CS 1100: Web Development: Client-Side Coding / Fall 2016
Assignment 6: Rearrange the course schedule

Due: Tuesday, October 18, 2016, by 11:59 p.m.

Assignment Description

The purpose of this assignment is to practice the following:

- creating variables that have meaningful names;
- assigning values to variables;
- concatenating strings;
- accessing HTML elements and modifying their inner HTML content;
- designing a program that modifies a web page;
- and learning a new and useful CSS selector in the process.

Background. Your professor has realized he made a mistake when developing the schedule for this course. Initially he planned to cover HTML and CSS (which is 9 sessions) in five weeks. Now he has realized that five weeks is far too long a time to spend on HTML and CSS with such excellent and admirable students! One week should be enough.

Your task. Write a JavaScript program that rearranges the current schedule so that sessions 1, 2, 3 are offered on Mon. 08/22, sessions 4, 5, 6 are offered on Wed. 08/24, and sessions 7, 8, 9 - on Fri. 08/26. Here's a screenshot of what the result should look like (the only modifications are in the Sessions & Labs column in week one):

Schedule

Week	Date	Readings	Sessions & Labs	Homework
1	Mon, 08/22		Session 1: Introduction to the course Session 2: Internet & WWW: essential background Session 3: Introduction to HTML & CSS (part 1)	
	Wed, 08/24	Shay, chapter 1 (In Practice: optional)	Session 4: Introduction to HTML & CSS (part 2) Session 5: (lab assigned) Lab 3 part 1: Common CSS properties and selectors download lab files Session 6: Review of lists and tables; CSS box model	
	Fri, 08/26		Session 7: Introduction to CSS layout Session 8: Approaches to layout Session 9: Responsive design: Displaying web pages on any device	Homework 1 *available on eLearning*
2	Mon, 08/29	Shay, chapter 2 (In Practice: optional)	Session 3: Introduction to HTML & CSS (part 1)	
	Wed, 08/31	Shay, chapter 1 (review Working with Selectors) Shay, chapter 3 (Common CSS	Session 4: Introduction to HTML & CSS (part 2)	Homework 2

Implementation Plan

Download the provided file hw6.html. This web page references a JavaScript file, hw6.js (<script> tag at the bottom of the page). Your task is to create that file.

Here's an outline of the steps you need to take:

1. Figure out the CSS code to select the HTML elements, the content of which you'll need to access and/or modify.
2. Create 9 selectors for the elements that contain the 9 session descriptions and store them in variables.
3. Use the variables you've created to select these elements and store them in 9 other variables.
4. Repeat steps two and three for the 3 elements the content of which you will need to modify (week one).
5. Combine the contents of the 9 elements (containing session descriptions) into 3 strings and assign them to 3 variables. Use "
" to separate session descriptions.
6. Use these variables to modify the content of the 3 elements in week one.

Implementation Guidelines

Note: I will usually leave it up to you to design a solution. However, this is your first formal programming assignment, so I will guide you through it.

Make sure to use good and consistent variable names: this will help you a lot, especially in this assignment! The variable names I provide in this description are only examples of "templates" for the names.

Step 1

Often we don't have the luxury to select elements by their type, class or ID, especially when we write smart JavaScript programs that manipulate web page content. There are many powerful CSS selectors that can help us in these situations, and one of them is the **:nth-child(an+b)** pseudo class.

This selector matches a number of child elements whose numeric position in the series of children matches the pattern ***an+b*** (or just ***n*** when ***a=1*** and ***b=0***). For example, you could use it to select every fifth row in a table, every third cell in a row, or every forty-second paragraph on a page.

The querySelector method returns only the first occurrence of an element that matches our selector - which is exactly what we need. Consider this: if you look at the HTML code of hw6.html, you'll see that:

- to access the text "**Session 1: Introduction to the course**", you need to get at the **fourth** TD element in the **second** TR element (fourth cell in the second row of the only table on the page);
- to access the text "**Session 7: Introduction to CSS layout**", you need to get at the **third** TD element in the **eleventh** TR element (fourth cell in the second row of the same table);

How do we select the second table row? As simple as this:

```
tr:nth-child(2)
```

How do we select the fourth table cell? As simple as that:

```
td:nth-child(4)
```

But how do we select the fourth table cell ***in*** the second table row? You already know! a TD is always a decsendant of a TR - so we can use the descendant CSS selector:
https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_selectors

You've used it not so long ago! Recall selecting only *those* list items that belonged (i.e., were descendants) to a specific unordered list and not any list on the page.

To save time, use the browser console to test your selectors.

If you are curious, here's more information on nth-child selector:
https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child
http://www.w3schools.com/cssref/sel_nth-child.asp

Step 2. Create CSS selectors to access 9 **table cells** that contain session descriptions (their content starts with the word "Session"; most of them contain only the session number and topic, except the cell for session 5 which also contains text about a lab). These cells appear in rows 1, 2, 4, 5, 8, 10, 11.

Assign each selector to a variable with a meaningful name; this might look like this:
```
var selectorForA = "a CSS selector that you will use to select element A";
```

Here's a real example:
```
var pSelector = "p";
```

Thus, these 9 variables will contain **strings**.

Step 3. Use the variables you've just created as selectors to access the 9 table cells (TD elements) using the querySelector method of the document object.

Remember that querySelector returns **an object** that ***represents*** an HTML element and provides access to lots of useful methods and properties, including **innerHTML** - used to access or modify the innerHTML of that element!

Assign these 9 elements to 9 other variables with meaningful names, like this:
```
var elementA = document.querySelector(selectorForA);
```
where elementA is a meaningful name for a variable holding element A, and selectorForA is the CSS selector you used to find that element A in your HTML.

Thus, these 9 variables will contain **objects**.

Now you have access to 9 objects that contain (among other things) the HTML you will need to copy into the table cells for week one.

Step 4. Now do the same as you did in steps two and three to get access to the three "target" cells: the TD elements in week one, the contents of which you need to modify. But notice that you already have access to the first two cells from step 3! So you only need one more - for the Friday cell.

Your new variable will contain an **object**.

Step 5. Build up the HTML for each of the target cells.

We have 9 objects (from step three) which provide access to the HTML we need. Let's combine that HTML into 3 strings. Use "
" to separate session descriptions. For example:

var newHTML = text-from-element-1 + "
" + text-from-element-2 + "
" + text-from-element-3;

These 3 variables will contain **strings**.

Step 6. Use the innerHTML method to modify the content of the cells in week one. For example, you might write:

```
targetElementA.innerHTML = newHTML;
```

Submit your work

Submit your **hw6.js** file to eLearning. You do not need to submit anything else.

Grading

This assignment is worth **43 points** (which accounts for approximately **2.7% of your grade**).

1. Step 1: selectors are correct **(10 points)**
2. Step 2: functionality is correct / guidelines followed **(5 points)**
3. Step 2: good variable names **(2 points)**
4. Step 3: functionality is correct / guidelines followed **(5 points)**
5. Step 3: good variable names **(2 points)**
6. Step 4: functionality is correct / guidelines followed **(5 points)**
7. Step 4: good variable names **(2 points)**
8. Step 5: functionality is correct / guidelines followed **(5 points)**
9. Step 5: good variable names **(2 points)**
10. Step 6: functionality is correct / guidelines followed **(5 points)**

What's a good variable name? Check your readings, and sessions slides. Also, consider this: a good variable name helps us understand the program. Your program has variables that contain data that appears to be similar, but is very different: you will have variables that contain:

- a selector for selecting element X
- the HTML contents of element X
- an object that represents element X

...and many of those!
Your variable names should ***consistently*** **distinguish** between these different types of values that they hold.

To earn credit, your program should work. If your program does not work at all (i.e., does nothing to the page), I will only give partial credit for correct selectors in step 1 (if they are correct).