|  |
| --- |
| University of South Wales |
| Advanced Internet and Mobile Computing |
| IS3S664 |

|  |
| --- |
| Baber M (FCES)  1-10-2020 |

Table of Contents

[1 – Introduction 2](#_Toc29542827)

[2 – Design 3](#_Toc29542828)

[3 – JavaScript / jQuery 5](#_Toc29542829)

[4 - PDO 7](#_Toc29542830)

[5 - XML 7](#_Toc29542831)

[6 - Evidence of testing using different browsers. 7](#_Toc29542832)

[7 - References 7](#_Toc29542833)

[8 - Software 9](#_Toc29542834)

[9 - Appendix 9](#_Toc29542835)

[All your code and details of database tables and XML files should be included as appendices 9](#_Toc29542836)

# 1 – Introduction

This report will look over creating a website which makes use of front-end and back-end functions using JavaScript, jQuery and PHP to manipulate the page layout. Starting with the design, the implementation and how the functions are navigating the DOM tree to send and receive data.

# 2 – Design

Before anyone can create a website, it would be ideal if they had an idea of what they want it to look like. This would have website layout and basic functions with a brief explanation of what functions they should carry out (e.g. print hello on click).

The design of this website was intended to be as simple as possible to not distract from the main operations of the JavaScript. Below was the original design which was drawn up with a little bit of information about what each button is doing and how it is displayed.

A close up of a piece of paper

Description automatically generated

After looking at the design, the next stage was to start the implementation of HTML & CSS design.

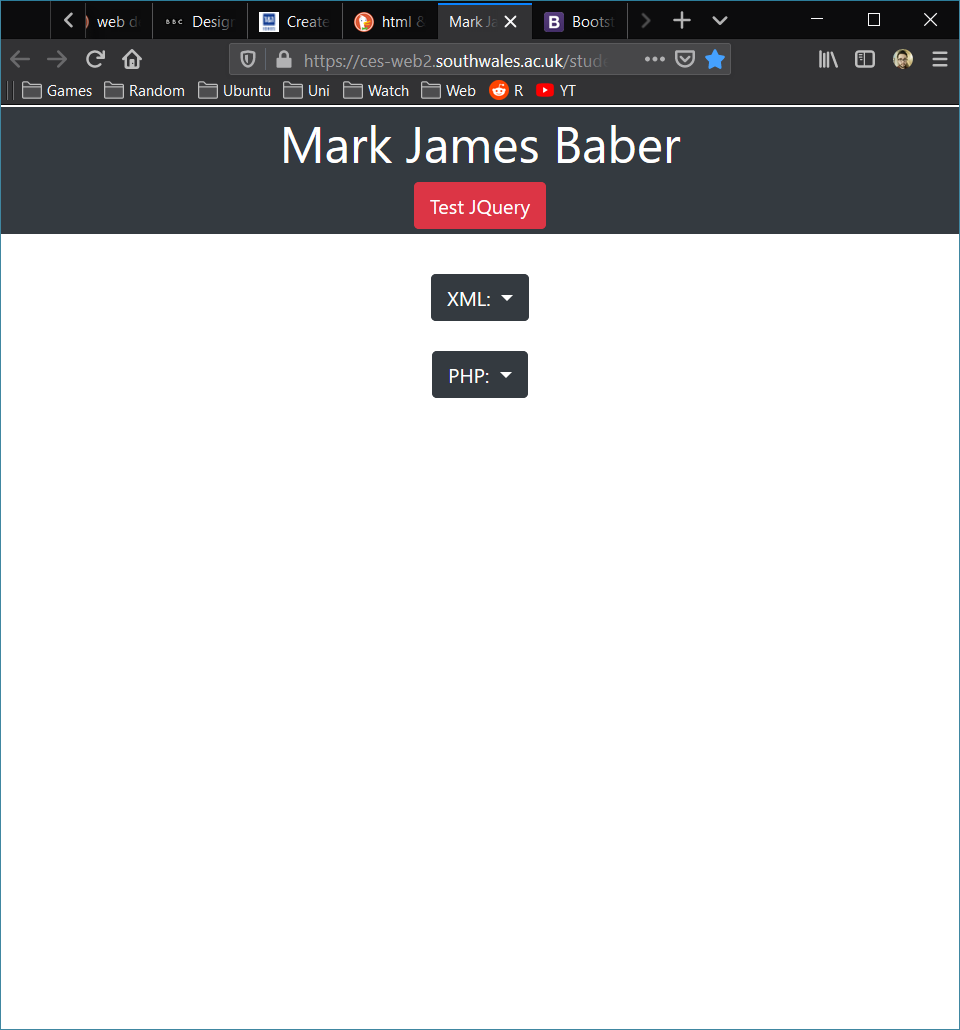
To start this process there was a frame-work used called Bootstrap which will take care of a lot of the CSS design, which can be used to speed up the process of creating a new website. This was used due to the fact the designer had used it for a few other websites in the past.

Starting with the default page and 3 buttons was quite simple without functionality, back to the design each section is within a div element to make sure we can edit each part separately if needed. For example, if we wanted to replicate the main header from the landing page, this is all the code you would need.

*<div container>*

*<h1>Mark James Baber</h1>*

*</div>*



(Landing Page)

From the landing page, the user has a choice of 2(3) buttons to choose from, the XML and PHP buttons were implemented to get data from their tables (XML & PHP). To get these buttons working correctly, there was some JavaScript & jQuery being used to create a drop-down effect. The user would then be prompted with the 2 types of files available to them. This will be covered more in the next section.

# 3 – JavaScript / jQuery

During the testing phase, there was a lot of errors being made with how to import the jQuery library. Due to this, there has been a button added called *Test JQuery* at the top of the page. This was implemented to alert the user is anything went wrong with the jQuery library. This simple function works something like this:

*onPageLoad – if jQuery library detected – alert (Library loaded fine)*

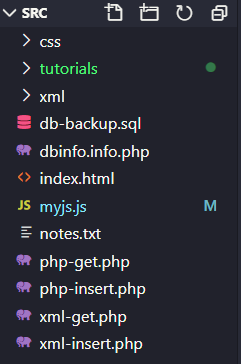
*else - alert (Something went wrong).*

With 2 alerts being called with 1 function, it would get annoying every time the page loaded to alert the user, which is why it is now waiting for the click of a button.

The drop-down buttons also have JavaScript functionality, to start with both buttons have the same effects. When both buttons are click, their contents will start hidden with a delay and a slide down to make the button more modern. This can be seen quite easily within the jQuery code as some parts is quite easy to read. Below is a snippet for the first button.

*$('#dropdown-xml').hide().delay(100).slideDown(250);*

For the first option on the drop-down after the fancy effect, is the XML – Pets button, as the name states it will get the pets from the xml file. In order to do this the file structure should be as follows

By having the xml folder contain the xml files, it doesn’t allow any errors from the server. As some servers have rules set up to make sure files can’t be manipulated at the root file.

So, in order to make sure the button XML – Pets gets the pets, there was JavaScript and PHP being implemented. To start off the function it uses a function called getJSON which calls a php file *xml-get.php.* This PHP file has a lot of information in there for error checking, so the user can track errors and easily report them. The file also contains what the source file should be called, which is stored as a variable in the PHP, which checks that a file exists within a folder called *"xml/".$filename.".xml"*. By using the filename within a variable, this allows the use of the same code for multiple files and tables.

After some extensive error checking, if no errors are found the file is returned to the JavaScript function, which now states what the file name is *sourceName: "pets"*. Again, this is great so there isn’t much to be changed for other files and tables. After getting the xml file back, the function will then go through another form of error checking to make sure the data is in fact an XML and doesn’t have any syntax errors. This was where the function started to become quite sophisticated, the function starts off by declaring a variable called table which contains the html for a table header and the first row. For this example, it was for pets, so the row looks like this:

*<thead><tr><th scope="col">Pet Name</th><th scope="col">Pet Type</th><th scope="col">Pet Breed</th><th scope="col">Pet Owner</th></tr></thead><tbody>';*

The function then checks the entries of the XML using the forEach function, this goes through the XML for all pets with the data entries within each pet. For example,

*Pets: pet1 – pet1.name – pet1.type – pet1.breed – pet1.breed – pet1.owner*

In order to carry out this function smoothly, there needed to be a nested loop getting the entries (pets) and values (petName). By putting this in a nested loop the function can carry out a loop of pets with the respected values and due to this being within a loop, it builds the rest of the table around each entry to make it dynamic. This data is then placed within a div on the index.html page using a small bit of jQuery,

*$('#dataTable').html(table);*

This was how the first get function ended.

xmlGetPets()

phpGetPets()

xmlPetsInsert()

phpPetsInsert()

hideForms() for each table

hideSearch()

showForms() for individual tables

showSearch() for individual tables

disabledButtons() for each

enabledButtons() for individual table

# 4 - PDO

The PHP documentation describes PDO as something which represents a connection between PHP and a database server (PHP, 2020). Which is almost like a variable that stores all of your database information that can be used to carry out a bunch of commands which you pass it. Within this website, there was use of PDO to set our SQL within a variable that will ready the command, execute the command and even give error information.

Within this website, the PDO was started off as a PDO::\_\_construct which is what stores our database details. Below is the PDO variable used within the project, note it doesn’t give out any of our personal information.

*$pdo = new PDO("mysql:host=$host;dbname=$database;charset=utf8", $username, $password, [PDO:: ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, PDO::ATTR\_EMULATE\_PREPARES => false]);*

The next part of the PDO is the prepare section which can have SQL stored within, this is useful as it doesn’t allow people to use SQL injection within the browser. After preparing the PDO to carry out the required SQL, the user can then use the command execute to start off the process.

SQL Injection is carried out when a user can type an SQL command within the URL to get or delete your database tables without the need to login to it.

# 5 - XML

# 6 - Evidence of testing using different browsers.

# 7 - References

Bootstrap - <https://getbootstrap.com/docs/4.4/getting-started/introduction/>

W3schools - <https://www.w3schools.com/howto/howto_js_dropdown.asp>

jQuery - <https://api.jquery.com/>

PHP - <https://www.php.net/manual/en/class.pdo.php>

# 8 - Software

Windows 10 – 1909 (OS Build 18363.535)

Firefox - 72.0.1 (64-bit) – Windows

Google Chrome - Version 79.0.3945.117 (Official Build) (64-bit) – Windows

Visual Studio Code – 1.41.1

XAMPP – 3.2.4

GitHub Desktop – 2.2.4

# 9 - Appendix

# All your code and details of database tables and XML files should be included as appendices