**Coursework for Internet and Web Technologies module (2018/19)**

The purpose of this coursework is to help you learn about using Javascript, the DOM and jQuery to process JSON data. The coursework will be assessed and counts 10% of the final mark for this module.

**The task**

The JSON data you will be working with represents results of tennis matches played at Wimbledon in 2013. Specifically, you will use results for the men's and women's tournaments, which are stored in [wimbledon-men-2013.json](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/coursework/wimbledon-men-2013.json) and [wimbledon-women-2013.json](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/coursework/wimbledon-women-2013.json), respectively. For each match (represented by elements of the "match" array), the file contains information about the round ("round" key) in which the match took place (possible values are 1 to 7), and the two players involved ("player" array). Each player has their name recorded ("name"), the outcome ("outcome") of the match for them (with values "won" or "lost"), and the number of sets won ("sets-won"), with possible values being 0 to 3 for men and 0 to 2 for women. This is followed by the number of games they won in each of the sets played, represented by the "set" array (which will contain up to 5 elements for men and up to 3 for women).

The product of the coursework should be a HTML page which a user can use to query information about the results of matches. In other words, the HTML page should allow the user to select which results they are interested in (i.e., to choose one of the two files mentioned above), and then provide an interface through which the user can query the data. The results should be displayed on the same page. You should use JavaScript, jQuery and HTML forms to implement your solution, which should work with *both* Firefox *and* Chrome. The techniques you need to use are covered in[Client-side processing](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/client/client.html). Extra information is given [below](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/coursework/cw1.html#hints).

The components of the task are as follows:

1. Create a web page consisting of an HTML form with various form fields for user input, as well as an area for outputting results. The results should appear in a table on the same page. The table should have columns for round, player, and the results for up to five sets. Each pair of rows in the table represents the result of one match, with the name of the winner appearing in bold. So for the first two results in the men's tournament, the table might look as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Round** | **Player** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** |
| 1 | B.Becker | 4 | 3 | 2 |  |  |
| 1 | **A.Murray** | 6 | 6 | 6 |  |  |
| 1 | J.Ward | 7 | 4 | 6 | 6 |  |
| 1 | **Y-H.Lu** | 6 | 6 | 7 | 7 |  |

1. The first result above shows that A.Murray beat B.Becker in round 1 by three sets to love, 6-4, 6-3 and 6-2. There should be a button on the page which when clicked results in the table being filled with appropriate rows. When reloading the HTML page, the form fields should be cleared (reset).
2. By means of a drop-down list, the user should be able to select which set of results (i.e., which file) they wish to query. If the querying button is clicked with no further restrictions being entered into the form by the user, then all the match results from the selected tournament should be returned in the table.
3. The user should be able to enter the name of a player, as well as one of the conditions 'equals', 'contains' or 'none'. If 'equals' is selected, only the results of matches in which the player with exactly the given name participated should be returned. For 'contains', the value entered needs only to match a substring of the player's name. For example, the user may not remember that Murray's first initial is 'A', so can search using the string "Murray". The match should be case-sensitive in each case (so avoiding the need to transform the strings). For 'none', no restriction is placed on the player name.
4. The user should be able to enter a value for the number of sets (e.g. 4) as well as one of the conditions 'equals', 'greater than' or 'less than'. So if the user enters 4 for the number of sets and 'greater than' for the condition, only the results of 5-set matches are returned.
5. The user should be able to enter a value for the round (e.g. 6) as well as one of the conditions 'equals', 'greater than' or 'less than'. So if the user enters 6 for the round and 'equals' for the condition, only the results of the two semi-final (i.e., round 6) matches are returned.

For each of (3), (4) and (5), you might use a combination of a drop-down list for the user to select the condition and a text box for them to enter a value.

The user should be able to make selections from any combination of items (2) to (5) above, so, for example, to ask for all matches played by A.Murray lasting more than 3 sets in the men's tournament.

I would recommend developing your solution in stages. This is particularly important since debugging Javascript can be very time-consuming and frustrating. All browsers do now provide a "developer console" or something similar which you can use to view error messages, set breakpoints allowing you to view the values of variables, and so on.

Start by simply trying to read one of the JSON files and outputting a single value. Then develop a solution which will output all the results for one of the tournaments. This can then be modified to add the functionality for choosing a file and then gradually adding all the conditions to be checked. At each stage add only small amounts of code before retesting your solution.

**Handing in the coursework**

The deadline for submission is *6pm on Tuesday 19th March 2019*. Please submit the coursework via Moodle *as a single zip file* containing a single HTML file, along with any separate Javascript files you use. You should not submit any instructions or explanations in a separate file. Instead, the interface should be self explanatory and the code should be commented appropriately.

Remember that plagiarism is taken very seriously by the Department and the College (see the relevant section in your programme booklet). Consequently, you are required to state the following in your HTML submission (either as a comment in the source code, or visible on the displayed page): *I confirm that this coursework submission is entirely my own work, except where explicitly stated otherwise.*(You are welcome to reuse code presented during lectures without acknowledgement, but any other code that is not yours should be acknowledged.) Your solution may be submitted to an online plagiarism detection service. The College's disciplinary procedure will be invoked in any cases of suspected plagiarism.

The College policy with regard to late submission of coursework is described in the MSc/MRes programme booklet. No extensions will be granted. The cut-off date for submissions is *6pm on Tuesday 26th March 2019*. Submissions after this date will not be marked. Those submitted after 6pm on the 19th and before 6pm on the 26th March, where mitigating circumstances are not accepted, will receive a maximum mark of 50%.

**Marking guide**

Your program should be properly structured and should include comments and some simple error checking. The user interface does not need to be elaborate, but it should be clear to a user how to use it.

Marks will be awarded out of 20. The areas in which marks will be awarded and the maximum mark possible in each case are as follows:

|  |  |
| --- | --- |
| friendliness of the user interface | 2 |
| code structure and documentation | 2 |
| error handling in the code | 2 |
| part 1 | 2 |
| part 2 | 3 |
| part 3 | 3 |
| part 4 | 3 |
| part 5 | 3 |

Full marks for the first 3 items above will not be awarded if only a partially working solution is submitted.

Comments on your coursework, along with the mark you were awarded, should be returned to you within 4 weeks of the cut-off date.

**Hints and potentially useful information**

* Firefox and Chrome have different behaviour when retrieving a JSON file. Both work fine if you perform the retrieval over the Internet, e.g. if your HTML and JSON files are in your DCS web space and you load the HTML file over the web (by using its URI). Firefox will also work fine if you load the file *locally*, i.e. you load your HTML file by double-clicking on it. Chrome will return an error if you do this and then try to retrieve the JSON file.
* For debugging Javascript, it can be helpful to use the console.log function. You enter this in your script (e.g. console.log("Reached line 7");) and it will output whatever is specified to the console (which can be viewed using the Web developer tools of the browser and switching to the console). The function can take multiple arguments: strings, variable names, etc.
* To reserve an area for output, you can have a part of your document identified for example as follows
* <div id="resultArea">
* </div>

and then append a document fragment to this div element or use the innerHTML property.

* Remember that + is used for string concatenation.
* To test whether two values are equal, use ===; to test whether two values are unequal, use !==; to test whether *both* of a pair of conditions is true, use &&; to test whether *either* of a pair of conditions is true, use ||.
* To test whether a string contains a given substring, you can use the indexOf(.) method of Javascript. For example, myString.indexOf(mySubstring) returns the starting index where mySubstring occurs in myString (starting from index 0), or returns -1 if mySubstring does not occur in myString.
* If you need to use quotes inside quotes, the inner pair will have to use a different symbol from the outer pair.
* If you use the jQuery [$.each](http://api.jquery.com/jquery.each/) function to iterate over an array, you should be aware that the callback function is passed an array index and a corresponding array value each time. By contrast, the example used on [slide 32](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/client/client.html#(32)) iterates over an object.

The links on the [Links to more information](http://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/client/client.html#(34)) page might also be useful.