CS 203, Emmanuelle Renauld
Assignment 6: More Javascript

All labs must be submitted by the next Sunday

Part 0. Setting up your GitHub branch

At the beginning of every lab, you should set up your GitHub repo. Follow instructions in lab 0.

1. Fix my comments on your previous pull request.
2. IF YOU HAVE NOT RECEIVED A COMMENT SAYING YOUR PULL REQUEST IS GOOD FROM LAST LAB, COME SEE ME. I MAY NOT HAVE RECEIVED YOUR LAB.
3. If you have received an ok, merge your previous pull request online.
4. On your computer, go to your master branch and pull the changes.
5. You may delete last week's branch
6. Create a new branch lab6 and go on it.

If you want, you can cleanup your website (optional):

1. We no longer need the table on the main page (index)
2. We no longer need the calculators page

Part 1. Setting the nav menu with Javascript

You must have noticed by now that adding a new page to your website requires changing your navigation menu on each page, which is difficult to maintain. Today, we will remove the copy-pasted menu on all pages, and instead, create it once and import it on each page. Note that we will do it today with JavaScript to practice our JavaScript skills, but considering that JavaScript can be deactivated on a page, this is not the best option. Later, we will learn to do it with PhP instead.

1. Copy-paste everything that was inside your nav menu (ex, from index.html) in a new nav.html page. For now, remove the class "current_page" that we created before to highlight the current page.
2. On each page, remove the nav menu, and replace it by the code below. Here, our navigation menus will be called main-nav, and we will call a JavaScript function setNav that we will soon create. Note that giving an ID to your nav is not required, but makes it a lot easier to access the right node in the DOM using JavaScript.

```
<nav id="main-nav"></nav>
<script>
setNav()
</script>
```

3. Create a nav.js page, create a function setNav inside, with the following code:

```
fetch("nav.html")
 .then(r => r.text())
 .then(html => "YOU NEED TO
CHANGE THIS");
```

Explanations:

- *fetch creates a request to fetch your html file.*
- *r is the response from the request to the nav.html file. r.text() reads the file's content. So* .then(r => ...) *means: do this when you have received r.*
- .then(html => …) *means: do this when the html content (from nav.html) is ready.*

4. Change the section “YOU NEED TO CHANGE THIS”. The line of code that goes there should be a single line of code doing the following: it will find the element associated to ID “main-nav” and change its innerHTML to the value of the variable html.
5. Import nav.js it at the top of each page (head section) (using <script>)

SEE NOTE ON NEXT PAGE

Note. This will NOT work. You can check the console in the "inspect" panel in the browser. This is because your browser has rules to prevent "fetch requests" to get a file on your own computer. A fetch request is not the same as simply opening the file directly in your browser. It asks the web server to give you access to the file. We will learn all about this in class tomorrow. However, you can test this right now by opening a pull request and setting your website to access your new lab6 branch right now. Your website will then be allowed to fetch the nav.html page from GitHub.

Hint. Meanwhile, while doing the next steps and debugging, you can add back the content of the nav on your index page, just while you prepare your code.

6. Let's now set back the "current_page" highlight. We want to automatically add the current_page class to the tab that contains a link to the same pas as where we are currently.
    a. In the setNav function, add an input argument current_path.
    b. In each call to setNav, we want to send the current_path. This could be done manually, or you can use the code below to find the whole (relative) path automatically.

```
const current_path = location.pathname;
setNav(current_path)
```

    c. In your JavaScript file, add this function at the top. This will help you make sure that we always use relative path (to be sure we will not try to compare a relative path with an absolute path).

```
function splitAtRoot(path){
    const url = new URL(path, location.origin);
    const pathFromRoot = url.pathname;
    document.write("<br>----> path from root: " + pathFromRoot)
    return pathFromRoot
}
```

    d. In your setNav function, apply this to the current_path your received:

```
current_path = splitAtRoot(current_path);
```

    e. In your setNav function, we will now loop on all elements in the nav menu. If it points to the same page as current_path, we will apply the current_page class to that element. Use the code below and complete it. Use child.href, apply the splitAtRoot function on it to clean it, and then verify if it is the same as

current_path. If so, you can add the current_page class with: child.classList.add("current_page"); .

```
nav = "CHANGE THIS TO GET THE NAV FROM ID"
for (let child of nav.children){
    if (child instanceof HTMLAnchorElement) {
            ...
```

Part 2. Create a html form

We will now add a form on your page. It will be a quiz, of a type that we could find in a magazine. Ex: Which type of lover are you? Which type of student are you? You know, the type of quiz we did as teenagers, where it said "If your total score is more than 20, then this is the conclusion". Or, "If you choose more diamonds than clovers, then you are a …". Be creative!

1. Add a new page "my_form" and set it up: add its link in nav.html, and format your page with the head section, nav menu and footer.
2. Add a form element.
3. Add a fieldset element surrounding the whole form and add a legend associated to it.
4. Each other input element should have a label associated to it.
5. Add a text element to receive the name of the user.
6. Add an email element.
7. Then asks your questions. They may be checkboxes, radiobuttons, numbers, or anything! Ask at least 5 questions.
8. Add a submit button.
9. **Change your CSS stylesheet to make it look nice. I want more than a basic styling. Ex:**

Part 3. Add some JavaScript

We will do most of our form management in PhP. But we can already add some checks. Luckily for us, since html 5, some checks are made for us (ex, the email verification with a email input). Try using a wrong address and hitting submit (even if you did not add a onClick= or action=attribute), you will see an error from the browser:

We could still check some aspects.

1. Create a onsubmit=validate() attribute that will call a function. Add that function either in your html file or in a separate JavaScript file.
2. Check that the name and email are not empty. If they are, create a warning (alert() or window.alert). You may add other checks required by your own questions.
3. Make sure that if the warning is shown, the data does not get deleted. Change the call to "onsubmit=return validate()"
    a. Add this at the beginning of validate, to prevent trying to send the data to PHP if it fails: event.preventDefault();
    b. Add a “return true” if everything is fine, else return false;