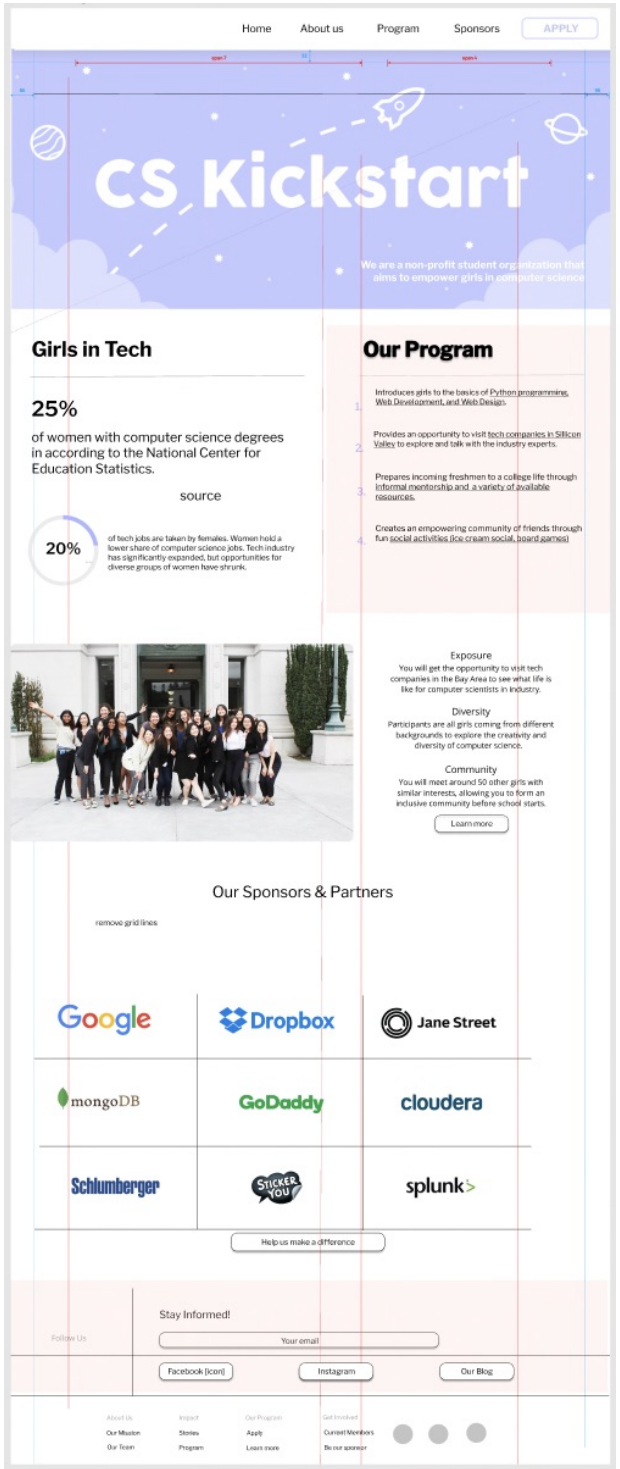
CS Kickstart's Web Design Guide

(also see index.html indexstyle.css)



1. Creating a Prototype

Before beginning any project, you should have a plan. In web design, planning is usually done through applications like Adobe XD (free with your Berkeley email) or Figma (free, online, and collaborative), though prototypes can also be drawn by hand or on Powerpoint.

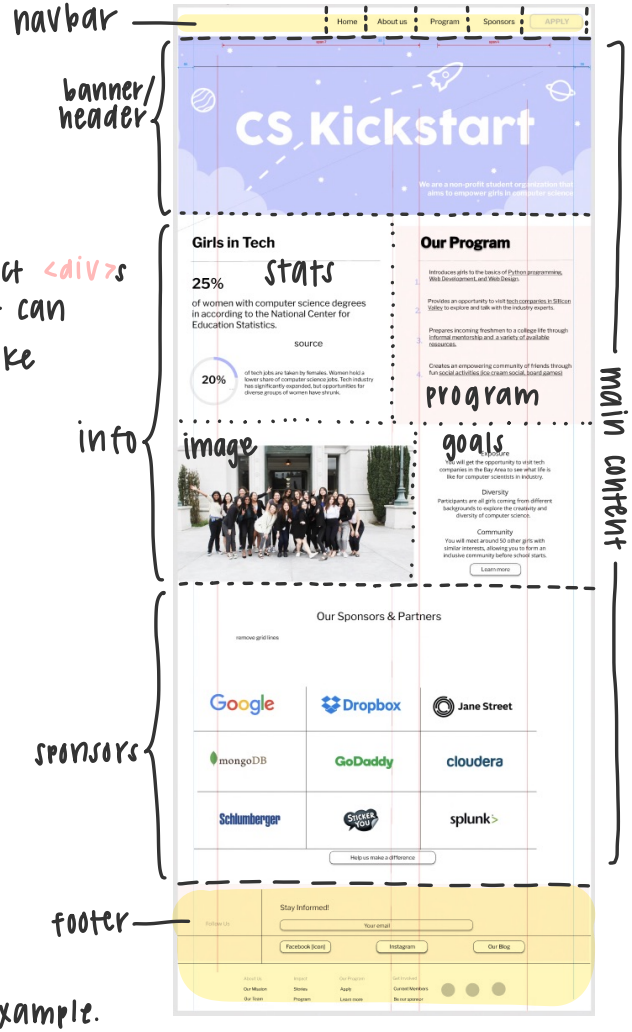
The image on the left is the prototype (specifically a high fidelity mockup) we will be using in this walkthrough. This will be a tutorial on how I approach web design, but is definitely not industry standard (or anything).

2. Dividing up the Page

A website usually contains 3 large sections:
- navigation bar (navbar)
- main content
- footer

that are usually created with their own distinct s in an HTML file (index.html). The main content can then be further divided into subsections to make organization and positioning much easier.

The images on page 4 translate this to HTML.



3. Filling in the Content

Fill in all the s with your webpage content using <h1>...<h6>, <p>, <img>, and <a> tags.

When opening your HTML file in a browser, the page does not need to be pretty / formatted yet. That comes later with CSS.

See the image on page 4 & index.html as an example.

4. Make it ~pretty~!

In your style.css file, add styling to your HTML elements!

Let's start by creating variables for our colors!

```css
/* define variables */
:root {
  --font-color: #333;
  --font-color40: rgba(51, 51, 51, 0.4);
  --pink: #FDCFC1;
  --pink60: rgba(253, 207, 193, 0.6);
  --purple: #DADFFF;
  --purple60: rgba(218, 223, 255, 0.6);
  --shadow: rgba(0, 0, 0, 0.2);
}
```

rgba ↓

By assigning hex and rgba color values to variables, we're able to easily access them later. Usually designers try to avoid black, which is why our var(--font-color) is #333333, a dark gray.

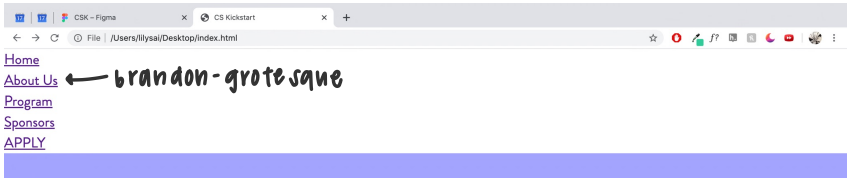
When working in CSS, don't forget semicolons at the end of a line!

```css
body { /* applies to the entire page, unless overwritten */
    margin: 0px; ← the body automatically has margins
    font-family: brandon-grotesque, sans-serif; ↑ back-up font
    font-weight: 400; ← regular, bold, etc.
    font-size: 3vh;
    color: var(--font-color);
}
↑ font color

h2 { ← Header <h2> tags
    font-family: neue-haas-grotesk-display, sans-serif;
    font-weight: 700;
    font-size: 5vh;
    line-height: 2vh; ← affects spacing between lines of text
    text-transform: uppercase;
}
↑ makes text uppercase

h3 {
    font-size: 4vh; ← vh is a unit that scales to the webpage (viewport) height. 1vh = 1% of webpage height.
    line-height: 1vh;
}

h4 {
    font-size: 3vh;
    line-height: 0vh;
    color: var(--font-color40);
}
```

I then begin setting fonts for text. Here, I set all of the body's text to default to brandon-grotesque of size 3vh.

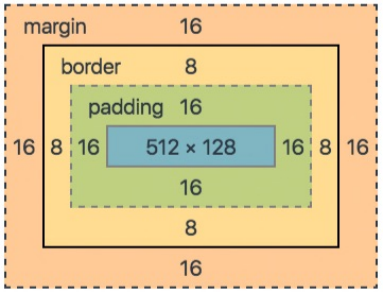
But since I want headers to be different, I set h2 to neue-haas-grotesk-display with size 5vh.

Sizes in CSS require units! Common ones include px (pixels), % (of the parent's size), and vh (% of the website height) / vw (% of the website width) for this website we will use vw and vh so the page responds to different browser sizes.

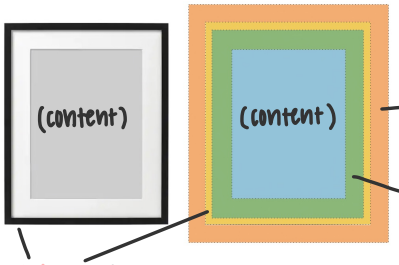


Now, we'll work on the spacing and sizing of elements.

The box model on the right is how HTML elements are displayed in the browser.



Imagine two frames side by side.



Margin: clear area outside the border (transparent)

Padding: clear area around the content (transparent)

Border: frame that goes around the padding and content

```css
.button { ↓ sets border design, thickness, & color
    border: solid 0.11vw var(--purple);
    border-radius: 1vw; ← rounds corners
    box-shadow: 0vw 0.1vw 0.3vw var(--shadow);
    padding: 0vw 3vw 0vw 3vw; ↑ applies a drop shadow
    font-weight: 500;
    color: var(--font-color);
    text-align: center;
}
↑ 3vw of padding on left & right
```

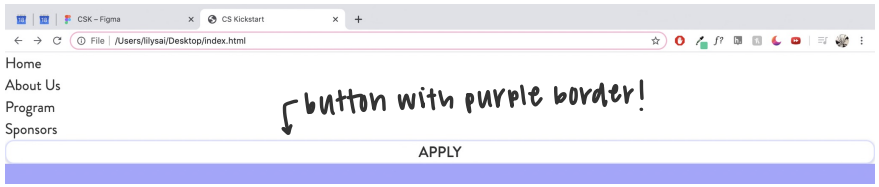
This block of CSS code sets the design for all elements with the button class, assigning border, border-radius, box-shadow, and padding to the elements.

```css
.button:hover {
    background-color: var(--purple);
}
```

Because users interact with buttons, we want to give them feedback when the elements are active or being hovered over. This is where pseudo-selectors come in! They are used to define what happens during a certain state of an element. For our buttons, we change the background-color of the button to purple, so people realize it is an interactive element.

```css
a {
    text-decoration: none;
    color: var(--font-color);
}
```

Also we can remove the underline from links by setting text-decoration to none.



5. Flexboxes (Responsive Design)

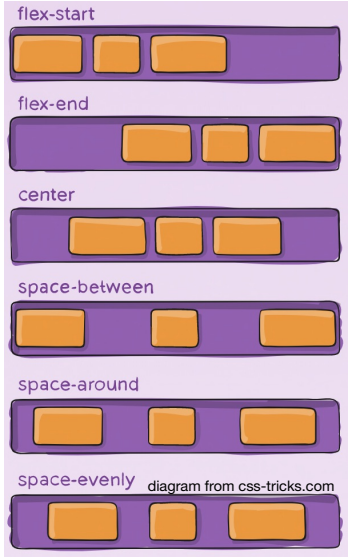
Now that our elements are individually pretty, let's organize the page. Flexbox is a CSS web layout model that allows elements to be arranged based on screen size. Let's apply this to the navbar so that it stretches across the top of the page.

```
/* navbar */
#navbar {
    display: flex;                  → elements aligned in a row. main axis is x
    flex-direction: row;            → aligns elements to the right
    justify-content: flex-end;
    position: fixed;
    z-index: 1;
    background-color: white;
    width: 100%;
    box-shadow: 0vw 0.2vw 0.5vw var(--shadow);
}

#navbar a {                         → adds spacing between the links
    padding: 1vw 2vw 1vw 2vw;
}
```

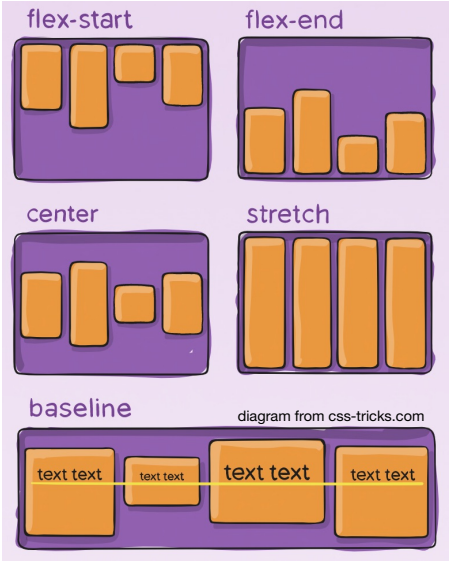
To use flexbox, display must bet set to flex in the parent container, so for us that is a div containing all the links.

flex-direction is the order in which the elements in the container will be aligned. (row, row-reverse, column, column-reverse)

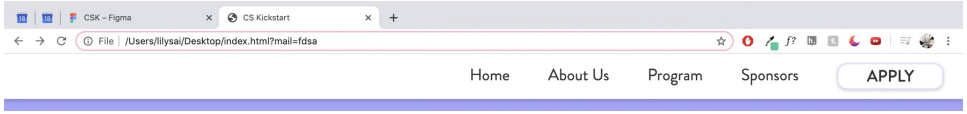
justify-content aligns on main axis



align-items aligns on cross axis



flex-wrap allows the items to wrap onto the next line if it runs out of space.



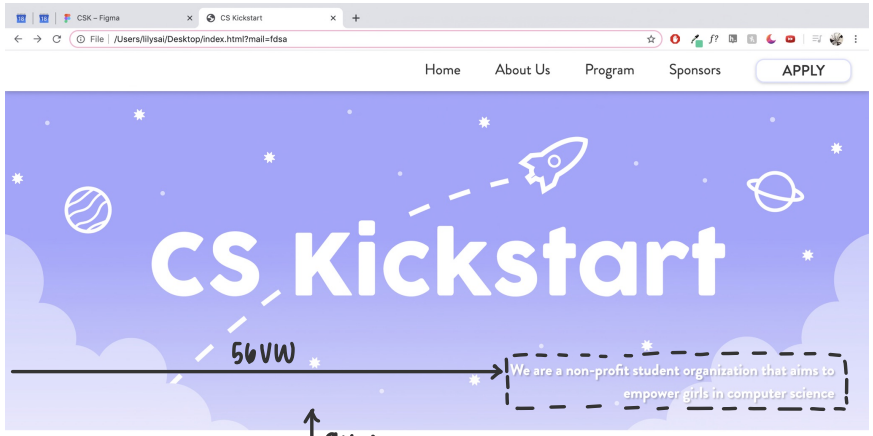
6. Positioning

Generally I avoid positioning items manually, but there are certain specific scenarios that require positioning. The ones we will tackle today are fixing the navbar to the top of the page and overlaying text on an image.

The navbar can be fixed to the top of the page using position: fixed. This means that no matter how we scroll, the navbar will stay on the screen exactly where we have placed it. However to ensure that it stays infront of other elements, we need to set z-index to a larger value than the other elements, moving it to the top. (see the code on the left as an example).

```
/* header */
#header img {
    padding: 4vw 0vw 0vw 0vw;
    width: 100%;
}

#header div {
    font-size: 1.8vw;               ↓ similar to box-shadow
    text-shadow: 0.15vw 0.15vw 0.15vw var(--shadow);
    width: 40vw;
    text-align: right;
    color: white;
    position: relative;             } positioning
    left: 56vw;
    bottom: 9vw;
    font-weight: 600;
}
```

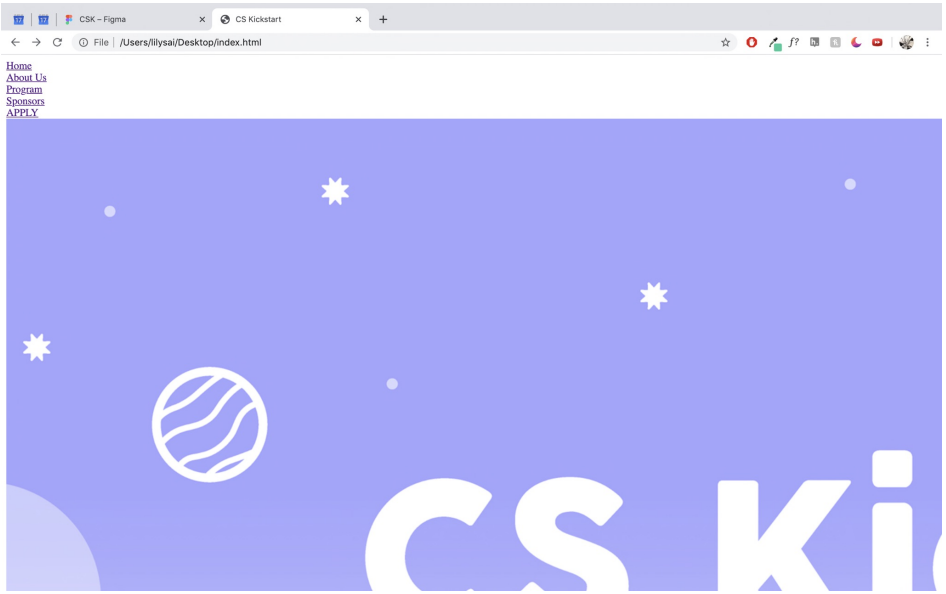
position:
- static: the element's default position on a page
- absolute: if the child is absolute, the parent ignores it and the child element is positioned relative to the page
- relative: positioned relative to it's default static position on the page and moved around using left, right, top, bottom

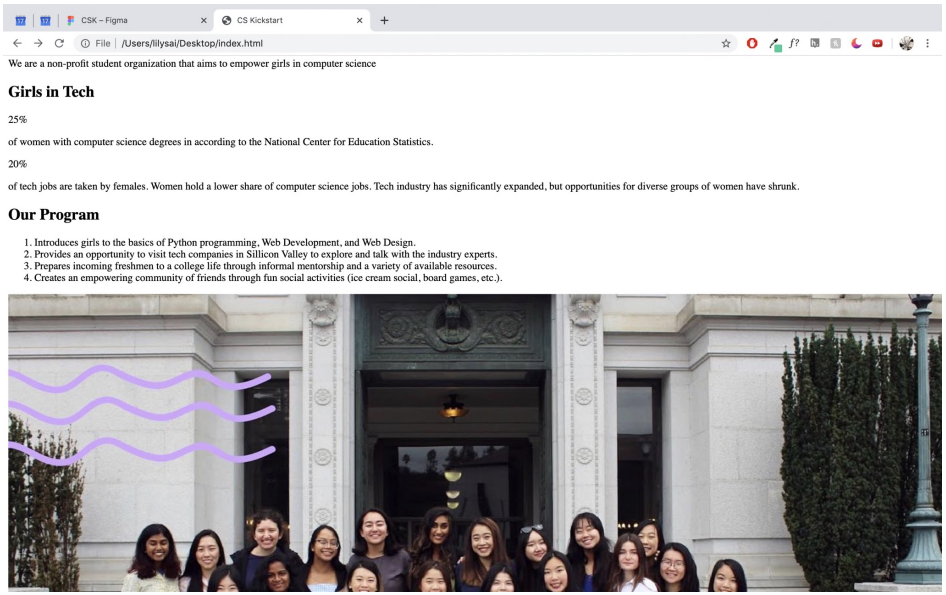


2. Dividing up the page

```
8   <!-- this is where all visible content goes -->
9   <body>
10      <div id="navbar">
11          <a href=""><div>Home</div></a>
12          <a href=""><div>About Us</div></a>
13          <a href=""><div>Program</div></a>
14          <a href=""><div>Sponsors</div></a>
15          <a href=""><div class="button">Apply</div></a>
16      </div>
17
18      <div id="main-content">
19          <!-- header -->
20          <div></div>
21
22          <!-- info -->
23          <div>
24              <!-- stats -->
25              <div></div>
26
27              <!-- program -->
28              <div></div>
29
30              <!-- image -->
31              <div></div>
32
33              <!-- goals -->
34              <div></div>
35          </div>
36
37          <!-- sponsors -->
38          <div></div>
39      </div>
40
41      <div id="footer">
42          <!-- stay informed -->
43          <div></div>
44
45          <!-- footer links -->
46          <div></div>
47      </div>
48  </body>
```

3. Filling in the content





Also see index.html!