PIC 40A: Homework 5 (due 2/21 at 10pm)

It is important that you meet the following requirements.

- You must upload your files to **Gradescope** before the deadline.
- You must upload your files to the **PIC server** in the appropriate directory before the deadline.
- Both submissions must be **identical** (down to the character).

  Never make changes to the PIC server submission after the deadline.

  (We can see when a file was last modified.)
- You must tell us (me and the grader) your **PIC username**.
- You must **validate** your HTML using https://validator.w3.org/.

In this assignment you will submit five files...

1. `README.txt`. This will contain your PIC username.
2. `login.html`, `login.js`. `index.html`.

   These are files that you worked on for homework 3. Updates will me made to these files. Don't worry about the other files from homework 3 & 4. They will be incorporated into the final project at the end of the class.
3. `username.js`. This is a new file.

As mentioned above, you should submit all files to Gradescope before the deadline.
You should also submit the files to the PIC server. Save them in the directory

`/net/laguna/???...???/your_username/public_html/HW5`

(in the folder `HW5` within `public_html`). We should all be able to view your live webpage at

www.pic.ucla.edu/~your_username/HW5/login.html

Now, I am just left to tell you what I want the webpage to do. See the next page!

New Features

A `cookie` is a Document property that allows you to read and write cookies using a getter and setter. They come in a specific string format grouping corresponding name-value pairs. The nice aspects of `cookie` is we can begin to transfer information from one page to another page. We can also set cookies to expire so when you close your browser and then reopen it, the cookie will remain (note this won't work with incognito). You'll use `cookie` that will expire to store a `username`.

The following updates will be added to your pages...

login.html

1. Your login page should appear as before. The only difference is now it should also source `username.js`.

index.html

1. Your `index.html` should also source the same `username.js`.
2. A `<span>` element at the top (first child element of `<header>`) should be added with the `id` attribute set to the value "greeting".
3. Upon visiting this page after logging in with a username, the user will see a greeting "Hello, {}!". Where {} is the username they typed in the username text box of `login.html`. This part should be done by the new script `username.js`.

login.js

1. This script should fill the text box with the user's username if it is stored in `document.cookie`. (You will want to make of the function in `username.js`, so you should include your JavaScript files in a sensible order when they're sourced on your HTML pages.)
2. It should make use of two event listeners as in Homework 3 with the added feature that provided that an appropriate username has been chosen, it should also...
    - Create a new cookie with `name` equal to the string `"username"` and `value` equal to what the user typed. This cookie should expire in an hour. It should have the default `path`, i.e. do not specify anything about the path; `path=/` is incorrect. Leave `path` alone.
    - Redirect to `index.html`.

username.js

1. This script should contain a single function definition.

   The function should be called `get_username` and have no parameters.

   It should extract from `document.cookie` the `value` corresponding to the `name "username"` or return the empty string if there is not such a `name`.

   Note that `get_username` should account for when `document.cookie` returns `'username; username =johnnypickles'`. The similar HW2 question did not ask you to account for this scenario. Recall that the first name-value pair in this example has an empty name and value equal to `username`.

2. If we go to `index.html` after logging within the last hour, a greeting is added to the new `<span>` element with the specific username. For example, `Hello, johnnypickles!`

3. If we go to `index.html` but have not logged in or our login has expired, we are redirected back to `login.html`.

4. This script is imported both by `index.html` and `login.html`. You'll need to check the `pathname` of your page before adding the previous two features. Otherwise, you'll get a error when you load `username.js` into `login.html`.

Grading

- (4 pt) Trying to log in with an invalid username should not set a cookie. Either by pressing enter in the username box or by clicking login. It should display the correct alert.
- (4 pt) Trying to log in with a valid username sets a cookie. Either by pressing enter in the username box or by clicking login.
- (2 pt) The cookie that's set expires in one hour and no path is specified.
- (1 pt) If the username cookie is set, the textbox in `login.html` shows the username.
- (2 pt) When redirected to `index.html` there's a greeting of the user's username on top in the appropriate format. Similarly, landing on `index.html` after the username has been entered and within the hour it was set shows the greeting.
- (2 pt) If we go to `index.html` but have not logged in or our login has expired, we are redirected back to `login.html`. Also, same thing for going to

  `https://www.pic.ucla.edu/~your_username/HW5/`.
- (1 pt) `index.html` and `login.html` will source the same `username.js` and loading this file will not produce errors.
- (2 pt) `get_username` works correctly with changes made as specified.
- (1 pt) `get_username` function definition is not included in `login.js`. (No redundancy).
- (1 pt) The JavaScript has no uses of `var`.