PIC 40A: Homework 3 (due 2/7 at 10pm)

Like on homework 1, 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 nine files...

1. `README.txt`. This will contain your just your PIC username in the first line. Optional: A description of the location of the phishing link (if not obvious —you may change the location if you'd like when you submit your final project).
2. `login.html` and `logim.html`.
3. `login.js` and `logim.js`.
4. `index.html`
5. `phish.js`, `scarf1.html` and `scarf2.html`.

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/HW3`

(in the folder `HW3` within `public_html`). We should all be able to view your live webpage at

www.pic.ucla.edu/~your_username/HW3/login.html

Now, I am just left to tell you what I want `login.html`, `logim.html`, `login.js`, `logim.js`, `index.html`, `phish.js`, `scarf1.html` and `scarf2.html` to achieve. See the next page!

So you know where we're heading...

In this homework, we're going to see how someone could phish users' passwords using JS. We are also going to steal some code from another webpage.

Imagine that there is a website made by someone. We will refer to the creators of this website as `nice`. On their website, users can log in and they can post comments about anything that they want to. `nice` are not very careful and they allow their users to post raw HTML. On the home page, some recent users' comments are displayed.

In this situation someone can be `malicious` and post a comment which includes mischievous anchor elements. We'll pretend that someone has posted a comment as malicious666. malicious666 says: "Could anyone see how I can fix my scarf? Please help. I'm so sad. Here's a picture of the other side." The words "scarf" and "picture" will link to pages which look like they have failed to load. "scarf" will direct to an error page that looks like it was produced by the PIC servers. "picture" will direct to an error page that looks like it was produced by Google Chrome. Both of these error pages will load in a new tab, but here is the catch... They will cause the original home page to be redirected to a page that looks the same as the login page but which functions differently. When a user enters their login information to this new page, `malicious` will be able to acquire it*.

`login.html` and `logim.html`

To a casual observer `login.html` and `logim.html` will look exactly the same. That is, they should both look like `login.html` from HW1 with one improvement: add a `<label>` to the texts corresponding to the first two input elements.

So what is the difference?

- Typing in `login.html`'s username box and pressing Enter, or clicking "Login" will redirect to `index.html`.
- Typing in `logim.html`'s password box and pressing Enter, or clicking "Login" will not redirect to anything and the page will mock you.

`login.js` and `logim.js`

While `login.html` and `logim.html` will look exactly the same, they will have different functionality.

`login.html` will source `login.js` to do `nice` things. This includes,

- Use your HW2 function and two event listeners to accomplish the following. . .
  1. Clicking on the "Login" button validates the username in the textbox.
  2. Pressing the Enter Key while typing in the textbox validates the username in the textbox.
- Your `validate_username` function should be used here with a few changes:
  - If the username is invalid, the user is alerted on the page in a useful manner.

  - If the user enters an acceptable username, it will not show any message. Instead the page will be redirected to `index.html`. Once you add the Event Listener, you should remove the `onclick` attribute from `login.html`.

  Note: Nothing is done with what's typed the password box on this page. Pressing Enter in the password box does nothing yet.

`logim.html` will source `logim.js` to do `malicious` things.

- This time clicking on the "Login" button, or pressing the Enter Key while typing in the password box will cause the page to mock you. Add two event listeners for those interactions.
- The "mocking" will include (see demo):
  - Add an additional `<p>` element inside `<section>` and at the bottom stating: "Somebody knows the password you like to use is **{}**." Where **{}** is the password entered in the password box displayed in bold.
  - The main heading should be changed to say "HA". Then each time you continue to try to submit another "HA" is added with no spaces in between.

  Note: Nothing will be done with what's typed the username box on this page. Pressing Enter in the username box does nothing.

`index.html`

A new section should be added at the bottom of `index.html`.

- It should have the heading "Some recent posts by other users:".
- Inside it should have a post by user `malicious666`. The post should contain two links to `scarf1.html` and `scarf2.html`. Clicking on either of these links will open up a new tab while the original page will be redirect to `logim.html`. (You can call these two links something different but they have to correspond to the equivalent fake error pages in the next set of instructions.)

`phish.js`, `scarf1.html` **and** `scarf2.html`

- `phish.js` will be one line long. It'll use `window.opener.location` to redirect the opening tab. Both `scarf1.html` and `scarf2.html` should source this file.
- What about the rest of `scarf1.html` and `scarf2.html`? Well, first...

  Note: it is okay if `scarf1.html` and `scarf2.html` don't validate.

  This is because we're going to steal `scarf1.html` and `scarf2.html` from elsewhere.
- To get `scarf1.html` you can attempt to go to

https://www.pic.ucla.edu/~burnett/thisDoesNotExist

and use the JavaScript console to `console.log` the `outerHTML` of the only `<html>` element.

- To get `scarf2.html` you can attempt to go to

  http://www.utternonsense.notawebsite.com/afterForwardSlash

  and use the JavaScript console to `console.log` the `outerHTML` of the only `<html>` element.

  The file is long, but you should be able to find `www.utternonsense.notawebsite.com` eight times and replace each occurrence accordingly: sometimes you'll want `www.pic.ucla.edu`; sometimes you'll want the entire path `https://www.pic.ucla.edu/~your_username/HW3/scarf2.html`.

  Also, you'll want to edit the function `reloadButtonClick` so that the `location` is updated to `https://www.pic.ucla.edu/~your_username/HW3/scarf2.html` regardless of the value of the parameter `url`.

*Aside: A small lie was told here. In this assignment we are practicing modifying the DOM. Currently, we are only doing client-side scripting, which is all we can with Javascript alone. Meaning these modification happen only on the user's computer. In order for `malicious` to steal your information and put it on their server, they would need a server-side language like PHP. Javascript can submit the data to the host's server using an AJAX POST request, then PHP can handle saving the data to the host's computer. We'll practice with that during week 7.

See the next page for Grading details. . .

Grading

1. (2 points) The pages `login.html` and `login.html` look identical with HTML elements and details as specified.
2. (1 point) A valid username on the `login.html` page redirect to `index.html`
3. (1 point) On `login.html` hitting "Submit" validated the username.
4. (1 point) On `login.html` pressing Enter will typing in the username box validated the username.
5. (1 point) On `logim.html` hitting "Submit" mocks you.
6. (1 point) On `logim.html` pressing Enter will typing in the password box mocks you.
7. (4 points) "Mocking" in both ways described above.
8. (2 point) `index.html` should have a new section with the correct heading and two links. From `index.html`, clicking on the "scarf" anchor opens `scarf1.html` in a new tab and redirects `index.html` to `logim.html`.
9. (2 point) From `index.html`, clicking on the "picture" anchor opens `scarf2.html` in a new tab and redirects `index.html` to `logim.html`.
10. (2 point) `scarf1.html` looks like a realistic error message created by the PIC servers.
11. (2 point) `scarf2.html` looks like a realistic error message created by Google Chrome.

    (There is more to address here than for `scarf1.html`.)
12. (1 point) The reload button on `scarf2.html` appears to try to reload the page and appears to continue to result in the page not loading.