Intro to React

What is React, you may ask? Well, it's a "declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components". A React component class, or React component type takes in parameters, called props (short for "properties"), and returns a hierarchy of views to display via the render method. The render method returns a description of what you want to see on the screen. React takes the description and displays the result. In particular, render returns a React element, which is a lightweight description of what to render. Most React developers use a special syntax called "JSX" which makes these structures easier to write."

What's it good for? Simplifying some JavaScript projects into a single page that use repeated bits of code and want some functionality applied to them. In this project, we'll be setting up a basic system for storing and displaying a collection of information, such as a collection of DVDs.

Overview

First we will set up React, then create the Components we need (Collection and Disc), then create functions that will allow us to use them effectively.

Setting up a React App

Detailed instructions here

With Node.js installed, access the folder you want to create the app in in the terminal then run (with "my-app" replaced by your project's name, i.e. disc-collection):

```
npm install -g create-react-app
create-react-app my-app
cd my-app
```

This creates the basics of a project, and will pull up a sample of project in your browser.

Creating the Disc

The basic structure for React is already set up in App.js - a Component called App, with a render function inside. As you can see, this render function returns what looks like some HTML tags with some attributes. The "className" attribute is React's version of the basic HTML "class" attribute and must be used in place of it.
Let's recycle the code here but customize it to make it a Disc component.

1. Delete the logo import at the top, as well as the content within the div
2. Replace "App" with "Disc" after class and in the className of the div
3. Use your HTML knowledge to create some data points you'd like to catalog for each disc, and remember to use the className attribute instead of class for styling purposes.

a. I'm using a bootswatch library to help styling go smoother, you can do the same by adding these to the top of App.css

```
@import
url(https://bootswatch.com/4/litera/bootstrap.min.css);
@import
url(https://bootswatch.com/4/litera/_bootswatch.scss);
```

4. This is what my HTML structure looks like:

```
class Disc extends Component {
  render() {
    return (
      <div className="Disc card text-white bg-primary mb-3">
        <div className="card-header">
          <h2 className="Disc-title card-title">Sample Title</h2>
        </div>
        <div className="card-body">
          <ul className="Disc-info card-text">
            <li className="Disc-type">Type: DVD</li>
            <li className="Disc-year">Year: 1997</li>
            <li className="Disc-genre">Genre: Action</li>
            <li className="Disc-rating">Rating: PG-13</li>
          </ul>
        </div>
      </div>
    );
  }
}
```

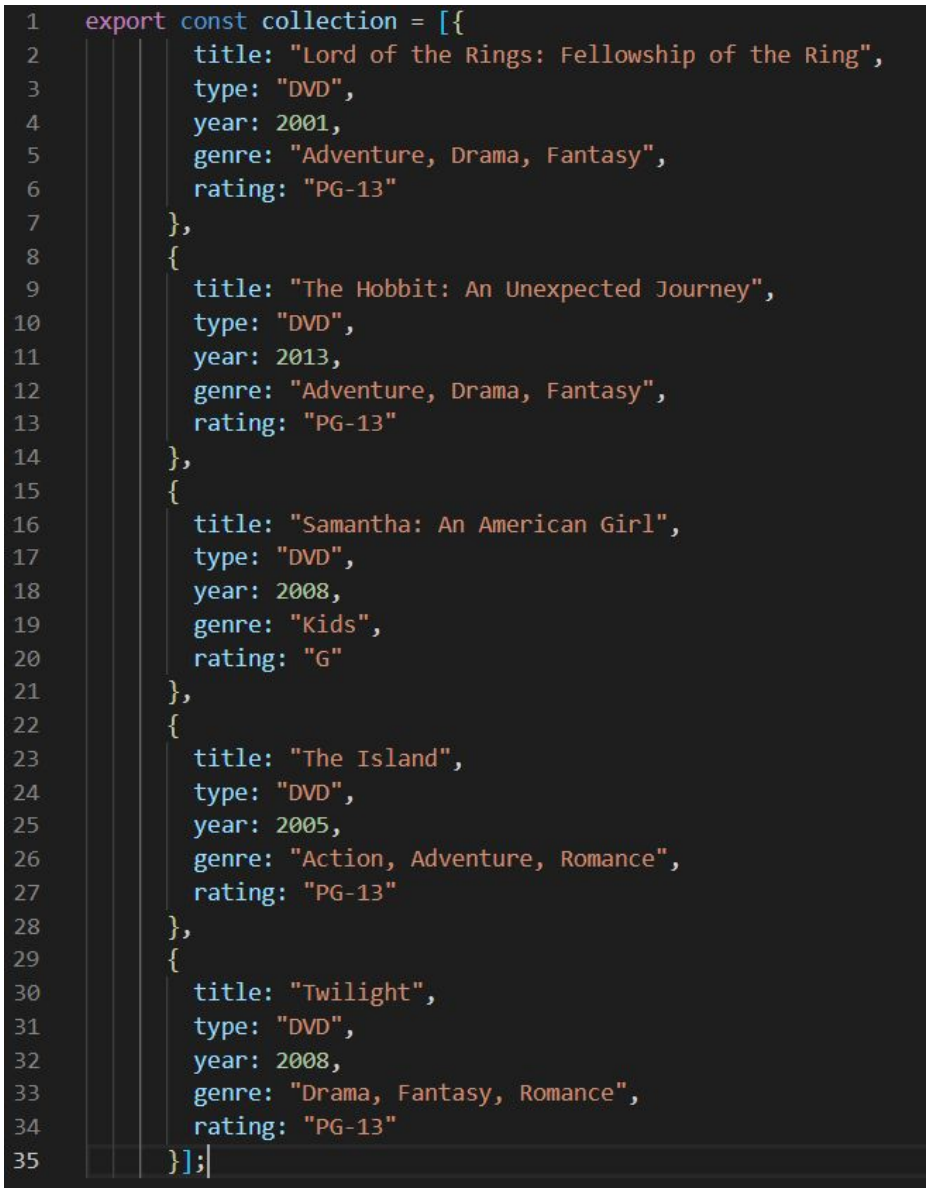
5. Nothing is showing up if you try to preview it right now, but don't worry, we'll fix that soon. Time to create the next component!

Creating the Collection

1. Create another component, but this time called Collection. This will house a group of the Disc components, so in the render function return a div with className Collection, and inside include any navs, headers, or other divs you want.
2. At the end of the file is a line that needs "App" to be changed to "Collection" so the file knows to render this component and not the old "App" one:
   `export default App; => export default Collection;`
3. Another addition to get it running again is to import the ReactDOM at the top of the file:

```
import ReactDOM from 'react-dom';
```

4. Obviously we want to have more than one disc in the collection, so let's make a file that we can house multiple discs in. In the src folder create a new file called data.js and set up an array of objects containing the information you defined in the HTML of the Disc component. Mine looks like this:



```
export const collection = [{
        title: "Lord of the Rings: Fellowship of the Ring",
        type: "DVD",
        year: 2001,
        genre: "Adventure, Drama, Fantasy",
        rating: "PG-13"
      },
      {
        title: "The Hobbit: An Unexpected Journey",
        type: "DVD",
        year: 2013,
        genre: "Adventure, Drama, Fantasy",
        rating: "PG-13"
      },
      {
        title: "Samantha: An American Girl",
        type: "DVD",
        year: 2008,
        genre: "Kids",
        rating: "G"
      },
      {
        title: "The Island",
        type: "DVD",
        year: 2005,
        genre: "Action, Adventure, Romance",
        rating: "PG-13"
      },
      {
        title: "Twilight",
        type: "DVD",
        year: 2008,
        genre: "Drama, Fantasy, Romance",
        rating: "PG-13"
      }];
```

5. Now to import this data into our app, we need to put this command at the top, which also declares it as a variable called DiscData

```
import * as DiscData from './data';
```

6. To use the data, we set the component's state variable to have a discs field with the value of DiscData.collection. We set the stat variable in the

constructor for Collection. This is what the file looks like at this point:

```
class Collection extends Component {
  constructor(props) {
    super(props);
    this.state = {
      discs: DiscData.collection
    };
  }

  render() {
    return (
      <div className="Collection">
      <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
      <h1 className="navbar-nav mr-auto">Tasha's Disc Collection</h1>
      </nav>
      <div className="cards">
      </div>
      </div>
    );
  }

}

export default Collection;
```

7. Now we need to be able to render a Disc inside the Collection, which just returns a tag version of the disc component with each property of the data fields you defined as an attribute set to the current disc's field:

```
  renderDisc(disc) {
    return <Disc key={disc.title} title={disc.title} type={disc.type}
    year={disc.year} genre={disc.genre} rating={disc.rating}/>;
  }
```

8. To go through all of the discs in our data.js file, we'll use the map function of React, which runs the given function for each item of the array, and then call our renderDisc function inside it as an anonymous function. Doing this in a container in the Collection's render function will give us all of the discs in the file:

```
        <div className="cards">
          {this.state.discs.map((disc) =>
            this.renderDisc(disc)
          )}
        </div>
```

9. To get the custom data to show up, we need to update our li in the Disc to call the specific props for each iteration of the data.js collection:

```
class Disc extends Component {
  render() {
    return (
      <div className="Disc card text-white bg-primary mb-3">
        <div className="card-header">
          <h2 className="Disc-title card-title">{this.props.title}</h2>
        </div>
        <div className="card-body">
          <ul className="Disc-info card-text">
            <li className="Disc-type">Type: {this.props.type}</li>
            <li className="Disc-year">Year: {this.props.year}</li>
            <li className="Disc-genre">Genre: {this.props.genre}</li>
            <li className="Disc-rating">Rating: {this.props.rating}</li>
          </ul>
        </div>
      </div>
    );
  }
}
```

10. Finally, to get this to all show up in a browser, we need to render the Collection tag on the DOM, so put this at the end of the file before the export command:
    `ReactDOM.render(<Collection/>, document.getElementById("root"));`
11. Now we get something in the browser! Cards for each object in the array now appear, with the custom data. From here, it's up to the imagination to add styles in the App.css file. There is a lot more that React can do, including sort and add new discs all without refreshing the page, making it a great tool for dynamic data. To see my code for implementing sorting, styling, and adding,

check out my GitHub repository below!

Resources/Further Reading

My Full Implementation: https://github.com/tashasimone/disc-collection
https://reactjs.org/tutorial/tutorial.html
https://www.tutorialspoint.com/reactjs/index.htm
https://scotch.io/tutorials/learning-react-getting-started-and-concepts
https://reactjs.org/docs/getting-started.html