Dynamic Web Development

Lecture 16 – Ajax 1

What is Ajax?

Asynchronous
Javascript
and
XML

A bit misleading, as it doesn't have to be asynchronous, and it doesn't have to involve XML.

The name for a collection of technologies that had existed for some time, but web developers had never combined them in this way.

The term was coined by Jesse James Garrett in:

"Ajax: A New Approach to Web Applications"

http://www.adaptivepath.com/ideas/essays/archives/000385.php

With traditional web page design, if you want something on the page to change, you have to refresh the entire page.

The browser issues an HTTP request.

The server issues an HTTP response.

The new information is displayed in the browser.

The key to Ajax applications, is the use of scripted HTTP requests.

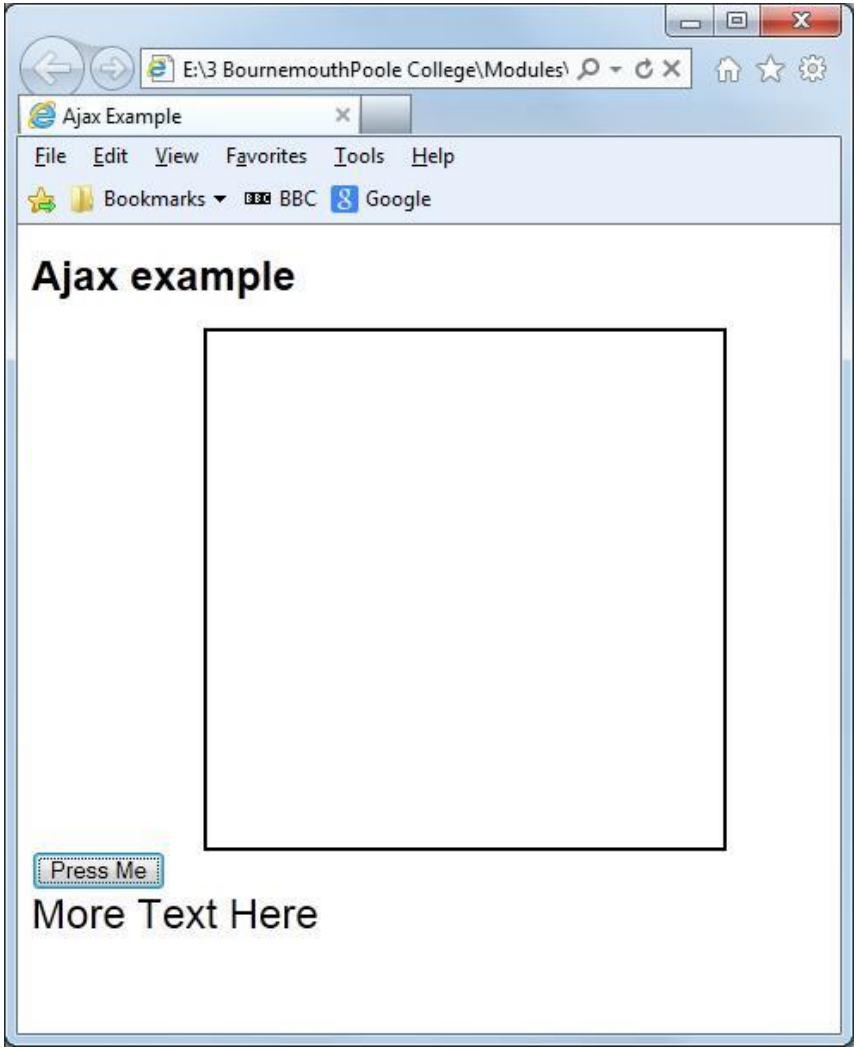
Since the amount of data transferred is often very small, and the browser doesn't have to parse and render an entire web page, response time is improved.

An html file

```
<!DOCTYPE html>

<html>
 <head>
  <title>Ajax Example</title>
  <link type="text/css" rel="stylesheet" href="Ajax03.css"></link>
  <script type="text/javascript" src="jquery-1.7.1.js"></script>
  <script type="text/javascript" src="Ajax03.js"></script>
 </head>

 <body>
   <h1>Ajax example</h1>
   <div id="window"></div>
   <button id="one">Press Me</button>
   <div id="text2">More Text Here</div>
 </body>
</html>
```

CSS file

```
h1
{
font-size: 18pt;
font-family: Arial;
}

div#window
{
margin-left:100px;
width: 300px;
height: 300px;
border: solid 2px black;
}

div#text2
{
font-size: 18pt;
font-family: Arial;
}
```


E:\3 BournemouthPoole College\Modules\
Ajax Example
File Edit View Favorites Tools Help
Bookmarks BBC Google
Ajax example
Press Me
More Text Here

testdata.txt

```
This is a test file containing some text.  I want it
to appear in a document when I click on a button,
without having to refresh the page.  I need to use a
remote http request to get the file. This is called
the Ajax technique.
```

In more complex Ajax applications, this data would be encoded in XML format, or JSON format.

XML (Extensible Markup Language) is a way of encoding documents electronically. The standards are laid down by the W3C.

JSON is a more lightweight format. It is not as structured as XML, and therefore easier for the browser to process.

Loading External Files From the Server

The XMLHttpRequest object is well supported in modern browsers and provides script access to the HTTP protocol.

You can:

- Make GET, POST and HEAD requests.
- Return the webservers response synchronously or asynchronously.
- Return content as text or a DOM document.

Despite the name, it is not limited to XML documents. It can fetch any text document.

The XMLHttpRequest object is a key feature of the AJAX architecture.

Using XMLHttpRequest

This is a three part process:

1. Creating an XMLHttpRequest object.
2. Specifying and submitting the HTTP request to the web server.
3. Synchronously or asynchronously retrieving the server's response.

Unfortunately, the XMLHttpRequest object has never been standardised, and the process of creating one differs depending on the browser you are using.

Luckily, once created, the API for using it seems to be the same on all platforms.

Creating an XMLHttpRequest object

```
function loadtext()
  {
    if (document.all)
      var request = new ActiveXObject("Msxml2.XMLHTTP");
    else
      var request = new XMLHttpRequest();
```

The document.all property is only recognised by Internet Explorer, and so can be used for browser sniffing.

It is possible that future versions of Internet Explorer will implement the W3C standard and make this check unnecessary.

However, most people tend to use the jQuery implementation of Ajax, which is browser independent, as we will see.

Submitting the request - Synchronous

Using a synchronous call means that execution of the JavaScript code will stop until a response has been received from the web server.

```
// Instruct it to get a specified file,
//using a synchronous call
request.open("GET", "testdata.txt", false);

// Send the request to the webserver
request.send(null);
```

Retrieving the Response - Synchronous

```
// Store the returned text string
var doctext = request.responseText;

// Find container div and copy the text into the DOM
var element = document.getElementById("window");
element.innerHTML = doctext;
```

Note the use of the innerHTML method. This saves you having to link the text into the required text node of the DOM.

More complex Ajax applications manipulate the DOM directly.

Ajax01.js

```
function loadtext()
  {
    if (document.all)
      var request = new ActiveXObject("Msxml2.XMLHTTP");
    else
      var request = new XMLHttpRequest();

    // Instruct it to get a specified file,
    //using a synchronous call
    request.open("GET", "testdata.txt", false);

    // Send the request to the webserver
    request.send(null);

    // Store the returned text string
    var doctext = request.responseText;

    // Find container div and copy the text into the DOM
    var element = document.getElementById("window");
    element.innerHTML = doctext;
  }
```

Submitting the request - Asynchronous

The problem with a synchronous request is that if the server is down, or busy, the Javascript comes crashing to a halt until the response is received.

Not good if you are halfway through updating your bank account or credit card details.

An asynchronous call means that execution of the JavaScript code will not wait - it will carry on even if no response has been received from the web server.

This means that we must specify what is effectively an event handler before we send the request to the server.

Creating an XMLHttpRequest object

The first part of the script is the same as before:

```
function loadtext()
  {
  // Do the same thing with an asynchronous call

  if (document.all)
     var request = new ActiveXObject("Msxml2.XMLHTTP");
  else
     var request = new XMLHttpRequest();

  request.open("GET", "testdata.txt", true);
```

Except this is set to true!

The XMLHttpRequest Object Reference

http://www.w3schools.com/dom/dom_http.asp

Methods

Method	Description
abort()	Cancels the current request
getAllResponseHeaders()	Returns the complete set of http headers as a string
getResponseHeader("headername")	Returns the value of the specified http header
open ("method","URL",async,"uname","pswd")	Specifies the method, URL, and other optional attributes of a request The method parameter can have a value of "GET", "POST", or "PUT" (use "GET" when requesting data and use "POST" when sending data (especially if the length of the data is greater than 512 bytes. The URL parameter may be either a relative or complete URL. The async parameter specifies whether the request should be handled asynchronously or not. true means that script processing carries on after the send() method, without waiting for a response. false means that the script waits for a response before continuing script processing
send(content)	Sends the request
setRequestHeader("label","value")	Adds a label/value pair to the http header to be sent

Properties

Property	Description
onreadystatechange	An event handler for an event that fires at every state change
readyState	Returns the state of the object: 0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
responseText	Returns the response as a string
responseXML	Returns the response as XML. This property returns an XML document object, which can be examined and parsed using W3C DOM node tree methods and properties
status	Returns the status as a number (e.g. 404 for "Not Found" or 200 for "OK")
statusText	Returns the status as a string (e.g. "Not Found" or "OK")

Like the objects that represent the browser, and the HTML tags, the XMLHttpRequest object has:

- properties
- methods

which we can access from our script.

XMLHttpRequest readyState Values

0	open( ) has not been called
1	open( ) has been called, but send( ) has not been called
2	send( ) has been called, but the server has not responded yet
3	Data is being received from the server
4	The server's response is complete

Note that different browsers differ in their handling of readyState 3

See Flanagan, D. (2006). JavaScript: The Definitive Guide. 5th ed. for full details of scripting http requests.

This is the name we gave to the request when we created it.

```
// W     the readystate property changes,
// i  will trigger this function

request.onreadystatechange = function()
        {
         if (request.readyState == 4)
           {
            // Get the data

            // Make use of the data here
           }
        }
```

Preparing to Receive the Response - Asynchronous

```
// When the readystate property changes, it will
// trigger this function

request.onreadystatechange = function()
        {
         if (request.readyState == 4)
          {
           // Get the data
           var doctext = request.responseText;

           // Make use of the data here
           var element = document.getElementById("window");
           element.innerHTML = doctext;
          }
        }

// Once the event handler is defined, it is safe to
//send the request
request.send(null);
```

Ajax02.js

```
function loadtext()
 {
    if (document.all)
      var request = new ActiveXObject("Msxml2.XMLHTTP");
    else
      var request = new XMLHttpRequest();

    request.open("GET", "testdata.txt", true);

    request.onreadystatechange = function()
       {
        if (request.readyState == 4)
          {
           var doctext = request.responseText;
           var element = document.getElementById("window");
           element.innerHTML = doctext
          }
       }

    request.send(null);
 }
```

Adding the jQuery library

There are two ways that you can link your pages to the jQuery library:

1. Download the library file from http://jquery.com/ and link to it as a local file.
2. Link to a remotely hosted copy of the library.

Once you have downloaded one of the library files, you need to add the following line at the top of your webpage (remember to update the version number):

```
<head>
  <script type="text/javascript" src="jquery-1.7.1.js"></script>
</head>
```

This tells the browser where to find the jQuery library routines.

Using jQuery

The jQuery library provides a number of Ajax functions that are essentially wrapper methods.

That is to say, they shield you from some of these inner workings of a scripted HTTP request.

Shorthand methods:

jQuery.load()

jQuery.get()

jQuery.post()

Full method:

jQuery.ajax()

$.load( ) Ajax03.js

```
$(document).ready(function(){
  $("button").click(function(){
    $("div#middle").load('test1.txt');
  });
});
```

Syntax is:

Wait until document has finished loading {
  When the button is clicked, run this function {
    Load into div#middle the file test1.txt
  }
}

Would this work?

```
$(document).ready(function(){
  $("button").click(function(){
    $("div#middle").load('gettext.php');
  });
});
```

and gettext.php contained something like this:

```
<?php
$text1='This is more text that ';
$text2='I want to send.';
echo "This is the text that I want to send";
echo $text1.$text2;
?>
```