React

By Utkarsh Mehta

Basics of React

React is a JavaScript library for building user interfaces.

React is used to build single-page applications.

React allows us to create reusable UI components.

React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React only changes what needs to be changed!

React finds out what changes have been made, and changes only what needs to be changed.

Setting up React

- Install `create-react-app`

  ```
  npm install -g create-react-app
  ```

- Run create-react-app to create a react project

  ```
  npx create-react-app test
  ```

- Change the directory

  ```
  cd test
  ```

- Run the app

  ```
  npm run start
  ```

Hello World Example

Introducing JSX

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.

JSX converts HTML tags into react elements.

Rendering Elements

ReactDOM.render method is used to render JSX by selecting a target element

JSX is rendered as a child of that element.

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

The result is displayed in the `<div id="root">` element:

```
<body>
  <div id="root"></div>
</body>
```

Components and Props

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

Components come in two types, **Class** components and **Function** components.

When creating a React component, the component's name MUST start with an uppercase letter.

Props are arguments passed into React components.

Props are passed to components via HTML attributes.

State and Lifecycle

React components has a built-in state object.

The state object is where you store property values that belongs to the component.

When the state object changes, the component re-renders.

The state object is initialized in the constructor.

Each component in React has a **lifecycle** which you can monitor and manipulate during its three main phases.

The three phases are: Mounting, Updating, and Unmounting.

When mounting a component

1. `constructor()`
    a. called before anything else, when the component is initiated
2. `getDerivedStateFromProps(props, state)`
    a. called right before rendering the element
3. `render()`
    a. outputs the HTML to the DOM
4. `componentDidMount()`
    a. called after the component is rendered

When Updating

1. `getDerivedStateFromProps(props, state)`
   a. first method that is called when a component gets updated
2. `shouldComponentUpdate()`
   a. method you can return a Boolean value that specifies whether React should continue with the rendering or not
3. `getSnapshotBeforeUpdate(prevProps, prevState)`
   a. you have access to the `props` and `state` *before* the update, meaning that even after the update, you can check what the values were *before* the update
4. `render()`
   a. Renders the component
5. `componentDidUpdate()`
   a. called after the component is updated in the DOM

When unmounting

The `componentWillUnmount` method is called when the component is about to be removed from the DOM

Handling Events

Just like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

React events are written in camelCase syntax:

`onClick` instead of onclick.

React event handlers are written inside curly braces:

`onClick={shoot}` instead of `onClick="shoot()"`.

Conditional Rendering

Rendering components or data by checking a condition.

We can use && (Logical And) || (Logical Or), ?: (Ternary operator), etc.

Lists and Keys

Rendering components by iterating a list.

We can use map, reduce & filter functions on list to render elements.

When rendering same component using a list we should assign a key attribute to avoid warnings and for optimizations.

Forms

Composition vs Inheritance

React uses components which can nested together to build UIs.

Components inherits React.Component.

Custom Components does not need inheritance except the React.Component one.

Composing the components is the best way to use React.

Build custom, independent & atomic components and compose them.

Thinking in React

Mindset is everything.

1. UI mindset
    a. UI should be broken down into components
2. State mindset
    a. What data is needed for component
    b. Define & Initialise states
3. Data Flow Mindset
    a. Listen for event to update data (Component to states)
    b. Use to states to show data (state to component)

Debugging

Using `debugger;` statement in code.

Open source tab in developer tools.

Controlling the flow of code.

Checking & updating the values of the variables.

Adding new breakpoints.

Hands on app.