NCC education

Awarding Great British Qualifications

Dynamic Websites

Topic 6:

Using Scripts (1)

Scope and Coverage

This topic will cover:

- An overview of Ajax and JavaScript
- Using JavaScript to create simple client side applications
- Using Ajax to implement simple server communication

Learning Outcomes

By the end of this topic students will be able to:

- Understand JavaScript and Ajax and how they work in web pages.
- Create interactive elements for web pages
- Create loops, arrays, arithmetic operations and strings.

Introduction

- In this lecture, we are going to introduce dynamic client-side scripting to improve our user interfaces.
  - We are going to do this through the software architecture known as AJAX.
- Ajax (Asynchronous JavaScript and XML) is a set of related tools that improve the responsiveness of presentation layers.

Ajax- 1

- Although XML is part of the Ajax title, it is not strictly required.
  - However, we will be using in the short term.
- AJAX is a ***grouping*** of technologies.
  - It is not programming language in itself.
- AJAX makes use of HTML and CSS for presentation.
  - It uses the ***Document Object Model*** to adapt and manipulate data.

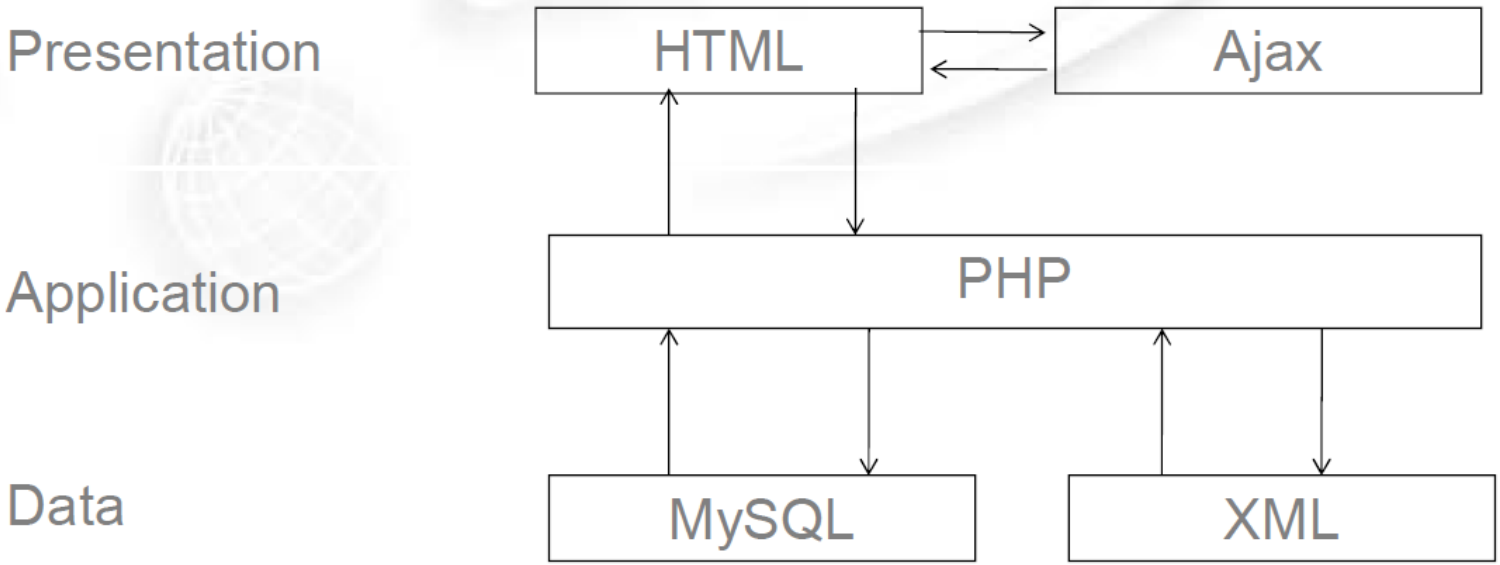
Ajax - 2

- Ajax makes use of a special kind of object called the XMLHttpRequest object.
  - This allows Ajax to communicate with the server and request HTML/PHP pages.
- Ajax does not allow direct access to the application layer.
  - It is contained within an HTML page, and so must communicate via that.
- This impacts on our dynamic web pages architecture.

Ajax - 3

- The Ajax processing loop is as follows:
  - The browser creates an XMLHttpRequest object and then sends that to the server.
  - The server processes the object and then sends back the response.
  - The browser processes the response using JavaScript.
  - The browser then updates the page content to reflect changes.
- The results is a very responsive web page.

Our Architecture So Far



JavaScript - 1

- The various technologies of Ajax are held together by a scripting language called JavaScript.
  - This has very little to do with Java, the programming language.
  - JavaScript allows us to locate code in a web page so that it can be executed by the browser.
  - This offloads some of the bandwidth and processing costs from the server to the client.
  - We can use this to create more responsive user interfaces.

JavaScript - 2

- Before we begin to unravel Ajax, we must first learn the syntax of JavaScript.
  - This is relatively straightforward.
- As with our PHP, we can incorporate JavaScript directly into a HTML page.
  - Using the <script> tag:

```
<script type="text/javascript">
    document.write("<p>Hello world!  It's " + Date()
+
          ".</p>");
</script>
```

- JavaScript's can be placed in external files

The Document Object Model - 1

- Core to JavaScript is the concept of the Document Object Model (DOM).
- The DOM is a way of representing XML documents as a tree,
  - And by extension, HTML/XHTML documents.
- There are several ways of parsing XML documents into workable structures.
  - The one we used for our PHP RSS parser is called the SAX method; DOM is another.

The Document Object Model - 2

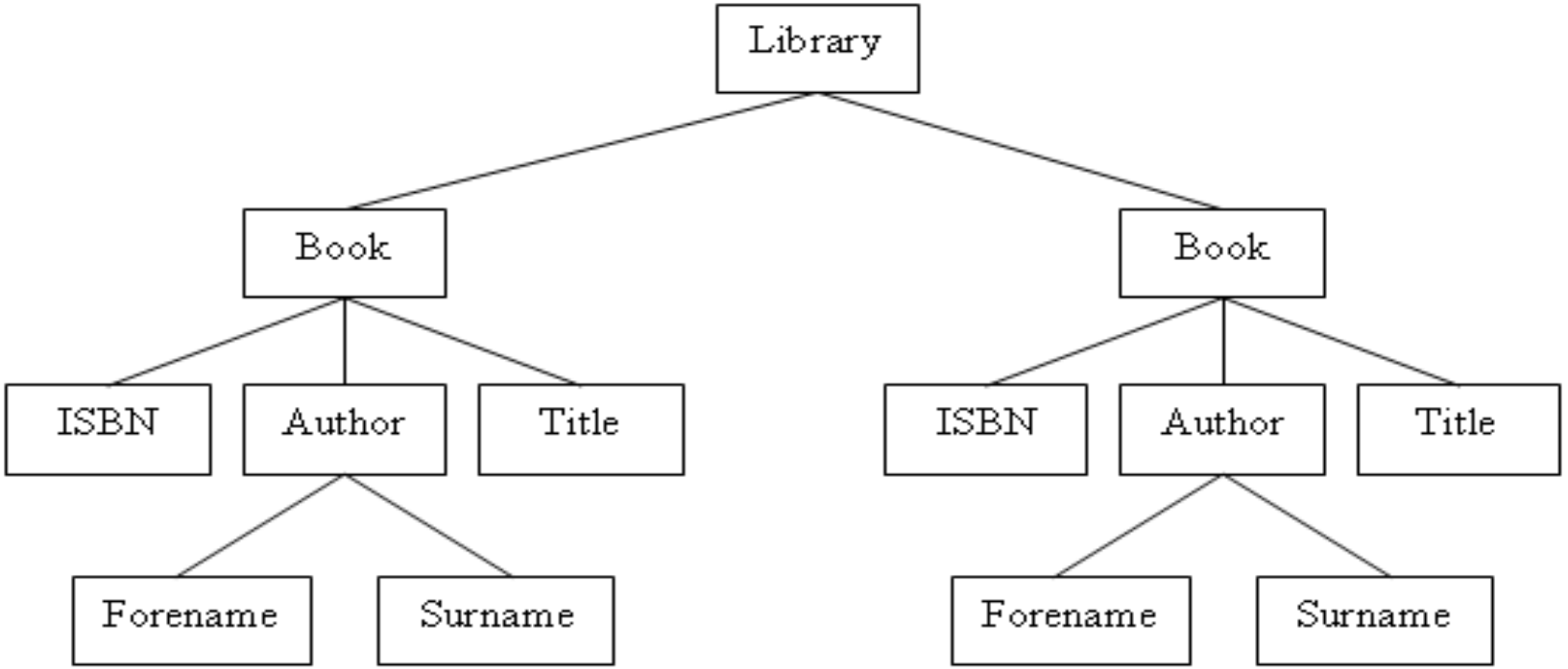
- The DOM creates a structure that is easy for developers to navigate.
  - Hints as to the structure of a document are provided.
- Any time that tags are nested, they become ***nodes*** in the tree,
  - Nodes in turn may contain nodes of their own.
- We can then explore branches and iterate over nodes as needed.

The DOM - 1

```
<?xml version='1.0' encoding='utf-8'?>
        <library>
                <book status = "on shelf">
                        <ISBN>012345678</ISBN>
                        <title>Component Based Solutions Module
Book</title>
                        <author>
                                <forename>Michael</forename>
                                <surname>Heron</surname>
                        </author>
                </book>
                <book status = "on loan">
                        <ISBN>2222222</ISBN>
                        <title>The Javanomicon</title>
                        <author>
                                <forename>Michael</forename>
                                <surname>Heron</surname>
                        </author>
                </book>
</library>
```

Source: Used with permission from http://www.monkeys-at-keyboards.com/java3/8.shtml

The DOM - 2



Source: Used with permission from http://www.monkeys-at-keyboards.com/java3/8.shtml

JavaScript and the DOM

- Every HTML page in JavaScript is represented as a document object.
  - This is available for us for free, we do not need to do anything to create or access it.
- We can make use of a number of useful methods.
  - getElementByld is the most useful of these to begin with.
  - It gives us access to the element matching the ID we provide.

Manipulating HTML

- Much of the power of JavaScript comes from allowing us to modify HTML elements once they have been loaded into the browser.

```
<p id = "test"> </p>

<script type="text/javascript">
 document.getElementById("test").innerHTML=
    "<p>Hello world!</p>";
</script>
```

JavaScript

- Inner HTML represents the textual content of an element.
  - This is not part of the DOM specification, but it is almost universally supported by browsers.
- We can use this to dynamically adjust our web page content as required as seen on the previous page.
- We can also bind JavaScript functions to be invoked when we interact with HTML elements, such as buttons.
  - This is where much of the real power lies.

JavaScript Functions - 1

```
<html>
  <body>
    <h1>My First Javascript</h1>

    <script language="JavaScript">

      function displayText(form) {
        var text = form.myText.value;
        alert ("You typed " + text);
      }

    </script>

    <form name = "testForm">
      <p>Enter some text</p>
      <input type = "text" name = "myText">

      <input type="button" value="Press Me" name="button"
          onClick="displayText(this.form)">
    </form>
  </body>
</html>
```

JavaScript Functions - 2

```
<script>
function myFunction() {
    var txt;
    var person = prompt("Please enter your name:", “Mickey Mouse");
    if (person == null || person == "") {
        txt = “Prompt has been cancelled.";
    } else {
        txt = “Hi " + person + "! How can we help you today?";
    }
    document.getElementById("demo").innerHTML = txt;
}
</script>
```

Ajax

- Our first steps in Ajax require us to navigate an awkward element of dynamic websites.
  - Browser compatibility.
- Every browser handle the initial parts of setting up an XMLHttpRequest object differently.
  - And we need to honour each of these methods.
- This makes the first part of setting up our Ajax quite verbose.
  - Mostly, everything that follows is standardised.

The Request Object

```
var request;

if (window.XMLHttpRequest) {
  // Code for modern browsers
  request=new XMLHttpRequest();
}
else {
  // code for older versions of Internet Explorer
  request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Server Communication

- Communication with the server is handled via GET/POST requests.
  - We do this through the ***open*** and ***send*** methods of the XMLHttpRequest object we created.
    - request.open(“GET”, “test.txt”, trust);
    - request.send();
- There are three parameters to open:
  - The method of the request
  - The URL of the request
  - Whether to handle the request asynchronously

Receiving the Response

- As with our XML parser, we handle this through the use of a ***callback.***
  - We do not know when the server is going to give us the response, so we provide a function to be called as and when it happens.
- To do this, we create a function and bind it to the ***onreadystatechange*** property of our XMLHttpRequestobject.
- We do this ***before*** we send a request.

The Server Response

- The server response is made up of three key parts.
  - ***The ready state***
    - This will be a number of 0 to 4.
      - » Only 4 means “it’s compete”
  - ***The request status***
    - 200 means “ok” here.
      - » This is the standard HTTP status code.
  - ***The response text***
    - This is what the server gave us back.
      - » It is also what we will usually need to parse with JavaScript.

A Simple Ajax Application - 1

- For the following, we have a small text file on the server.
- When we press the button, we replace the contents of a <p> tag with what is in the text file:

```
<button type="button" onclick="setupAjax()">
  Press me!</button>
<p id = "contents">Contents Go Here</p>
```

A Simple Ajax Application - 2

```
<script language="JavaScript">
    function setupAjax() {
      var request;

      // Our usual setup code would go here - omitted for brevity.

      // Here, we set up the code to handle the response from the server
      request.onreadystatechange=function() {
        if (request.readyState==4 && request.status==200) {
          document.getElementById("contents").innerHTML=
            request.responseText;
        }
      }

      // Here we send the request to the server to GET the file called
      // test.txt.
      request.open ("GET", "test.txt", true);
      request.send();
    }
</script>
```

Ajax Applications

- For our Ajax application, we are binding event handlers to that they occur at particular points in a document's lifetime.
  - There are many places where this can be done.
- onLoad and onUnload handle when a page is loaded or unloaded.
- onFocus, onBlur and onChange permit you handle when there is interaction with a form element.

A second Ajax Application - 1

In our second example, we are going to bind mouseover events to an Ajax function:

```
<a href = "http://en.wikipedia.org/wiki/Moon" onmouseover = "setupAjax('moon.txt')">
  <img src = "http://astro.unl.edu/misc/moonpics/moon09.jpg" />
</a>

<a href = "http://en.wikipedia.org/wiki/Sun" onmouseover = "setupAjax('sun.txt')">
  <img src = "http://cdn.wn.com/pd/94/d8/29a86975e94d1fb77ad2f1374572_grande.jpg" />
</a>

<a href = "http://en.wikipedia.org/wiki/Earth" onmouseover = "setupAjax('earth.txt')">
  <img src = "http://www.officialpsds.com/images/thumbs/Planet-Earth-psd20005.png" />
</a>


<p id = "contents">Contents Go Here</p>
```

A Second Ajax Application - 2

```
function setupAjax(url) {
      var request;

      if (window.XMLHttpRequest) {
        // Code for modern browsers
        request=new XMLHttpRequest();
      }
      else {
        // code for older versions of Internet Explorer
        request = new ActiveXObject("Microsoft.XMLHTTP");
      }

      request.onreadystatechange=function() {
        if (request.readyState==4 && request.status==200) {
          document.getElementById("contents").innerHTML= request.responseText;
        }
      }

      request.open ("GET", url, true);
      request.send();

    }
```

Variables in Array -1

- Collection of variables stored in an array (list)
- Each variable called by position in a list
- The array has name
- Items within array are numbered from 0 onwards

```
<script>
var mycars = new Array();
mycars[0] = “VW";
mycars[1] = “Jaguar";
mycars[2] = “Mini";
</script>
```

Variables in Array - 2

```
<script>
var cars =[
    “VW",
    “Jaguar",
    “Mini"
];
document.getElementById("demo").innerHTML = cars;
</script>
```

Java Script Arithmetic Operations

```
<script>
function calculation(){
var price=200;
var quantity=50;
var total= price * quantity;
document.getElementById (“count”).innerHTML = total;
}
</script>
```

Ajax and PHP

- We have still to discuss how we can make use of the results from PHP applications in our Ajax applications.
  - This will be done in another topic.
- For the current architecture we have, we use the responseText property to incorporate our results.
  - This is the raw text of a returned response.
- The responseXML property will be more useful when it comes time to integrate across layers.

Ajax Versus Server Requests

- The biggest benefit that comes from handling user input in this way is a greatly improved experience for users.
  - There is nothing in the two example programs from this lecture that we could not already do in terms of functionality.
- We no longer have to wait for a new webpage to be served in order to see the results of our interaction.
  - It can be done, in the background, creating the impression of a seamless application.

Conclusion

- Ajax gives us an effective way to provide the user interface in a dynamic application.
  - It is a framework built of several elements.
  - It is not a programming language in itself.
- JavaScript in the language that binds the Ajax family of technologies together.

  We bind our Ajax functionality into JavaScript event handlers.

In this way, we can create more dynamic user front-ends.

Terminology

- *Ajax -* A family of internet technologies designed to improve the responsiveness of web-based user interfaces.
- *JavaScript -* The scripting language that is used to bind together the various dynamic elements of a client-side interface
- *Document Object Model -* The tree-based representation of an XML document that allows us to manipulate elements of a loaded webpage.

References

- W3schools.com, 2017. [online] Available at www.w3schools.com

Awarding Great British Qualifications

Topic 6 – Scripts (1)

Any Questions?