Recap

Welcome back to Week 2 of our QWEB educational modules. Quickly we'll recap what we did last week:

1. Gave an introduction to QWEB and our goals here
2. Provided a very abstracted view of what the internet is
3. Did some HTML!
4. Ended off with a club page for a people watching club.

Today

Today we're going to further edit that same club page from two weeks ago. We'll work on a little bit more **HTML** and add some **CSS** to style our page.

HTML

Let's start with a few more HTML elements that may be useful to you. We'll only spend a few more minutes on this as our hope is to jump right into CSS today.

First we can open up `club1.html`, and we'll see the exact same website from the end of the last lesson. Once again you can find these files on our github which we provide a link to on our discord server.

Provide how to find that link

From here you can go into each week and see the lecture files that I use, I've been calling these club1, club2, etc. You can also find some extra files we've written for documentation and such. Generally, having a quick peak at this page after a lesson is a good idea!

Let's first create a new file called `club2.html` and grab our previous code from club1.html. From here we are going to introduce a new tag called the *header*. Let's go into our body tag and place our title within this header tag.

```
<header>
    <h1>PeopleWatching Club!</h1>
    <h3>We are the People watching club, we love People.</h3>
</header>
```

On first glance, this has no change to our actual website. The *header* tag is similar to the *div* tag in that it doesn't actually change anything but allow us to better group our HTML based on it's function.

We'll also create a an actual footer for our website. Last time we created this simple thank you for visiting section at the bottom, so we'll improve on this section. First we'll remove the div at the bottom of our page and replace it with a footer tag.

Generally footers on a website have some copyright information and maybe some contact information. Let's start with some contacting, first we'll create a link and provide an email address for a user to contact us. We'll also provide a telephone number. From here we'll add some break lines, and add some text about copyright. We'll also add this copyright text at the bottom and include `&copy;` which will provide the copyright symbol on your site.

```
<footer>
        <a href="email:contact@gmail.com">contact@gmail.com</a>
<br>
        <a href="tel:123-456-7890">123-456-7890</a>
        <br>
        <br>
        This site was written by: the members of the Queen's
People watching club.
        <br>
        copyright 2022&copy;
</footer>
```

If we check out our site, we now have a more official looking footer!

Contact Us Form

Next we're going to add a second page to our website, a contact us page. Let's create a folder called club3 and copy over our club code into `club.html`. Then create `contact.html` and start to build out this form. We haven't done any HTML input, so let's start here. First, we're going to want to create all the base HTML that we would normally:

```
<!DOCTYPE html>
<html lang="en">
<head>

</head>
<body>

</body>

</html>
```

In our header, we'll add a title. Then in our body we're going to create a navbar. Navbars are the top of a website that allow you to click through the different pages on a website and generally navigate the site.

We'll also create a selection tag to hold our form. Generally, the section defines a *section* in a document. Generally it makes it easier to style things, and organize our HTML. We'll then define the form tag in order to hold our form's information. We'll create a quick heading, and a div to hold all of our forms' information.

From there we're going to define an input, you'll notice that the input tag does not have any space for text within it. We'll first define a **type**, in this case *text*. We'll also define a **id**, in this case *Name*. This will allow us to reference this value later with some text to label it, as well as potentially with JavaScript (we'll get there!). We will then define a name, in this case *name* since we are going to ask for the user's name.

Then from there we can look at our site and realize we've just made a box to put text in, we don't actually know what text should be placed within! We are then going to create a label to add text to this blank input box. We'll add a **for** value with the name of *Name* in order to link it to our input value's ID.

Furthermore, we'll continue to do the same for email and message.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Contact Us</title>
</head>

<body>
  <nav>
    <a href="club.html">Home</a>
    <a href="contact.html">Contact us</a>
  </nav>
  <section>
    <form>
      <h3>Questions? Contact us!</h3>
      <div>
```

```
        <label for="Name">Name:</label>
        <input type="text" id="Name" name="name">

        <br>

        <label for="Email">Email:</label>
        <input type="email" id="Email" name="email">

        <br>

        <label for="Message">Message:</label>
        <textarea name="message" id="Message" cols="30" rows="10">
</textarea>

        <br>

        <div>
          <p>How do you rate your experience?</p>

          <label for="Ok">Ok</label>
          <input type="radio" id="Ok" name="experience"
value="Ok">

          <label for="Good">Good</label>
          <input type="radio" id="Good" name="experience"
value="Good">

          <label for="Amazing!">Amazing!</label>
          <input type="radio" id="Amazing!" name="experience"
value="Amazing!">
        </div>
      </div>
      <button>Submit</button>
    </form>
  </section>
</body>
```

```
</html>
```

We are also going to define a new div here in order to create some selector buttons. We'll create some text here about rating the experience. Then we'll define inputs with the *radio* type - this adds these selector buttons where only one can be selected at once. We'll also add the value attribute in order to provide what this selector actually means.

We'll then fill out this form, open up the inspect element and open up the network tab. Here is where we'll see all of the **requests** that our web page makes. We are going to use it in order to see our form's request, since we don't currently capture it anywhere.

We'll open up our page and realize hey - our form works and is there! And we even have a navigational element in order to get back to our home page. Unfortunately we don't have a way to get from the home page to the contact page, so let's quickly add the same navigation bar to the `club.html` that we created earlier.

There we go - we've used some more HTML in order to create a separate page, and link them together.

CSS

CSS stands for Cascading Style Sheets and is the main way for a developer to be able to style their website. By styling we mean changing fonts, colours, image sizes, tables, etc. CSS will allow you to change pretty much anything that is displayed on your website in a near infinite number of ways.

Whether it's a programming language or not is actually quite the discussion, so we're not really going to get into that, but suffice to say it will be something that along with HTML you come back to often even when you're eventually developing in React.

Let's first open up a new folder called `club4` and copy over our club and contact pages to start styling them. We'll start by creating a style tag in our `<head>` tag. This is where we define our styling.

```
<style>
        /* This styling  */
        h3 {
                background-color: antiquewhite;
                font-family: sans-serif;
                padding: 10px;
        }

</style>
```

From here we can say hey we want to change all the h3s so we'll first type that. Open up some curly braces and here is where we can type our CSS. Generally it's a property followed by a colon followed by a value. In this case, we've changed the background colour of this element, changed the font family, and added some extra padding.

This may seem like a hassle as look we've changed **all** the headings to be this style! What if we only want one of them to be this way? From here, we can introduce the `<span>` tag. The span tag allows us to apply styling to a specific element or spot on our website. For example, if we grab the header for our page and change the first word, "People", to blue:

```
<h1> <span style="color:blue;">People</span> Watching Club!</h1>
```

Now we have the first word being blue, but the rest of it is fine!

You may also think that this is a bit cumbersome, we have to go through our entire HTML file and change all the styling of each individual element?

CSS also allows us to create stylesheets. Let's create one in this folder called `style.css`. In this file we are able to define CSS just like we did within the style tag.

We briefly said earlier that hey, you can grab the name of an HTML eleemelementapply some CSS to all instances of it on your site. There are other of selecting elements on your website to style with HTML. We are going to show some of these off within this file:

```
body{
        color: orange;
        font-family: sans-serif;
}
```

The body tag will change the styling for our entire website. In this case all of our text has been changed to orange and the `sans-serif` font family.

We'll notice that hey we saved our stylesheet, and it's not actually doing anything. We first have to tell our HTML where our stylesheet actually is. We'll go into our `club.html` file and add a link to our stylesheet, so our HTML page can import that styling. Think of this kind of like a python import statement.
`<link rel="stylesheet" href="style.css">`

We can also choose to apply styling based off of an HTML element's class. Any HTML element can be provided a class and all we have to do is find that element within our file and add that class attribute. For example, we can add the class "navbar" to our navbar in our `club.html` file.

From there if we head back to our `style.css` file we can style these by adding a period followed by the class name.

```css
.navbar{
    background-color: gray;
    width: 100%;
    padding: 10px;
}
```

We can then see the changes on our site.

We also have one more main way of selecting elements, and that's by their id. We saw earlier when we were creating a form that we can add an ID attribute to HTML elements. We can select based on ID by using a hashtag. If we were to say:

```css
#contact-form{
    background-color: rgb(41, 41, 139);
    color: white;
    padding: 10px;
    margin: 0px;
}
```

We can then go into our `contact.html` page and add an ID to the `<form>` of `contact-form` and see that our CSS styling will be applied to that element!
We'll realize that nothing happens... This is because we didn't link our stylesheet to our contact page. Let's link it

```html
<link rel="stylesheet" href="style.css">
```

We'll then switch to `club5` where we'll start with the final versions of `club.html` and `contact.html`. We're going to get into some of the cooler styling that CSS can provide.

First let's define some simple styling as a brief review for CSS.

```css
body{
    padding: 0px;
    margin: 0px;
    background-color: white;
    font-family: sans-serif;
}

.navbar{
    background-color: gray;
    width: 100%;
    padding: 10px;
}

.nav-link{
    color: white;
    font-size: 24px;
    text-decoration: none;
    padding-left: 40px;
}
```

Recall that `body` selects the entire body of the website. The `.navbar` tag will select anything with the class of `navbar`. Same with the `nav-link`.

Next we can define what happens when you hover over a specific element. It turns out, with CSS we can change the styling of our elements when we hover over them!

```css
.nav-link:hover{
    color: rgb(204, 85, 0);
}
```

We can then define the `.nav-link:hover`. `.navlink` will grab all of the elements with the `nav-link` class. Then `:hover` will change that styling when we hover over it.

We can see another example of this hover effect by using:

```css
.blog-image-link:hover {
  padding-left: 10px;
}
```

We can also select specific elements that are nested within each other. For example, we can grab anything with the `contact-form` class, and then pick the headings within there by using `#contact-form h3`.

```css
#contact-form h3 {
  background-color: inherit;
  margin-left: 20px;
  padding-top: 20px;
  border: 1px solid rgb(96, 96, 227);
}
```

We can combine the two ideas from earlier by also including a hover attribute:

```css
#contact-form button {
  background-color: rgb(87, 181, 91);
  color: white;
  font-family: sans-serif;
  padding: 5px 10px 5px 10px;
  border: 0px;
  margin-left: 20px;
}

#contact-form button:hover {
  background-color: rgb(49, 116, 52);
}
```

We could also define a cool thing called a gradient using CSS. Without using any Javascript we can create a linear rainbow gradient.

```
.rainbow {
  width: 24.5%;
  background: linear-gradient(90deg, red, orange, yellow, green,
blue, purple);
  background-clip: text;
  -webkit-background-clip: text;
}
```

The angel is the gradient line's angle of direction. We can see that it's moving at a 90 degree angle. We have then listed our colours. We've also defined some selectors to ensure that the gradient is assigned to the text.

We also notice that... there is no change. We've created a background gradient of a rainbow, but we still have the text colour being fully shown. If we define the hover value to make the colour of our text transparent, we'll be able to actually see the gradient!

```
.rainbow:hover {
  color: transparent;
  transition: 200ms ease;
}
```

Now we'll notice when we hover over that text it will become a rainbow!

We can check out the final versions of our program with `club.html` and `contact.html`. It will import from the styling page, and we can see that our website looks like the following!

Now we've gone over the different ways in which we can select different CSS elements. If we open up the CSS folder we'll notice some pre-defined files. We know that we can define CSS within HTML files by doing it in-line, where we place it right on the element, as well as creating a style tag at the top of our file.

If we check out `inline_messy.html` we'll find a version of the club website using entirely inline styling. On first glance it looks exactly like the final club page that we looked at earlier. But if we look a little closer we'll notice some of the reason that we don't only use inline styling. Our previous rainbow hover effect is not possible because we cannot define these CSS hover effects inline.

While lacking functionality is a good enough reason to not pursue inline CSS another one is that it's simply messy. If we look at this file it's hard to read and difficult to find the sections within for our website. While this is *kind of* a matter of personal preference, in general when we can split things into different files it will make things more readable in the long term.

We can also check out `messy.html` in which we've just dumped our entire CSS styling page into a style tab at the top of the page. While this works exactly as importing a stylesheet you can see upon opening up this HTML file why we might not want to do this. You can imagine in a professional setting that this HTML file would become extremely large.

Then we'll show of a current version of a portfolio from one of our directors, Sky: https://schuylergood.com/

Notice how this entire website is written in HTML & CSS. I highly suggest you start writing your own personal site or personal portfolio as we move through this process. That way, by the end, you'l have some web development skills and a website!