Intro to building web-apps with React

Hosted By Kushal Jaiswal

What is going to be covered today?

- Basics of React
  - Creating a react app
  - JSX and what it is
  - ReactDOM
  - Components, props, state and lifecycle
  - Conditional rendering and handling events
- Building a meme Generator
  - Building a basic ui using what we learned
  - Making request to an API
  - Randomly generating memes on click of button

RECAP: HTML, CSS and JavaScript

HTML

- Markup language to create basic web pages.

CSS

- Stylesheet language for styling HTML documents and telling the browser how they would be displayed.

JS

- Scripting language used to enhance and add functionality to HTML elements and is generally embedded within the document.

Hello World!

The simplest React fragment, displays Hello, world! On the screen.

```
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

Imports and Exports in React

- Really similar to VanillaJS, import and export.
- Came with ES2015, and can have a default export or several named exports.
- Allows us to organise components in a folder structure.

Import

```
import { PARA_NAME } from ADDRESS
```

```
import GIVEN_NAME from ADDRESS
```

Export

```
export { PARA_NAME }
```

```
export default GIVEN_NAME
```

What is JSX?

- A template language, but it comes with the full power of JavaScript, and is used to render React elements in the DOM.
- React doesn't require using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.
- JS expressions and functions can be embedded inside HTML in React.

Creating a react app with basic template

npx create-react-app myApp; cd myApp; npm start

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Components and Props

- Components are basic react functions
- They accept input called ‘props’
- Components can be functional as well as class based.
- Components can be rendered by JS elements.
- Components can refer to other components in their output
- Components can be split into smaller components and are always pure.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

State and Lifecycle methods.

- State is similar to props, but it is private and fully controlled by the component.
- State can be used with both class and functional components.
- React component lifecycle has three categories – **Mounting, Updating and Unmounting**
- **Lifecycle methods tell React what to do when the component is in either state of its lifecycle.**
- **State should not be modified directly but by using a function called setState()**
- **setState updates are merged.**

```
// Wrong
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

```
// Correct
this.setState((state, props) => ({
  counter: state.counter + props.increment
}));
```

Handling Events

- Handling events are really similar compared to DOM.
- We need to bind the functions otherwise this will be undefined during function call.

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // This binding is necessary to make `this` work in the callback
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

ReactDOM.render(
  <Toggle />,
  document.getElementById('root')
);
```

Conditional rendering

- What component should be rendered based on conditional operators.

```
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}
```

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  // Try changing to isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Conditional rendering (cont.)

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Hello!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          You have {unreadMessages.length} unread messages.
        </h2>
      }
    </div>
  );
}

const messages = ['React', 'Re: React', 'Re:Re: React'];
ReactDOM.render(
  <Mailbox unreadMessages={messages} />,
  document.getElementById('root')
);
```

- We can also use inline && for conditionally rendering.
- Moreover, we can also use ternary operators for the same(? :).

Bonus: Making AJAX calls to APIs

- Many libraries are available to make Ajax calls(Axios, jQuery AJAX).
- We will learn about built in fetch().
- It gets JSON file across a network and displays to the console.
- Fetch actually returns a JS Promise, which resolves to a Response object.
- We use json() to extract the body content, as Response contains the entire HTTP response.

```
fetch('http://example.com/movies.json')
  .then(response => response.json())
  .then(data => console.log(data));
```

Let’s Start Coding!

We will be creating a meme generator

The app will make calls to an API to generate the memes.

The memes would be displayed in a user friendly UI.

Resources

(For all your future exploitative endeavours)

https://reactjs.org/docs/getting-started.html - React introduction

https://scrimba.com/learn/learnreact/welcome-to-an-introduction-to-react-coaaf455789c2a9addc20dd24 - Live coding based React learning

https://www.npmjs.com/ - NPM registry for different packages and libraries to use with React.

https://github.com/public-apis/public-apis - A collection of free public APIs.

(Bonus)

https://www.geeksforgeeks.org/top-10-extensions-for-reactjs-in-vscode/ - Useful VS Code extensions for working with React.

(or ask questions on Discord)