

Class:	COSC 231.10 - CRN: 11664 - TUE/THU 13:00-14:50
Lab Number:	04 (Part 04)
Due Date:	2023-10-28 SAT T 23:59:00 Z-04:00
Submission:	Upload zipped/compressed source code to Canvas
Topic:	JavaScript functions, show/hide HTML elements (via CSS and JS), floating elements
Updated:	2023-09-27 T 07:46:56 Z-00:00

Preface:

Lab 04 is a multi-part lab.

Part 00 through Part 04 are a progression of lab exercises designed to build a demo website, piecewise, and in a fashion that uses automation is much as possible. Part 00 and Part 01 focus on using a script/program to generate multiple pages as uniformly as possible. Part 02 introduces CSS styling to the site and adds a main/home/index page for the site with a uniform navigation menu across all site pages. Part 03 adds three additional specialty pages to the site (one page each for maps, charts, and weblinks) and adds a little more styling. As the class continues to learn new material, Part 04 and beyond continuously add CSS styling and JavaScript functionality via external files to improve features of the site.

Each of these lab part could pose a challenge from one week to the next and in a manner that could introduce either missing code or errors that could carry through to all future lab parts. While the student must learn how to minimize errors in similar development scenarios, it is not the goal of the instructor to penalize the student more than once for their errors; therefore, each new part has a mechanism to avoid this and keep project development as consistent as possible throughout the lab series.

Each of these of these lab parts offer a proposed solution that is available to view after the submission deadline. If for any reason you are not satisfied with your submission for any particular lab part, you are welcome to use the proposed solution as the basis for the next lab (e.g., if you completed just half of a lab part or you did something incorrectly, you are welcome to use the proposed solution to the lab part as the basis for the next lab part in order to keep your work as similar to the overall project requirements through to the end of the series of labs).

It is the hope of the instructor that the student continues to use automation to generate the site pages of these lab parts, as much as possible, from beginning lab part 00 to the final part. While a student could simply grab a lab part solution and start adding code for the next part, it is far preferable to take the time to incorporate the text in the files into the automation program that was started in Lab 04 parts 00/01 and continuously improve upon it to produce uniform, machine-generated HTML/JS/CSS files.

In other words, if you are not happy with the outcome of a lab part, please feel free to use the proposed solution file to one part as the basis for the next part, but it is considered wiser not to use those files directly (instead they should be incorporated, as much as possible, into a program you write that generates equivalent files).

Description:

Whereas the focus of Lab 04 Part 03 added more pages and styling to the overall site, the focus of Lab 04 Part 04 turns to the use of JavaScript functions to alter the CSS of a document in order to show/hide the navigation menu (on every page) and the lengthy, hourly data tables in the monthly pages. Although every HTML file in Lab 04 Part 04 is to be modified (from Part 03), the changes are all rather slight.

In the realm of CSS, seven rules are added to the `wolfcreek.css` file. A few of those rules regard overall styling on each page, but most of these styling rules target specific elements within the navigation menu. These rules add some visual style to the nav menu. In conjunction with the JavaScript functions that are added to this lab, they help make allow the user to toggle the nav menu between visible and invisible. For the most part, it is JavaScript functions that are doing this hide/show toggling, and their function calls are enabled simply by adding some attributes to a few elements in the HTML.

On the backend (hidden from the viewer's browsing), the JavaScript functions are the more important ingredients of this lab. Lab 04 Part 04 requires the next file for this project – `wolfcreek.js` – to be placed in the previously created `js/` subdirectory.

The next section describes additions to the `<head>` of each HTML page. After that, the following section describes additions to the uniform site navigation menu found at the top of each HTML page. Next, there is a description of a minor addition to the body of each of the “monthly” pages. After the HTML descriptions, there is a description of additional styling that must be added to the CSS file. Finally, there is a description of the three functions to be added to the new `wolfcreek.js` file.

Updates to the `<head>` (in all pages):

In the `<head>` of all sixteen HTML pages, add a script link to the external JavaScript file `wolfcreek.js` (make sure you remember what sub-directory the JS file is in).

Updates to the site navigation menu (in all pages):

In the navigation menu on each of the sixteen HTML pages, locate the button with id “`nav_button`” and add the `onclick` attribute with the value being a call to a function call that is, the function labeled “`showhide_nav_menu`”; pass “`this`” as the single argument to the function call. The function definition will be described later in this document in the JavaScript section.

Required modifications to the 12 “monthly” pages:

Each of the sixteen “monthly” pages have a button the page with a unique id that is labeled “`showhide_button_01`” through “`showhide_button_12`” (notice that each of these buttons, one per page, also has a class “`major_format`”). Locate this button on each of the twelve pages and add an `onclick` attribute with the value being a call to a function call that is, the function labeled “`showhide_table`”; pass “`this`” as the single argument to the function call. The function definition will be described later in this document in the JavaScript section.

CSS (external) styling to add to **wolfcreek.css**:

For Lab 04 Part 04, you are to add seven new rules to your stylesheet **wolfcreek.css**. Some of these rules have just a single selector for the corresponding declaration block; however, two of the rules have multiple selectors to which the declarations (CSS rules) will apply.

- Rule 01: to any element that contains the id “**top**”, add the following declaration:
 - Set the bottom padding to 300 pixels.
- Rule 02: to elements that contains the class “**navbarouter**”, add the following declarations:
 - Set the overflow property to hidden.
 - Set the position property to fixed.
 - Set the top property to 4%.
 - Set the left property to 4%.
- Rule 03: to anchor elements that are descendants of any element that contains the class “**nav_list**”, add the following declarations:
 - Set the float attribute to left.
 - Set the display attribute to grid.
- Rule 04: to any element that contains the class “**nav_list**”, add the following declaration:
 - Set the clear attribute to left.
- Rule 05: to paragraph elements that are descendants of any element that contains the class “**nav_list**”, add the following declarations:
 - Set the color to **#f2f2f2**.
- Rule 06: to anchor elements that are descendants of any element that contains the class “**navbar**”, add the following declarations:
 - Set the color to **#f2f2f2**.
 - Set the text alignment to center.
 - Set the top and bottom padding to 10 pixels.
 - Set the right and left padding to 12 pixels.
- Rule 07: to anchor elements with a pseudo class “**hover**” that are descendants of any element that contains the class “**navbar**”, add the following declarations:
 - Set the background attribute to **#ddd**.
 - Set the color attribute to the color black.
 - Set the border to 1 pixel solid **#3f3f3f**.

Note: to keep your CSS organized, consider adding a comment before and after all your rules for this particular lab that indicate these are the styling rules added for Lab 04 Part 03. This ensures that you can keep track of issues, should any arise in subsequent labs with your styling.

This concludes the description of additions to the **wolfcreek.css** document.

JavaScript functions (external) to add to wolfcreek.js:

There are three functions to add to the (newly-created) external `wolfcreek.js` file. One of them is rather trivial and the other two are somewhat more complex. Of the two more complex functions, they are performing very similar actions albeit with slight differences.

The first function is labeled `print_stars` and takes no arguments. Its body is simply the following single statement: `console.log("*****");`

The second function is labeled `showhide_table` and has one parameter called `elt`. The third function is labeled `showhide_nav_menu` and also has one parameter called `elt`. Both the second and third function perform similar tasks. Both of their single arguments, when called in a page, are passed `this` as the argument, which passes the object (element) that called the function, and that object which is passed is then referenced in the function via its parameter name `elt`; thus, `elt` in both functions is a reference to the object which called the function (a button, in both cases).

Specific requirements to incorporate into the second and third functions:

To get started with writing, calling, testing, debugging and employing these two function, a few statements are mandatory to add to each function. The two functions `showhide_table(elt)` and `showhide_nav_menu(elt)` must both perform the following tasks:

- The first three statements in each function must be the following:

```
print_stars();  
console.log("DEBUG - Before: " + elt.innerHTML);  
console.log("DEBUG - ID: " + elt.id);
```
- The last two statements in each function must be the following:

```
print_stars();  
console.log("DEBUG - After: " + elt.innerHTML);
```

For both functions – the `showhide_table(elt)` and the `showhide_nav_menu(elt)`, you will be leveraging this reference to the button element in the page to show/hide portions of the page. In the case of the `showhide_table(elt)` function, you are employing the function in the twelve “monthly” pages to show/hide the lengthy table of hourly data on the page. In the case of the case of the `showhide_nav_menu(elt)` function, you are employing this function in all sixteen pages of the site to show/hide the navigation menu. One additional feature of these two functions is that a call to the function (which will show/hide a menu or a table) will also change the text label on the button itself – the `innerHTML` string of the button; your function will be toggling the text back and forth (with each click of a button) from “Show Table” to “Hide Table” and back to “Show Table” (for `showhide_table(elt)`), and from “Show Menu” to “Hide Menu” and back to “Show Menu” (for `showhide_nav_menu(elt)`).

JavaScript functions (external) to add to wolfcreek.js (continued):

The bodies of both of these functions may be generally categorized as having two parts. In the first part of both functions, you are leveraging your reference to `elt` to get further references to other elements in the document so that their visibility may be toggled (i.e., to hide/show them). In the second part of both functions, you are employing a conditional `if-else` that is simply testing the current `innerHTML` string value of the button (`elt`) to see if the button is displaying (when it was clicked) “Show Table”/“Show Menu” or “Hide Table”/“Hide Menu”. If a button’s label is presently displaying “Show Table/Menu”, then you know that the table/menu is presently hidden; if a button’s label is presently displaying “Hide Table/Menu”, then you know the table/menu is presently showing/visible. It’s in this second part (the `if-else` conditional code blocks) that you are hiding/showing menus/tables via references to various elements that you created in the first part of each function. So, in other words, the second part of the functions are making things appear/disappear, and it’s the first part of the function which is simply establishing the object references to those objects/elements whose visibility needs to be toggled.

The aforementioned mandatory debug statements aside, the two functions will now be described individually, in greater detail (albeit omitting reference to the debug statements in their description).

Specific requirements for the second function - `showhide_table(elt)`:

In this function, there are two things you are toggling via the `if-else` blocks. The first thing you are toggling is the label on the button itself – back and forth between “Show Table” and “Hide Table”. The second thing you are toggling is the long table of hourly readings in the page. Unfortunately, the button and the table are not located together inside a parent `<div>` element; however, the table is inside a `<div>` element, and that `<div>` is the next element document, so you can get a reference to this `<div>` via the DOM `nextElementSibling` property of the button. With a reference to the `<div>`, you can then get a reference to its child element – the table.

In this first half of this function – setting up the references, you must create a JavaScript variable called `btn` that stores a reference to the `id` of the object `elt`. (thus, `btn` stores a reference to `elt` and you shall get this via `elt.id`). Next, create a JS variable called `div` that stores a reference to the `<div>` element after/below the button on the page (this is where `nextElementSibling` comes into use). With the variable `div` storing a reference to the element, create a third JS variable called `table` which, as the name suggests, is a reference to the table that is the child element of the `<div>`. Finally, create a JS variable called `label` and initialize it to the current value of the button’s `innerHTML`.

In the second half of this function – the `if-else` blocks, you perform the actual toggling of two of the four variables mentioned above. The `if` conditional Boolean must test if the label is equivalent to the string “Show Table” (which means the table is presently hidden). The `if`-block has two statements in it. In the first statement, set the button’s `innerHTML` to “Hide Table”. In the second statement, set the visibility of the table to “visible”. The `else`-block also has to complimentary statements. In the first statement, you must set the button’s `innerHTML` to “Show Table”. In the second statement, set the visibility of the table to “hidden”.

JavaScript functions (external) to add to `wolfcreek.js` (continued):

Specific requirements for the third function - `showhide_nav_menu(elt)`:

In this third function, you are faced with a similar issue. You must find a way to reference the nav menu via the reference that you have to the button object in `elt`. While the nav menu does have a single outer `<div>` element (the one with id “`navmenu_div`”) that is a parent to both the button and the nav menu portions that we wish to hide/show, we will not be using that parent div to make our magic. Instead, we will employ in this function the same technique that we did in the second/previous function of using the button’s DOM `nextElementSibling` property to access the `<div>` that we wish to hide (the one with id “`navbar`”).

In the first half of this function, you must create a variable `btn` that stores a reference to the `id` of `elt`. (`btn` is a reference to `elt` via `elt.id`). Next, create a second variable `navbar` that stores a reference to the `<div>` element after/below the button on the page (this is where `nextElementSibling` comes into use). Next, create a third variable `label` and initialize it to the current value of the button’s `innerHTML`. Finally, create a fourth variable `btn_parent`, which is a reference to the button’s parent (id “`navmenu_div`”).

In the second half of this function – the if-else blocks, you perform the actual toggling of all four of the variables mentioned above. The if conditional Boolean must test if the label is equivalent to the string “`Show Menu`” (which means the menu is presently hidden).

Add the following lines of code to the if-block:

```
btn.innerHTML = "Hide Menu";
navbar.style.visibility = "visible";
btn_parent.style.backgroundColor = "#989A98";//#navmenu_div
btn_parent.style.opacity = .9;//#navmenu_div
btn_parent.style.border = "1px solid grey";//#navmenu_div
btn.style.backgroundColor = "#a07574 ";
btn.style.border = "1px solid #3f3f3f";
```

Add the following lines of code to the else-block:

```
btn.innerHTML = "Show Menu";
navbar.style.visibility = "hidden";
btn_parent.style.backgroundColor = "transparent";
btn_parent.style.border = "0";//#navmenu_div
btn_parent.style.opacity = 1;//#navmenu_div
btn.style.backgroundColor = "#5f8a8b";//Steel Teal
```

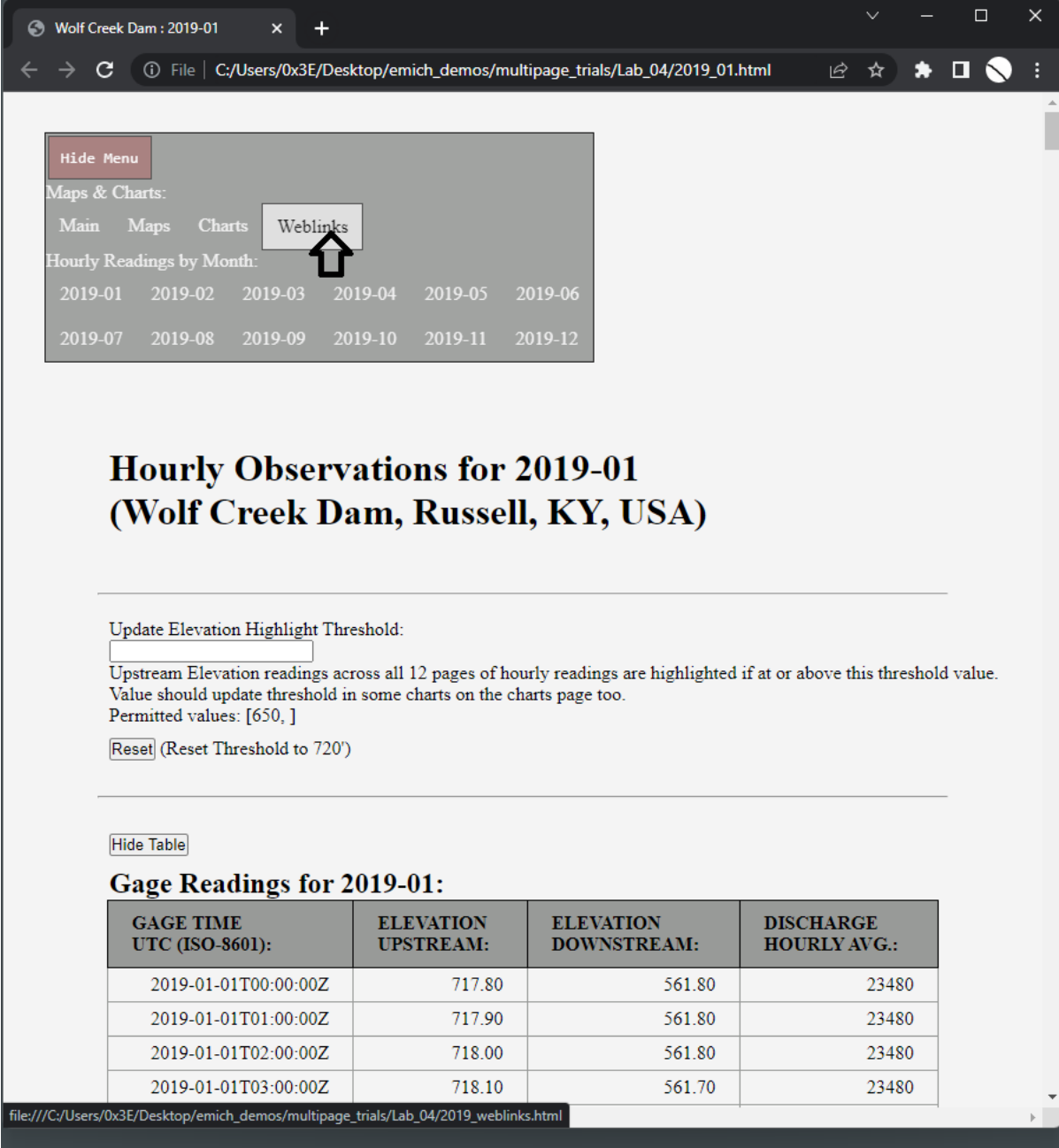
Note: keep your JS organized; consider adding comments before/after your 3 Lab 04 functions. This ensures that you can keep track of issues, should any arise in subsequent labs.

Note: it is imperative that to add comments before function definitions which explain the function in general (perhaps explaining pre-/post-conditions, possibly return values, etc.) and in the function bodies, describing what particular lines or blocks of code do, what elements a code blocks or statement might be referencing, etc.

This concludes the description of additions to the `wolfcreek.js` document.

Screenshots:

The following screenshot exhibits the hover effect when the mouse hovers over one of the sixteen anchors in the nav menu.



The screenshot shows a web browser window with the title "Wolf Creek Dam : 2019-01". The address bar displays the file path: "C:/Users/0x3E/Desktop/emich_demos/multipage_trials/Lab_04/2019_01.html". The main content area features a navigation menu with a "Hide Menu" button and a list of anchors: "Main", "Maps", "Charts", and "Weblinks". The "Weblinks" anchor is highlighted with a white background and a black border, and a mouse cursor is hovering over it. Below the navigation menu, there is a section titled "Hourly Observations for 2019-01 (Wolf Creek Dam, Russell, KY, USA)". This section includes a form for updating the elevation highlight threshold, a "Reset" button, and a table of gage readings for 2019-01.

Hourly Observations for 2019-01
(Wolf Creek Dam, Russell, KY, USA)

Update Elevation Highlight Threshold:

Upstream Elevation readings across all 12 pages of hourly readings are highlighted if at or above this threshold value. Value should update threshold in some charts on the charts page too.
Permitted values: [650,]
 (Reset Threshold to 720')

Hide Table

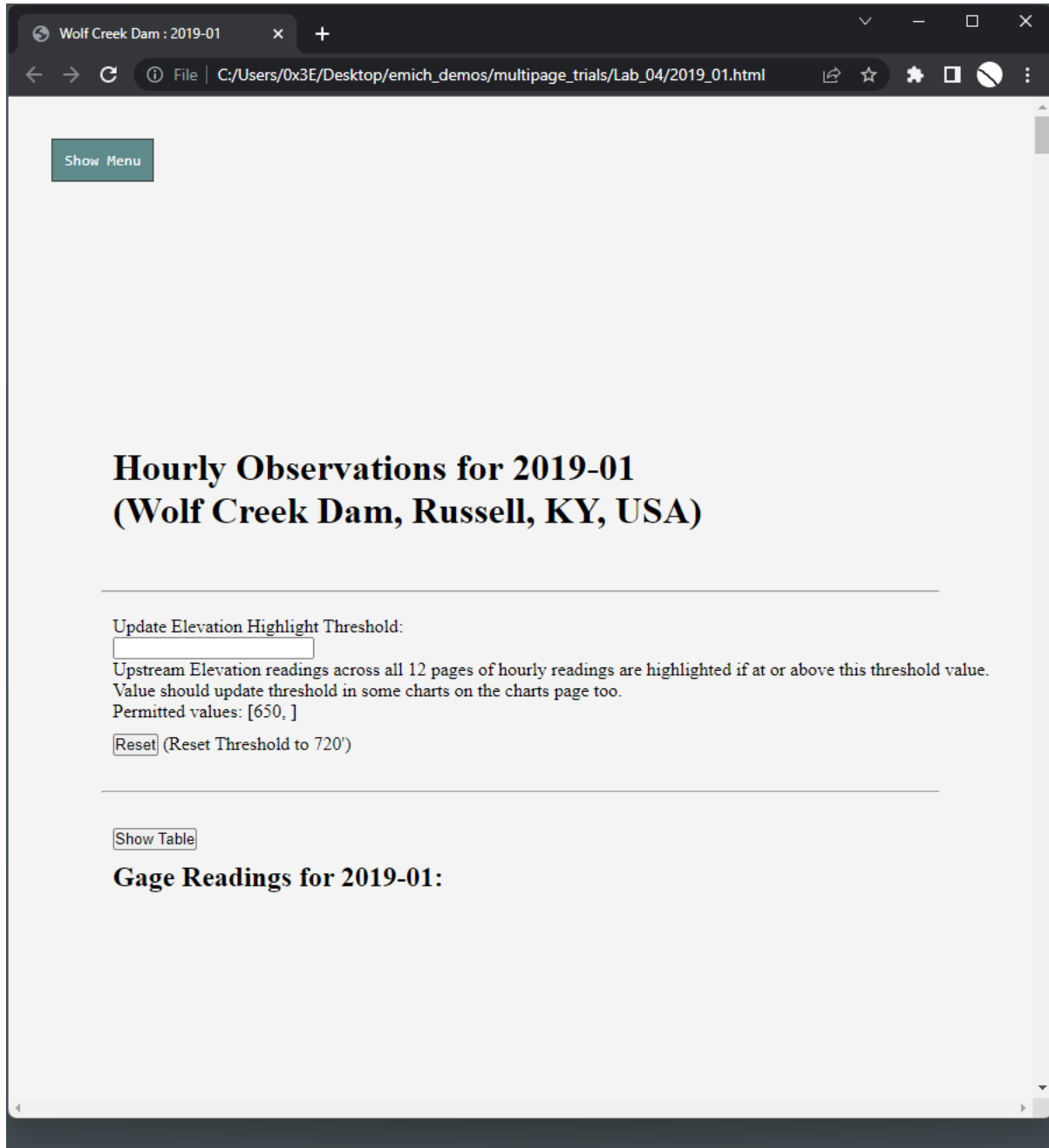
Gage Readings for 2019-01:

GAGE TIME UTC (ISO-8601):	ELEVATION UPSTREAM:	ELEVATION DOWNSTREAM:	DISCHARGE HOURLY AVG.:
2019-01-01T00:00:00Z	717.80	561.80	23480
2019-01-01T01:00:00Z	717.90	561.80	23480
2019-01-01T02:00:00Z	718.00	561.80	23480
2019-01-01T03:00:00Z	718.10	561.70	23480

file:///C:/Users/0x3E/Desktop/emich_demos/multipage_trials/Lab_04/2019_weblinks.html

Screenshots (continued):

The following screenshot exhibits a typical monthly page when both the menu is hidden (or appears collapsed) and the hourly table data is hidden.



Note: for more information about the behavior of these buttons hiding/showing menus/tables, please see the video description for Lab 04 Part 04 on Canvas.

Note on Using Comments:

In comments at the top of your JavaScript file, and somewhere within the `<head>` element of each of your documents are comments that look similar to below; please ensure that you have finished commenting with your full name, EID and a proper description and any requirements specification:

Class: COSC 231.10

Name: *Your Full Name*

EID: *Your EMU EID*

Lab: 04 (Part 04)

Date: 2023-10-28 SAT T 23:59:00 Z-04:00

Requires: [write any sub-dir path/file/system requirements this lab is dependent on]

Description: *Provide a description of what this file is/does/displays.*

Furthermore, you must add any helpful comments into the body of your code that help describe it. For example, because tables and their content can get messy to read, use whitespace/newlines and comments to delineate the lists that are in the table.

```
<table>
  <tr>
    <td>blah blah blah</td>
    <td>
      <!-- list of my favorite fruits -->
      <ul>
        <li>apples</li>
        <li>oranges</li>
      </ul>
      <!-- list of my favorite fruits -->
    </td>
  </tr>
</table>
```

Important Notes:

Use proper indenting of elements. I do not care whether you use tabs for indentation or some number of spaces (e.g., 2 or 4 or 5 or 8), but be consistent throughout your document. Child elements should be indented more than their parent element. The reason I don't care about how much spacing you use for indentation is because the web-programming world has some even more divergent opinions on readability and white-spacing, so I'll leave it to you to find your own style/preference.

Note on Directory and File Names:

The directories and files you zip/archive/compress to submit are the following:

```
./css/wolfcreek.css
./js/wolfcreek.js
./2019_01.html
./2019_02.html
./2019_03.html
./2019_04.html
./2019_05.html
./2019_06.html
./2019_07.html
./2019_08.html
./2019_09.html
./2019_10.html
./2019_11.html
./2019_12.html
./2019_charts.html
./2019_maps.html
./2019_weblinks.html
./index.html
```

Submission:

Zip just the aforementioned files (and only relevant sub-directories specified – i.e., the directories for JS, CSS) and upload to Canvas.

For Part 03, the archive/zip/etc must be called:

`COSC231_10_202401_Lab_04_Part_04_{Your_Surname}.zip`

Grading:

1. (10%) Correctness of HTML elements – i.e., did you use all described elements?
2. (40%) Correctness of JavaScript functionality – i.e., do the functions work correctly?
3. (35%) Correctness of CSS styling.
4. (15%) Proper commenting and consistent pattern of indentation.