Recap

Last week we went over version control using git and linking it to a remote repository → specifically GitHub. We then created our first react project using `npx create-react-app` and showed off how components work in order to reuse our code in multiple locations.

Today

How a React Project works

Today we're going to take a brief recap of everything that we've done previously by walking you through how we're going to create a club's home page in react.

First, we're going to run `npx create-react-app club0` in order to create a brand new react project called `club0`. We covered this last time but remember that this is the command you're going to want to run when you create a react project. I've already prepared a blank react project to work from so we don't actually have to wait for this to run. If you didn't attend last time, you'll need to install node.js before you're able to run this command in your terminal.

From here we'll run our example react-app using `npm start`. Let's dive deeper into how the structure of our project actually works. This may seem boring at first glance but fully understanding this will help us develop going forward. Last time we breifly showed you how components work in order to get you interested in why we use react.

First, if we go into the `public` folder and then `index.html` we'll find the very first HTML file that is called when our site is created. If we went

into this file and changed the tab title for example, we'd see it update on our actual site. Now that we've established that our react app starts from this file, we're also briefly going to look at this div created within the `body`. It turns out that this is the "root" div, essentially everything created with React is going to be stored within this div, and we could apply styling to it if we so chose.

This folder also has some icons, and a `robots.txt` file but we're not going to worry about those right now. If we then checkout `index.js` found within the `src` directory we'll see the first Javascript that is ran when our app is started. First we import some things from React, we grab our css file from `index.css` (recall that we no longer have to link the CSS and HTML manually, it can be done for us with that import statement!). We then import our App component from `App.js` and these web vitals. For now we're going to delete this web vitals import, remove the extra function call and the file to make this a little bit simpler.

Last time we brushed over this root constant but it's the only thing that `index.js` actually does. What we do is we use our ReactDOM in order to create our root item, we'll then pass it in the actual HTML componenet on our page to link it to. In this case we're going to grab the element with the ID of root. Turns out that is the empty `div` that we saw in `index.html`!

Now that we've grabbed our root element on our site we want to actually "render" something within it. We can call `root.render` in order to do so. We then provide it with what we actually want to render. Notice how if we delete this and replace it with something else, our website will update and most importantly these elements will be placed within that root div!

Let's restore what was there before. The React.Strictmode tag is only for development mode, and it just tells React that it's okay to display errors to us. Then we provide our App component, and recall last time that a component is just a set of defined HTML, and CSS usually written in JSX that allows us to reuse it and call it anywhere. The best way that I can display that this is true would be to just grab App and display it twice. We'll see that our web page is just mirrored twice. And it turns out that `App` isn't actually anything special, it's just a component defined in another file in order to simplify `index.js` and make our lives easier.

If we then peak at `App.js` we'll start to see what we saw last time, and the fact that this really is just a component. We import our CSS which will automatically apply the stlying, and we grab the logo which is stored in the `logo` variable. From there, we create this function called App, define some JSX within it, and export it for use within `index.js`.

Simple Club Website

From here, as much as this is mostly review we are going to work in order to create a simple version of our club website in React. We'll run `create-react-app` once again in order to create `club1`. I've already ran it before hand, so we'll open up that folder. I've completed some minor changes in order to set our site up for our development.

1. Removed the extra icons and files from `public`
   1. Deleted the links to extra things, and renamed the title within this file to `People Watching Club`
   2. Removed the `App.test.js`, `logo.svg`, `reportWebVitals.js`, and `setupTests.js` files

   If you want to start from this same position, you can find this under the `week5` folder on our Github, in the `club1_before` folder.

From here, our plan is to build out a home page today, so we're going to make a new folder called pages, and create a file within called `Home.js`. We also have some graphics that we're going to use for this home page, I'll just copy them in from the final version of our site. We've stored them in an `img` subdirectory. Within it we have a `carousel` directory with a few images in it. We'll see those later. We will also grab our CSS folder which has a few CSS files in it that we'll look at later. For now, know that all of this is just us already having some images and pre-written CSS, you are capable of grabbing all of this.

First we're going to start by attempting to create an image carousel. Essentially a few images that a user can click through on our home page.

We'll first start by creating a `Home` function:

```
function Home() {
}
```

Within it we are going to create the bare-bones of our image carousel and some other random text and information. We'll start by returning some JSX, and we'll use this empty component `<></>` that we haven't shown before. Essentially, this is kind of like a div, but it won't take on any of the custom styling that we may have defined for divs.

```
function Home() {
  return (
    <>
      <section className="carousel">Carousel</section>
      <section className="about-section">
        <div className="info-card">
          <div>
            <span className="section-header">About QPWC</span>
            <p>
```

```
              At QPWC, our mission is to turn everyone into first
class
              stalkers. When you first join our squad we provide
you with free
              binocolurs to get started! Not hooked yet, well see
our upcoming
              events below!
            </p>
          </div>
        </div>
      </section>
    </>
  )
}
```

Now that we've defined the home page, we need a way to actually view it. Our site is still displaying the previous template. Let's head into `App.js` and remove all of the JSX that is currently being returned. Instead, we want to return our home page, `<Home />`. We'll also need to import it using `import Home from './pages/Home'`. Remember that we need to include an export statement in our `Home.js` file, `export default Home`.

Suddenly, our site works! Our text is displayed! We want to apply the CSS styling that we defined within the `CSS` folder, so we're going to need to import it in our `Home.js` file: `import '../css/Homepage.css'`. We'll notice the second we save our file that our site updates to use our styling.

Currently, we've just defined our carousel as some text. We're probably going to want to create a component for it just in case we want to use it somewhere else on our site. We'll create a components folder and create `Carousel.js`.

```
const Carousel = (props) => {
}
```

We've included props because we're going to want to take some props containing the images that we want to display.

State

We are then going to briefly introduce something called `state`. This is how we can create a variable that our users can actually interact with. We define our state like: `const [state, setState] = useState(0);`. We are then going to need to import setState:
In this case, `state` is our variable in order to access the value
setState is actually a function that we can call in order to actually set the state
`0` is the starting value of our state
We are using destructuring in order to take out the `state` and `setState` values directly into variables. What `useState` actually returns is an array with two values in it, we then are automatically turning it into these two constants.
We can then return some JSX, and in this case we're actually just going to create a `div` and within it we'll include the `state` value to see it:

```
import { useState } from 'react'

const Carousel = (props) => {
  const [state, setState] = useState(0)

  return <div>{state}</div>
}
```

We will then display the Carousel on our Home page by importing it: `import Carousel from '../components/Carousel'` and then add it to

replace the text: `<Carousel />`. We'll need to make sure we export our component as well using `export default Carousel`.

We'll then see on our site that we have a 0 at the very top which is displaying the current state.

We can then define a button in order to increase this number. Recall that we can use `onClick` in order to define what happens when you click that button. We have to define a function here to be called, or provide what function we want called. In this case we are going to set our state to 1 whenever we click this button. We'll call the setState function, and then set it equal to 1. Now when we check out our site, if we click this button we see that our value updates to `1`.

```
import { useState } from 'react'

const Carousel = (props) => {
  const [state, setState] = useState(0)

  return <div>{state}
  <button onClick={() => {setState(1)}}>Click to increase!
</button>
  </div>
}
export default Carousel
```

We'll notice that hey we aren't actually increasing it, as if we continue to press the increase button nothing happens. So let's try to make this more dynamic. We'll replace that 1 to `state + 1`. Recall that `state` holds the current value of the state, and then we'll increment it by 1 `setState(state + 1)`
Our button will now continue to increase!

We'll remove our example and start to define how our carousel is going to work. We'll create a state `const [currentIndex, setCurrentImg] = useState(0);`. We're setting it to 0 by default as the first image. We'll then return some JSX:

```
import { useState } from 'react'

const Carousel = (props) => {
  const [currentIndex, setCurrentImg] = useState(0)

  return (
    <div className="slider-styles">
      <div className="carouseInner">This is going to store our
image
      <div className="left">This will be our left button</div>
      <div className="center">
        <div className="text">
          <h1>
            QUEEN'S UNIVERSITY
            <br />
            PEOPLE WATCHING CLUB
          </h1>
          <p>This will hold our image's text</p>
        </div>
      </div>
      <div className="right">This will be our right button</div>
          </div>
    </div>
  )
}
export default Carousel
```

We are then going to want to start to make this carousel actually work, and to do so we need to define our image prop. We'll head back to our home page and first import our three images:

```
import carouselImg1 from "../img/carousel/carousel1.png"
import carouselImg2 from "../img/carousel/carousel2.png"
import carouselImg3 from "../img/carousel/carousel3.png"
```

Next, we'll create some json in order to hold all of this information:

```
const carouselImages = [
    {
      id: 1,
      src: carouselImg1,
      alt: "carousel image 1",
      subtitle: "Have a seat and watch the world go by...",
    },
    {
      id: 2,
      src: carouselImg2,
      alt: "carousel image 2",
      subtitle: "It's a good day, let's see...",
    },
    {
      id: 3,
      src: carouselImg3,
      alt: "carousel image 3",
      subtitle: "Perfect sunny day to watch people...",
    },
  ]
```

Now we'll pass in our `carouselImages` as props:

`<Carousel images={carouselImages} />`

Heading back over to our carousel, we'll have to figure out a way to dynamically display these items based off the index. At the very beginning, we'll just want to display our image to see how this works. First, we'll print out what `props.images` actually looks like and check out our console: `console.log(props.images)`. We'll see our array with our images within it!

Then we can get at an individual image using: `<img src={props.images[0].src} />`
Our image shows up! Let's delete that and begin to build our carousel. First, we're going to want to set the background image, since in a carousel things are kind of in the background:

```
<div
        className="carouseInner"
        style={{ backgroundImage: `url(${props.images[0].src})` }}
      >
```

We are setting the background image, we are then calling a function called url which takes in a url and formats it correctly. Then we grab the url to our image.

Now we've setup our background image! Our site looks a bit interesting but let's start by defining one of our buttons:

Our left button is going to decrease the image index that we're currently on, unless we're at the far left image in which case the button won't change anything. First we'll define or onClick:

```
<div
          className="left"
          onClick={() => {
            currentIndex > 0 && setCurrentImg(currentIndex - 1);
          }}
        >
```

we use this neat inline if statement where the second value will only run if the first value is true. Then we set the actual value to `&#60;` which is a left arrow! We click our button and... it doesn't do anything! Let's start by setting our value to `1` and see what happens when we click that back button.

We'll go down to our right button and define it in a similar function!

```
<div
          className="right"
          onClick={() => {
            currentIndex < images.length - 1 &&
setCurrentImg(currentIndex + 1);
          }}
        >
          &#62;
        </div>
```

We'll change the text to update to the image as well.

`<p>{props.images[currentIndex].subtitle}</p>`

We now have a basic image carousel and a very rudamentory version of our site!