the shortcut

React Tutorial
Lesson 5

Today's lesson will cover

- Async/Await
- Forms
- React Legacy: Class Components and State
- Styling in React
- CSS in React
- CSS Modules
- Styled Components
- SVGs in React

Intro to React

Async/Await

- You'll work with asynchronous data often in React, so it's good to know alternative syntax for handling promises: **async/await**
- To include error handling as before, the try and catch blocks are there to help. If something goes wrong in the try block, the code will jump into the catch block to handle the error. then/catch blocks and async/await with try/catch blocks are both valid for handling asynchronous data in JavaScript and React.

Forms in React

- Forms aren't much different in React than HTML. When we have input fields and a button to submit data from them, we can give our HTML more structure by wrapping it into a form element with a `onSubmit` handler. The button that executes the submission needs only the "submit" type.

React’s Legacy: Class Components and State

- A typical class component is a JavaScript class with a mandatory render method that returns the JSX. The class extends from a React.Component to inherit (class inheritance) all React's component features (e.g. state management for state, lifecycle methods for side-effects). React props are accessed via the class instance (this)
- If no side-effects and no state were used in legacy apps, we'd use a function component instead of a class component. Before 2018--before React Hooks were introduced--React's function components couldn't handle side-effects (useEffecthooks) or state (useState/useReducer hooks). As a result, these components were known as functional stateless components, there only to input props and output JSX
- Before React Hooks, class components were superior to function components because they could be stateful. With a class constructor, we can set an initial state for the component. Also, the component's instance (this) gives access to the current state (this.state) and the component's state updater method (this.setState)

Styling in React

- There are many ways to style a React application, and there are lengthy debates about the best styling strategy and styling approach
- Two alternatives for more advanced are **CSS-in-CSS (CSS Modules)** and **CSS-in-JS (Styled Components)** strategies
- If you don't want to build common UI components (e.g. button, dialog, dropdown) from scratch, you can always pick a popular UI library suited for React, which provides these components by default, such as:
  - Ant Design
  - Chakra UI
  - Tailwind UI
  - Semantic UI
  - Material UI
  - React Bootstrap

CSS in React

- Common CSS in React is similar to the standard CSS you may have already learned.
- Each web application gives HTML elements a class (in React it's `className`) attribute that is styled in a CSS file later.
- We can also use the native style attribute for HTML elements. In JSX, style can be passed as an inline JavaScript object to these attributes. This way we can define dynamic style properties in JavaScript files rather than mostly static CSS files. This approach is called **inline style**, which is useful for quick prototyping and dynamic style definitions. Inline style should be used sparingly, however, as a separate style definition keeps the JSX more concise.
- Without CSS extensions like Sass (Syntactically Awesome Style Sheets) inline styles can become burdensome, though, because features like CSS nesting are not available in native CSS.
- We can also pass the `className` attribute as a prop to React components.

CSS Modules

- CSS Modules are an advanced **CSS-in-CSS** approach. The CSS file stays the same, where you could apply CSS extensions like Sass, but its use in React components changes.
- To enable CSS modules in create-react-app, rename the src/App.css file to src/App.module.css. This action is performed in the command line from your project's directory
    - `mv src/App.css src/App.module.css`
- CSS Modules--like any other CSS-in-CSS approach--can use Sass for more advanced CSS features like nesting.
- The advantage of CSS modules is that we receive an error in the JavaScript each time a style isn't defined. In the standard CSS approach, unmatched styles in the JavaScript and CSS files might go unnoticed.

Styled Components

- With the previous approaches from CSS-in-CSS, Styled Components is one of several approaches for **CSS-in-JS**
- Styled Components because it's the most popular. It comes as a JavaScript dependency, so we must install it on the command line:
    - `npm install styled-components`
- As the name suggests, CSS-in-JS happens in your JavaScript file
- When using Styled Components, you are using the JavaScript template literals the same way as JavaScript functions. Everything between the backticks can be seen as an argument and the `styled` object gives you access to all the necessary HTML elements (e.g. div, h1) as functions. Once a function is called with the style, it returns a React component that can be used in your App component
- Advanced features like CSS nesting are available in Styled Components by default

SVGs in React

- To create a modern React application, we'll likely need to use SVGs. Instead of giving every button element text, for example, we might want to make it lightweight with an icon.
- You can download the icon from here as SVG.
- Regardless of the styling approach you are using, you can give your SVG icon in the button a hover effect too.
- The create-react-app project makes using SVGs straightforward, with no extra configuration needed. This is different if you create a React project from scratch with build tools like Webpack, because you have to take care of it yourself.

the shortcut

Self Study Assignments

To Dos

- Unless you have your own project to work on to practice what we have learned, complete the Pokemon API app with React from here: https://learn.chrisoncode.io/webinars/javascript-to-react
- Start working on React or Fullstack JavaScript project, OR complete FCC Frontend Library Certification : Bootstrap, React, Redux, React & Redux, projects; APIs and Microservices Certification : https://www.freecodecamp.org/learn
- Refer to Reactjs official documentation as the best place to learn underlying main concepts of React with examples: https://reactjs.org/docs/hello-world.html