrays

By storing the employee names in an array we create a structured container that we can loop over easily.

First let’s look at how we create an array:

```
var myEmployees = new Array("Mike","Mary","Joe", "Sue");
```

We now have an array with 4 elements. Each element has an index and a value. The index of the first element is 0, the next index is 1, then 2 and so on.

To find the value for any element we use the name of the array followed by the element’s index in square brackets:

```
document.write(myEmployees[1]); // ‘Mary’
```

Arrays

To loop over an array we can use the following syntax:

```
var myEmployees = new Array("Mike","Mary","Joe", "Sue");

for (var id in myEmployees) {

  document.write(myEmployees[id] + ‘<br>’;);

}
```

This is a special kind of for loop that will automatically loop over every element in the array, irrespective of how many there are.

On every iteration of the loop the value will be retrieved from the next element in the array and assigned to the variable.

Exercise

Create a new file HTML file named javascript-arrays-1.html

Make sure that your external javascript file is referenced in the <head> section .

Immediately after that add a <script> element in the head section with a statement that creates an array of between 3 and 5 place names.

In your external script create a function named createEmployeeList()

The function should loop through the array of places and output them in the form of an ordered list.

Finally, add another script element in the <body> of you document with a statement that will call the function.

Summary and Q&A

In this module we continued looking at JavaScript, and in particular some basic event handling.

We also covered the switch conditional structure and the while loop.

Finally we looked at the use of arrays for storing multiple related items in a structured way.

Labs

Your instructor will upload the labs for this module to Basecamp after the session ends.

Questions

Do you have any questions before we move on to the next module

?

Don’t worry, if you think of something later then just ask!