# **Coding Activity 3.1**

# **Add A React Template To Your Project**

**Learning Outcome Addressed**

 4.Set up a React project

**Add A React Template To Your Project**

### Adding React To Your Page In 1 Minute

You've seen examples of what you can do once you have React set up in your application. Now you can go through the step-by-step tutorial to turn your regular HTML and JavaScript files into a functional React app that can be modified to your needs.

This process is based on documentation taken directly from the React website, where instructions tell you how to add React to your Application. As you continue to code, this will become a familiar process for you. When working with new tools, programmers often don't learn every quirk of a new language before working with it. It's more common for a developer to start from a template or baseline project, and then edit that project for their use case.

#### Task instructions

In this task, you need to copy and paste code blocks according to the instructions in order to get a React Demo working.

**Step 1: Add a DOM container to HTML**

The first step involves providing the div that will be the container for your React component. It also needs to have a unique ID attribute to allow React to find it.

Add an empty <div> tag to mark the spot where you want to display something with React:

<!-- ... existing HTML ... -->  
<div id="like\_button\_container"></div>  
<!-- ... existing HTML ... -->

**Step 2: Add the script tags** Now, add three <script> tags to the HTML page right before the closing </body> tag as shown below.

  <!-- ... other HTML ... -->  
   
  <!-- Load React. -->  
  <!-- Note: when deploying, replace "development.js" with "production.min.js". -->  
  <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>  
  <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>  
  <!-- Load our React component. -->  
  <script src="addReact.jsx"></script>  
</body>

The first two tags load React. The third points to your .jsx file shown in the file tree to the right. Make sure the file name in the third script src attribute matches the file name of the file you'll populate in Step 3.

**Step 3: Create a React component**

Go to the file named addReact.jsx in the file tree to the right. It should be empty. You can add the following starter code:

'use strict';

function LikeButton()

{

const [state,setState] =React.useState({liked:false});

if(state.liked){ return( <div> You liked this </div> ) }

else{<button onClick={()=>setState({liked:true})} > Like </button>}

}

This code defines a React component called LikeButton. Don't worry about understanding what it does yet. You'll get more hands-on experience using React in future modules.

After the starter code you just pasted into the addReact.jsx file, add the following two lines at the very bottom.

// ... the starter code you pasted ...  
   
const domContainer = document.querySelector('#like\_button\_container');

ReactDOM.render(e(LikeButton), domContainer);

This code finds the <div> tag you added to the index.html in the first step.

Once you've completed these three steps, you'll see a **Like** button displaying in your web browser. Upon clicking it, the button will be replaced by text until you refresh the page to re-initialize your React app.

Hint:

* This activity is closely based off of the following [*link*](https://reactjs.org/docs/add-react-to-a-website.html#add-react-in-one-minute) on the reactjs.org official website. To learn more about React or this activity, feel free to explore the site!

Task

Follow the instructions exactly to get a React Demo up and running.