CSY2028
Web Programming
Topic 19

Tom Butler
thomas.butler@northampton.ac.uk

Topic 19

- Quick javascript recap
- AJAX – Requesting data from and sending data to the server using JavaScript

Ajax

- The usual process for a PHP page is:
  - The user connects to the server requesting a specific page e.g. index.php
  - The server runs the PHP code
  - HTML is generated and sent to the browser
  - The browser displays the HTML
  - The user clicks on a link e.g. contact.php and the entire process starts again

Ajax

- This is the traditional way websites work, however most modern websites are considerably more *interactive*
- When using Facebook, Twitter, Google or any other big website you'll see that you tend to stay on *one page* which constantly gets updated without needing to be refreshed

AJAX

- Ajax stands for *Asynchronous Javascript and XML*
- Ignore the name, all it means is using Javascript to request information from the server while the page is open
- Ajax is the technology that makes this type of website possible

Ajax

- Using ajax, the browser can use Javascript to make a request to the server *in the background*
- This can be done without the knowledge of the user
- The page doesn't go blank or show the user that anything is loading (spinning icons, progress bars, etc)

Ajax

- When using Ajax the user experience can be different:
  - They navigate to a page, e.g. index.php
  - The server generates some HTML and sends it to the browser
  - When a user clicks on a link or submits a form, instead of navigating the browser to another page, the browser sends a request to the server *in the background*
  - The response from the server is stored using Javascript which can then place the response on the page somewhere

Ajax

- Ajax can be used to request anything from the server
- Usually Ajax is used to request:
  - HTML code
  - JSON data (Javascript Object Notation)

Quick Javascript recap

- Javascript is included on web pages using the <script> tag
- The <script> tag is placed inside the <head> element
- The src attribute is the name of a file containing your javascript code

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

<script> tag

- HTML files are processed from the top to the bottom
- This means the <head> and <script> tags are loaded before the <body> element

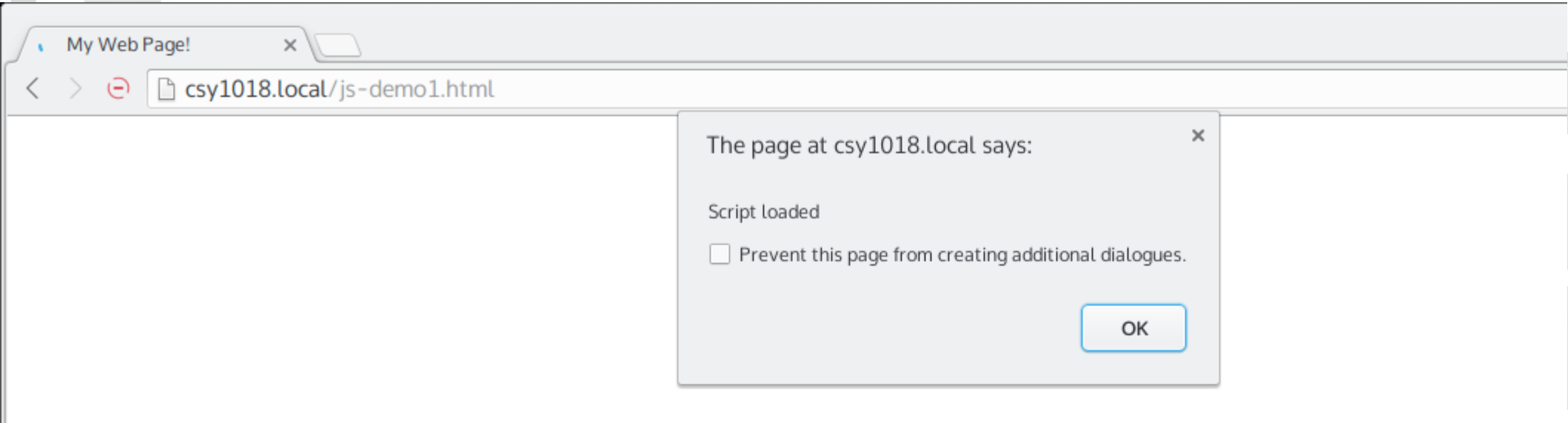
```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Javascript

- If you add some code to the javascript file it is run before the elements on the page exist

```
alert('Script loaded');
```



Javascript

- Because of this, if you want to write code that uses the elements in some way (e.g. changing the content) you must run the code to do this **after** the page is loaded
- To do this, you need to write a function and tell the browser to run it when the page loads

Javascript

- This is done using the inbuilt function document.addEventListener function

```
function myLoadFunction() {
        //code to run when the page loads
}


document.addEventListener('DOMContentLoaded', myLoadFunction);
```

DOMContentLoaded
means when the content on the
Page is loaded (the elements exist)

The name of the function

Selecting elements in Javascript

- Javascript contains functions for selecting HTML elements so you can change properties on the (css, attributes, etc)
- The simplest way is to give an element an ID in the HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1 id=”pageheading”>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Selecting elements with Javascript

- Once an element on the page has an ID, you can use the javascript function document.getElementById() to select it and store the *element* in a variable

```
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1 id="pageheading">
        Page heading
    </h1>
    <p>Page content</p>
  </body>
</html>
```

```
var element = document.getElementById('pageheading');
```

Selecting elements

- Once you have an element you can make changes to it
- E.g. to update the content you can use:

```
var element = document.getElementById('pageheading');
element.innerHTML= '<strong>HTML to add</strong>';
```

Click Events

- You can call element.addEventListener to add an event to a specific element
- addEventListener takes two arguments:
  - The name of the event, e.g. 'click'
  - The name of a function to call when the event occurs

Javascript Click Event Example

```
function myClickEvent() {
  alert('The button was pressed');
}

function myLoadEvent() {
  var element = document.getElementById('elementId');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded', myLoadEvent);
```

When the button is clicked, run the function myClickEvent

When the page loads, run myLoadFunction

Ajax

- Javascript can be used to make requests in the background to retrieve some information from the server
- To use AJAX you need to create an XMLHttpRequest Object

```
var xmlHttp = new XMLHttpRequest();
```

XmlHttp

- The XMLHttpObject has a method called *open*
- This takes three arguments:
  - Method – Either "GET" or "POST" (like HTML forms)
  - The URL to load
  - Whether the request is "asynchronous", true or false. In most cases this can be set to true.
- E.g.

```
var xmlHttp = new XMLHttpRequest();
xmlHttp.open('GET', '/page-on-server.php', true);
```

Request/response

- There are two parts to an Ajax request:
  - Sending the request to the server
  - Waiting for the response from the server
- Before sending the request, you have to set up a *listener*
- The listener is a function that runs when the server sends the response

Capturing the respose

- This creates a function that is run when the response is sent from the server

```
xmlHttp.onreadystatechange = function() {
        if (xmlHttp.readyState > 3) {
                //Code to run when the response is received
                alert(xmlHttp.responseText);
        }
};
```

- The variable xmlHttp.responseText stores the text e.g. HTML code that the server sent to the browser

Sending the response

- After configuring the request with *open* and configuring what to do with the response using *onreadystatechange* you can send the request to the server
- This is done using the *send()* method:

```
var xmlHttp = new XMLHttpRequest();

xmlHttp.open('GET', '/page-on-server.php', true);

xmlHttp.onreadystatechange = function() {
        if (xmlHttp.readyState > 3) {
                //Code to run when the response is received
                alert(xmlHttp.responseText);
        }
};

xmlHttp.send();
```

Ajax Requests

- Ajax requests usually happen after an event, e.g. when clicking on a button or link
- To do this, you can put the Ajax code in an event listener:

```
function myClickEvent() {
      var xmlHttp = new XMLHttpRequest();

      xmlHttp.open('GET', '/page-on-server.php', true);

      xmlHttp.onreadystatechange = function() {
         if (xmlHttp.readyState > 3) {
                  //Code to run when the response is received
                  alert(xmlHttp.responseText);
         }
      };

      xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded', myLoadEvent);
```

Ajax Example

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="ajaxdemo.css"/>
    <script src="ajaxdemo.js"></script>
  </head>
  <body>
    <button id="clickme">Click me</button>
    <div id="content">

    </div>

  </body>
</html>
```

```javascript
function myClickEvent() {
  var xmlHttp = new XMLHttpRequest();

  xmlHttp.open('GET', '/page-on-server.php', true);

  xmlHttp.onreadystatechange = function() {
     if (xmlHttp.readyState > 3) {
      //Code to run when the response is received
        alert(xmlHttp.responseText);
      }
  };

  xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded',
                          myLoadEvent);
```

```javascript
function myClickEvent() {
  var xmlHttp = new XMLHttpRequest();

  xmlHttp.open('GET', '/page-on-server.php', true);

  xmlHttp.onreadystatechange = function() {
     if (xmlHttp.readyState > 3) {
      //Code to run when the response is received
        alert(xmlHttp.responseText);
      }
  };

  xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded',
                          myLoadEvent);
```

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="ajaxdemo.css"/>
    <script src="ajaxdemo.js"></script>
  </head>
  <body>
    <button id="clickme">Click me</button>
    <div id="content">

    </div>

  </body>
</html>
```

When the page loads, the element with the ID `clickme` is assigned a click event

Click me

```javascript
function myClickEvent() {
  var xmlHttp = new XMLHttpRequest();

  xmlHttp.open('GET', '/page-on-server.php', true);

  xmlHttp.onreadystatechange = function() {
     if (xmlHttp.readyState > 3) {
      //Code to run when the response is received
        alert(xmlHttp.responseText);
      }
  };

  xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded',
                          myLoadEvent);
```

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="ajaxdemo.css"/>
    <script src="ajaxdemo.js"></script>
  </head>
  <body>
    <button id="clickme">Click me</button>
    <div id="content">

    </div>

  </body>
</html>
```

When the button is pressed
The function `myClickEvent`
is run

Click me

```
function myClickEvent() {
  var xmlHttp = new XMLHttpRequest();

  xmlHttp.open('GET', '/page-on-server.php', true);

  xmlHttp.onreadystatechange = function() {
     if (xmlHttp.readyState > 3) {
      //Code to run when the response is received
        alert(xmlHttp.responseText);
      }
  };

  xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded',
                          myLoadEvent);
```

The browser loads 'page-on-server.php' in the background

```
<h2>Ajax Request</h2>

<p>This page has been loaded from the server</p>
```

The function is run and xmlHttp.responseText is set to the contents of the response

Debugging

- The Javascript console can be used to monitor Ajax request
- Press F12 on the browser (or Tools → Developer tools) and it will open the *Developer Console*
- The network tab shows any ajax requests that have been made. You can click on one of the requests to view the response

127.0.0.1:8080/ajaxdemo.html

Click me

Elements Console Sources Network Timeline Profiles Resources Security Audits

View: Preserve log Disable cache No throttling

Name

ajaxdemo.html
demo.css
ajaxdemo.js
page-on-server.php

Headers Preview Response Timing

```
1 <h2>Ajax Request</h2>
2
3 <p>This page has been loaded from the
```

4 requests | 1.2 KB transferred | Finish: 5.84 s | DOMContentLoaded: 18...

Writing content to the page

- A common use for Ajax is updating content on the page (instead of just issuing alerts!)
- To do this, you can use the `innerHtml` property to write content to any element:

```
var element = document.getElementById('content');
element.innerHTML= '<strong>HTML to add</strong>';
```

Writing content to the page

- This can be done in the ajax call:

```
function myClickEvent() {
  var xmlHttp = new XMLHttpRequest();

  xmlHttp.open('GET', '/page-on-server.php', true);

  xmlHttp.onreadystatechange = function() {
     if (xmlHttp.readyState > 3) {
        //Code to run when the response is received
        var element = document.getElementById('content');

        element.innerHTML = xmlHttp.responseText;
      }
  };

  xmlHttp.send();
}

function myLoadEvent() {
  var element = document.getElementById('clickme');
  element.addEventListener('click', myClickEvent);
}

document.addEventListener('DOMContentLoaded',
                          myLoadEvent);
```

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="ajaxdemo.css"/>
    <script src="ajaxdemo.js"></script>
  </head>
  <body>
    <button id="clickme">Click me</button>
    <div id="content">

    </div>

  </body>
</html>
```

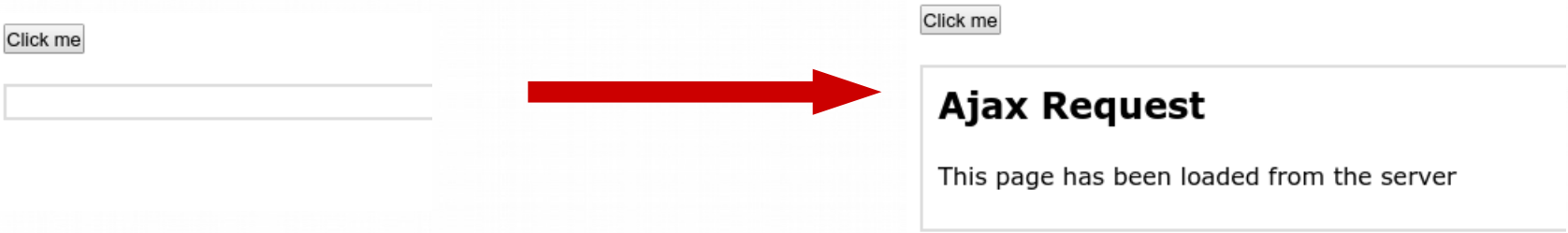
```javascript
var xmlHttp = new XMLHttpRequest();

xmlHttp.open('GET', '/page-on-server.php', true);

xmlHttp.onreadystatechange = function() {
  if (xmlHttp.readyState > 3) {
     //Code to run when the response is received
    var element = document.getElementById('content');

    element.innerHTML = xmlHttp.responseText;
   }
};

xmlHttp.send();
```



Sending POST requests

- POST requests require some POST data
- On a form, this data comes from a HTML form
- For AJAX requests, the data needs to be sent as part of the request
- You need to manually build the data strucuture

```javascript
var xmlHttp = new XMLHttpRequest();

xmlHttp.open('POST', '/login.php', true);

xmlHttp.onreadystatechange = function() {
  if (xmlHttp.readyState > 3) {
     //Code to run when the response is received
    var element = document.getElementById('content');

    element.innerHTML = xmlHttp.responseText;
   }
};

var data = new FormData();
data.append('username', 'student');
data.append('password', 'secret');

xmlHttp.send(data);
```

```php
<?php
var_dump($_POST);
```

```
/srv/http/public_html/index.php:2:
array (size=2)
  'username' => string 'student' (length=7)
  'password' => string 'secret' (length=6)
```

JSON Data

- JSON is a data structure format which stands for Javascript Object Notation
- It’s commonly used to send data from PHP to the Javascript via Ajax

Creating JSON data in PHP

- Any data structure in PHP can be converted to JSON using the json_encode() function

```php
<?php

$array = [
  'one' => 'orange',
  'two' => 'banana',
  'three' => 'apple'
]
echo json_encode($array);
```

```
{"one":"orange","two":"banana","three":"apple"}
```

JSON in javascript

- If your PHP script prints JSON rather than HTML, the javascript can understand this format and replicate the *data structure* you had on the server
- By using JSON.parse(xmlHttp.responseText) you can use the data structure as an object in your javascript code

```
var xmlHttp = new XMLHttpRequest();

xmlHttp.open('GET', '/fruit.php', true);

xmlHttp.onreadystatechange = function() {
  if (xmlHttp.readyState > 3) {
        var jsonString = xmlHttp.responseText;

        var fruits = JSON.parse(jsonString);

         alert(fruits.one); //prints “orange”
         alert(fruits.two); //prints “orange”
    }
};

xmlHttp.send();
```

```
<?php

$fruits = [
   'one' => 'orange',
   'two' => 'banana',
   'three' => 'apple'
]
echo json_encode($fruits);
```

```
{"one":"orange","two":"banana",
"three":"apple"}
```

JSON

- By using JSON you can copy data strutures from PHP directly into your javascript code without much effort
- These can be multi-dimensional arrays or objects
- You can even use json_encode on arrays that have come from the database!

Exercise 1

- 1) Create a HTML page with a <button> element. When the button is clicked send an Ajax request to a page on the server e.g. ex1.php. The page should print the date in the forma dd/mm/YYYY. Print the returned data using an alert() box with the text: "Today's date is: ..."
- Sample output :
  - "Today's date is 16/03/2017"

Exercise 2

1. Take one of your databases from previous weeks for storing users (or create one if you don't have one already. Fields: ID, Username, Password, Firstname, Surname, Email) and make sure there at least 3 records in the table.

2. Create a PHP script with a form that contains a <select> element and a submit button. The <select> element should contain a list of all of the user's from the database in it's <option>s. When the submit button is pressed, send an Ajax request to the server and generate a HTML table that shows all the information about the selected user. Write the generated HTML to the page underneath the form. You should be able to view information about each user without the page refreshing.

2a. Remove the button and make the request when the selcect box is changed using the javascript “onchange” event

Exercise 3

- Download game.zip and add Ajax requests so multiple players can log into the game and see each other's characters:
  - Hint: Use a database table to store each player's head/body/and xy coordinates
- Request the data about all players from the server in the timeout and redraw the players
- When a player moves, send their new coordinates to the server
- Upload your game to the university server
- For a demo visit http://www.eng.nene.ac.uk/~tom/game.html