CS 328 - Week 14 Lab Exercise - 2025-05-01/02

Deadline

Due by the end of lab.

Purpose

To practice using unobtrusive-style client-side JavaScript.

How to submit

Submit your files using **`~st10/328submit`** on nrs-projects, each time entering a lab number of **94**.

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, **BOTH** of you should submit appropriate versions of these files using **`~st10/328submit`** on nrs-projects, with a lab number of **94**.

Problem 1

The assigned zyBooks Chapter 7 - "JavaScript Fundamentals" sections 7.1, 7.7, 7.8, and 7.9 give an overview of programming language features in JavaScript. But this text does not cover **unobtrusive-style JavaScript**, which I think is important to know about.

We have discussed this and given an example in class, and this problem will walk you through another small example using this style.

Problem 1 part a

Start with the following HTML document 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
- it contains the **URL** that can be used to run your document.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- 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 whose content includes a

button labelled "Click me!" that does nothing when clicked, and a labeled textfield that lets you enter stuff but, again, that's it.

Problem 1 part b

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

`<!-- ADD the parts described in WEEK 14 LAB EXERCISE - PROBLEM 1 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 `</head>`**.

For CS 328, you are expected to use ***only*** the following **two** styles of tags for your JavaScript, which should appear ***only*** 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 that, for **external** JavaScripts, it is also USEFUL to add **ONE** of these ADDITIONAL attributes (written using the course-required strict style):

    `async="async"`

    `defer="defer"`

    - BOTH of these tell the browser it is SAFE to continue **parsing HTML** while this external JavaScript is being downloaded; (that is, both are assuring the browser you are using good modern practice of NOT modifying the DOM until the page is loaded)
    - **`async="async"`** means that this external JavaScript can be downloaded and at the same time as HTML parsing **and** other JavaScript downloads (for example, it does not depend on anything from a previous JavaScript)
    - **`defer="defer"`** means that, while it can be downloaded at the same time as HTML parsing, the JavaScripts need to be executed in the order their script elements appear (for example, because one uses a function from an earlier one)

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

Client-side JavaScript provides a function **`alert`** that expects a string and, when executed, IF the executing browser supports pop-up windows, it will create a pop-up window including the given string and an OK button.

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

`<!-- ADD the parts described in WEEK 14 LAB EXERCISE - PROBLEM 1 here -->`

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

```
<script type="text/javascript">
    "use strict";
    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`**.

Problem 1 part c

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** into your file 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

/*===
    SIDE 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, returns nothing, and has
        side-effects:
        *   causes a complaint to appear in a pop-up if
            browser has pop-ups enabled and Name textfield is
            empty when this is called

        *   flips the content of 2 of the list items

    adapted by: Sharon Tuttle and YOUR NAMES
    last modified: 2025-05-01
===*/

function changeDoc()
{
    // calls document object'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="text", the corresponding DOM
    //     object has a data field named value whose value is the
    //     current value of that textfield's value attribute,
    //     (which is the current content of that textfield!)

    // complain if nothing is currently in namefield

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

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

    // get the DOM objects for last 2 list items

    let li2 = document.getElementById("cs-luminary-2");
    let li3 = document.getElementById("cs-luminary-3");

    // innerHTML is a DOM data field whose value is that element's
    //     content (between its start and end tag)
    // [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 list items

     let tempContent = li2.innerHTML;
     li2.innerHTML = li3.innerHTML;
     li3.innerHTML = tempContent;
}
```

Now, go to the next task to **call** this JavaScript function **`changeDoc`** from **`js-play1.html`**.

Problem 1 part d

Now, back in your **`js-play1.html`**, add the following two **`script`** elements AFTER your **`script`** element from **Problem 1 part b**. (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

        defer="defer" is here to specify that this external JavaScript
```

```
        file needs to be loaded before the script after it is
        executed -- its function will be called in that next script
-->

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

<script type="text/javascript">
    "use strict";

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

        SO: 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!
    ===*/

    // 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 this 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 LAST two list items' content has been FLIPPED.
- And, if you DO enter a name and then CLICK the button:

  - you should JUST see that the LAST two list items' content is flipped again.
- Remember that you may be able to see **JavaScript error messages** in your browser's **JavaScript Console**.
  - For example, if you are using the Chrome browser, you can see Chrome's JavaScript Console using:

    View->Developer->JavaScript Console

Submit your files **`js-play1.html`** and **`changeDoc.js`**.

Problem 2

Note that external JavaScript files can contain more than function definitions -- they can contain JavaScript statements as well.

In this problem, you'll put **all** of your JavaScript in a file **`328lab14.js`**.

Problem 2 part a

- FIRST: copy the files from **`~st10/for-328lab14-ex`** into your nrs-projects Week 14 Lab directory:

```
cp ~st10/for-328lab14-ex/* .   // don't forget the space and period!
```

  - You should now have copies of the files:

    `328lab14.php`

    `lab14-functs.php`

    `lab14.css`
  - This is a small working two-state PHP postback application, currently with a button that does nothing and no client-side JavaScript.
- Edit your copy of **`328lab14.php`** so that its opening comment block includes **YOUR name(s)**, and the **URL from which YOUR version of this application can be run**.
  - Then -- run it!
  - Confirm that, between PHP and HTML, it has certain, ah, input requirements for its form.
  - Confirm also that the "MUCK..." button currently does NOTHING.

You should now be ready to start adding some client-side JavaScript to this application.

Problem 2 part b

The goal here is to add a little unobtrusive-style client-side JavaScript so that the "MUCK..." button does something.

- Create a new file **`328lab14.js`**, and put **EXACTLY** the following as its **FIRST** line:

```
"use strict";
```

  - This says to use strict-style for your JavaScript -- this can help you in finding errors.
- **AFTER** this, put a blank line and then **comment(s)** containing at least **your name(s)** and **today's date**.
- In this file, write a small function **`changeIt`** that expects nothing, returns nothing, but has the following side-effects as its actions:
  - it declares a JavaScript variable and sets it to the JavaScript DOM object corresponding to some

element in the page created by **`328lab14.php`**

- it **CHANGES** this element's content to a noticeable value of your choice -- if this function is successfully run, that element's displayed contents should change to what you put here.

- In this file, **AFTER** the function definition for **`changeIt`**, **also** put a JavaScript **statement** that sets the **`onload`** attribute of the JavaScript DOM **`window`** object to an **anonymous function** that does the following actions:
  - declares a JavaScript variable and sets it to be the JavaScript DOM object corresponding to the **`button`** element in **`328lab14.php`**
  - sets this button object variable's **`onclick`** attribute to be the function **`changeIt`**.
- Edit your **`328lab14.php`** so that, at the **end** of its **`head`** element, it includes:
  - the class-approved **`script`** element to include your external JavaScript **`328lab14.js`** -- and ALSO include the attribute **`async="async"`**, since it is safe to load and execute this external JavaScript while loading and parsing this document's HTML.
- See if your button in **`328lab14.php`** now changes your chosen element's content when it is clicked! Debug your JavaScript until it does.
  - Remember that you may be able to see **JavaScript error messages** in your browser's **JavaScript Console**.
  - For example, if you are using the Chrome browser, you can see Chrome's JavaScript Console using:

    View->Developer->JavaScript Console

Problem 2 part c

Fun JavaScript fact #1: JavaScript's **`String`** object has a method **`indexOf`** that expects a string to look for, and returns the index of its first occurrence in the calling string. It returns **-1** if the given string does not appear in the calling string.

Fun JavaScript fact #2: JavaScript has a modulo operator, **`%`** , that returns the remainder from integer division of its operands. So, for example, **`(13 % 3) === 1`**

The goal here is to now get a little bit of client-side JavaScript successfully validating a form (and **refusing** to submit it if it doesn't meet its validation requirements).

- In your file **`328lab14.js`**, write a second function **`meetsSpecs`**. This function expects nothing, tries to make sure some aspect or aspects of a form about to be submitted are OK, and returns **`true`** if they ***are*** OK, and returns **`false`** if they are ***NOT*** okay.
- DECIDE on **at least ONE** of the following aspects of this form you want to validate:
  - making sure that the user has entered a (non-empty) string **with NO blanks** in the first textfield, and/or
  - making sure that the user has entered an **EVEN** integer in the number field
- For each form widget you choose to validate, declare a JavaScript variable and set it to be the JavaScript DOM object corresponding to that form widget.
- Write an **`if`** statement to do the desired check of the value in that JavaScript variable's **`value`** data field for each form widget you choose to validate.
  - iI it **FAILS** that check, use the JavaScript **`alert`** function to print a suitable complaint in a JavaScript pop-up window, and **return `false`** to prevent the form from being submitted.

  - Otherwise, **return `true`**.
- In your file **`328lab14.js`**, in the anonymous function that you set to be the value of **`window`**'s **`onload`** attribute, now **also ADD** the following to what you have from **Problem 2 part b**:
  - declare a JavaScript variable and set it to be the JavaScript DOM object corresponding to the **`form`** element in the initial state of **`328lab14.php`**
  - what if we are here for this PHP application's second state? Then there will be **NO** such form, and the previous statement would set that JavaScript variable to **`null`**!
  - SO: write an **`if`** statement that checks to see **IF** this JavaScript variable is **NOT** equal to **`null`**, and only if it is **NOT** equal to **`null`** should it:
    - set the **`onsubmit`** data field for this JavaScript object representing the form to the function **`meetsSpecs`**.
    - (that is, we **don't** want to try to change a document element that **isn't** currently part of the document!

      But if this form **IS** part of the current document, add a little event hander to check its entered contents and only allow it to be submitted if it passes those checks.)
- See if trying to submit **`328lab14.php`**'s first state's form with a "bad" input into your chosen textfield now causes your JavaScript pop-up to display, and for the form NOT to be submitted!

  Debug your JavaScript until it does.
  - Remember that you may be able to see **JavaScript error messages** in your browser's **JavaScript Console**.
  - For example, if you are using the Chrome browser, you can see Chrome's JavaScript Console using:

    View->Developer->JavaScript Console

Submit your files **`328lab14.js`**, **`328lab14.php`**, **`lab14.css`**, and **`lab14-functs.php`**.

BEFORE you leave lab:

Make sure that you **both** have copies of the files:

- **`js-play1.html`** and **`changeDoc.js`**
- **`328lab14.js`**, **`328lab14.php`**, **`lab14.css`**, **`lab14-functs.php`**

...and you BOTH submit these using **`~st10/328submit`** on nrs-projects, with a lab number of **94**.

- I will leave it up to the navigator to decide if they would like to UPDATE their copies 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.