Day 7 – Data Driven

**Make sure everyone is comfortable to move on**

**Everything for today will be in the Day 3 folders**

Today we are going to learn how to work with data in React. So far, we have hard coded everything but if you think about it, that’s not how we would do things right? We would want to pull data from a database. Well, today we are going to make cards and render some data inside those cards.

**Open Starters Day 3 cards.**

Frist we are going to go over some things from yesterday. Let’s start a new project and call it cards. Get rid of anything we do not need in App.js, and let’s start with a nav bar. I also have some images in an image folder that we are going to be using for this project.

Make a **components folder** and make a new file named **NavBar.js**. Import React, add a nav element, and in that nav element we will add an img tag. I am going to use the AirBnb logo image. Let’s import the image and add it to the src like an object. You have to surround objects with {} in React.

The image is going to be huge but don’t worry we will fix it with some css. Don’t forget, we need to render our navbar in app.js. Add a nav element to our css file, and set the height of the av element to 70px. Let’s also set the navbar to display: flex. This will set the nav element to use a flexbox and will automatically resize what we put into the navbar to be within its boundaries. As you can see, the image has now resized itself to the navbar because of that flexbox.

Now because it is in a flex box all we have to do is add some padding of 20px for the top and bottom and 36px for the sides and now it looks pretty normal. Let’s add a box shadow to make the navbar standout a bit. box-shadow: 0px 2.98256px 7.4564px rgba(0, 0, 0, 0.1).

Now let’s build a hero component. Hero, in web design basically means when websites have an oversized image at the top of their website.

Make a new file under components called **Hero.js.** Import React, lets do a section element, and in that well add an img tag, a h1 tag, and a paragraph tag. In the img tag we are going to use the phot-grid image. We add whatever we want into the h1 and p tags but I’m just going to add “Online Experiences” into the h1 and “Join unique interactive activities led by one-of-a-kind hosts—all without leaving home.” Into the p tag. Now lets add some classNames so we can style these elements with css.

section {

  padding: 20px;

}

.hero {

  display: flex;

  flex-direction: column;

}

.heroPhoto {

  max-width: 400px;

  align-self: center;

}

.heroHeader {

  margin-bottom: 16px;

}

.heroText {

  margin-top: 0;

}

Now we are going to start on some new stuff. We are going to make a card and we are going to first hard code some data for AirBnb in a card component and then we will fix it later. I have two images that we are going to use for this card, an image of Katie Zeferes and a star image.

We need to first add a div tag with className of card. Then an img tag with the image of Katie hard coded and give it a className of image. Now create another div tag and call it cardStats. Add another img tag in the new div and put the start image in it and name it cardStar. Now add 3 span elements in the new div. Add 5.0 in the first, (6) • in the second and USA in the third.

Outside of that div, lets add two p tags with “Life Lessons with Katie Zaferes” in the first and “From $136 / person” in the second. Now we can add the css.

.card {

  width: 175px;

  font-size: 12px;

}

.cardImage {

  width: 100%;

  border-radius: 9px;

}

.cardStats {

  display: flex;

  align-items: center;

}

.cardStar {

  height: 14px;

}

.bold {

  font-weight: bold;

}

.gray {

  color: #918E9B;

}

Now that we have our card, you can see how we would only be able to use this one time because we hard coded our data.

**Show example of cards on the web. (YouTube or Amazon or whatever you can think of)**

In order to reuse a component in React, we have to use something called props or properties. It just means we are going to be passing information into whatever component we need to later. Kind of like functions and arguments or parameters for JavaSrcipt.

**Open Starters props**

Let’s look at this contact for cats. There are 4 cards but right now everything is hard coded. So just like before, we’re going to start by making a contact component and then we’ll get into how to use props.

So, if we just move one of the cards to our new component, and then add in 4 instances of that component to app.js, you’ll see that now we have the same information 4 times. That’s because we are still hard coding information. This is were props comes in. Let’s add an image prop to the first contact instance. Then we’ll add name, email, and phone number props. Let’s finish the others the same way, with their information.

Notice now that none of the information has changed in our browser. That is because we have not changed anything in our contact component. We are still displaying that hard coded data. So, in the contact component, just like how you would pass in data to a function, we are going to pass in props. Now we can take props and call our properties that we set in App.js.

export default function Contact(props) {

    return (

        <div className="contact-card">

            <img src={props.image} alt={props.name}/>

            <h3>{props.name}</h3>

            <div className="info-group">

                <p>{props.phone}</p>

            </div>

            <div className="info-group">

                <p>{props.email}</p>

            </div>

        </div>

    )

}

**Break.**

**Open cards again.**

Now that we have an idea of how to work with props, lets go back to our cards app. Just like how we passed props for the cat contacts, we are going to pass props for our card.

export default function App() {

            // <Hero />

    return (

        <div>

            <Navbar />

            <Hero />

            <Card

            img={photo}

            rating={5.0}

            reviewCount={6}

            country="USA"

            title="Life Lessons with Katie Zaferes"

            price={136}

            />

        </div>

    )

}

export default function Card(props) {

    return (

        <div className="card">

            <img src={props.img} className="cardImage" />

            <div className="cardStats">

                <img src={star} className='cardStar' />

                <span>{props.rating}</span>

                <span className="gray">({props.reviewCount}) • </span>

                <span className="gray">{props.country}</span>

            </div>

            <p>{props.title}</p>

            <p><span className="bold">From ${props.price}</span> / person</p>

        </div>

    )

}

Now, nothing has changed in terms of what we are seeing in the browser but, this is a better way of making sure you can reuse your components. We are still technically hard coding information in App.js so we’re going to go over how to pull information from another file.

First, make a new folder called data and add a data.js file into that. I already have data that I set up that I am going to use here.

Now that we have our data, lets import it, and then use it. To use it, we’re going to use a JavaScript method to iterate through our data. .map is used with arrays and if you notice, our data is in an array.

So we are going to make a new constant for the data we are pulling in. Call it cards and make it equal to data.map() and for each item in the array we are going to return our card component. Then we are going to make sure the information being passed is not hard coded.

export default function App() {

    const cards = data.map(card => {

        return (

        <Card

            img={card.img}

            rating={card.rating}

            reviewCount={card.reviewCount}

            location={card.location}

            title={card.title}

            price={card.price}

            />

        )

    })

    return (

        <div>

            <Navbar />

            {/\* <Hero /> \*/}

            {cards}

        </div>

    )

}

Now we are just going to add some css so are cards are not just stacked on top of each other. As a bonus, I will just give you the css.

Since we are done with our project I wanted to talk about how this is a better way of coding this information. Since we are mapping our data, if we were to add more to our data, it will just add more cards. Or if we change something about the data, it will be changed in real time.