**Passing Information**

Welcome to the second part of the Ravenous project! Take a minute to review what you accomplished in the first part of this project. You:

* Used create-react-app to start your project
* Built all of the components you’ll need for the remainder of the project
* Simulated what Ravenous should look like after returning search results

In this project, you’ll complete the second part of Ravenous: **Passing Information to Components**.

So far, you’ve hard coded business information and rendered it manually. Now we’ll focus on how information (like business information) can pass from component to component with less hard coding. This is crucial as we build towards the goal of constructing a Yelp-like clone.

A few notes before getting started:

* You should expect to spend more than 1 day on this specific project. It’s OK if it is not all finished in one day.
* If you don’t understand how to implement a certain part of the project, we’ll provide guidance as needed. However, you should expect to search Codecademy for the exercises that will provide you with the relevant information.

Let’s get started!

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**0/10Complete**

Mark the tasks as complete by checking them off

**Get Started**

**1.**

Future parts of this project will implement the Yelp API. That specific functionality will require that you pass information from the Yelp API to multiple components. In this project, you’ll build the initial portions that will set your project up for this functionality in the future.

**2.**

In another terminal window, cd into the Ravenous directory and type the following command:

npm start

This command will start a development server and open up a preview of your app in a browser tab. As you build, the preview will automatically update when you save your work. This is a great way of seeing your progress in real-time.

**Create businesses Array**

**3.**

When the Yelp API is implemented, a list of businesses will be returned to **App.js**. Therefore, we should create a hard-coded list of businesses there and then pass them to the appropriate components.

Open **Business.js**. Cut the entire business object and paste it into **App.js.**

Make sure to paste it right after the imports and above the App class declaration.

**4.**

Take a look at **BusinessList.js**. What do you notice about the contents of the BusinessList div?

There’s a lot of repetition here. Specifically, the <Business /> component is repeated six times. Recall that we manually did this in the BusinessList component so that we can simulate a list of returned business. We’ll refactor this to remove some of the component repetition.

Go back to **App.js**. Under the business object, create a businesses array. The array should hold six references to the business object.

Hint

const businesses = [

business,

// Make sure to have six in total

];

**Setting a BusinessList Prop**

**5.**

Remember, our goal is to remove the repetition of components inside of the <BusinessList /> component. Let’s add some more functionality to aid this goal.

In **app.js**, add a businesses property to the <BusinessList /> component inside of the render() method. Set the property equal to the businesses *array*. (Don’t forget your curly braces!)

Hint

Here’s an example of a component with a property:

<ComponentName propName={value} />

**Add Functionality to the BusinessList Prop**

**6.**

When the businesses prop is set in the BusinessList component, there should be some functionality behind it. Specifically, it should iterate through the businesses array. Recall that you created an array of businesses in **App.js**.

Open **BusinessList.js**. Remove everything inside of the BusinessList div (the repeated components).

Inside of the BusinessList div, access the businesses prop and call the .map() method on it.

Hint

<div className="BusinessList">

{

this.props.businesses.map();

}

</div>

**7.**

Inside of the map() method, pass a callback function with one parameter called business.

**8.**

The callback function should return a <Business /> component. The returned <Business /> should have a property called business. Set the property equal to the parameter of the callback function. (Don’t forget your curly braces and semicolon!)

**Modifying Business.js using Props**

**9.**

Let’s review what you’ve done so far:

* Moved business information to **App.js**
* Created a list of businesses using an array, rather than repeating components in **BusinessList.js**
* Built functionality for the BusinessList component to iterate through the array and return Business components

You’re just about done! With the changes you’ve made, however, the Business component will have to access business information differently. This is what you’ll build next.

Open **Business.js**. What do you notice about how information is accessed inside of the return statement?

Because the business object was removed, statements like the following no longer have any meaning:

{business.imageSrc}

Business information now has to be accessed via the business prop you set in **BusinessList.js**. Modify all statements (like the one in the example above) by prepending them with this.props.

**Completion and Reflection**

**10.**

Congratulations! You completed the second part of the Ravenous project. Let’s briefly review what you did:

* Moved business information to the container component (App)
* Passed information from parent components (App) to child components (BusinessList, Business)

This is typical of the flow of information in React apps. You’ll continue to build on this structure as you move on to future parts of this project.

Finally, reflect on the knowledge you used in this project. As you do so, consider the following questions:

* The “Let’s Go” button doesn’t do anything at the moment. How might you simulate a query to the Yelp API?
* A user may decide to search with a different sorting option (for example, “Highest Rated”, or “Most Reviewed”). How can you handle this change in *state* using React?

We’ll expand the search bar in the coming weeks to address the questions above (and more).