CS 328 - Week 2 Lab Exercise - 2024-01-26

Deadline

Due by the end of lab.

Purpose

To practice creating **and validating** a strict-style HTML document, and to practice with some HTML elements.

How to submit

Submit your file for this lab using `~st10/328submit` on nrs-projects, each time entering a lab number of **82**.

Requirements

- You are required to work in **pairs** for this lab exercise.
  - This means **two** people working at **ONE** computer, one typing ("driving"), one saying what to type ("navigating"),

    while **BOTH** are looking at the **shared** computer screen and **discussing** concepts/issues along the way.
- Make sure **BOTH** of your names appear in each file submitted.
- When you are done, before you leave lab, somehow e-mail or copy the lab exercise file so that **BOTH** of you have copies, and **BOTH** of you should submit this file using `~st10/328submit` on nrs-projects, with a lab number of **82**.
  - Because these happen to be files the nrs-projects web server has to be able to reach, the navigator should be able to get a copy of a file from the driver using an approach like this:
    - Assume the driver has username `ab12`, and the navigator has username `yz89`.
    - Also assume the driver created, in their `public_html` directory, a sub-directory named `328lab02`, and this sub-directory contains a lab file `lab2.html` to be submitted for the lab exercise.
    - The NAVIGATOR `yz89` can now:
      - log in to THEIR nrs-projects account

        ```
        cd public_html
        mkdir 328lab02   # or other name they choose
        chmod 711 328lab02
        cp ~ab12/public_html/328lab02/lab2.html .  # note the space & period
        ```
      - And now the navigator `yz89` has their own copy of `lab2.html`.
      - The navigator `yz89` should now UPDATE their `lab2.html` so its opening comment contains a working URL for THEIR copy of `lab2.html` (for example, changing the `~ab12` to `~yz89`)

Problem 1

Now, as a pair:

- Following the class coding standards, create a strict-style HTML document named **`lab2.html`**
  - Create this in the driver's `public_html` directory on nrs-projects.

    (In Problem 2, you will copy this to the navigator's `public_html` directory on nrs-projects.)
  - It is your choice if this is within the `public_html` directory itself, or in a sub-directory of `public_html`
    - (but if it is in a sub-directory of `public_html`, make sure that sub-directory is also world-executable!
    - `711` would be appropriate permissions for such a sub-directory.)

- Use the **basic HTML document structure** given at the end of class on Wednesday, January 24.
  - (You can find this at the end of the "Week 2, Lecture 2 projected notes" in the "In-class examples" on the public course web site.)

add a comment

- After your `html` element's opening tag and before your `head` element's opening tag,

  ADD a **comment** containing your names, AND the **(absolute) URL** I can use to view this `lab2.html`
  - Double-check that the URL indeed works! (successfully displays your page when pasted in a browser)
  - (You *will* lose some credit if this URL does not work when it is pasted it into a browser during grading!)

add a `title` element

- Consider the following information about the **HTML `title` element**:
  - It *can* have content.
  - It should appear *within* the `head` element (it should be a child element of the `head` element).
  - from https://html.spec.whatwg.org/multipage/semantics.html#the-title-element :

    "The `title` element represents the document's title or name. .... The document's title is often different from its first heading, since the first heading does not have to stand alone when taken out of context.

    There must be no more than one `title` element per document."
  - Its content is document metadata, data about the data -- it is not part of the document's content itself. Many browsers will use its content in the tab displaying the document.

  Knowing the above, now ADD an appropriate `title` element in your document `lab2.html`.

add an `h1` (level-1 heading) element

- Consider the following information about the **HTML `h1` (level-1 heading, top-level heading) element**:

- It can have content.
- It should appear *within* the `body` element (it should be a child element of the `body` element).
- It represents a top-level heading for content.

Knowing the above, now ADD an appropriate `h1` element in your document `lab2.html`.

add a `p` (paragraph) element

- Consider the following information about the **HTML `p` (paragraph) element**:
  - It can have content.
  - It should appear *within* the `body` element (it should be a child element of the `body` element).
  - It represents a paragraph in one's content.

  Knowing the above, now ADD an appropriate `p` element in your document `lab2.html` that happens to include BOTH of your names.

add a `ul` (unordered list) or `ol` (ordered list) element

- Consider the following information about the **HTML `ul` (unordered list) and `ol` (ordered list) elements**:
  - These have content, but that content is limited instances of the **HTML `li` (list item) element**!
  - These should appear within *within* the `body` element (should be a child element of the `body` element).
  - `ul` should be used for an unordered list (for example, a bulleted list) in one's content, and

    `ol` should be used for an ordered list (for example, a numbered list) in one's content.
  - The `li` element used for list items can have content, a desired list item.
  - You can have sublists, but the syntax is just a little persnickety, so ask me if you are interested.
  - (How should you change or specify the desired style of bullets or of numbering? You should use CSS, as we will discuss later in the semester.)

  Knowing the above, decide on a topic and ADD an appropriate `ul` or `ol` element containing at least three appropriate `li` elements related to that topic in your document `lab2.html`

add an `a` (anchor) element

- Consider the following information about the HTML `a` (anchor) element:
  - It can have content.
  - It should be within an appropriate "containing" element, for example `p` or `li`.
  - This element is used to indicate text that is hyperlinked to other text or to another document.
  - It has an **attribute** `href` - hypertext reference - whose value is typically a URL (although some other options are also possible).
    - This URL can be **absolute**, starting with a protocol such as `https://`,

      ...or it can be **relative**, the path to the desired other document relative to THIS HTML document

(in this lab's case, relative to `lab2.html`).

- NOTE: AVOID starting a relative URL with a forward slash! Linux systems such as nrs-projects will treat those differently than you want.

- Its content should *describe* the document being linked to.

Knowing the above, decide on something you would like to link to (for example, a web site you like, or the CS 328 public course web page, or a document on the web, etc.) and:

- Decide where it makes sense to put this -- it can be in your current `p` element, in your current list element, or in a new `p` or list element that you add.
- Add an appropriate `a` element in the location you decide on, linking to your chosen site or file.

optional additions

- You may include additional elements if you would like, but note that strict-style HTML is a CS 328 style requirement for full credit.

before you go on

- Make sure your `lab2.html` works!

  That is: make sure it successfully displays when you paste its absolute URL from its opening comment into a browser!

validate your document

- NOW: add the following, to be able to VALIDATE your `lab2.html`, (and at least check to SOME extent if it is meeting strict style):
  - CHANGE the start tag of your `html` element to:

    `<html lang="en" xmlns="http://www.w3.org/1999/xhtml">`

  - WITHIN your `head` element, add this `meta` element:

    `<meta charset="utf-8" />`

  - SAVE a copy of your page as follows -- at the nrs-projects prompt, type the Linux command:

    `cp lab2.html lab2.xhtml`

  - Go to: **`https://html5.validator.nu/`**

    and paste in the absolute URL of the **`lab2.xhtml`** version of your file

    and click the "**Validate**" button.

    - SAD-BUT-TRUE THING: this validator seems to get overwhelmed frequently -- if you get an error "Bad Gateway 502", that's likely the culprit.
    - Wait a little bit, then try again. If necessary, let me know, submit what you have by the end of lab, and if you try later and find there was something you needed to fix, resubmit your fixed file and let me know, so that version will be graded instead.

- (if it validates, the driver should be set to submit their `lab2.html` --

  if not, **correct** this `lab2.html` and repeat the above until its `lab2.xhtml` copy successfully validates.)

Problem 2

Now, make a version of `lab2.html` in the navigator's `public_html` directory on nrs-projects. (This is to double-check yet again that everyone's nrs-projects account are appropriately set up for web pages!)

- You only need to make ONE change to this version of `lab2.html` -- it should have a comment containing the (absolute) URL I can use to view *this* version of `lab2.html`, the version in the navigator's nrs-projects account.
  - Again, double-check that the URL in this comment indeed works!

And, EACH of your should submit your version of `lab2.html` using `~st10/328submit` with a homework number of **82**.

Before you leave lab:

Make sure that you both have a file `lab2.html` with the correct URL in a comment for successfully executing your copy in your nrs-projects account, and that EACH of you submits this file.