Day 5 – Static Page:

**Everything for today will be in the Day 5 folders.**

**Download Live server in vs code to run react**

**Get everyone set up in vs code and open Render folder in Starters Day 1 folder**

So, this will be a bit different than the “correct way” to set up react but it’s easy and goes along with what we have learned so far. We’ll set it up the correct way eventually but right now this is what we will do.

We’re going to add script tags to our html to add react, remember I said that this is not the “correct way” but the “easy way”? Well, I want to get your hands dirty first and then we can learn the “correct way”.

Link to the CDN Links page on react: <https://reactjs.org/docs/cdn-links.html> & <https://reactjs.org/docs/add-react-to-a-website.html#quickly-try-jsx>

Put the script tags in the head of the HTML file.

In the JS script tag, add type=’text/babel’

Add a div tag with an id called root. This will serve has our container for our entire webpage.



That’s it for set up right now.

In the JS file we’re now going to add something called ReactDOM and using .render() to render something to the browser. It lets us render something that is similar to the HTML that we worked with last week. Render takes two arguments, what you want it to render and where you want it to render.

Now we are going to add code inside the parentheses for the render method. Let’s add a H1 tag and hello inside that. Ok, so now remember I said that render take two arguments so now we are going to add the where. Add a comma after the h1 tags and add a selector.



Click on your live server button at the bottom to see the results. Let’s practice a bit more…How would we add an unordered list with two list items?



**Open Components folder in Starters Day5 folder**

React is composable and that means we take little parts and put them together to make a whole project. Back in the day, websites were usually just one large HTML file. (**Show the nav code in components to see how long it is**) This is just for the navigation bar. Can you imagine how much more code would need to go into actually making the rest of the webpage? A lot, right? Now, imagine, having thousands and thousands of lines of code for one webpage. That would be so hard to troubleshoot and maintain. But, React is component based and that means we can build the components separately, in different files and even folders, and put those components together to make our projects. This is to help make your code easier to maintain and read.

Let’s try building our own component. I’ve included the nav component for you to see and I want you to try, on your own, for a few minutes to make your own component. Call it Main and return a paragraph tag that says ‘Hello!’. Make sure you remember to render it bellow the Navbar component inside the render method. Notice how there are div tags surrounding the nav component, we’ll go over that in a bit but for right now, just leave it.

**Open JSX folder in Starters Day5 folder**

So, if you have been wondering how we are writing HTML inside of JS, we’ll talk about it. It’s actually called JSX, JavaScript XML. The React team created JSX to work with React. This nice thing is we can just write HTML that we are used to writing, with a few minor differences. (show vs code JSX folder)

The commented-out code is how we would add what we did in JSX using vanilla JS. Notice how much more you would have to code if you used vanilla JS. It doesn’t seem like a lot but imagine you had more than one element. Now our 4 lines of code can turn into way more, depending on how complex your page will be.

I want to show you guys the difference between these two blocks of code. I’m going to uncomment out the JS, take our JSX out of the render method and make it into a variable. Then, I’ll add a console log for element. Let’s reload our page and see what is in the console.

So, as you can see, the one on top is our vanilla JS. It is showing the h1 tags, class name is header and it is telling us ‘Hello!’. But, do you see the one below it? It says Object. JSX returns a JS object. If we expand the object, we can see the different elements of it. The ones that we are going to look at are the ‘type’, and ‘props’. Notice that ‘type’ is set to ‘h1’ and ‘props’ has an array with className set to ‘header’ and children set to ‘Hello!’. This corresponds with what we can see in our file. React interprets these objects created by JSX to show what we have on our page.

**Open components again.**

Let’s look at why I surrounded the components we were working on earlier with div tags. With JSX you can only return one parent element. If I take out the div tags, you will notice we now have an error and if I hover over that error, it will tell us “JSX expressions must have one parent element”. That doesn’t mean we can’t have multiple elements. You saw from our original example that we had two components or elements. You can have as many as you want as long as you wrap them in a parent element. It doesn’t even have to be a div tag, it can be any tag as long as it is the parent and the other elements are the children.

You can also save all of that JSX as a variable and then add that variable name to the render method just like how we added the JSX.

So, let’s try something. I want you to create a page with JSX. You can use whatever wrapper you want, but it has to include an h1 element with the website name, and an unordered list with list items for menu, about, and contact. (**Website1 folder in Finished Day1 folder**)

**Break**

Let’s look at how to set up React the “correct” way. You will already have Node.js and npm installed on these machines so just keep that in mind if you wanted to set this up at home. First let’s make a folder called Project 1. Open that folder in vs code and open the terminal. In the terminal, we need to type in npx create-react-app and then the name of our app. We’ll just call it project1. Now we wait. Once it is done let’s open that folder in vs code.

So, before we move on, open the terminal in vs code and type in **npm run build**. This is the command that you will need to use compile your web app. It takes a little while when you run that command but you have to make sure your code is compiled before your run it. After it has been complied, type in **npm start** into the terminal. This command will actually run your web app in the browser for you.

If you notice, even though we have not added any code ourselves, there is something happening in our project. That is because when we use create-react-app it makes everything we need to start our app and gives you instructions on what to do next. This page is telling us to open the App.js file, so let’s do that. This is the code for the page we are looking at. **Go over what is in the file quickly.**

Let’s play with it a bit. Change the text that is shown on the screen. Once you save what you have in your file, the web page will automatically recompile the code and update what is in the browser. (I have autosave on so it does it every time I change something.)

So, lets make our first project. I want you guys to make a “My Favorite Things” page. It must have a heading (<h1>) and an unordered list (<ul> and <li>). The list will have 5 items consisting of your top 5 favorite things. You can choose whatever you want and if you want to mess with the css file you can as well. But that is up to you, just make sure you keep the import statement for the css file if you do.

If for some reason you think you did it right but it is not showing up in the browser that way you want, go back into the terminal and press Ctrl+C and it should then ask you if you want to “terminate batch job?” press y and then run npm run build again. Then npm start again. I am not sure but sometimes this happens and you just have to recompile.

**End the day**