Trainer guide
Examples and exercises
React-1 – Components

Contents

React-1a | Example: A static page in React

Link to environment

This example shows that you can build a static app in React in much the same way as you can assemble a static page with HTML.

src/App.jsx

- Look at the "App" function.
    - `App` contains all the content that is displayed on the page. There is a header, some main content, and a footer.
- `App` is a React component.
    - Components are how we build UIs in React.
    - To create a component, all you need to do is write a function that returns markup.
- What's returned from the App function looks a lot like HTML.
    - For example, `nav`, `h1`, and `img` are all HTML tags.
    - This syntax is called JSX. JSX is a syntax extension to JavaScript that allows us to write our components in a familiar, HTML-like way.
    - JSX elements can also receive the same attributes that their respective HTML elements can - for example, the <img> element has attributes like `src`, `alt` and `width`.
    - Although JSX and HTML are very similar HTML and JSX are not identical. For example, instead of the "class" attribute you'd see in HTML, we instead use "className" in JSX to apply the "spinning" class to the image. The reason we don't use "class" in JSX is primarily because "class" is a reserved keyword in JavaScript.
    - Also, as we'll see later, JSX is very powerful. It allows us to embed any JavaScript that we want inside our markup.

src/index.js

- index.js contains some important setup.
- We import the main App component from App.jsx.
- Then, React needs a "root" HTML element in which to render the app.
- In this case, we're finding the element with an ID of "root" and rendering our app there.
- However, you don't need to worry about understanding the nitty-gritty of this, because generally when you create a new React app, index,js is created for you, ready to go. You'll probably never need to touch it throughout the development of your app.

public/index.html

- *Point out the div with an ID of "root".*
- This is where the React app will render.

src/App.jsx

- *Encourage learners to mess around with the contents of <App>.*

React-1b | Exercise: Making your own components

Link to environment

This exercise gives learners practice on extracting and defining their own components.

src/App.jsx

- In the previous example, everything on the page was contained in a single component called App.
- But the beauty of React is that we can organise our app into lots of smaller, more logical components.
- As you can see, all of the code related to the main content has been organised into a component called MainContent.

Exercise

- Your task is to take the header content and turn it into its own component called Header.
- Then, do the same with the footer content.
- In total, App should contain 3 custom components (Header, MainContent, Footer)
- Note: React components must start with a capital letter.

Solve the exercise live before moving on.

Solution reference

React-1c | Exercise: Placing components in separate files

Link to environment

This exercise gives learners practice on organising their components into separate files.

src/App.jsx

- Up until now, we've placed all of our components in a single file, App.jsx
- However, to keep things organised, components are often placed inside their own files
- Notice that we're referencing this component MainContent, but it's not inside this file.
- *Point out "import MainContent"*
    - We're importing this component from another file.

src/components/MainContent.jsx

- Here's where we've defined the MainContent component.
- Note that we use the .jsx file extension instead of the .js file extension to indicate that this is a component file.
- To use this component elsewhere in the app, we need to export it as a module with the export keyword.
- We also need to make sure to import the React module, seen at the top of the file.

Exercise

- Your task is to take the Header component and place it in its own file within the /src/components folder.
- Then do the same with the Footer component.

Solve the exercise live before moving on.

Solution reference

React-1d | Example: Embedding JavaScript expressions in JSX

Link to environment

This example shows how JSX allows you to embed any JavaScript expression within your component's markup.

src/App.jsx

- This is a simple application that displays the current date.
- Within our App component, we get the current date as a string.
- "date" is a regular JavaScript variable.
- To inject this into our markup, we simply place it within curly brackets.
- You can put **any** JavaScript expression you want in your markup by placing it within curly brackets