

6G5Z2107: Web Design and Development

Lab 04 – jQuery, AJAX and XML

Matthew Crossley

Objectives

- Be able to construct dynamic webpages based on files obtained from a server

The aim of today is to reinforce the concepts we covered in this week's lecture, and those we've developed over the past two weeks. You can, and should, refer to the slides used in the lecture, and you may even wish to challenge yourself by running through the quiz again. This week draws together everything we've covered so far this term and is a bit of a challenge – get ready...!

IMPORTANT - Before you begin!

As emphasised in the lecture, now that we are working with XMLHttpRequests, we cannot test our code without running it through a server, as only certain protocols (such as http) will allow for data to be fetched. We will be running our code using the **XAMPP Apache** server, which allows your computer to act as a local server.

There is a portable version of XAMPP available on the university's network drive¹. Copy this entire directory to a **USB stick**, or, to **the computer's D drive**. We will be working in the **htdocs** folder. If you are working on the **D drive** and not your USB stick, make sure to copy the contents of your **htdocs** folder to your **H drive** at the end of the session, as the D drive is machine-specific and will not move with you!

Once you have XAMPP running (run setup_xampp.bat once copied, then launch xampp-control.exe, and start **Apache**), point your browser to **localhost:80** to test everything is working okay. This is essentially a mirror of your **htdocs** directory, so any files you put in this directory, you can access through your browser. For example, if I put a file called **exercise_one.html** in my **htdocs** folder, I could access it through my browser by navigating to **localhost:80/exercise_one.html**

Note that if you are using Notepad++, launching files directly into the browser will **not** use this method, and will no longer work!

¹ S:\Faculty_Of_Science_Engineering\School_ComputingMathDigitalTech\6G5Z2107 Web Design and Development\xampp-portable-win32-1.8.2-6-VC9\xampp\

Exercise 1: This Menu is Not Very Refreshing

In this week's lecture, we were introduced to jQuery's `.load()` method, which can be used to obtain data from a server – and then incorporate it into a page without a refresh. That is exactly what we will be doing in this first exercise!

Your tasks are two-fold:

1. You have been given a template page (**exercise_one.html**) and the contents of a menu (**menu.html**). You will need to load the contents of the menu into the template page, inserting them into the appropriate div (the one with the `outermenu` id).
2. Once the menu is in place, we want to add events to the menu. You will need to inspect **exercise_one_menu.html** and consider what your page will look like once the contents of this have been loaded into your template page. You will need to add events, which then load the other pages...

- news.html
- aboutus.html
- contactus.html

... into the content div, when the appropriate menu items are clicked.

You will need to use a combination of `load()`, callback methods and event handling to solve this exercise.

When completed, your page may look something like the below:



Exercise 2: A Player Library

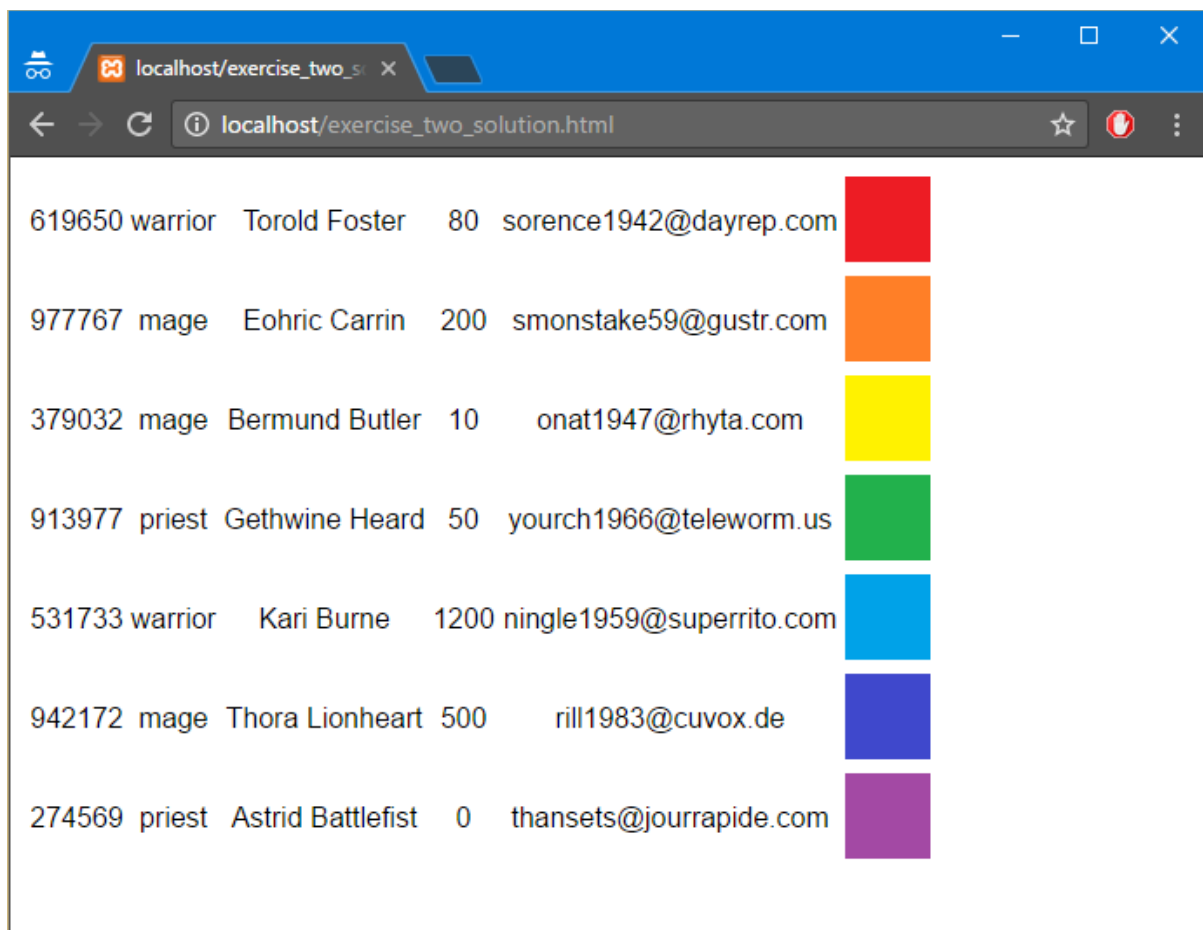
For your second exercise, you have been given an XML file containing some player data, similar to that we examined in the lecture (**exercise_two.xml**). It has some different players, and some extra data, in it.

Your task is to take the template (**exercise_two.html**) and add jQuery/JavaScript to show the contents of the XML file in a table. You will need to **get** the file, and then write some code to parse and handle the XML file, and then write the output to the empty table that has been included in the template.

You will also need to spend some time examining the XML file, and becoming familiar with the format and the data held within it, in order to complete this exercise.

Everything you need to complete this exercise is available in both the lecture notes and the quiz, but you may need to adapt the code and extend it accordingly.

Once completed, your page may look something like the below:



The screenshot shows a web browser window with the address bar displaying 'localhost/exercise_two_solution.html'. The page content is a table with 7 rows of player data. Each row has five columns: ID, class, name, level, and email. To the right of each row is a small, solid-colored square. The colors of the squares are red, orange, yellow, green, blue, dark blue, and purple, respectively.

619650	warrior	Torold Foster	80	sorence1942@dayrep.com	
977767	mage	Eohric Carrin	200	smonstake59@gustr.com	
379032	mage	Bermund Butler	10	onat1947@rhyta.com	
913977	priest	Gethwine Heard	50	youch1966@teleworm.us	
531733	warrior	Kari Burne	1200	ningle1959@superrito.com	
942172	mage	Thora Lionheart	500	rill1983@cuvox.de	
274569	priest	Astrid Battlefist	0	thansets@jourrapide.com	

Extension Exercise 3: A Pretty, Filtered Player Library

The page we made in Exercise 2 is okay, but isn't very usable. Let's say we have 10,000,000 players (like a certain popular Massively Multiplayer Game once had). Would we want to display them all on the same page at once? Probably not.

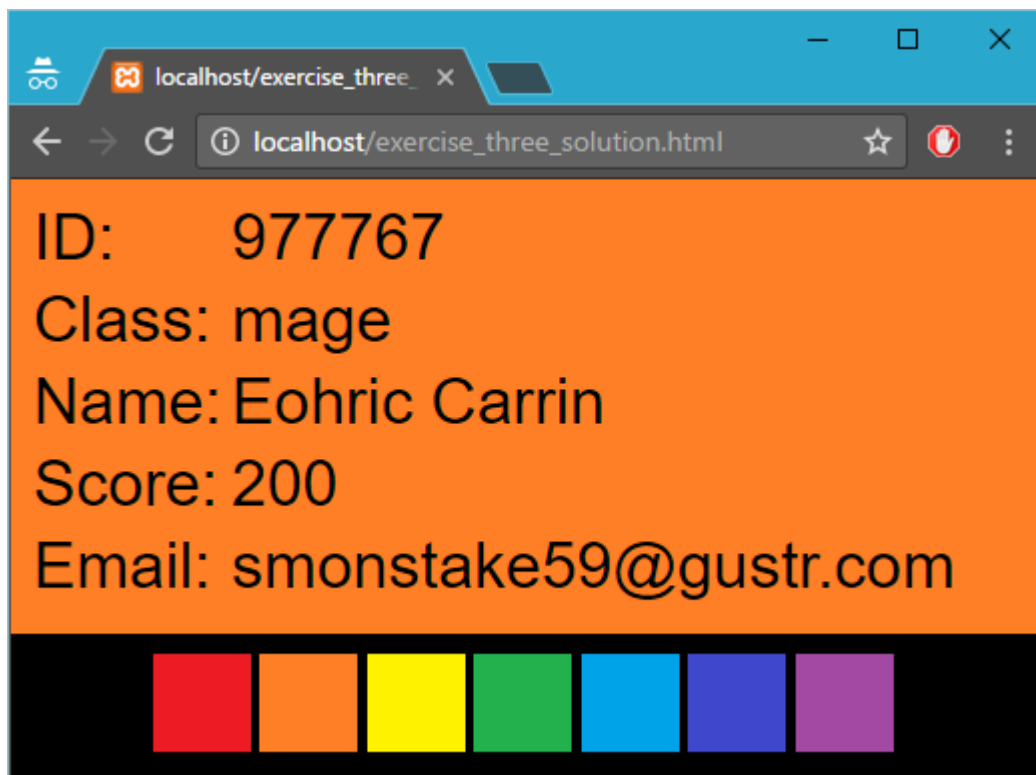
Your task for this extension exercise, is to think about how we could display **a single player** at any given time. This is a difficult task, and will require everything we have learnt so far all working together. A recommended approach is to consider the following:

1. We will need a menu with which to select which player is currently on display. We can do this by getting the XML, iterating through all players, and displaying only certain information on the page. For example, we might display only the avatar of each player in a big list.
2. Once we have a menu, we need to add some kind of event handling to each, which identifies which player each represents (in part, this is similar to Exercise 1 of this week). How could we identify each player uniquely? At this point, you might consider refreshing your memory on how to write your own **JavaScript functions**, although it is not wholly necessary.
3. Once you have an event set up for your menu, you will need to write the handler. One way of doing this – and there are many – is to again get the XML file, and iterate through all the players, this time outputting the full player details if, and only if, the player matches the one we are looking for.

This is but one approach. There are many ways to tackle this problem. Another idea is to maybe pre-load all of the players into JavaScript objects, store them into an array, and allow our menu to select players this way.

If you have an idea, but are not sure how to implement it, feel free to discuss it with your tutor.

When completed, your page may look something like the below:



Help! I don't know what to do!

Don't panic!

There are plenty of resources available, including the following, which are all great starting points:

- The lecture slides (available on Moodle) which recap what we covered in the lecture
- w3school's jQuery & AJAX tutorial:
http://www.w3schools.com/jquery/jquery_ajax_intro.asp
- jQuery Essential Training on [Lynda.com](https://www.lynda.com) – covers more than you need to know!
 - Particularly: <https://www.lynda.com/jquery-tutorials/Convenience-functions/494389/539736-4.html>
- XML Essential Training on Lynda.com: <https://www.lynda.com/XML-tutorials/Welcome/145930/164596-4.html> - tonnes of detail specifically on XML if you want to really delve into it!

The online resources are the same as last week, but you want to look at different sections of them this time! Particularly the AJAX Operations section.

Exercises 1 and 2 can be solved using much of the code included in the lecture notes – but it's important that you understand what's going on, too.

If you are having difficulties, please discuss with your tutor, who will be able to guide you through some of the early steps to solving the puzzles.

I fancy a challenge, what do you recommend?

Have a go at the extension exercise (#3). You will need to stretch your JavaScript legs (again) to solve some of this, but don't panic – it's just combining everything we've covered so far with what we covered last year.

If you want to push yourself, make sure you partition your code nicely into JavaScript functions instead of trying to cram it all into one big chunk and see how well-formatted you can make your webpage. There are lots of sensible ways you could make Exercise 3 more impressive!