React Fundamentals

Component Lifecycle

Topics

- Day 2 Review
- Understanding the Component Lifecycle
  - componentDidMount()
  - componentDidUpdate()
  - componentWillUnmount()

Think, Discuss, and Share

What are props?

Think, Discuss, and Share

What is the difference between a functional component and a class component?

Component Lifecycle

What is the Component Lifecycle?

What is the Component Lifecycle?

- Components have a ‘lifecycle’ that consists of three phases:
  - **Mounting**
  - **Updating**
  - **Unmounting**
- Each phase makes use of other built-in React methods to manage its operation.

Mounting

Methods called during mount:

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. **componentDidMount()**

Updating

Methods called during update:

1. getDerivedStateFromprops()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. **componentDidUpdate()**

Unmounting

Methods called during update:

1. **componentWillUnmount**

Component Lifecycle

componentDidMount

componentDidMount()

- Runs after component is mounted
- Typically used to fetch any data the component needs to perform its role
- x

```
class Jeopardy extends React.Component {
  state = {
    question: {}
  }

  getNewQuestion() {
    axios.get('http://jservice.io/api/random')
    .then(result => {
        this.setState({ question: result.data[0] })
    })
  }

  // calls this.getNewQuestion() after component mounts
  componentDidMount() {
    this.getNewQuestion()
  }

  render() {
    return (
      <div>
        {JSON.stringify(this.state.question)}
      </div>
    )
  }
}
```

Component Lifecycle

componentDidUpdate

componentDidUpdate()

- Runs after every component update
- Receives previous props and state as arguments
- Often paired with a conditional to limit execution to certain updates

```
class Counter extends React.Component {
  state = {
    count: 0
  }

  handleIncrement = () => {
    this.setState({ count: this.state.count + 1 })
  }

  // updates document title after count changes
  componentDidUpdate(prevProps, prevState) {
    if(this.state.count !== prevState.count) {
      document.title = `Count - ${this.state.count}`
    }
  }

  render() {
    return (
      <div>
        Count: {this.state.count}
        <button onClick={this.handleIncrement}>+1</button>
      </div>
    )
  }
}
```

Component Lifecycle

componentWillUnmount

componentWillUnmount()

- Runs right before component is removed

- Typically used for cleanup to avoid memory leaks:
  - Clear timers
  - Cancel network requests

```
class Clock extends React.Component {
  this.state = {
    date: new Date()
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.tick(),
    1000
    )
  }

  // clears interval when component is unmounted
  componentWillUnmount() {
    clearInterval(this.timerID);
  }

  tick() {
    this.setState({ date: new Date() })
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    )
  }
}
```

Let’s try it!

DevelopIntelligence
A PLURALSIGHT COMPANY