JavaScript Assignment 1

420-320-DW

Prepared by Nasreddine Hallam, Modified by Myles Burgess

Synopsis

The purpose is to dynamically construct and add elements into a web page using JavaScript. Specifically form and table elements. The form elements will be used to get user input to determine the structure and styling of the table elements.

Objectives

Using flex to layout the GUI

Coding using JS DOM structure

Coding using JS Event handling

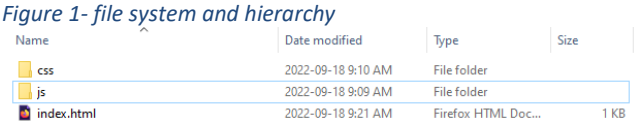
Coding using object literals

Keywords: html, CSS, JavaScript, DOM, event programming, object literal, HTML collection, array.

Part1 - Project files system and hierarchy

- If you have not already done so, create a course folder for this JavaScript course on your H: drive (or any drive on your Personal computer, depending on where you normally edit your files).
- Inside the course folder create a folder for Assignment1.
- Inside Assignment1 folder, create the directories css and js.

Figure 1- file system and hierarchy



Part 2 - HTML and CSS

- The HTML file pictured in Figure 2 has been supplied to you.
- You need to complete the head section. **You cannot modify the HTML document in any other way.**
- All CSS must be in an external stylesheet. **No inline or embedded styles are allowed.**
- All JavaScript must be in an external document. **No inline or embedded JS code is allowed.**

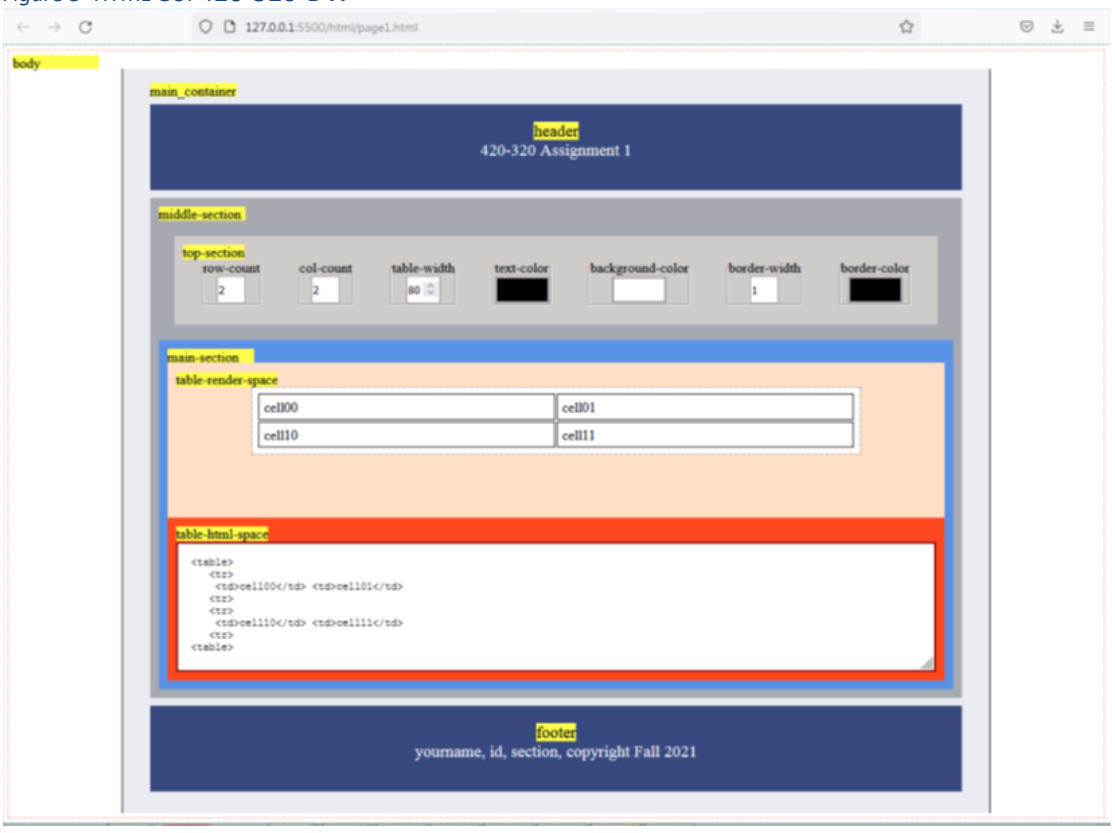
Figure 2- main HTML GUI

- You are required to style the page and should refer to Figure 3 below for guidance
- Write you CSS-rules in a file named style.css. The colors are approximate, but layouts must be observed.
- You must use flex layout. **You cannot use floating and positioning.**

IMPORTANT: The yellow (stickers) in figure 3 identifying each container are there for reference only, do not code them.

- The section **main_container**:
  - 80% wide and centered.
  - Contains three sections: header, **middle-section**, footer.
  - Header and footer are self-explanatory
- Section **middle-section**: contains two sections: top-section and main-section.
  - Section **top-section**: will contains the form for user interaction.
  - Default values for number of rows, number of columns, and table-width are shown below in Figure3.
  - Default colors for text-color, table-background and border-color are black, white and black respectively.
  - These form elements (widgets) will be dynamically generated in your JavaScript.
- Section **main_section**: has two sections: table-render-section and table-html-section.
  - Section **table-render-section**: once a form element has its value set/changed or loses the focus (up to you to determine the right event), a table is dynamically built based on rows and columns, some css settings applied, and the whole table is rendered (visualized) in this section.
  - Section **table-html-section**: each time a table is created and rendered in the previous section, the corresponding table html code is generated and displayed in a "textarea" of this section.

Figure 3- HTML GUI 420-320-DW



Part 3- The JS part

- **Form Generation:** The first part of the JavaScript program will deal with building the form seen in figure 3 in the section called top-section.
  - Store all the attributes required by the various form elements (input tags) in an array of object literals. Cycle through the array and pass the attributes to the createFormElement() function. The following is a list of the attributes that each form element should have:
    - row_count - type (text), id(row_count), size(2), value(2)
    - column_count - type (text), id(col_count), size(2), value(3)
    - table_width - type (number), id(table_width), size(3), value(100, min(2), max(100)
    - text_color - type (color), id(text_color), size(7), value(#000000)
    - background_color - type (color), id(background_color), size(7), value(#FFFFFFF)
    - border_color - type (color), id(border_color), size(7), value(#000000)
    - border_width - type (text), id(border_width), size(2), value(1)
  - **createFormElement() (Required)** This function should be called from the main function and be passed the attributes required to make one of the form elements. It will build the element and then return the constructed element back to the calling function where it will be appended to the form.

- **Table Generation:** The program will require a function that will build a table dynamically based on the values stipulated in the form. This function will be called from the main function the first time and use the default values of the form to build and render the table. Later on this same function will be called by events that occur when a value in the form is changed (more on events later) in order rebuild and render a new table based on the new form values.
    - The row_count form field will determine how many rows need to be added to the table
    - The col_count form field will determine how many columns to place inside each row of the table
    - The table_width form field will determine the width of the table as a percentage of the container it will occupy
    - The text_color form field will determine the color of the text used inside the table
    - The background_color form field will determine the background color of the table
    - The border_color form field will determine the border_color of the table and the td elements of the table
    - The border_width form field will determine the width of the border lines for the table and the td elements of the table
  - **drawTable()** (Required) This function will initially be called from the main function to build the table with the default values in the form. Subsequently it will be called by events triggered by changing the values of the form. This function must do the following:
    - Dynamically create a table element along with the row and td elements of the table
    - Add cell contents to each td element in the format celli,j where i is the row index and j is the column index (example: the first cell in the first row will have the text cell00, the second cell will have the text cell01)
    - Render the table in section table-render-section of the web page
    - Style the table based on the values from the form
    - Generate a string containing the table html code(which has proper indentation)
    - Build a new textarea element containing the string as a text node
    - Add the new textarea element to the section **table-html-section**
    - Whenever the table is redrawn the old table and textarea elements should be removed from the web page and then replaced with the newly constructed elements

- **Events: The program will require events that will change the table design when any of the values in the form are manipulated.**
    - On either row_count or col_count you need to use a keyup event, you determine which one.
    - Except for the one element using the keyup event above, all other form elements should all have change events.
    - Up to you to determine if each element will have a registered event or if you use event delegation.
    - Up to you if each element will have an event handler or if event handlers will deal with multiple events.
    - Whenever a single form element is modified you must redraw the new version of the table
    - You must use event listeners. **You cannot use property events.**

- **Miscellaneous**
    - You must comment your JS code

Non-coding requirements

This is an INDIVIDUAL assignment, which must be done individually (sharing code is considered plagiarism, a form of cheating)

Submissions

- This assignment is due: September 27, 2022, at day-end
- Place all the files used to make your Web site into a compressed folder and Submit the compressed folder to LEA assignment 1.

    To place your Files into a compressed folder, open File Explorer, select all the files and/or folders that make up your web site, then right click on the highlighted area, then place your mouse over **Send to** from the drop-down menu and then select **compressed (zipped) folder** from the side menu. When asked to name the folder, change the default name supplied to Your name and assignment number.

    Example: Myles_Burgess_Assignment1. You must place all of your files and folders in a windows compressed folder **not a zip that you generate** using Windows zip or any other compression program you have.

Appendix

The initial CSS style for the table should be as follows:

```
table{
   border: 1px dotted black;
   padding:5px;
   background-color:white;
   margin: 0 auto;
}

td{
   border: 1px solid black;
   padding:5px;
}
```