# *Web Programming III (420-H30-HR)*

# *Assignment 1 – AJAX*

Date assigned: Wednesday, September 9, 2020

Date due: **Wednesday, September 23, 2020**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Read JSON file
* Write new record to JSON file
* Perform error checking on values entered

To do:

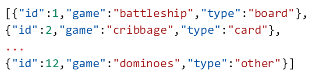
**General idea of the assignment:**

Create a web application that allows the user to display a list of all the game playing members. The user can filter the list based on a specific game or a specific game type. The user must be able to add a new player to the members and include one or more games that the player has played.

**Details:**

Your data files look like this:

games.json – A listing of 12 different games each with a type. You cannot change this file for the assignment. This lists the games that are in the user set. You must use this file to get the games (all 12 of them) AND the game types (4 of them right now). These values can NOT be hardcoded in your code and must be read from this file. You will also have to sort these lists and get the unique ones out. Simple array of JSON objects in the following format:



players.json – A listing of the players who are actively playing the games. An array of JSON objects with the format as below. You will need to scan, sort, display and update this file. I will provide the PHP program you need to update the file (more below). A sample record looks like:  
  
You can see the key/value pairs listed. The games\_played is an array of games the player has played and the last time they played them. If they has only played one game, games\_played is just a JSON object.

The avatar is an image from a site that produces random avatars based on a string passed. The form is https://robohash.org/*<<somestring>>*.png ?size=*##*x*##*&set=set*n* where *<<somestring>>* is any characters; ## is the size you want the image in pixels and *n* is the set the images are from. So <https://robohash.org/amcdonald.png?size=60x60&set=set1> produces .

You will need to produce this URL whenever a new player is added (see below).

Your program must contain the following components:

**An HTML page to View Players**

This page is called gamePlayers.html which displays the information from the players.json file automatically. This must be formatted professionally and properly and the avatar must be displayed (don't show the link). The list is sorted by the last\_name and then the first\_name of the player. All information must be displayed.

The user must be able to filter the people based on the game or the type of game. This can be done using a dropdown list or other means if you have a better idea. The list of games and types MUST retrieved from the games.json file.

This must all be done using the JavaScript fetch command. You cannot use XHR or jQuery. You can make it async if you want (I would) or do it with regular promises. All errors must be caught and handled professionally (a nice message to the user not just a console.log dump).

If the user chooses to filter, then only those records matching the filter are displayed. So, if the user chooses the board game type, only people who have played at least one board game are displayed. The user can further filter by game (or filter by game immediately) and only the players who have played that game are displayed.

**Functionality to Add Player**

You must include functionality to allow the user to add players to the players.json. This can be a separate page or (better) the same page with a form hidden in a div that is displayed when the user chooses to add a player. Storing the information is done using the PHP program addPlayer.php which is called using the POST method.

You must perform basic validation on the fields to ensure all the required fields are there. You also must validate that the wins and losses values are numbers. ALL ERRORS MUST BE DISPLAYED AT THE SAME TIME.

When adding a player:

1. You must create a unique id number that is one greater than the maximum id number in the file (there is an array function for this)
2. The enrolled field must default to today, but can be changed.
3. The avatar URL must be formulated as their username .png and size 60x60. Use any set 1-4 (randomly or consistently it doesn't matter).
4. The addPlayer php program does not do ANY validation so you will need to validate the record before posting.
5. When a player is added you must be able to add one or more games for the player. The list of games and types comes from games.json and cannot be hardcoded in your page.

You must send the information to the program using a fetch command with POST. You must stringify the json object which is the player JSON object before sending it to the server. Do NOT send all the information, just the record that was added.

When the record is added the list of players is updated automatically. Two ways, force the request after add or (better) set a timer (timeout) to request the list of players every 10 seconds.

Remember, use the power of the developer options in the browsers to know what is being sent as parameters. YOU CANNOT USE A FORM WITH A POST METHOD. YOU **MUST** USE THE POST METHOD USING FETCH TO SEND THE INFORMATION ASYNCHRONOUSLY.

**Functionality to Increase Wins and Losses**

The user must be able to add wins and losses to the person. This can be does as an increment of 1 at a time or by entering a value and adding that to the wins and losses. The php program updateCount.php must be sent the entire JSON record for the user using a POST command.

**The Final 5**

Doing everything about perfectly will get you a maximum 95%. To get the final 5% you need to do at least 2 of the following:

1. Allow the user to change the sort order to sort by first name, game type or game,
2. Automatically generate the username based on the first\_name last\_name as first\_initial and last\_name. Allow the default value to be overwritten,
3. Validation on fields such that: names only contain letters (if you want to add a single space, quote and dash that is great); dates are valid dates, email is a valid email,
4. Ensure that the date a game was played was not before the person enrolled,
5. Have a ranking display for wins/total and sort from top to bottom.

**Other Things**

1. There are marks for interface design. Use appropriate colours, labels, headers, etc. This does not mean make it garish or having flashing lights, it means make it professional. Design for mobile first.
2. Make the code efficient; there are marks for efficiency.
3. You must use HTML5 and (proper) CSS techniques for this assignment. You are welcome to use grid layout if you wish.
4. The application **must** be responsive at the bootstrap breakpoints which are: >=768px, and >=1200px (you do NOT have to do the other ones but you may). That is, the page must adjust at each of those breakpoints. You do NOT have to use bootstrap. Remember it needs to look good on your phone so make sure that you try it on the phone and that it has proper font sizes and positioning.

**To submit**

When you have completed the assignment zip the YourUserName\_H30A01 folder containing all the files for the assignment and copy it to the course page.