Lab 5: JavaScript

Due. Wednesday, July 10th, 2019

In this lab, you will continue working on your website from the previous lab. You will add client-side javascript to upgrade the website's functionality. You do not need to modify any of the HTML files for today's lab — the files provided in this lab have been updated to include the changes you made to the website during the last lab.

What To Do

Part 1: Client-side javascript

Step 1. Download the website template files here. The website directory structure looks as follows:

```
|--Lab_Website
      |--views
            |--login.html
            |--register.html
            |--home.html
            |--team_stats.html
            |--player_info.html
      |--resources
            |--css
                  |--signin.css
                  |--my_style.css
            |--img
                  |--beproudcapaabanner.jpg
                  |--cu_boulder_logo.jpg
                  |--favicon.ico
                  |--temp_image.svg
            |--js
                  |--my_scripts.js
```

The only file you will need to modify today is my_scripts.js. **If you are unclear about how your javascript will change your webpages, inspect the HTML files and look for clues.**

Step 2. You will now update the registration page to hide/show sections of the form based on the user's radio button selection.

In my_scripts.js, write a method that will accept the id of an HTML tag and a toggle value as input and then sets the HTML tag's CSS visibility to hidden/visible and the height to 0/auto.

parameters:

`id` - the CSS id of the HTML tag being updated

`toggle` - an int, either 0 to hide the HTML tag or 1 to make the HTML tag visible

method stub:

```
function viewStudentStats(id, toggle) {
     /* Get the HTML element with the given id */
     /* If toggle is 0 set the HTML element's css visibility to
     hidden and the height to 0, but if toggle is 1 set the css
     visibility to visible and the height to auto. */
}
```

Step 3. The home page now contains a row of HTML buttons. Each button, when clicked, should change the webpage's background color.

In my_scripts.js, write a method that will accept a CSS color and set the HTML body's background color to this color.

parameters:

`color` - a CSS color

method stub:

```
function changeColor(color) {
     /* Set the HTML body's background color to color */
}
```

Step 4. You will now update the statistics page. The javascript for this page won't be interactive. Instead, the javascript will start when the page is loaded. It will determine the winner of each game and calculate the total wins/losses for the Buffs this season.

In my_scripts.js, write a method that will iterate through the stats table and update the winner column for each game listed in the table and update the win-loss ratio for the Buffs (second table)

parameters:

none

method stub:

```
function loadStatsPage() {
        /* Read through each row of the table & determine which
        team won the game */
        /* Update the winner column to the name of the winning
        team */
        /* Keep track of the number of wins/losses for the Buffs
*/
        /* Update the second table to show the total wins/losses
        for the Buffs */
}
```

Step 5. Now, update the player information page. The player information page will have two pieces of functionality. First, you need to populate the dropdown menu to allow the user to pick which player's information to view. This should happen when the page loads. Second, when a player's name is selected, you need to show their information and image.

In my_scripts.js, write two methods: one that will populate the dropdown menu and one that will update the player's info, calculate the average passing, rushing and receiving yards.

parameters:

None

method stub:

```
function loadPlayersPage() {
        /* Iterate through the players and do the following for
        each player:
                1. Create an anchor tag
                2. Set the href to "#", this will make sure anchor
                   tag doesn't change the page
                3. Set the onclick to call the switchPlayers
                   method with the player's array index as the
                   parameter
                4. Set the anchor tag's text to the player's name
        */
        /* Update the innerHTML of the dropdown menu. The id for
        the dropdown menu is player_selector */
}
```

parameters:

`playerNum` - the index of the player in the players array

method stub:

```
function switchPlayers(playerNum) {
        /* Update the HTML span tag on the player's information
        page and calculate the average passing, rushing, and
        receiving yards. */
    }
```

HINT: The onload event handler executes automatically after the entire page has loaded. However, one cannot simply stack onload event handlers like this:

```
window.onload = function ...
window.onload = function ...
```

After the first onload event handler is executed it will be replaced when the second onload event handler executes. To get around this, you can encapsulate your function calls inside a new function like this:

```
function start() {
    func1();
    func2();
}
window.onload = start;
```

What To Turn In

Submit your my_scripts.js to Canvas.