Learn React for Modern Web Applications

Basic React Syntax and Concepts

Why React

What is it about React that make it worth learning?

- React is an open-source, front-end JavaScript library that was originally created by Facebook and still continues to be maintained by them as well as the open source community.
- React helps developers quickly create maintainable performant applications.
- React uses modular components to increase code reuse, which can drastically reduce the amount of code we end up writing.
- React applications are written in JavaScript, JavaScript is both very popular and relatively easy to learn.
- React has been around long enough to be stable and has an active ecosystem.

React Boilerplate Generators

- Setting up a React app from scratch takes time, so we generally run a script that set it all up for us.
- Create React App

  Before create react app, we need to install latest nodejs

  node –v

  npm –v
  to check version.
  npx create-react-app my-react-app —-use-npm
  cd my-react-app
  code .
  entry is src/index.js

Write your first JSX

- JSX goes beyond HTML
- JSX allows us to use other components we've defined as tags
- As we saw in the index.js file, we're importing our App component and then using JSX syntax to render it to the page.
- We will learn how to define our own components, use this JSX syntax to display the components we create inside the app component.
- We can assign JSX expressions to JavaScript variables, and then use JavaScript logic to help structure and restructure our pages.
- Wrap it in curly braces to insert into our JSX.

React Components

- Components are React's basic units of organization.

They're reusable collections of elements and functionality that usually represent conceptually distinct pieces of our application

- Class component syntax
  class myComponent extends React.Component{}
- Functional component syntax

Components that we define in this way are just functions that return some JSX that will be displayed in the browser when that component is rendered. This should be the best way to create component so far.

We're using a separate component, so our app is more modular and more maintainable, this is one of the major benefits of using React.

- JSX that React components return is only allowed to contain one top level element
  Use React.Fragment or div tag to wrap multiple elements.

Pass props to components

- Component can receive an object containing all the props that its parent component is passing to it as an argument
- Use object destructuring to get all the props your component uses up at the top of your component definition.
- Destructure the props inside the function's parenthesis

Render components conditionally

- If (someCondtion) return null // won't render anything.

  It's perfectly okay to return null from a component, returning undefined will throw an error

Display lists of components

- Create a List component, ie PeopleList
-
```
export const PeopleList = ({people}) =>
(
  <>
  {people.map(person  => (
  <div>
    <h3>{person.name}</h3>
    <p>Age: {person.age}</p>
  </div>
  ))}
  </>
)
```
- Create a ListItem component, ie PeopeListItem

```
export const PeopleList = ({people}) =>
(
  <>
  {people.map(person => <PeopleListItem
person={person} key={person.name} />)}
  </>
)

  export const PeopleListItem = ({person}) =>(
    <div>
      <h3>{person.name}</h3>
      <p>Age: {person.age}</p>
    </div>
  )
```

Handle Clicks and other events

- Event handling in React is similar to regular HTML and JavaScript

<button onClick = {}> click Me! </button>

In React, the value that we pass to this onClick prop is not a string. Instead, we use curly braces to insert a function that should be called when the button is clicked.

If we want delay the execution of a function, you need to pass the function without the parentheses after it.

Style components in React

- CSS modules

  Most common way.
  import App.css

  The styles from that CSS file are scoped to that component and all of its child components.

  If we wanted to add a custom styling , we can do is inside our source directory to create a new CSS file.

- Styled-components

  npm install styled-components

Css in javascript, allow us to write css inside our javascript

- Inline styles
  <p style={{color:'red', fontSize:'96px'}}>Big red Text</p>

Learn React for Modern Web Applications

State and the Component Lifecycle

Use State in components

- Add state to our components
- React hooks
  Modern React syntax, the main use of hooks is to make it easy for us to add state and side effects
- useState hook, allows us to add state to our components

  Defining constants using the array destructuring syntax, and the first constants is the current value of our state. second constant is a function that we can call to change the value of the state. Create this state by calling useState with whatever we want the initial value of our state (ie. NumberOfClicks) constant
- React will only re-render if the props, or a value from one of the hooks changes. This allows React to only re-render what needs to be re-rendered.
  React doesn't need to re-render our entire application every time one little variable somewhere changes. It only re-renders the components that are affected by that variable changing.

Decide where to put State

- If only one component uses that state, of course we should put the state into that component.
- If other components need access to the value of another component State, we need to decide where to put the State
- React components have what's called a unidirectional data flow, parent components can pass props to their children, but child components can't directly pass any data back to their parent component.
- See demo for "CounterButton" and "CongratulationsMessage" share to use same State "numberOfClicks"

Use Lifecycle in components

- Component lifecycle revolves in useEffect React hook

It will get called when our component is first rendered, and whenever the data of the component updates.

The second argument is an array of values that we want our useEffect hook to watch, if it is empty, only run once, in the real world, this functionality is really helpful for situations such as when we're fetching data from the server.

The function that we pass to useEffect can return another function and useEffect will call the function that we return when our component unmounts, This is useful in situations such as if our components subscribes to an observable or some other event stream when it's first rendered. Our component needs to unsubscribe from that when it's unmounted.

Be careful about changing the value of a state variable from inside this function that we pass to it. If our useEffect hook makes a change to some state variable, and it's set to rerun every time that state variable changes, this can easily cause an infinite loop.

Learn React for Modern Web Applications

Learn Route in React

Install and setup React Route

- npm install react-route-dom
- Setup some pages to navigate between pages
- import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

  Use Route path property to set router path, and element property to set the route page.

```
<Router>
  <Routes>
  <Route path="/"
  element=<HomePage/> />
  <Route path="/counter"
   element = <CounterButtonPage/> />
   ...
  </Routes>
</Router>
```

Add Link to the App

Instead of using Go to my page which caused our app to reload every time we navigate between pages We want our URL transitions to be smooth and not trigger reloads by using a link component without refreshing our application.

import { Link } from 'react-router-dom';

Add under the Router tag
<Link to = "/counter">Go to Counter Page</Link>

Use URL parameters

Extra info contained in URL

URL Parameters are found as segments of the URL path.

https://www.linkedin.com/in/**Gerry-Liu/**

import { useParams } from 'react-router-dom';

Change route path to "<Route path="/counter/:name" ..."
Use "const {name} = useParams()" to get the parameter

Use URL query parameters

Extra info contained in URL

Query parameters are in different part of the URL after ?
www.google.com/search?**q=ibm+market+price**

npm install query-string

```
import { useLocation } from 'react-router-dom';
import queryString from 'query-string';


  const location = useLocation()
```

Query parameters are in location search property.

```
const startingValue = queryString.parse(location.search).startingValue
```

Implement "not found" pages

- Create the actual "not found" page
  Add "NotFoundPage" under the last Route

```
<Route path="*"    element = <NotFoundPage/> />
```

Redirect (Navigate) with React-Router

- import { Navigate } from 'react-router-dom';
- <Navigate to="/" />;