Lab 2: CSS

The goal for this lab is to use CSS to change the visual of the website. Some pieces of this website won't be completely done. We're not looking for the exact values, instead, we'd like you to play around with the look and feel of the site. Before implementing anything, you should read through the entire lab handout. Then come up with questions to ask the TA. Then come up with a *written* plan. If you don't do this, you will struggle. The written plan helps the TA and I help you. There are many solutions in CSS, a written plan helps us take your solution to where it needs to be.

Deliverable

1. 3 webpages: a Home page, a Grades page, a Calendar page all uploaded to the ECC servers.
2. Your lab should be accessible at a path similar to:
    a. http://students.engr.scu.edu/~aelahi/coen-161/lab-2/index.html (this page can be a straight copy from Lab 1)
    b. http://students.engr.scu.edu/~aelahi/coen-161/lab-2/grades.html
    c. http://students.engr.scu.edu/~aelahi/coen-161/lab-2/calendar.html

Task

The end goal of the labs will be to create pages whose **visual style** resembles the following screenshots. Your pages **DO NOT** need to look exactly like the following screenshots but some exact numbers have been given. If you have a question, please ask the TA.

Common Setup

This section covers the common layout necessary for both of the pages.

base.css (Guided)

1. Start by creating a new CSS file called base.css. Then import this into the grades.html page. grades.html is the easiest page to start with since it's the least complex.
2. Reset the margin and padding on **every element** to 0.
3. For the body element
    a. Create a CSS custom property (referred to as accent color) that will be the background color for the side nav. The purple-ish value shown in the screenshot is #9b59b6.

   b. Create 3 more custom properties called border-radius-sm, border-radius-md, and border-radius-lg. These can be whatever values you want. I used 4px, 8px, 16px.
   c. Create 3 custom properties called padding-sm, padding-md, and padding-lg. These values can be whatever you want. I used 0.5rem, 1rem, 3rem.
   d. Do the same for margin.
   e. Use the accent color for the background and as the accent-color for any forms
   f. Set the height and width to the entire viewport and disable scrolling
   g. Make the body a flex container that lays its content out in a row
4. For the main element
   a. Set the border radius **only** on the left side border to border-radius-medium.
   b. Set the padding for the top and bottom to padding-md. The left and right padding should be padding-lg
   c. Make this flex item grow to take up all available space and never shrink

nav.css (Guided)

All of CSS is global so you should be sure to constrain your selectors such that the ones here only apply to a single element on the page.

1. Create another file called nav.css
2. Start by styling the nav.
   a. Because the body is a flex-container, you should be able to use the flex-basis property. Have the nav take up 20% of its parent's width.
   b. Set the background color to the accent color defined in (3.1 above)
   c. Because we're using a dark-ish color, using black for the text color will not provide enough contrast. Instead, change the color of the text to white.
3. Next we'll style the header
   a. Make sure that within the nav container, the text in the h1 is aligned to the center
   b. Make the font a little bit bolder by using a heavier font-weight
   c. Set some bottom padding
   d. Set the font size to the largest size
4. For each of the list items
   a. Set only a bottom border for each list element. The bottom border can be 1px solid and some color.
   b. You'll have to remove the bullet points from each list item as well
   c. When the list item is hovered, the background of the list item should match the color of the bottom border
   d. For each of the anchors inside of the list item
      i. Make sure that they take up the entire width of their parent
      ii. To add some visual balance, you'll want to add top and bottom padding

        iii. Then, center each anchor both horizontally and vertically

e. For each span

    i. Make sure they grow as necessary but should start by taking up 60% of the available space.

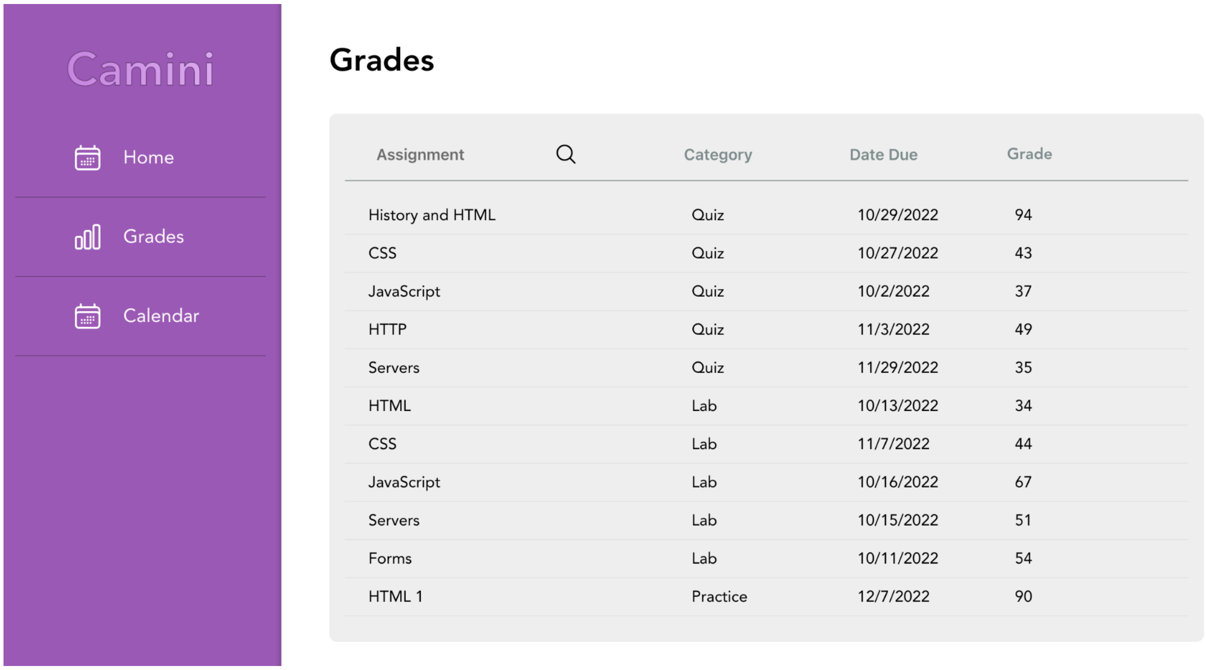
    ii. You may want to up thee font size

f. To style each of the images, you can use the following code block.

```
nav li a .svg-wrapper {
  flex: 0 0 40%;
  padding-right: var(--padding-sm);
  display: flex;
  justify-content: flex-end;
  align-items: center;
}

nav li a .svg-wrapper svg {
  flex: 0 0 40%;
  color: white;
  height: 32px;
}
```

grades.html

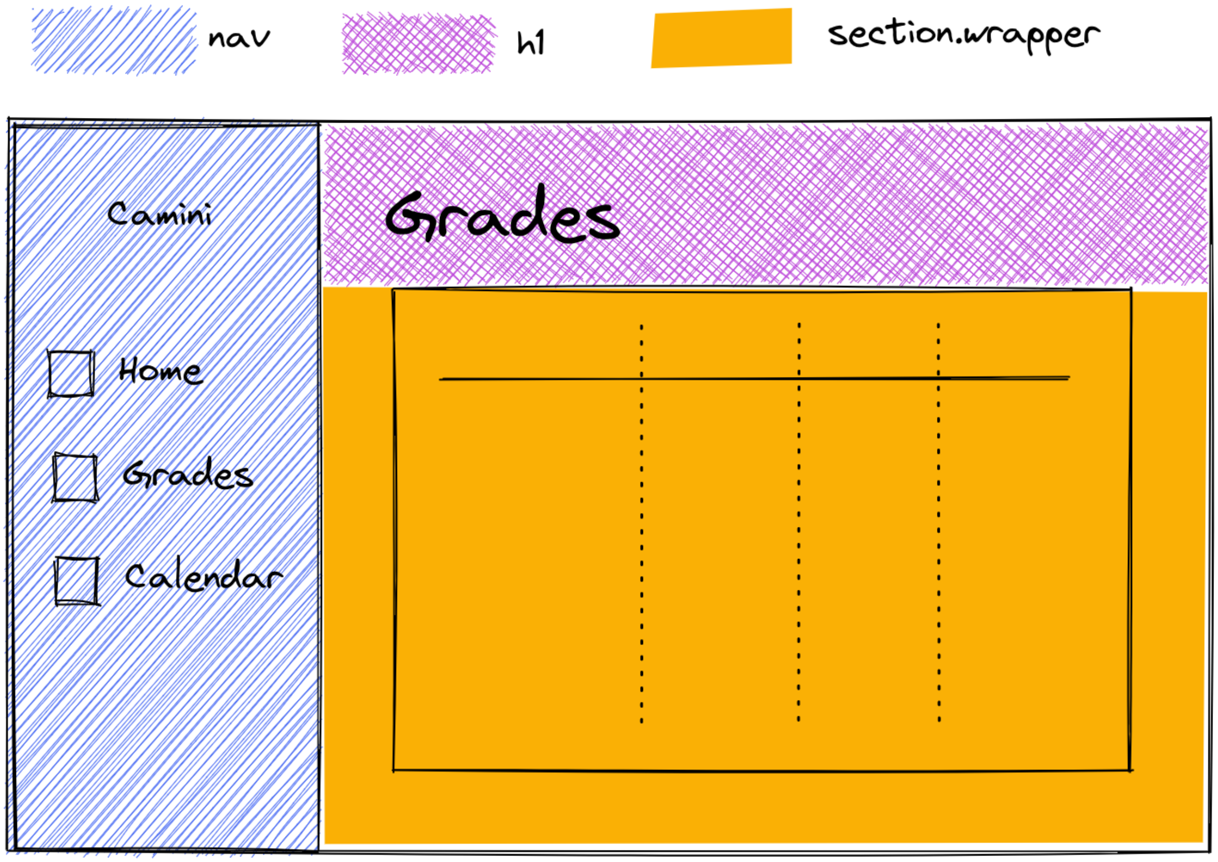


Grades

Assignment	Category	Date Due	Grade
History and HTML	Quiz	10/29/2022	94
CSS	Quiz	10/27/2022	43
JavaScript	Quiz	10/2/2022	37
HTTP	Quiz	11/3/2022	49
Servers	Quiz	11/29/2022	35
HTML	Lab	10/13/2022	34
CSS	Lab	11/7/2022	44
JavaScript	Lab	10/16/2022	67
Servers	Lab	10/15/2022	51
Forms	Lab	10/11/2022	54
HTML 1	Practice	12/7/2022	90

Boxing grades.html

To understand grades.html, it's best to start at the three main elements of the page:



We created all the necessary styles for the nav in the previous section. You can style the h1 as you want. This section focuses on the section.wrapper. Tables have unique default properties for the display property. The default helps to keep the alignment of tables but doesn't play nicely with some properties like margin and property. We'll be recreating the same effect with display: flex.

Before you get started on the page, make sure to link the CSS files from above. Then create a new one for grades.css and link that too. You should not have any <style> tags in your html and no element should have a style attribute.

1. Set the entire table to have display: flex
    a. Each flex-item (the thead and tbody) should be arranged in a single column, so apply the flex-flow: column property
    b. Set the height to 85%
    c. Add some amount of margin, border-radius, and padding
    d. The background for the table should be a distinct color from the background of the main.
2. A table is made up of two sections: a table head (thead) and a table body (tbody). Each row in either the thead or tbody are a tr (table row) element. Unfortunately, each section

uses a different for each cell. The body uses td for each cell and the head uses th for each cell.

3. thead
    1. Add a border-bottom to the thead and give a small amount of margini
    2. Change the text color so it's distinguishable from the body text.
4. tbody
    1. The tbody should be scrollable, so apply the overflow property.
    2. Make the tbody take up all available space by setting the flex-grow property to 1.
5. Creating alignment without display = table
    1. In flex, to make things take up an exact amount of space, we use the flex-basis property. The first column (with an assignment name) might be longer than any of the other three columns.
    2. Make the flex-basis = 40% for **any tr's first-child**. Make the flex-basis = 20% for all other children of a tr.
        1. Why do we use these values? There's 4 columns - assignment, category, date due, grade. We want our percentages to add up to 100.
    3. For each of the td and th's, set the display to flex.
        1. Then vertically align the contents of each td/th by using align-items: center.
        2. When display = flex and flex-flow = row (the default), the value of align-items applies to an element's children in the y-direction. justify-content applies to an element's children in the x-direction.
6. The assignment header is an input with a search box. Notice that there's a div.search-wrapper before the input
    1. To position the magnifying glass image correctly, you'll have to make the search wrapper position: relative.
        1. Once the .search-wrapper has position: relative, the img can have position: absolute and it'll now be positioned relative to the search-wrapper.
        2. Set the width and height of the .search-wrapper img to 24px
        3. Vertically center the image within its parent.
            1. The easiest way to do this is to make the parent have display = flex, align-items = center
    2. To make the search input shed its default styles, try applying the following styles and see how the input changes
        1. border: 0;
        2. background: inherit;
        3. border-radius: var(--border-radius-md);
        4. padding: var(--padding-sm) var(--padding-sm) var(--padding-sm) var(--padding-md);
        5. font-weight: inherit;

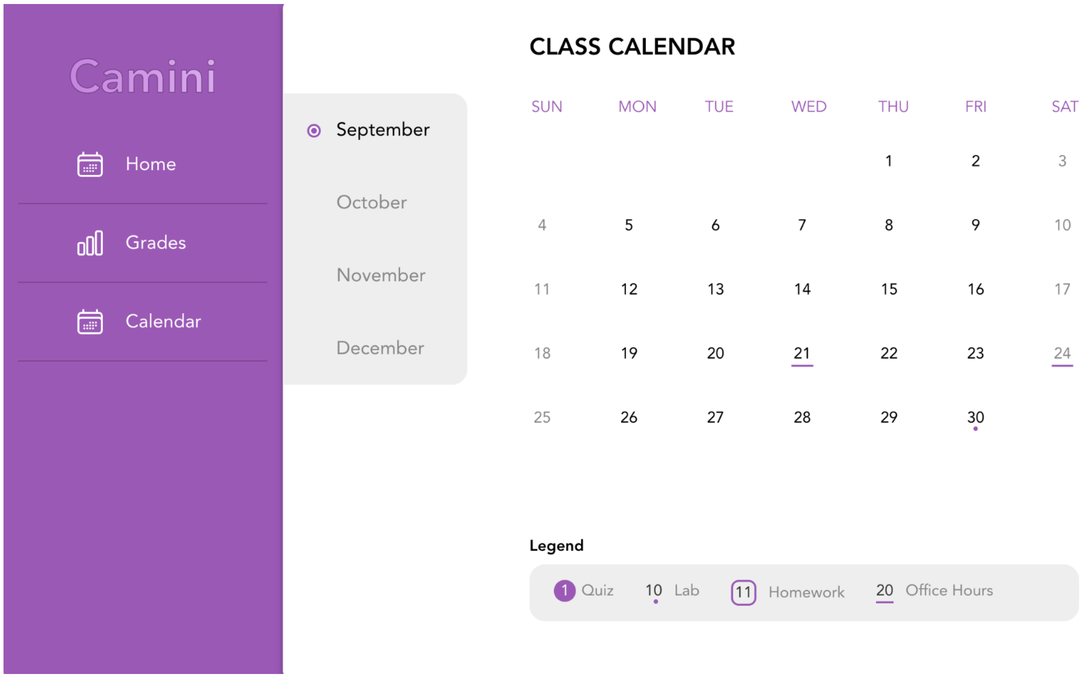
vi. font-size: inherit;

c. When the input is hovered, change the background color of the input

7. Styling the th's which have buttons inside
    a. You'll have to get rid of a lot of the default button styles. This is pretty similar to what you did for the search input above. Try applying those properties in sequence to remove the button's default stylings.
        i. One extra thing you'll want is cursor: pointer so the button looks like its clickable.
    b. When you hover either of the th's which are buttons, the background should change to a white-ish color (like #fafafa) and then update the text color so it's readable

calendar.html (Unguided)

The unguided part of this lab is to recreate the calendar.html page as shown below.



SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

If you've gotten this far, there is only one CSS concepts that you haven't already used that's necessary for the calendar.html:

1. The ::before and ::after pseudo-elements. You'll need a combination of position: absolute to style each of the items in the legend.

Tasks

1. Open a paint-like program (or excalidraw.com) and box the page, similar to the image shown for grades.html.
    a. You can include the nav if you want, but it should be the exact same from the grades.html page.
2. Once you've cut the page, you'll have a base to start from. From here, the rest of the lab is all on you. Your solution might have slight differences, but each element should have a passing resemblance to the picture above.

Grading

Criteria	Points
grades.html is visually similar to the image (just follow the directions)	+30
calendar.html > "box"-ing	+10
calendar.html > elements placed similarly to	+20
calendar.html > calendar days are aligned with headings	+10
calendar.html > calendar heading is visually distinct from the days + capitalized (even if the HTML is not)	+5
calendar.html > **ONLY** the selected month appears with the radio checked with color	+5
calendar.html > legend	+20
Deductions	Points
style tag or style attribute is present in HTML	-10

What does similar mean?

- Colors may be different
- Fonts and their weight might be different but MUST be changed from their defaults where appropriate (for things like headings)
- Spacing (margin, padding, border) MUST be changed from the defaults where appropriate

(things shouldn't look squished)

- The alignment of items on the page should match the image.
- Each element in the image should be placed roughly where it is in the image.
- **If you have a question, ask the TA or the lecturer.**