the shortcut

React Tutorial
Lesson 3

©2020 The Shortcut. All rights reserved

©2020 THE SHORTCUT

Today’s lesson will cover

- React Fragments
- Reusable Components
- Imperative React
- Inline Handlers in JSX
- Asynchronous Data and Conditional Rendering
- Advanced State - Using Reducer function

the shortcut

Intro to React

©2020 The Shortcut. All rights reserved

React Fragments

- A common pattern in React is for a component to return multiple elements. Fragments let you group a list of children without adding extra nodes to the DOM.

- You can use <></> the same way you'd use any other element except that it doesn't support keys or attributes.

```
const Fragment = () =>(

     <>
       <td>Hello</td>
        <td>World</td>

     </>
   )
```

- A fragment wraps other elements into a single top-level element without adding to the rendered output. So if you prefer to omit the wrapping <div> elements, substitute them with an empty tag that is allowed in JSX, and doesn't introduce intermediate elements in rendered HTML.

©2020 THE SHORTCUT

Reusable React Component

- Generally speaking we make a component more reusable by turning it from specific to more generic. That can be achieved by offering an API for the component. In React, a component's API is its props
- Generic React components are commonly widely used in the entire application because they are highly reusable.
- Another benefit is having the implementation of one widely used component of your application at one place. In case something changes in the implementation details for this component, you can adjust it at this one place and it takes effect everywhere in your application.

Component Composition

- What's composition in general? It's the ingredients and the arrangement of these ingredients to create something bigger out of it. It's the samples in a piece of music that make up a track. It's the fruits that are used for the perfect

- React has a powerful composition model, and it is recommended using composition instead of inheritance to reuse code between components

- There is one property (React prop) that helps us out with this dilemma for our React component: the React children prop. It's one special prop provided by React to render something within a component whereas the component isn't aware ahead of time what it will be.

- Composing React Components doesn't end here. There are two other advanced React patterns that are used for component compositions as well:
  - Render Prop Components
  - Higher-Order Components

©2020 THE SHORTCUT

Imperative React

- React is inherently declarative, starting with JSX and ending with hooks. In JSX, we tell React *what* to render and not *how* to render it. In a React side-effect Hook (useEffect), we express when to achieve *what* instead of *how* to achieve it. Sometimes, however, we'll want to access the rendered elements of JSX imperatively, in certain cases.

- Read/write access to elements via the DOM API:
  - measure (read) an element's width or height
  - setting (write) an input field's focus state

- Implementation of more complex animations:
  - setting transitions
  - orchestrating transitions

- Integration of third-party libraries:
  - D3 is a popular imperative chart library

Inline Handler in JSX

- Inline event handlers, also called inline handlers, give us lots of new options by using an event handler directly in JSX
- In general, developers are lazy people, so often inline event handlers are used to avoid the extra function declaration outside the JSX. However, this moves lots of business logic into the JSX which makes it less readable, maintainable and error prone.
- Inline handlers are also used to pass a parameter to a more universal handler which is defined outside of the JSX

©2020 THE SHORTCUT

React Asynchronous Data and Conditional Rendering

- Sometimes we must render a component before we can fetch data from a third-party API and display it
- To understand handling asynchronous data, read more about JavaScript Promises
- Handling asynchronous data in React leaves us with conditional states: with data and without data
- In JavaScript, a `true && 'Hello World'` always evaluates to 'Hello World'. A `false && 'Hello World'` always evaluates to false. In React, we can use this behaviour to our advantage. If the condition is true, the expression after the logical `&&` operator will be the output. If the condition is false, React ignores it and skips the expression.
- Conditional rendering is not just for asynchronous data though. The simplest example of conditional rendering is a boolean flag state that's toggled with a button. If the boolean flag is true, render something, if it is false, don't render anything.
- This feature can be quite powerful, because it gives you the ability to conditionally render JSX. It's yet another tool in React to make your UI more dynamic. And as we've discovered, it's often necessary for more complex control flows like asynchronous data

©2020 THE SHORTCUT

React Advanced State

- Basically reducers are there to manage state in an application. For instance, if a user writes something in an HTML input field, the application has to manage this UI state (e.g. controlled components).
- The reducer function is a pure function without any side-effects, which means that given the same input (e.g. state and action), the expected output (e.g. newState) will always be the same. This makes reducer functions the perfect fit for reasoning about state changes and testing them in isolation.
- The concept of a Reducer became popular in JavaScript with the rise of Redux as state management solution for React. But no worries, you don't need to learn Redux to understand Reducers. Foremost, there are two important things to understand in general:
- **The state processed by a reducer function is immutable.** That means the incoming state -- coming in as argument -- is never directly changed. Therefore the reducer function always has to return a new state object.
- Since we know about the state being a immutable data structure, we can use the JavaScript spread operator to **create a new state object from the incoming state and the part we want to change** (e.g. count property).

the shortcut

Self Study Assignments

©2020 The Shortcut. All rights reserved

©2020 THE SHORTCUT

To Dos

- 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