BUILDING FRONT-END APPLICATIONS WITH REACT

Welcome to the first installment of the Apollo project! You’ll build a website called “Apollo”, a Yelp-like clone.

This is a working sample of Project: [Apollo](https://jkey774.github.io/codecademy-ravenous/)  
In total, there will be four parts to this project:  
• Creating Static Components  
• Passing Information to Components  
• Setting the State of Apollo Components  
• Interacting with the Yelp API

Here’s a quick overview of how Apollo should function:  
• As a user, I should be able to search for restaurants.  
• As a user, I should be able to view a list of restaurants returned by  
the Yelp API.  
• As a user, I should be able to sort through restaurants using a filter.

The four steps will test your knowledge of JavaScript and React, all with the goal of building a Yelp-like clone.

In each step, you’ll be presented with the intended, final outcome  
(of that project) in the browser.

If you want to get a feel for what Apollo can be, visit the Yelp website and search for restuarants in your area.

Today, you will start by building the first part of Apollo: Creating Static Components.  
Let’s get started!

1.Start by creating a React app using the create-react-app package in your  
preferred terminal. The name of your app should be Apollo.

2.In terminal window, cd into the Apollo directory and type the following command: npm start

3.Replace the current favicon located at public/favicon.ico with the icon in your Project folder. Make sure you save the new image as favicon.ico.

4.By default, create-react-app will generate a sample application. We’ll need to add a reset.css file to ensure our app is styled the same in every browser.  
Take a look at the Project folder sended to you. Copy reset.css in to the public/ directory of react-app.

5.Link to reset.css in the index.html file that’s also located in the public/ directory. Keep index.html open.

6.You’ll also need to add a couple of Google fonts that the app will use.  
In index.html, add the following links to add the necessary Google fonts:  
<link href="https://fonts.googleapis.com/css?family=Work+Sans:300,500,600"  
rel="stylesheet">  
<link href="https://fonts.googleapis.com/css?family=Poppins:600"  
rel="stylesheet">

7.Great, now let’s move on to some actual React code! Apollo will be composed of four different components interacting with each other:  
• <Business />  
• <BusinessList />  
• <SearchBar />  
• <App /> (created by default with create-react-app)  
You’ll create all four components in this project, one by one. You can expect them to be static components for now — you’ll add additional functionality later.  
First, you’ll need to structure the Apollo directory differently.  
cd into the src/ directory. Create a new folder called components. You’ll store all of your components in this directory.

8.Each component should have its own folder inside of components/. Inside of the components/ directory create a new folder called App.  
Move App.js and App.css to the App/ folder and update the path in index.js accordingly.  
Remove App.test.js from the src/ folder, as you will not use it in this project.

9.Next, let’s create the <Business /> component.  
Inside of components/, create a new folder called Business. cd into the Business/ directory and create two files:  
• Business.js• Business.css

10.Great! Now you’re ready to begin creating your first React component. The purpose of the <Business /> component is to represent how a business (a restaurant) in Apollo will be formatted and styled. For now, we’ll hard code a single business listing. Later, you’ll update it dynamically using the Yelp API.  
Open Business.js in your text editor.  
Create a JavaScript object called business. Copy the following keys and values and add them to business:  
imageSrc: ./pizza.webp',  
name: 'MarginOtto Pizzeria',  
address: '1010 Paddington Way',  
city: 'Flavortown',  
state: 'NY',  
zipCode: '10101',  
category: 'Italian',  
rating: 4.5,  
reviewCount: 90

Note: Copy “pizza.webp” file to your **puclic/** folder.

11.Perfect. The information in this object will be consumed by the <Business/> component. To build the component using React, you’ll need to import the React library.  
At the very top of your Business.js file (line 1), import React.

12.Next, below the business object, create a React component called Business. The component should extend React.Component.

13.  
Add a .render() method to the Business component.

14.Inside of the .render() method, add a return statement with JSX that renders this HTML.:

<div className="Business">

      <div className="image-container">

        <img src={imageSrc} alt={category}/>

      </div>

      <h2>{name}</h2>

      <div className="Business-information">

        <div className="Business-address">

          <p>{address}</p>

          <p>{city}</p>

          <p>{state} {zipCode}</p>

        </div>

        <div className="Business-reviews">

          <h3>{category}</h3>

          <h3 className="rating">{rating} stars</h3>

          <p>{reviewCount} reviews</p>

        </div>

      </div>

    </div>

• Do not change the class values, as we will use them in the next step to add style to the business component.  
• Replace the relevant information with references to properties in the business object (i.e. {business.thisProperty}).

15.You added a bunch of className attributes to the HTML elements in the Business component, but they won’t be very useful if we don’t style those elements with CSS!

Copy Business.css file in to the Business/ directory.

16.At the top of Business.js (line 2), import the Business.css file.

17.You just built your first React component! Congratulations! All that’s left to do is to make this component available to the rest of the Apollo app. At the bottom of the file (last line), export the Business component.

18.In the components/ directory, create another folder called BusinessList. Inside of BusinessList, create two files:  
• BusinessList.js• BusinessList.css

19.Add the following CSS to BusinessList.css:  
BusinessList {  
display: flex;  
justify-content: space-around;  
flex-wrap: wrap;  
margin: 4.4rem 10%;}

20.At the top of the BusinessList.js file, import the following (in this order):  
• The React library  
• BusinessList.css

21.The point of the <BusinessList /> component is to simulate what a returned list of businesses would look like in Apollo (after querying the Yelp API, for example). To help this simulation, <BusinessList /> will make use of the <Business/> component repeatedly. To use the <Business /> component, you’ll have to import it. On the next line, import the <Business /> component.

22.Let’s start building the component. Use the React library to create a component called BusinessList.

23.Inside of the component, add a render() method.

24.Inside of the .render() method, add a return statement with JSX that renders this HTML.:

<div className="BusinessList">

     <Business />

  </div>

25.That’s it for the BusinessList component! This list component will need to be rendered again by another component, so you’ll need to export it. At the bottom of the file, export BusinessList.

26.To search for businesses (restaurants) in Apollo, you’ll need a search bar. We’ll implement this with a SearchBar component.  
In the components/ directory, create another folder called SearchBar.  
Inside of SearchBar, create two files:  
• SearchBar.js• SearchBar.css

27.Copy SearchBar.css file in to the folder. There are the two images you’ll need for the Project also.  
The “mobile” version should be used in any media queries you write.

28.  
Great! Now you’re ready to start building this component. Start by opening SearchBar.js. At the top of the file, import the following (in this order):  
• The React library  
• SearchBar.css

29.The search bar will communicate with the Yelp API, but you’ll build the  
functionality to communicate with the API in a later steps. Today, you’ll build part of the structure that’s needed to communicate with the Yelp API.  
Specifically, the search bar should allow users to search businesses by:  
• Best Match  
• Highest Rated  
• Most Reviewed  
To achieve this, you’ll create an object with keys and values that conform to what the API expects to receive (as shown in the documentation provided above). Let’s see what this looks like.  
Start by creating an object called sortByOptions.

30.The object should have three keys (in this order):  
• Best Match  
• Highest Rated  
• Most Reviewed  
Make sure the keys are strings.

31.Next, set the values. Use [the documentation](https://www.yelp.com/developers/documentation/v3/business_search) to help you set the values of those keys. The sort\_by entry in the table of the “Parameters” section will be helpful. The values should be strings.

32.Let’s start building the search bar component. Use the React library to create a component called SearchBar. Don’t add a render() method just yet.

33.Create a method called renderSortByOptions() in the body of the component declaration.

34.The purpose of renderSortByOptions() is to dynamically create the list items needed to display the sort options (Best Match, Highest Rated, Most Reviewed).

The method should iterate through the keys and values of the sortByOptions object and return a list item. The list item elements should use the keys as an attribute, and the values as content. Let’s start building it out.  
Start adding a return statement (no parentheses).

35.To iterate through the object, you’ll need to start by accessing the keys of the sortByOptionsobject. Call the keys() method on the JavaScript Object library. Pass in sortByOptions as the argument.

36.Now that you have access to the keys, you’ll iterate through them using the map() method. Call the map() method by chaining it to the end of the line you just wrote.

37.Pass a callback function to the map() method as an argument. The callback function should have one parameter called sortByOption. The callback function should also use arrow syntax.

38.renderSortByOptions() {  
return Object.keys(sortByOptions).map(sortByOption => {  
});  
}

39.Now let’s store the object values in a variable. Inside of the callback function, access the sortByOptions values using the sortByOption parameter of the callback function. Store values in variable called sortByOptionValue.

40.On the next line, return a <li> element. The list item should have an attribute called key set to sortByOptionValue (don’t forget to use curly braces to inject JavaScript).

The content of the list item should be sortByOption. Again, use curly braces to achieve the JavaScript injection.

41.Great, you’re done with renderSortByOptions()! You’ll use this method when building the structure of the search bar component.

42.Let’s build the structure of the <SearchBar /> component. Start by adding a render() method.

43.Inside of the .render() method, add a return statement with JSX that renders this HTML.

<div className="SearchBar">

        <div className="SearchBar-sort-options">

          <ul>

            {this.renderSortByOptions()}

          </ul>

        </div>

        <div className="SearchBar-fields">

          <input placeholder="Search Businesses" />

          <input placeholder="Where?" />

        </div>

        <div className="SearchBar-submit">

          <a href='www.#.com' >Let's Go</a>

        </div>

      </div>

44.  
That’s it for the SearchBar component! This list component will need to be rendered again by another component, so you’ll need to export it. At the bottom of the file, export SearchBar.

45.It’s time to put all of the components together!  
Open the App.js file in the default App/ directory.  
At the top of App.js, import the the following (in this order):  
• The React library  
• App.cssMake sure that App.css has the file you have in your Project folder.

46.You’ll need to import the <BusinessList /> and <SearchBar/ > components too. Next, import BusinessList and SearchBar. You’ll have to use the correct path to import them.

47.The create-react-app command creates a default App component for you. It includes a render() method along with a return statement. We won’t need the default App component, so let’s make some modifications. Delete everything inside of the return statement.

48.Inside of the App.js .render() method, return JSX that renders this HTML.

<div className="App">

        <h1>Apollo Project</h1>

        <SearchBar />

        <BusinessList />

      </div>

49.By default, the create-react-app command adds a default export in the class  
declaration of the App component. To be consistent across with how components are exported in the Apollo project, remove the export default from the beginning of the class declaration. Instead, export App at the bottom of the file (you’ve done this for all components so far).

50.Congratulations! You completed the first part of the Apollo project. Let’s briefly review what you built:  
• A Business component that simulates a single business.  
• A BusinessList component that simulates a list of single businesses.  
• A SearchBar component that will be used in the future to search for business.  
The App component renders a SearchBar component and a BusinessList component. The final product is a simulation of how the Apollo project will look and work. Some functionality (like querying the Yelp API) is currently missing, but you’ll build it in the upcoming steps.