CS 328 - Week 13 Lab Exercise - 2024-04-19

Deadline

Due by the end of lab today. (Submit whatever you have by the end of lab, even if incomplete.)

Purpose

To start learning about client-side JavaScript while also getting some practice using it.

How to submit

Submit your files using `~st10/328submit` on nrs-projects, with a lab number of **93**.

Requirements

- You are required to work in pairs for this lab exercise.
- Make sure **BOTH** of your names appear in **EACH** file you create for this lab exercise!
- When you are done, or before you leave lab, somehow e-mail or copy the lab exercise files so that BOTH/ALL of you have copies, and EACH of you should submit these files using `~st10/328submit` on nrs-projects, with a lab number of **93**.

Your Tasks

The NOW AVAILABLE zyBooks Chapter 7 - "JavaScript Fundamentals" will give an overview of the programming language features in JavaScript. But this text does not cover **unobtrusive-style JavaScript**, which I think it is important to expose you to.

You will walk through a small example that uses this style (and we'll discuss more about unobtrusive-style JavaScript and the DOM next week).

Task 1

Let's start with an HTML documemt with NO client-side JavaScript affecting it yet.

Copy over **`js-play1-start.html`** into the navigator's directory for today's lab, giving your pair's copy the name **`js-play1.html`**:

```
cp ~st10/js-play1-start.html js-play1.html
```

Edit this so that:

- the **`adapted by:`** part in its opening comment block contains **YOUR** names, and
- the **`PUT URL to YOUR COPY HERE`** in the opening comment block contains a **working URL** to your **`js-play1.html`**, and
- in the **`p`** element whose content starts with **`Experimenting...`**, replace the **`PUT YOUR NAMES HERE`** with **YOUR** names.

Make sure this URL works when pasted into a browser. You should see a page with a button labelled "Say the magic word!" that does nothing when clicked, and a labeled textfield that lets you enter stuff but, again, that's it.

Task 2

See, at the end of **`js-play1.html`**'s **`head`** element, the comment:

`<!-- ADD the parts described in the WEEK 13 LAB EXERCISE handout HERE -->`

This is where we will be putting JavaScript parts in CS 328 -- at the **END** of the **`head`** element, **after** all of your **`link`** element(s) and **before** the **`head`** element's **ending tag**.

For CS 328, you are expected to use only the following **two** types of tags for your JavaScript embedded within a document's `head` element:

- **`<script type="text/javascript">`**
  `//` JavaScript statements here
  `</script>`

or

- **`<script src="file-or-URL" type="text/javascript"> </script>`**

  ...for external JavaScripts.

(Note, then, that since the `script` element CAN have content, it is **NOT** a void element -- if it has no content, that's fine, but using strict-style it must still have an ending tag.)

So -- here's a quick first client-side JavaScript function: `alert` expects a string, and when executed, if the browser supports pop-ups, it will create a pop-up with the given string and an OK button.

In your **`js-play1.html`**'s **`head`** element, right after the comment:

`<!-- ADD the parts described in the WEEK 13 LAB EXERCISE handout HERE -->`

...add this **`script`** element, substituting a greeting including your names where indicated:

```
<script type="text/javascript">
    alert("greeting including your names");
</script>
```

Save, and try out your resulting `js-play1.html` from a browser -- if it has pop-ups enabled, you should see a pop-up with the greeting including your names each time you request or reload your `js-play1.html`.

Task 3

Next, you will create an **external JavaScript** file.

Create a file **`changeDoc.js`** with the following contents, replacing *YOUR NAMES* with your names.

The idea is for you to hand-type in at least the JavaScript statements below and consider what their effect is. (You are only **required** to include the **BOLDFACE** parts below -- the un-boldface parts are comments explaining some aspects of JavaScript.)

```
"use strict";  // this should be the FIRST line, to help finding errors

/*===
    NOTE: in an external JavaScript, you DON'T surround the contents
```

```
        with script tags! (important difference from PHP)
===*/

/*===
    adapted from "Web Programming: Step by Step", 2nd edition,
        pp. 235-238
    function: changeDoc: void -> void
    purpose: expects nothing and returns nothing, but has the
        side-effects of:
        *   causing a complaint to appear in a
            pop-up if the browser has pop-ups enabled and
            nothing has been entered in the Name textfield
            when this function is called

        *   flipping the content of two particular li elements

    by: Sharon Tuttle
    adapted by: YOUR NAMES
    last modified: 2024-04-19
===*/

function changeDoc()
{
    // calling the document's getElementById method
    //    to get the DOM child object corresponding to the
    //    element in this document with id="enteredName"

    let nameField = document.getElementById("enteredName");

    // for an input element of type="textfield", the corresponding DOM
    //     object has a datafield value whose value is the value of the
    //     textfield's value attribute -- same as the current contents of
    //     that textfield!

    // complain if no name entered in namefield currently

    if (nameField.value === "")
    {
        // you may change/customize this complaint if you would like

        alert("HEY! you were ASKED to enter a NAME!");
    }

    // get access to the DOM objects for two particular
    //     li elements
```

```
    let li1 = document.getElementById("cs-luminary-1");
    let li2 = document.getElementById("cs-luminary-2");

    // innerHTML is a DOM data field whose value is that element's
    //     content (between its start and end tag, remember?)
    // [it is fine to set innerHTML to text content,
    //     it is considered BAD STYLE to set innerHTML to *element*
    //     content...]

    // SWAP the contents of these two li elements

    let tempContent = li1.innerHTML;
    li1.innerHTML = li2.innerHTML;
    li2.innerHTML = tempContent;
}
```

Now, go to the next task to call this JavaScript function.

Task 4

Now, back in your **`js-play1.html`**, add the following two **`script`** elements AFTER your `script` element from Task 2. (Again, you are only required to type in the **BOLDFACE** parts below -- the un-boldface parts are comments explaining some aspects of JavaScript.)

```
    <!--
        this external JavaScript contains the function changeDoc
    -->

    <script src="changeDoc.js" type="text/javascript"
            defer="defer"></script>

    <script type="text/javascript">

        /*===
            it is considered good style to NOT muck with the DOM
            until the document has loaded;

            we will set the window object, representing the browser
            display, to have an onload attribute,
            so that desired document changes are not made until
            the document has been loaded

            that is, set the onload attribute of the window object so
            that it is a function that sets the event handling for the
            desired elements in this window when this window exists!

            And: JavaScript has anonymous functions!
                function()
                {
                    ...
```

```
            }
            ...can be the value of a JavaScript variable!
    ===*/

    // specify button's desired action when clicked

    window.onload = function()
                    {
                        let myButton =
                            document.getElementById("magicButton");

                        // add changeDoc -- the FUNCTION, NOT the
                        //     result of calling it -- as the value
                        //     for a button's onclick attribute

                        myButton.onclick = changeDoc;
                    };
</script>
```

Save, and try out your resulting **`js-play1.html`** from a browser.

- If pop-ups are enabled in your browser, then if you have not entered a name and click the button, you SHOULD get a pop-up complaint, and after you close that pop-up you should see that the list items' content has been flipped.
- And, if you do enter a name and then click the button, you just see that the list items' content is flipped again.

Submit your resulting **`js-play1.html`** and **`changeDoc.js`** with a lab number of **93**.

BEFORE you leave lab:

Make sure that you **both** have copies of **`js-play1.html`** and **`changeDoc.js`**, and ou BOTH submit these using `~st10/328submit` on nrs-projects, with a lab number of **93**.

How the navigator can get today's lab files:

- Because this happen to be a file the nrs-projects web server has to be able to reach, the navigator should be able to get a copy of these files 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 `328lab10`, and this sub-directory contains the files to be submitted for the lab exercise.
    - The NAVIGATOR `yz89` can now:
      - log in to THEIR nrs-projects account

```
cd public_html
mkdir 328lab13   # or other name they choose
chmod 711 328lab13
cp ~ab12/public_html/328lab13/js-play1.html . # note space & dot!
```

```
cp ~ab12/public_html/328lab13/changeDoc.js . # note space & dot!
```

  - And now the navigator `yz89` has their own copy of these files.

- I will leave it up to the navigator to decide if they would like to UPDATE their `js-play1.html` so their opening comment includes the URL to *their* copy, or if they want to leave the URL for the driver's copy.
  - HOWEVER: remember that you *will* lose some credit if this URL does not work when I or the grader paste your submitted file's URL into a browser, in either case.