**CS628 Full-Stack Development – Web App**

**HOS02A: React Fundamentals**

Reviewed by Clark Ngo on November 8, 2024

Reviewed by Yogesh Goel on January 9, 2025

Reviewed by Naveena Moddu on April 17, 2025

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

#### **Readings and Examples:**

* Visit the [CS 628 Repository for Examples](https://github.com/samchung0117/cs628-examples).
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.

**Learning Outcomes**

* Section 1: Accessing GitHub Codespaces
* Section 2: Understanding Single Page Applications (SPAs) vs. Multi Page Applications (MPAs)
* Section 3: create-react-app
* Section 4: React Components
* Section 5: Stateless vs. Stateful Components
* Section 6: Understanding JSX
* Section 7: Understanding Styles
* Section 8: Understanding Events & Hooks
* Section 9: Pushing your work to GitHub

**Upload the following to your GitHub Repository generated from GitHub Classroom**

* + - 1. The screenshot of your ‘01 Counter’ as ‘01\_counter\_firstname\_lastname.png’ by using your first and last name.

1. The screenshot of your ‘02 Button’ as ‘02\_button\_firstname\_lastname.png’ by using your first and last name.
2. The screenshot of your ‘03 About’ as ‘03\_about\_firstname\_lastname.png’ by using your first and last name.
3. The screenshot of your ‘04 Avatar’ as ‘04\_avatar\_firstname\_lastname.png’ by using your first and last name.
4. The screenshot of your ‘05 Events’ as ‘*05\_events\_firstname\_lastname*.png’ by using your first and last name.
5. The screenshot of your ‘06 Hooks’ as ‘*06\_hooks\_firstname\_lastname*.png’ by using your first and last name.

**Section 1: Accessing GitHub Codespaces**

Refer to the steps from the [TA Center](https://cityuseattle.github.io/docs/git/github_codepsace/) to get started with this week’s module GitHub Codespace.

**Section 2: Understanding Single Page Applications (SPAs) vs. Multi Page Applications (MPAs)**

Single-page and multi-page applications are different approaches to organizing and presenting web content.

A single-page website consists of a single HTML page that contains all the content and functionality. Instead of loading new pages, the content is dynamically updated using JavaScript, providing a smooth user experience. React is a popular JavaScript library for building single-page applications.

In contrast, a multi-page website consists of multiple HTML pages, each representing a different section or functionality. Users navigate between these pages by clicking links or buttons that load new HTML pages. Each page has its own HTML, CSS, and JavaScript files. Multi-page websites are suitable for content-heavy sites or when Search Engine Optimization (SEO) is important.

SPAs include Facebook, YouTube, Twitter, GitHub, and numerous Google services. MPAs include eBay and Amazon. **Please visit one of each SPA and MPA and find any differences in the applications.**

* <https://www.airbnb.com/>
* <https://www.amazon.com/>

**Section 3: create-react-app**

To create a React app, you can use Create React App. It's a tool developed by the React team that sets up a new React project with all the necessary configuration and build tools.

Run the following command to create a new React app:

**>>npx create-react-app** *client*

Replace client with the desired name of your app. This command will create a new directory called client and set up the initial React project.

After the command completes, navigate into the newly created app directory:

**>> cd client**

Finally, you can start the development server and open the app in your browser by running the following command:

**>> npm start**

If you see an error about **web-vitals:**

A screen shot of a computer screen

Description automatically generated

Please cancel the running terminal with **Ctrl + C** then type and execute:

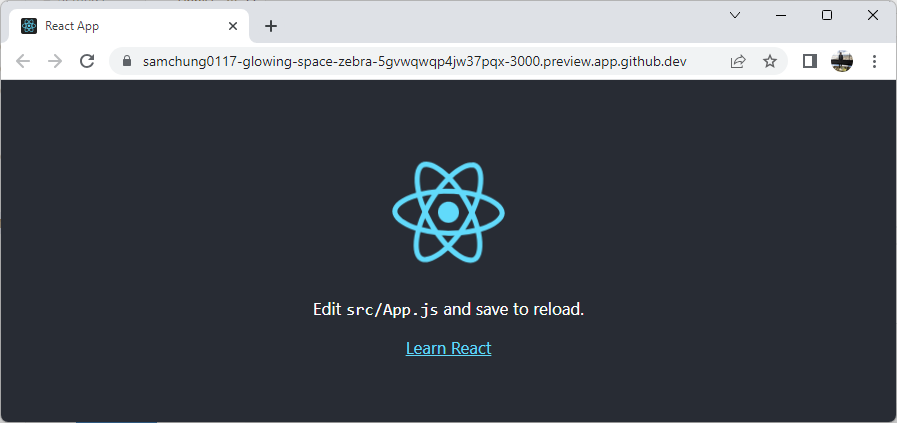
**npm install web-vitals**

Then try running again

**npm start**

This command will start the development server and automatically open the app in your default browser. Now you can build your React app by editing the source files in the '**src**' directory.

If the command does not work, install the dependencies to the local ‘**node\_modules**’ folder by typing ‘**npm install**.’



**Section 4: React Components**

There are two types of React components:

1. functional components
2. class components.

We recommend defining components as functions instead of classes. React still supports class components but does not recommend using them in new code.

**Class Components:**

Let us see what class components are for our understanding. Class components are defined as ES6 classes that extend the ‘**React.Component**’ class.

Create a file ‘Counter.js’ under the ‘src’ directory and type the following code in it.

A screen shot of a computer code

Description automatically generated with low confidence

Now update the code in your ‘App.js’ to match the following,  
  
A screen shot of a computer code

Description automatically generated with medium confidence

Now, run the development server by typing ‘npm start’ in your terminal and check the changes in your browser. Click on the ‘Increment’ button 10 times to update the ‘Counter’ to show 10.

Take a screenshot of your ‘01 Counter’ as ‘*01\_counter\_firstname\_lastname*.png’ by using your first and last name.

**Functional Components:**

Functional components are defined as JavaScript functions. They receive data through props (properties) and return JSX (JavaScript XML) elements. Functional components are simpler and easier to write and understand.

Update the code in your ‘App.js’ to match the following,

A screenshot of a computer program

Description automatically generated with medium confidence

Now refresh the browser windows where the development server runs to see the updated screen.

Take a screenshot of your ‘02 Button’ as ‘02\_button\_firstname\_lastname.png’ by using your first and last name.

**Section 5: Stateless vs. Stateful Components**

1. Stateless components, also known as functional components, are defined as JavaScript or arrow functions.
2. Stateful components, also known as class components, are defined as ES6 classes that extend the ‘React.Component’ class. They have their own internal state managed through the ‘**this.state’** object.

**Note**: With the introduction of React Hooks in React 16.8, functional components can now also manage state using the ‘useState’ and other hooks, blurring the distinction between stateless and stateful components. It is recommended to use functional components with hooks whenever possible, as they offer a more concise and flexible approach to managing the state.

**Example of a stateless component**: Replace the code in ‘App.js’ and observe the changes.

A screen shot of a computer code

Description automatically generated with low confidence

**Example of a stateful component**: It is the same example you have seen in the earlier examples in Section 4. Make changes to ‘App.js’ to match the following code and observe the changes.

A screen shot of a computer program

Description automatically generated with low confidence

**Section 6: Understanding JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript that allows you to write markup directly in your JavaScript code. It's a fundamental part of building user interfaces in React. JSX resembles HTML syntax, but it's syntactic sugar for creating React elements. All the examples you have seen earlier use JSX.

Replace your code in the ‘App.js’ with the following,   
  
A screenshot of a computer program

Description automatically generated with low confidence

We are creating two components called ‘AboutPage’ and ‘MyButton’ using JSX and reusing them in our ‘App.js.’ Observe the changes by refreshing the browser window running the development server.

Take a screenshot of your ‘03 About’ as ‘*03\_about\_firstname\_lastname*.png’ by using your first and last name.

**Section 7: Understanding Styles**

In React, you can assign a CSS class to an element using the ‘className’ attribute, which functions similarly to the class attribute in HTML. By specifying a class name, you can associate specific styles with the element using CSS rules defined in a separate CSS file. This approach allows for a separation of concerns, with the markup defined in JSX and the styles defined in a CSS file.

Alternatively, React also provides the option of using inline styles, where you can define styles directly within the JSX markup as JavaScript objects. By utilizing the style attribute on an element, you can pass in a JavaScript object that contains CSS property-value pairs. This approach is useful for applying dynamic or component-specific styles, as the styles can be determined based on the component's state or props.

In this example, let's focus on defining styles in a CSS file. This approach promotes modularity and reusability by keeping the styles separate from the component logic. Here's an example of using a CSS file to define styles for a component:

Create a file named ‘style.css’ in your ‘src’ folder and add the following code to it.

A picture containing font, text, white, line

Description automatically generated

Now update the ‘App.js’ with the following code:

A screen shot of a computer program

Description automatically generated with low confidence

Observe the changes by refreshing the browser window running the app.

Take a screenshot of your ‘04 Avatar’ as ‘*04\_avatar\_firstname\_lastname*.png’ by using your first and last name.

**Section 8: Understanding Events & Hooks**

In React, events are used to capture and handle user interactions, such as clicks, key presses, form submissions, and more. Hooks were introduced in React 16.8 as a way to use state and other React features in functional components. Hooks provide a simpler and more readable way to manage the state and lifecycle of functional components.

**useState**: The ‘useState’ hook allows functional components to manage the state. It returns a state value and a function to update that value. By calling the update function, React re-renders the component with the new state value.

**useEffect**: The ‘useEffect’ hook enables functional components to handle side effects, such as data fetching, subscriptions, or manually changing the DOM. It takes a callback function and an optional array of dependencies. The callback function is executed after each render, and the dependencies array specifies values that, when changed, trigger the callback function.

**Example of handling a button click event in React:**

Update your ‘App.js’ code with the following,

A screenshot of a computer program

Description automatically generated with medium confidence

Click on each button to observe the behavior.

Take a screenshot of your ‘05 Events’ as ‘*05\_events\_firstname\_lastname*.png’ by using your first and last name.

**Example of using ‘useState’ and ‘useEffect’ hooks in a functional component:**

Update your ‘App.js’ code with the following,

**A screen shot of a computer code

Description automatically generated with low confidence**

In this example, the App component is defined as a functional component using the ‘useState’ and ‘useEffect’ hooks. The ‘useState’ hook is used to declare a state variable called count and a function ‘setCount’ to update its value. The initial value of the count is set to 0.

The ‘useEffect’ hook is used to perform side effects in the component. In this case, it updates the document title with the current count value. The effect runs after every render because the count variable is specified as a dependency in the dependency array [count]. If the value of the count changes, the effect will be re-run. The effect also returns a clean-up function that will be called before the component unmounts, resetting the document title to its original value.

Refresh the browser window where the app is running to observe the changes.

Take a screenshot of your ‘06 Hooks’ as ‘*06\_hooks\_firstname\_lastname*.png’ by using your first and last name.

**Section 9: Pushing your work to GitHub**

[How to submit your work in GitHub](https://cityuseattle.github.io/docs/git/github_upload_files/)