ISTE-240: Web and Mobile II Assignment 3: HTML Forms and Navigation

Due: Wednesday, September 28, 2022

Overview: working with your Assignment 2 content, turn it into a multipage website, add a new webpage featuring an HTML form, and construct a working menu system to let the user navigate page-to-page.

Step 1. Copy and Rename

- In your local ISTE-240 file system, create a new folder: **exer/assignment03**
- Copy of your local **assignment02.html** file *and its assets* to the new folder^
- Rename the assginment02.html file to **index.html**
- If there were any references to "Assignment 2" in the copy of the HTML file, update them to "Assignment 3" (for example, if you put "Assignment 2" in the `title` tag, be sure to update it)

Step 2. Restructure the Content into Multiple Webpages

Assuming you had enough content in your original HTML file, you need to split the content into multiple webpages - at least three.

- If you didn't have enough content, you can gather some more (text and/or images)

Here's a suggested process...

1. Come up with a plan. Figure out how you want to split-up the content into multiple webpages. A typical way would be by the heading tags.
2. Make multiple copies of your **index.html** file - as many as you need based on the amount of content you have.
3. Rename the copies to filenames that make sense - based on the plan. For example, if you're working with a webpage that had headings like this...
    - **Introduction**
    - **Early Years**
    - **Life**
    - **Plays**

      ...then you would create three copies with filenames like this...
    - **early-years.html**
    - **life.html**
    - **plays.html**
      ...and the "Introduction" would be the **index.html** file
4. Carefully edit the HTML files to *remove* the content that doesn't belong, for each file.

5. Check each webpage; make sure they still work as stand-alone webpages
    - You may have to edit the HTML to move things around, add more content, and/or tweak the CSS ...anything you need to do to make each webpage look *reasonably* normal for what you'd expect web content to look like these days
    - Any questions about that^ use our #help channel in Slack; the instructor will let you know what's reasonable

Step 3. Create and Install a Navigation System

- On any of the webpages, in an appropriate area in the HTML (typically directly below, a HEADER element), add a NAV element.
- Construct an industry standard navigation element as shown in the presentation slides from the last class.
    - Use an unordered list
    - Add anchor tags inside each list item that link to each page, including the current one
- Copy the NAV element from the webpage where you're working, and paste it into the similar areas on each of the other webpages.
- Save everything (of course) and check your handiwork. The menu in the webpages should work as you'd expect
    - One link to every page
    - As you (as user) goes from page to page, the menu seems to stay static within the viewport (it should not move)

Step 4. Style the Menu

- Write appropriate CSS to style the NAV element as a horizontal navigation bar ("horizontal" is required for this assignment)
    - Note: the instructor demonstrated use of a separate **navigation.css** file to separate the styles for the menu. That is not required ...but it's a good practice.
- In the HTML of each webpage, add a class in the appropriate anchor tags in the NAV to indicate which is the "current" webpage
    - The class name can be whatever descriptive word you want; the instructor always uses "*is-current*" but other website developers often use "*active*" or other descriptive class names
- Add the ".is-current" (or similarly named) style to your CSS
    - The goal is to visually change the "current" menu item to indicate to the user, which webpage they're looking at

Step 5. Create a New Webpage with an HTML Form

- Make another copy of any of the webpages. Name it **contact.html** (or something like that ...whatever makes sense for your content)
- *Hollow-out* the content of the new **contact.html** file.
    - You'll probably want to leave the HEADER, maybe the FOOTER ...whatever makes sense for you
- Get the HTML Form we created in-class last week and extract the form (just the FORM tag and everything inside it), and paste it into the new **contact.html** file in a place that make sense for your content

- You can embellish the FORM - add content before, after, and even inside the form tag, as you see fit.

Step 5. Update the NAV elements in all the webpages

- Go back and edit the HTML in each webpage to add a link (list item and anchor tag) to point to the new **contact.html** webpage

Note: this process of manually updating the navigation element on each webpage is an old fashioned process. Nowadays, web developers use various technologies (like "PHP Includes") to do this. We'll start using that later in the semester.

Step 6. Power the HTML Form

- Open your **contact.html** file for editing
- Edit the FORM tag
  - Set the `action=""` attribute to `"form-processor.php"`
  - Make sure the `method=""` attribute is set to `"POST"`
- Check the elements in the FORM.
  - Make sure all the INPUT checkboxes and radio buttons have `value=""` attributes
  - Note all the `name=""` attributes on all the INPUT elements (you'll need them in the next part...)

- Download the **form-processor.php** file from this assignment in myCourses
- Move the file into your **assignment03** folder and open it for editing
- Edit the file where indicated by the comments in SECTION ONE, SECTION TWO, and SECTION FOUR; you need to look at each line of code in those sections and edit as needed to make it work with your HTML form
  - Your email (must be your RIT email - the server is set to reject anything going to anything but an RIT address)
  - Wherever you see `$_POST['']` you must insert the relative value in the `name=""` attributes from your HTML form
    - You may need to add or delete lines in the form processor as needed to match the elements from your HTML form
  - In SECTION FOUR at the bottom, you need to look at how your HTML pages in your website are coded and in effect, build a complete webpage down there in SECTION FOUR

When you've successfully connected and edited the Form Processor file, you'll notice that it does *not* work on your local file system because you don't have PHP running on your computer! To see the Form Processor in action, you need to install it on our web server which has PHP and a mail server already running.

Step 7. Hosting and Turn-in

- Install this assignment into the correct directory on **Solace** (also on Banjo, optionally)
    - Use industry standard folder and file naming conventions, and the file structure as defined previously
    - Be sure to set the file and folder permissions accordingly: **0644** (files); **0755** (folders), and...
    - IMPORTANT: set the permissions for the **form-processor.php** file to **0777** (world read/write/execute)
    - Test the contact form. Clicking the submit button on the form should take you (as a user) to the **form-processor.php** webpage, and (hopefully - if our mail server is running correctly), you will receive an email with all the data entered from the HTML form.
- Turn-in this assignment by posting working the links (URL) to the website in myCourses, in **Assignment 3** on or before the due date
    - NOTE: if you can't get the email part of this assignment to work, no worries. The instructor can't test that part anyway because they don't have access to your email.