ENSF-381: Full Stack Web Development Laboratory

Ahmad Abdellatif and Novarun Deb
Department of Electrical & Software Engineering
University of Calgary
Lab 3

Objectives

Welcome to the ENSF381 course lab! Below are the detailed instructions for creating and understanding CSS. The main objective of this lab is to understand how CSS attributes affect the layout and aesthetics of a webpage and to practice implementing styles systematically.

Groups

Lab instructions must be followed in groups **of two students**.

Submission

You must submit the complete source code, ensuring it can be executed without any modifications. Also, if requested by the instructor, you may need to submit the corresponding documentation file (e.g., word and image). Only one member of the group needs to submit the assignment, but the submission must include the names and UCIDs of all group members at the top of the code.

Deadline

Lab exercises must be submitted by **11:55 PM on the same day as the lab session**. Submissions made within 24 hours after the deadline will receive a maximum of 50% of the mark. Submissions made beyond 24 hours will not be evaluated and will receive a grade of zero.

Academic Misconduct

Academic Misconduct refers to student behavior which compromises proper assessment of a student's academic activities and includes: cheating; fabrication; falsification; plagiarism; unauthorized assistance; failure to comply with an instructor's expectations regarding conduct required of students completing academic assessments in their courses; and failure to comply with exam regulations applied by the Registrar.

For more information on the University of Calgary Student Academic Misconduct Policy and Procedure and the SSE Academic Misconduct Operating Standard, please visit: https://schulich.ucalgary.ca/current-students/undergraduate/student-resources/policies-and-procedures

Exercise 1: Styling a Webpage using CSS

Objective: In this exercise, we will learn to use CSS classes and selectors effectively to create reusable and modular styles, promoting efficiency and maintainability in web design. We will style a given HTML page to match a predefined visual design of the UoC history webpage: https://ucalgary.ca/about/our-history

1. Copy the HTML code provided in the supplementary material into your editor (e.g., VS Code, Sublime Text). This code gives you the structure of the webpage, but it does not include any styling beyond basic HTML defaults. You'll add the styles yourself.
2. Create a CSS file and save the new file named styles.css in the same directory as your HTML file.
3. Add a <link> tag in the <head> section of your HTML file to connect it to your CSS file. This allows the browser to apply your CSS styles to the HTML.
4. Remove the default margins and paddings applied by browsers to <body> element. Note the impact of this step on the space between the banner image and the page body.
5. Create a class for the second table (e.g., table2) and set it to add vertical spacing between rows using a spacing value of 20px. Apply the created style to Table 2 in the provided HTML file. This adds a visual separation between rows in the table that contains images and text, making the content easier to read.
6. Now, we need to apply background colours to the content boxes describing historical events. Thus, create three classes, one for each box. Change the text-align for each box to center and assign the following background colours:
    - Box 1: Dark Red
    - Box 2: Deep Pink
    - Box 3: Brown
7. After changing the background color for each box, we need to style their fonts by Create a class called box_text_color and apply it to all text inside the boxes. Use this class to:
    - Change the text colour to white.
    - Add padding of 10px around the text.
    - Remove margins.
8. In the last lab, all the images (except the banner image) contain the same width and height. This makes them difficult to maintain the webpage. For example, if we need to change the size of all images, we need to do it in three different places. In this step, we will standardize the size of images in sections like “Calgary's Urban Evolution”. Create a class named td_image_width and apply it to all image containers in the second table. In this class, set the width=570px. For all images in table 2, set the image dimensions to width=570px and height=400px and ensure the images are displayed as block elements.
9. We need to style the text that explains the history and milestones on the webpage to match the output. For all paragraphs, we need to set the font to Arial, font size to 16px, and apply a dark grey colour (#3d3d3d).
10. Open your HTML file in a browser to verify the applied styles. Focus on the appearance of tables, boxes, paragraphs, and images. In case you page does not match the output, revisit the related class and check for typos or incorrect values.

Submission:

1- Fill out the Answer_sheet.docx, including:
    a. The names and UCIDs of all group members.
    b. A screenshot showing the output of Exercise 1.
2- Create a new GitHub repository and upload the code for Exercise 1 to the new repository.

Exercise 2: Flexbox

Objective: In this exercise, we will explore Flexbox and apply its properties such as `flex-direction`, `flex-wrap`, `order`, `align-items`, `justify-content`, and `align-self`, enabling them to design responsive and visually appealing web interfaces.

1. Create a new exercise2.html file in your code editor.
2. Create a flex container that contains five items. Each item should have the following:
    - **An image**: Use placeholder images or relevant engineering images (e.g., schulich.png, EE_eng.png). Ensure you copy the image name with the extension.
    - **A level three heading**: Provide a title for each item (e.g., "SCHULICH School of Engineering").
    - **A description**: Add a brief description for each item (e.g., "UCalgary's hub for innovative engineering education.").

3. Use the following styles to create the layout:

```
.container {
    display: flex;
    border: 2px solid black;
    padding: 10px;
}
.item {
    flex: 1;
    padding: 20px;
    margin: 5px;
    background-color: lightblue;
    border: 1px solid darkblue;
    text-align: center;
}
.item img {
    width: 100px;
    height: 100px;
    display: block;
    margin: 0 auto 10px;
}
```

4. Save and open the file in your browser. Ensure that the layout reflects a professional and visually consistent design.
5. Take a screenshot of the layout and add it to the AnswerSheet.docx. Ensure your screenshot shows all five items.
6. After taking the screenshot, **remove any added styles for the exercise** to ensure you start fresh for the next steps.
7. Add the following styles to the .container class:

```
flex-direction: column;
```

8. Observe the arrangement of the items. Take a screenshot and add it to the AnswerSheet.docx. Then remove the added style.

9. Experiment with:

```
flex-direction: row-reverse;
flex-direction: column-reverse;
```

10. For each change, take a screenshot, insert it into the AnswerSheet.docx, and remove the added style afterward.
11. Add the following CSS to .container class

```
flex-wrap: wrap;
```

12. Reduce the width of the browser window and observe the default behaviour of the container.
13. Observe the layout when there is not enough space. Take a screenshot and add it to the AnswerSheet.docx. Then remove the added style.
14. Experiment with:

```
flex-wrap: wrap-reverse;
```

15. Observe the changes. Take a screenshot and add it to the AnswerSheet.docx. Then remove the added style.
16. Add the following styles to specific items:

```
.item:nth-child(1) {
    order: 3;
}
.item:nth-child(2) {
    order: 1;
}
.item:nth-child(3) {
    order: 2;
}
```

17. Observe how the visual order of the items changes. Take a screenshot and add it to the AnswerSheet.docx. Then remove the added styles.

Exercise 3: Navigation bar

Objective: In this exercise, we will use Flexbox to design a responsive navigation bar that efficiently arranges menu items. Moreover, we will implement a dropdown menu to enhance user interaction, ensuring smooth and intuitive navigation. This will help in understanding Flexbox properties for layout structuring and dynamic UI components.

1. Copy the HTML code and CSS (exercise3.html and style3.css) provided in the supplementary material into your editor (e.g., VS Code, Sublime Text). This code gives you the structure of the webpage.
2. Create a navbar using Flexbox with the following items:
   - SCHULICH School of Engineering (https://schulich.ucalgary.ca/).
   - Electrical Engineering (https://www.ucalgary.ca/future-students/undergraduate/explore-programs/electrical-engineering).
   - Chemical Engineering Department (https://schulich.ucalgary.ca/chemical-petroleum/home).
   - Biomedical Engineering Department (https://schulich.ucalgary.ca/biomedical/home).
   - Department of Geometric Engineering (https://schulich.ucalgary.ca/geomatics).
3. The background color of the navbar should change to #333 when the user hovers the mouse over it **(check the output in the supplementary material).**
4. Add a dropdown menu under the first item (SCHULICH School of Engineering). The dropdown **should only appear when the first navbar item is hovered (check the output in the supplementary material)**. The dropdown should contain the following links:
   - Home
   - About
   - Research

NOTE: All styles should be added to the style3.css **file.**

Hints:

1. Use Flexbox to lay out the navbar horizontally.
2. Add a hover effect that changes the background color of the links make background color #bdbbbb;

Submission:

1. In the Answer_sheet.docx you edited for Exercise 1, include a screenshot showing the output of each step in exercise 2. In addition, take a screenshot of the output of Exercise 3.
2. Upload the code for Exercise 2 and Exercise 3 to the same repository you created for Exercise 1.
3. Compress both Exercise 1, Exercise 2, and Exercise 3 into a single file and upload the compressed file to D2L. Also, upload the completed Answer_sheet.docx.