HTML, CSS, & Bootstrap.

FYI, comments in HTML:

```
<!-- I'm an HTML comment! -->
```

comments in CSS:

```
/* CSS comment :) */
```

HTML.

HTML is the structure of a website. Most websites you look at have a similar basic structure:

1. header / navigation bar
    - this is usually at the very top of the website
    - here you might see the logo of a company, as well as some links that would take you to different sections of the website
2. main content
    - this sits between the header and footer
    - this is the bread and butter of the website, so to speak. This is the content that the user visited our website for
3. footer
    - the footer is at the bottom of the site
    - it usually contains links to more obscure parts of the website, as well as links to the company's social media presence. It also usually has the copyright

The most common HTML tags to be aware of are:

div.

`<div>` stands for **division**, and its used to signify a section of a website. The idea is that all the content put in a `<div>` is connected to each other. If you put a series paragraphs and images in a `<div>` then presumably that means those words and images are somehow connected to each other.

A website is generally made up of 3 basic sections: a header / navigation bar, a main section, and a footer. Each of these sections would be likely* be contained within a `<div>`.

For example, you might have a few paragraphs of text that all relate to each other. All of this text would likely be placed in one single `<div>`. Perhaps following these paragraphs is a form where users can provide input. That form would likely be placed in another `<div>`.

The syntax for a `<div>`:

```
<div></div>
```

h1.

An `<h1>` is a header text. Use this to really draw attention to something. You usually use this only once per page to introduce the main theme of the page or website to the user.

```
<h1>Welcome to my site.</h1>
```

subheaders.

While `<h1>` is a header, we also have subheader tags: `<h2>`, `<h3>`, `<h4>`, and `<h5>`. They're basically smaller `<h1>`s, where `<h1>` is the biggest and `<h5>` is the smallest. They're meant to introduce smaller subsections of a website. On this cheatsheet, "HTML, CSS, and Bootstrap" is an `<h1>`, "HTML" is an `<h2>`, and "subheaders" is an `<h3>`.

p.

`<p>` is a paragraph. You would put a paragraph of text in a `<p>` tag. The text will by default be 16px, making it smaller than the header or subheader tags.

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed faucibus justo id lacinia volutpat. Etiam suscipit enim maximus ante pulvinar maximus. Sed quis ex in leo dictum viverra. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nullam congue odio dui, et porttitor lacus aliquet in. Sed semper risus in condimentum varius. Sed leo felis, eleifend quis dolor eget, tempus porttitor augue. Fusce eu finibus eros. Curabitur facilisis enim ac nibh mattis condimentum.</p>
```

a.

`<a>` is called an **anchor tag**, and we use anchor tags to either make requests to servers, or, more commonly, to simply link to another website or section of our own website. To make an anchor tag link to another website we have to pass it an attribute called `href`.

> side note about attributes: attributes basically exist to provide additional information. Some attributes can be applied to all HTML tags — class and id are examples of this — but some attributes, like hrefs, are specific to certain HTML tags. When working with an HTML tag you've never seen before, it's good to do a quick Google search to see if that HTML tag has any attributes you need to worry about!

```
<a href="/recommender">if you click me I'm going to take you the "recommender"
page of this website!</a>
```

```
<a href="https://google.com">if you click me I'm going to take you to a whole
different website</a>
```

img.

`<img>` exists to simply display on image onscreen. `<img>` must be passed an attribute called `src`, whose value should be the *relative path* that leads to the image you want to render. `src`'s value could also be a URL that points to an image online.

`<img>` tags do not need to be closed. It is something called a *void element*, which basically just means content is not passed to this tag. Because it can't have content, there's no reason to close it.

```
<img src='/path/to/img'>
```

If the image you want to render is in the same directory as your HTML file, then you can simply pass the name of the file to `src`.

nav.

`<nav>` is a new HTML tag that came out in 2016 with HTML5 (the newest edition of HTML). It is something called a **semantic tag**. What that means is that it behaves exactly like a `<div>`, but the tag itself provides meaning to screenreaders and other programmers who are reading your code. `<div>` is a generic term for a section, while `<nav>` is a more specific word that refers to a navigation bar.

```
<nav>
  <a href='/home'>LOGO</a>
  <a href='/recommender'>Recommender</a>
  <a href='/contact'>Contact</a>
</nav>
```

forms.

`<form>`s are most commonly used to capture input from users and send that data to a server so that server can do something with that data (like `INSERT` it into a database). By default, however, the `<form>` tag will take whatever input the user provided and add it as a query string to the URL.

A **query string** is a part of a URL that will store variable values. When we work with user input, the query string will contain the input we got from the user. A query string is whatever follows a `?` in a URL. So in the following URL

```
https://www.my-website.com/search?q=pandas
```

`q=pandas` is our query string. This means that we have some kind of input field and the user typed `pandas` into it. If we capture multiple inputs from the user — so let's say we had 3 input fields that the user types into — then whatever the user entered into all 3 of those inputs would end up in the query string:

```
https://www.my-website.com/search?q=pandas&oq=pandas&aqs=chrome
```

In this example `q=pandas&oq=pandas&aqs=chrome` is our query string, and `pandas`, `pandas`, and `chrome` are the inputs we got from the user.

The form tag by itself is quite simple:

```
<form></form>
```

But of course most `<form>`s are not empty, but contain input fields, or drop down menus, followed by buttons! So a `<form>` with input fields would look as follows:

```
<form>
  <input name="first" placeholder="first" type="text">
  <input name="last" placeholder="last" type="text">
  <input name="password" placeholder="password" type="password">
  <button>submit</button>
</form>
```

The `<input>` tag is just an empty white box where the user can type. There are 3 attributes that we generally use with `<input>`s: `name`, `placeholder`, and `type`. `name` will be the name of the variable put in the query string / sent to the server. `placeholder` will be the words the user will see in the input field before they click on it. `type` describes the type of the input field. The most common types are:

- `text`: means the user can type in the input field as usual.
- `password`: when you type into the input field, all you'll see are black dots
- `file`: will allow you to select a file to upload
- `checkbox`: will give you a check box that the user can tick.

Maybe in our form we want to have a dropdown menu? The HTML for a dropdown menu is:

```
<select name="ratings">
  <option value="1">movie1</option>
  <option value="2">movie2</option>
  <option value="3">movie3</option>
  <option value="4">movie4</option>
  <option value="5">movie5</option>
</select>
```

The value of the `value` attribute is what will be sent to the server / query string when the form is submitted. The words put between the opening and closing `<option>` tag is what the user will see when they click on the dropdown menu. Most of the time these would be the same as the values passed to `value`.

classes and ids.

`class` and `id` are attributes that can be passed to any HTML element. We use them to make it easier to target specific HTML elements in our CSS.

```
<div class="silly-muffin"></div>
<img id="my-silly-image" src="muffin.png">
```

The ONLY different between `class` and `id` is that the same `class` can be passed to multiple HTML elements, but the same is not true for `id`.

CSS.

CSS is the styling of a website. CSS turns the skeleton of our HTML into a good-looking website. The way CSS works is that we first have to target an HTML element or elements, and then we have to decide what styling we want that element to get. There are 3 basic ways to target HTML elements in our CSS:

1. by their tag
    - so I could target, for example, every single `<p>` in my website by doing the following:

```
p {
  /* all styling here will apply to EVERY p tag!*/
}
```

2. by their class
    - if I want to target an element or group of elements by their class, the syntax would be

```
.silly-muffin {
  /* any styling placed here would apply to every HTML element with
class='silly-muffin'*/
}
```

    - notice the `.`! The `.` means class!
3. by their id
    - if I want to target an element by its id, the syntax would be:

```
#my-silly-image {
  /* any styling placed here would apply to every HTML element with
id='my-silly-image'*/
}
```

    - notice the `#`! `#` means id!

CSS works by adding and manipulating properties on HTML elements. There's many, MANY properties in CSS but we're only going to focus on the most common ones:

font-family

`font-family` refers to the font that text has. We can change the font of text by changing its `font-family` property in CSS:

```
p {
  font-family: "Helvetica Neue";
}
```

font-weight

`font-weight` will make our font more or less bold. The lowest value that can be passed to `font-weight` is 100, and the highest (most bold) is 900.

```
p {
  font-weight: 300;
}
```

letter-spacing

This property affects how much space there is between letters in a group of text.

```
p {
  letter-spacing: 1px;
}
```

So here there would be a 1px gap between every letter in this `<p>` tag.

color

The `color` property affects **the color of text**.

```
p {
  color: red;
}
```

hex and rgb values can also be passed to `color`.

background

`background` will change the background-color of an element.

```
div {
  background: tomato;
}
```

width and height.

These will change the `width` and `height` of an element. Commonly used to resize images in CSS.

```
img {
  width: 250px;
  height: 300px;
}
```

Bootstrap.

Bootstrap is a CSS framework that makes it easy to format websites and make them look modern and professional. Bootstrap works by adding certain classes to elements that receive styling from Bootrap. The basic idea is that you add a bootstrap class (so `container` for example) to an element (like a `<div>`), and Bootstrap has a CSS file that includes styling for the `container` class.

Bootstrap is *style opinionated*. This means that Bootstrap will make a lot of decisions for you regarding styling. If you like the decisions Bootstrap makes for you, then this isn't a problem, but if you don't then you will have to write your own CSS to override the Bootstrap styling decisions. For this reason knowing the fundamentals of CSS is necessary for working with Bootstrap.

The way you discover Bootstrap classes and what they do is, truth be told, by Googling. So let's say you want to style a `<form>` using Bootstrap: you would Google `Bootstrap form` to see the page Bootstrap has for forms. From there you can browse the different forms Bootstrap has, and when you find the form you like you can copy the HTML into your HTML file, and modify as necessary.

However, some Bootstrap classes are very useful and are used often. Therefore I'd like to highlight them in the list below:

`.container`.

`.container` is probably the most common Bootstrap class. The `container` class is usually added to a `<div>` tag, and it adds margins to the left and right of the `<div>` to make the content it contains centered horizontally on the page. It's very common to wrap most sections of your website in a `<div class="container">`.

```
<div class="container">
  I'm horizontally centered content!
</div>
```

`.jumbotron`.

`.jumbotron` is a big box that's really useful for showcasing the most important message of your website. Marketers use `.jumbotron`s to showcase their key marketing message. You can use it to introduce the name or main theme of your website.

```
<div class = "jumbotron">
  Hey! Read me!
</div>
```

`.bg-[WORD], text-[WORD]`

So `bg` means `background` (as in background-color), and `text` refers to text color. So these two are classes we can use to assign Bootstrap colors to elements. The colors Bootstrap has can be found here: https://getbootstrap.com/docs/4.0/utilities/colors/

So you would use the classes as follows:

```
<p class='text-primary'>I'm blue text!</p>

<div class='bg-success'>I'm a green background!</div>
```

`.m[LETTER]-[NUM 1-5], p[LETTER]-[NUM 1-5]`

So `m` stands for **margin**, and `p` stands for **padding**. T

He values of LETTER are: `t`, `r`, `l`, `b`. The letters stand for `top`, `right`, `left`, and `bottom`.

And the numbers represent how big of a margin or padding to have. 1 is the smallest margin / padding, and 5 is the largest margin / padding.

So you would use Bootstrap margin and padding classes as follows:

```
<p>words words words words</p>
<p class="mt-4">get away from me, "words" p tag!</p>
```

navbars.

Navbars are navigation bars. Basically every website has them, and they're usually at the very top of the site . There's a LOT of Bootstrap classes we use to make navbars. The important classes to be aware of are:

`.navbar`.

`.navbar` will make everything all the text you put in it appear on one line. Without this class it's possible that all the links you want to put at the top of your page would be stacked on top of each other, not next to each other. `.navbar` will put the links next to each other.

`.navbar-brand`

`.navbar-brand` is used if you want to put a logo on the top left of the page, and you want to put a bit of emphasis on the logo. Usually used with anchor tags (`<a>`).

displaying the links in the navbar.

To actually display the links — so the things that the user can click on to be redirected to different pages of your site — we would essentially, in our HTML, created a bullet-pointed list of all of the links we want to have, and then add some Bootstrap classes to make them look like links, not bullet-points. The basic HTML, without Bootstrap, that we would use would be as follows:

```
<ul>
  <li>
    <a href="/recommender.html">Recommender</a>
  </li>
</ul>
```

`<ul>` is an **unordered list**, or a list with bullet-points. `<li>` represents every item in the list. So `<li>` is the words that would follow the bullet points. We wrap the link in an anchor tag, so that clicking on them redirects users to another page.

Now if you render this in the browser, you'll see an ugly bullet-point list on the top of your page. To use Bootstrap classes to make those go away, use the following classes:

```
<ul class="navbar-nav">
  <li class="nav-item">
    <a class="nav-link anchor-item" href="#">Recommender</a>
  </li>
</ul>
```

They remove the bullet points and add some margin to make everything look aligned.

The completed navbar HTML would be:

```
<nav class="navbar text-dark navbar-expand-lg ">
  <a class="navbar-brand">LOGO</a>
  <div>
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link anchor-item" href="#">Recommender</a>
      </li>
    </ul>
  </div>
</nav>
```

Bootstrap Cards

Bootstrap Cards are a really nice way to render lists of items. If each item in a list contains some words describing the item, an image, and maybe a button or an input field, it's a good idea to p

`.card`

This class adds a nice border to all the elements in the list (so the image, text, and input field)

`.card-img-top`

This will correctly position an image within the top portion of the card.

`.card-body`

Use this to nicely display text within the card. If you have a lot of text in your card, you'd want to put that text in a `<div>` with the `.card-body` class on it.

`.card-title`

Should be applied to an `<h*>` element. Will add a title to your card.

`.btn`

This class should be added to `<button>`s to make them more prominent and have nice rounded borders.

The completed card HTML would be:

```
<div class="card mr-5 mt-5" style="width: 18rem;">
  <img class="card-img-top" src="pup.jpg" alt="Card image cap">
  <div class="card-body">
    <h5 class="card-title">Saw</h5>
    <select name="rating-saw">
      <option value="1">1 - terrible</option>
      <option value="2">2 - bad</option>
      <option value="3">3 - ok</option>
      <option value="4">4 - good</option>
      <option value="5">5 - excellent</option>
    </select>
  </div>
</div>
```