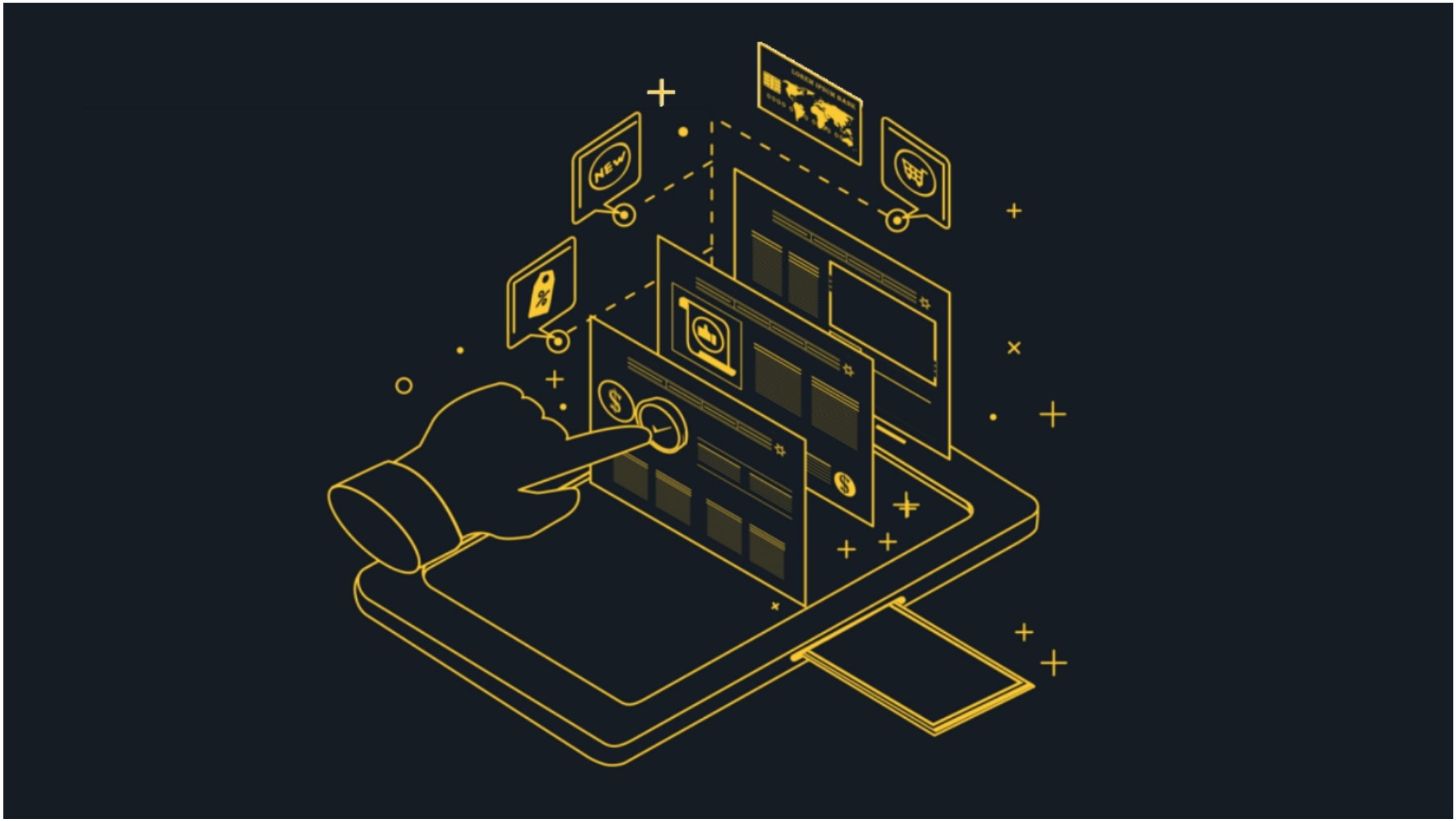
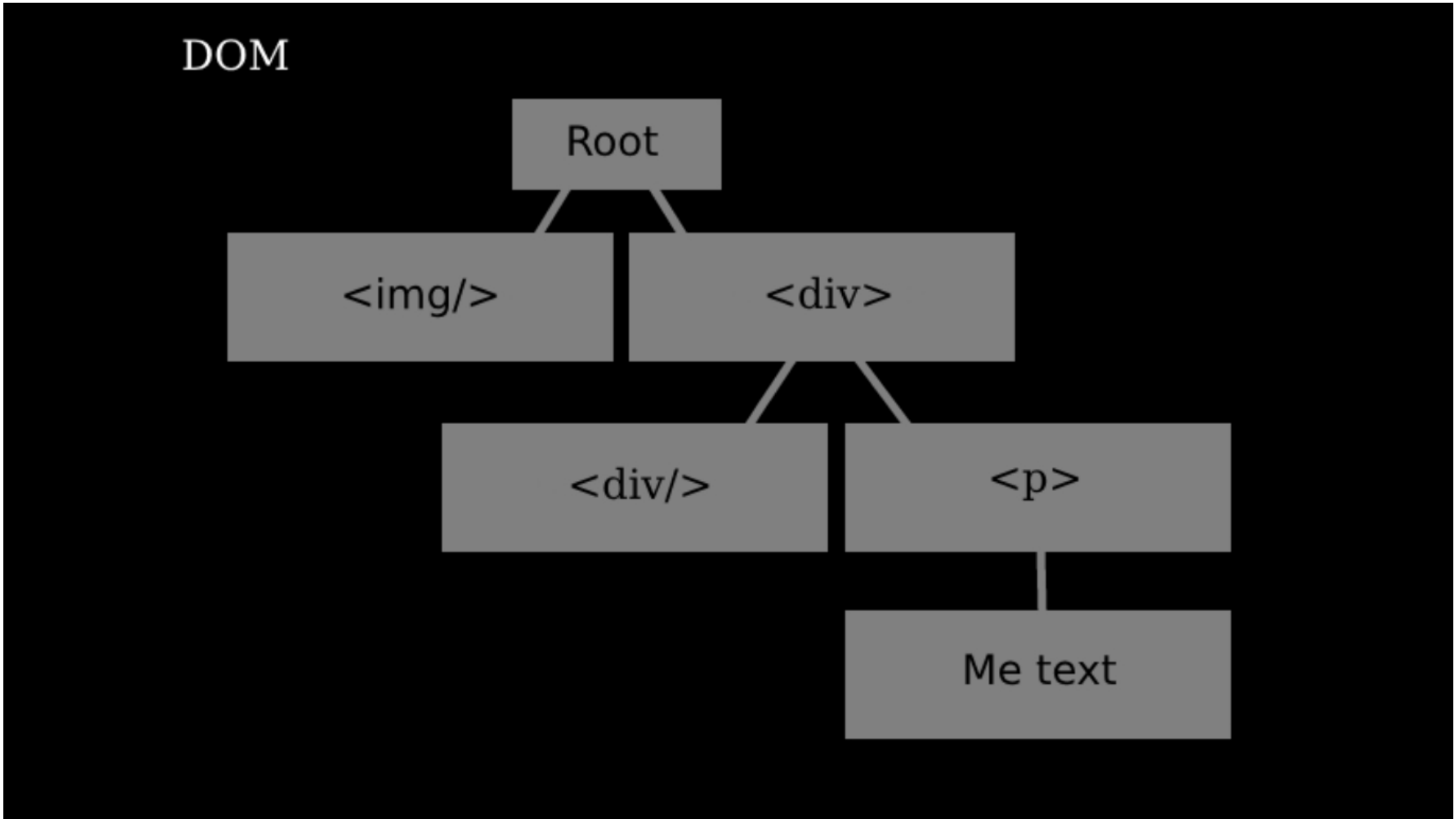
WTF is React?

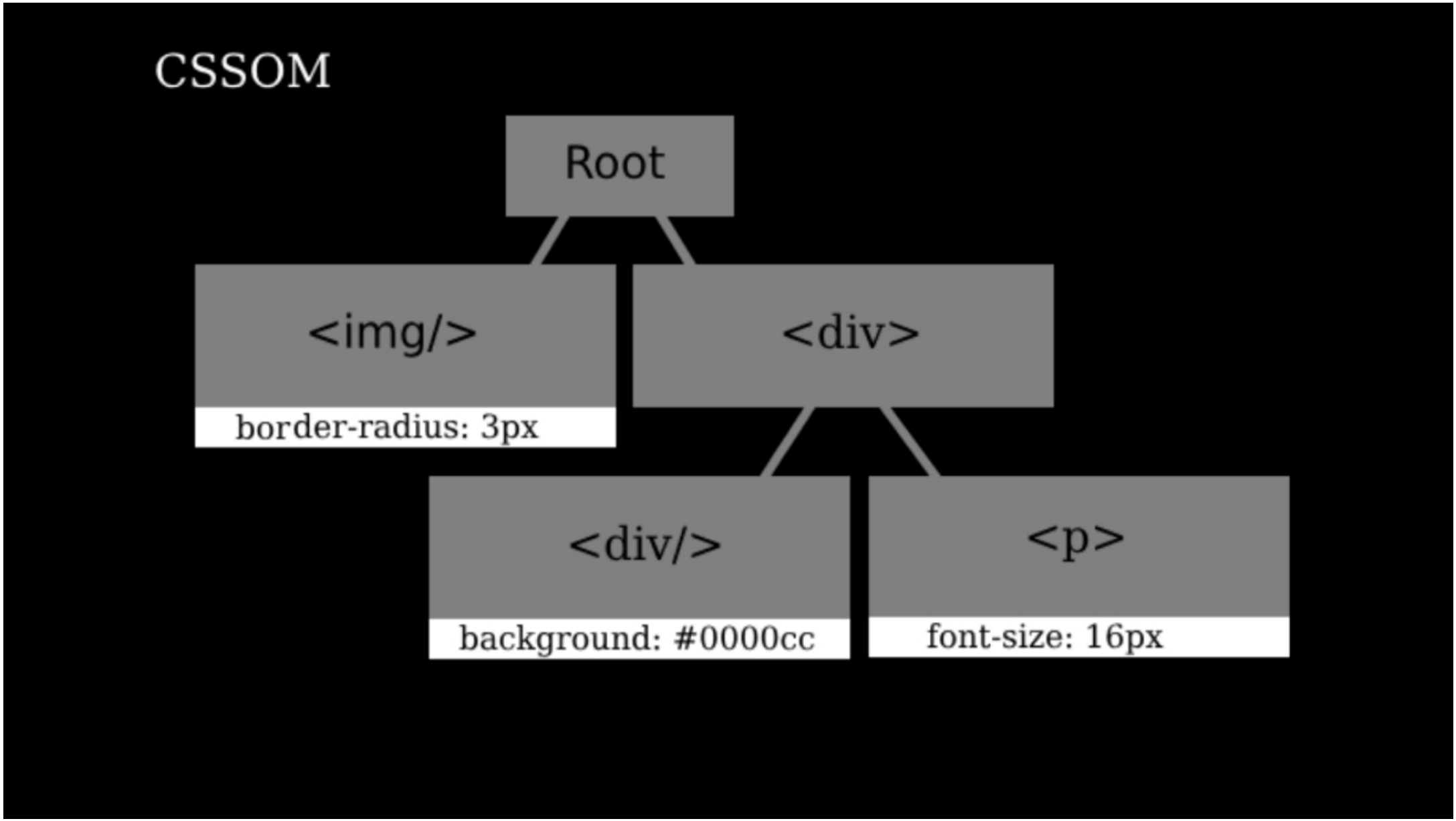
Don't act, react


NEW

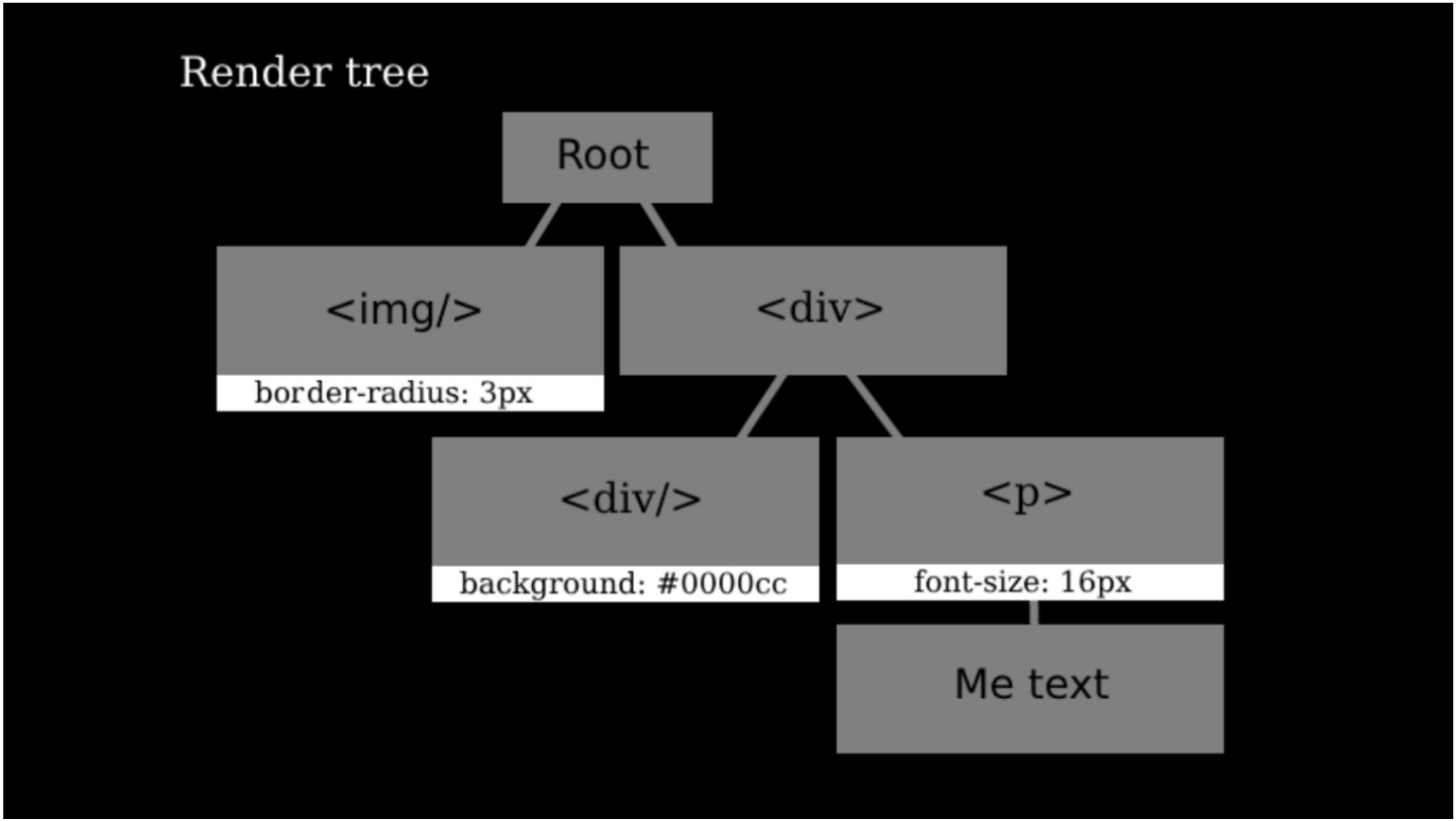
```
<body>
  <img class="image" />
  <div>
    <div class="nested-div" />
    <p class="text">Me text</p>
  </div>
</body>
```


DOM
Root
<img/>

<p>
Me text


CSSOM
Root
<img/>
border-radius: 3px

background: #0000cc
<p>
font-size: 16px

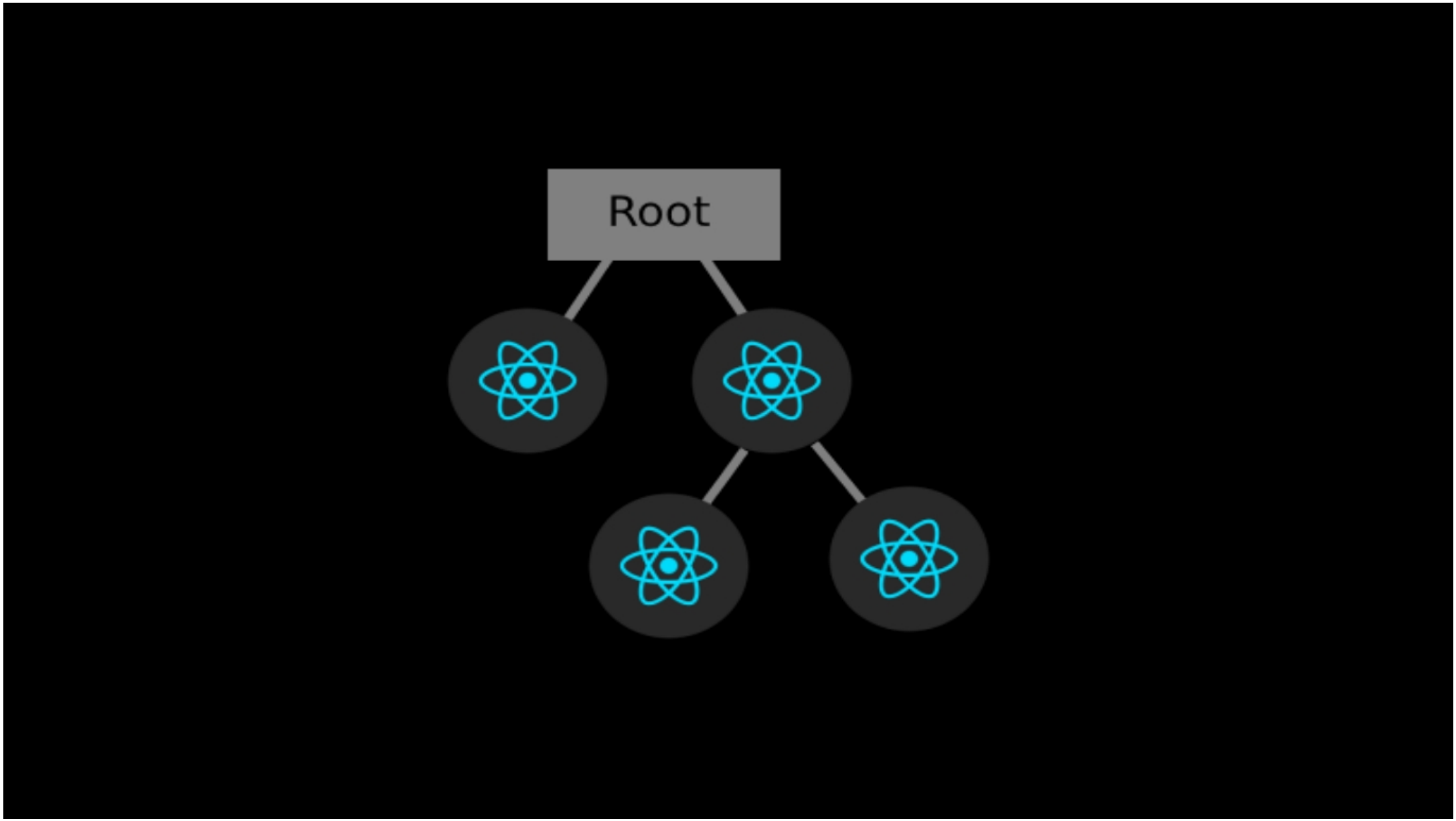

Render tree
Root
<img/>
border-radius: 3px

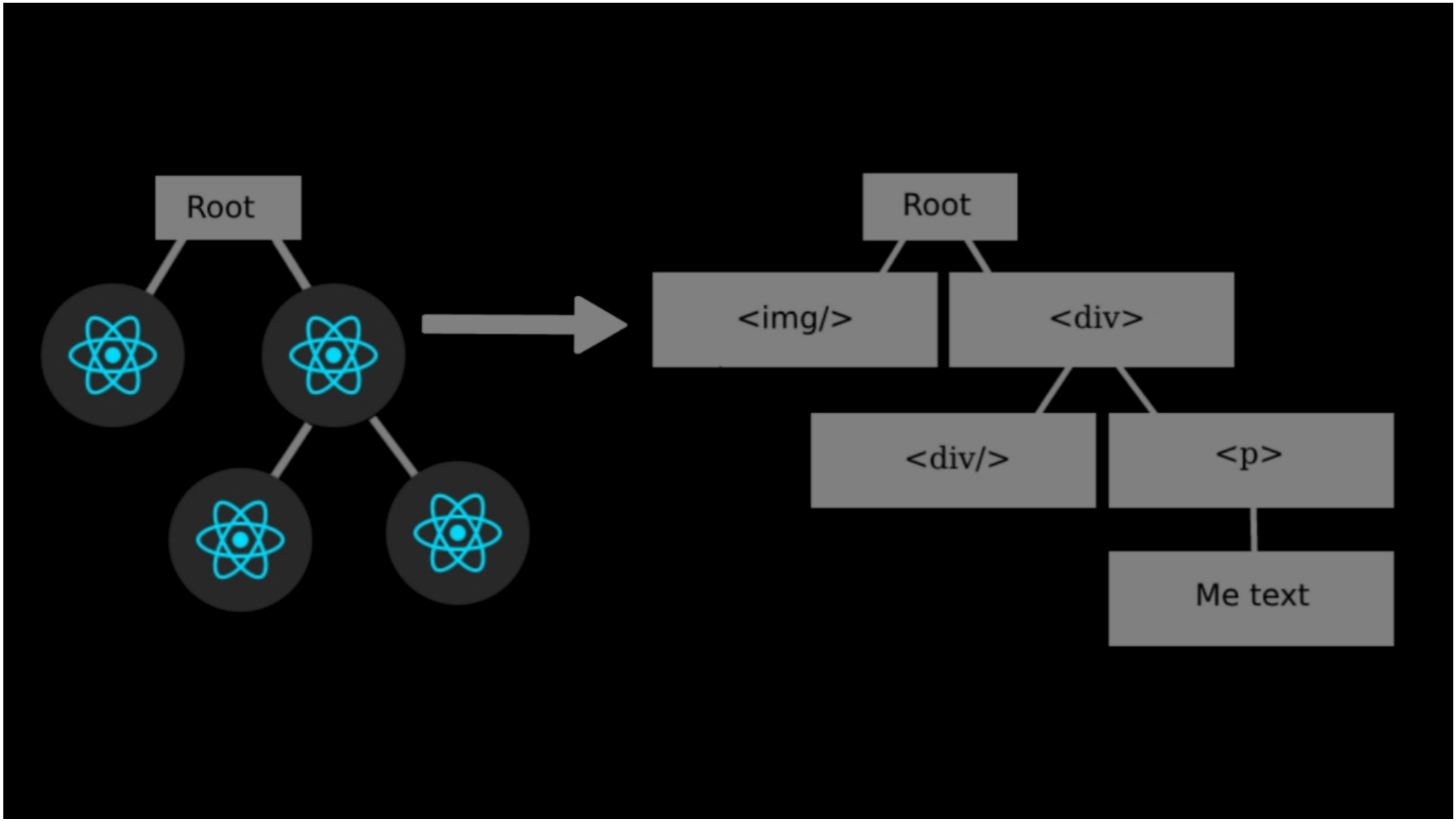
background: #0000cc
<p>
font-size: 16px
Me text

```
const header = document.getElementById('headerText');
if (header != null) {
  header.innerHTML = 'This is not a text';
}
```

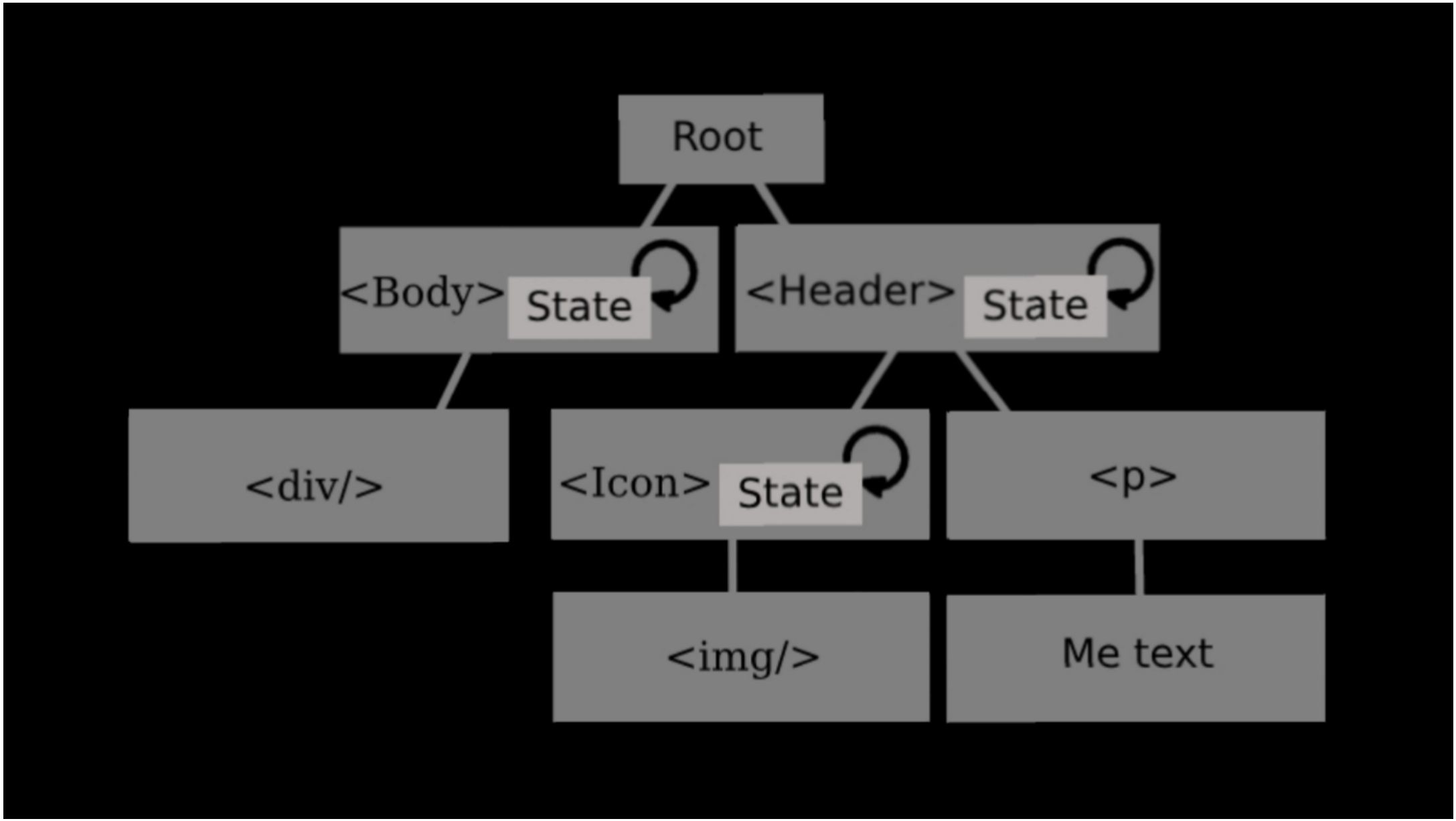
Think about how the UI should look at any given moment rather than how to change it

reactjs.org

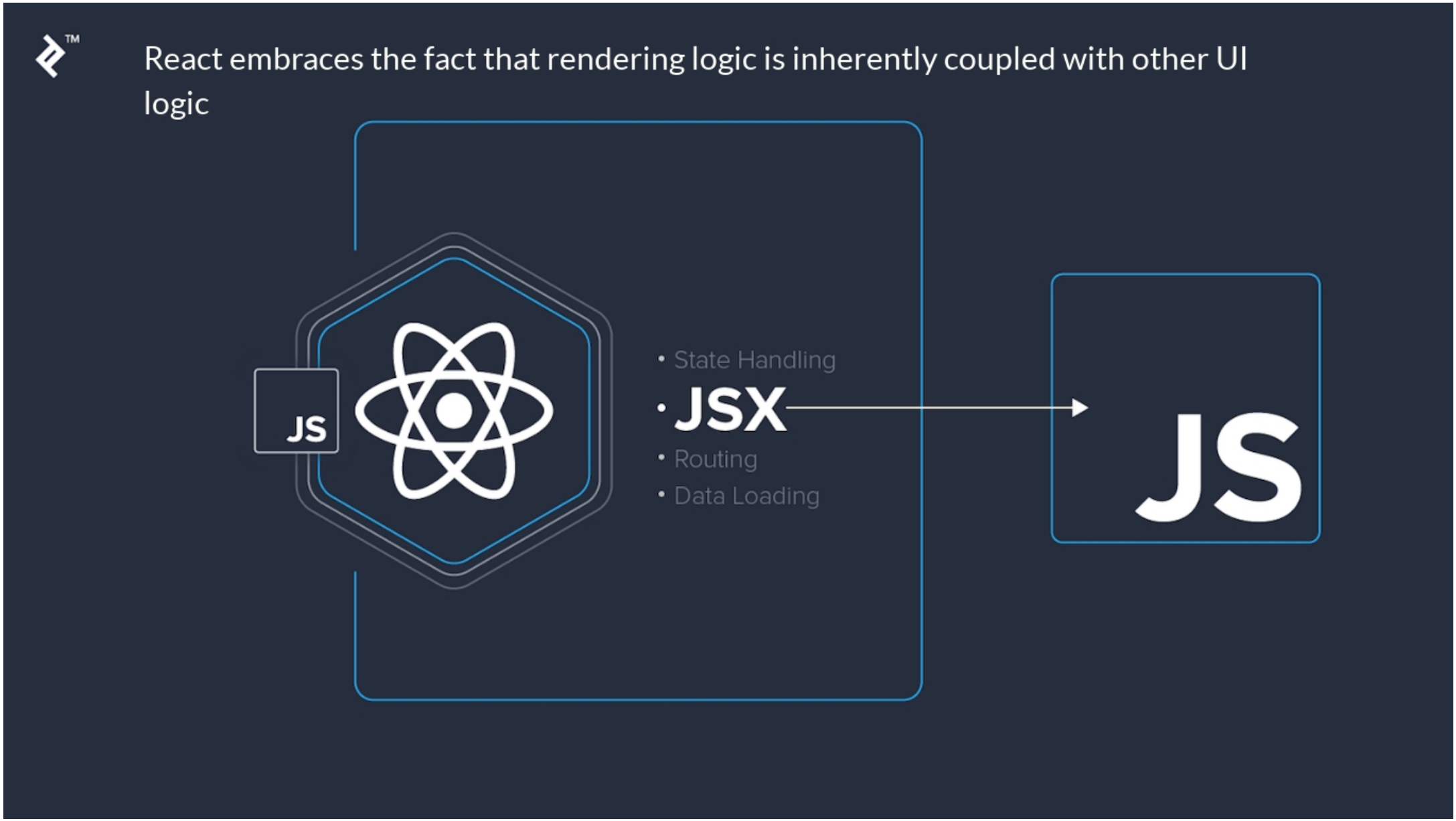

Root

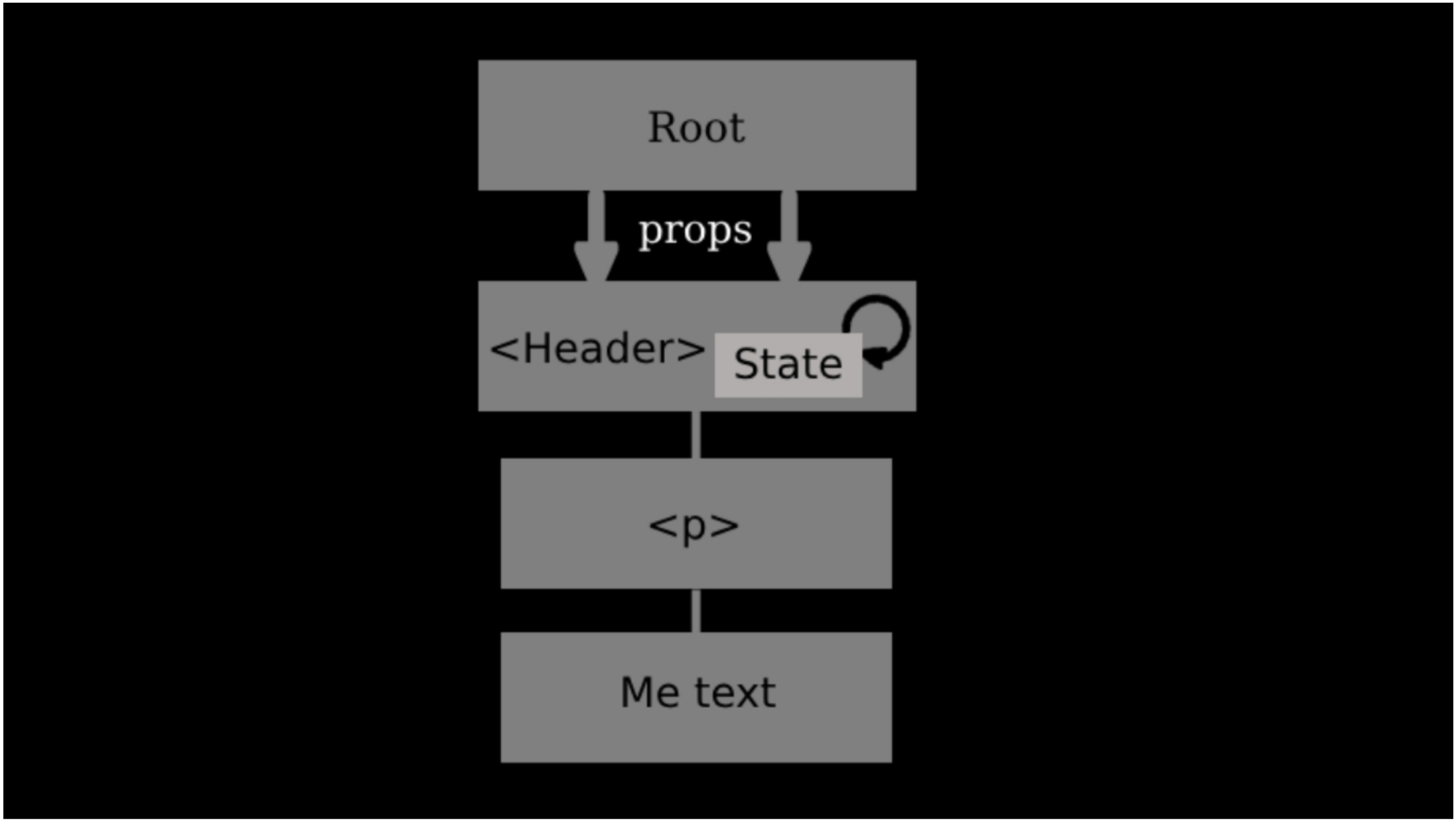

Root
Root
<img/>

<p>
Me text


Root
<Body>
State
<Header>
State

<Icon>
State
<p>
<img/>
Me text

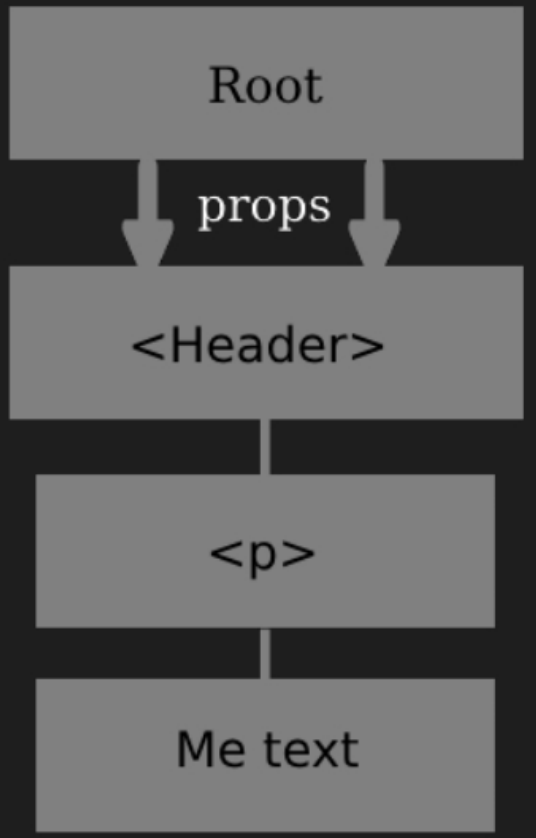

React embraces the fact that rendering logic is inherently coupled with other UI logic
JS
• State Handling
• JSX
• Routing
• Data Loading
JS


Root
props
<Header>
State
<p>
Me text

```
{
    headerText: "Me text"
}
```

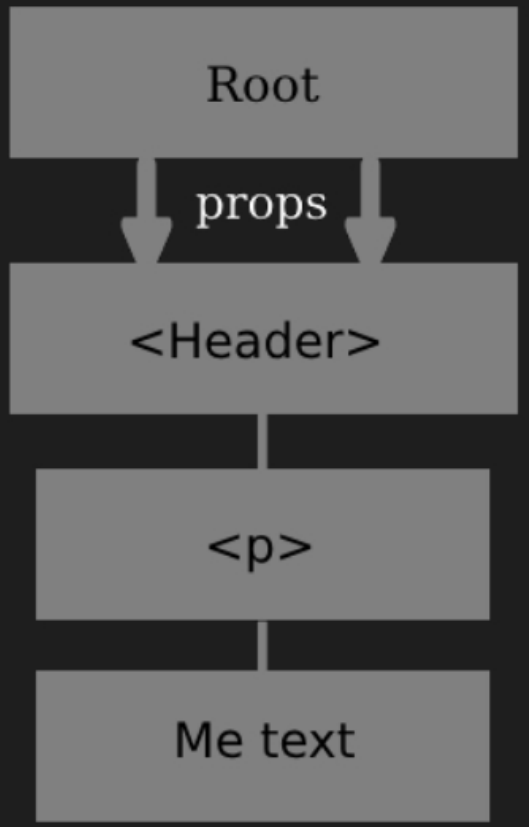
<Header headerText='Me text' />

```jsx
function Root() {
  return <Header headerText='Me text' />;
}

function Header(props) {
  return <p>{props.headerText}</p>;
}
```

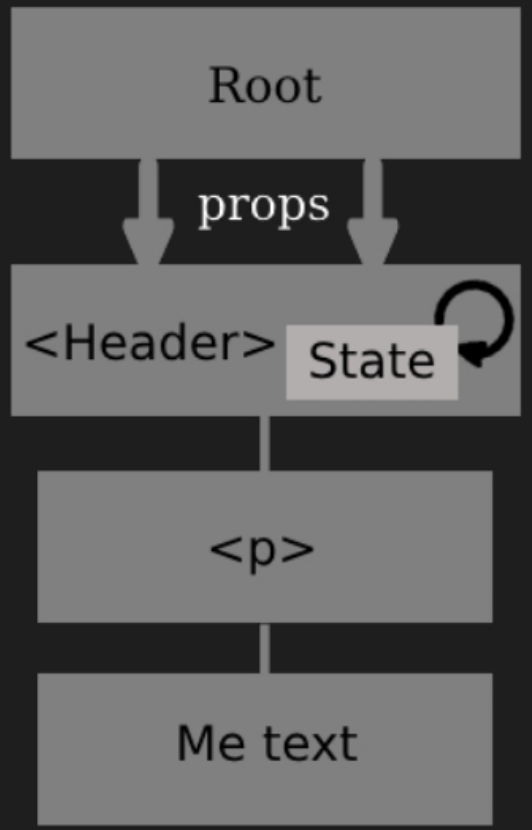


```
const rootElement = document.getElementById( "root" );
ReactDOM.render( <Root />, rootElement );
```

```
class Root extends React.Component {
  render() {
    return <Header headerText="Me text" />;
  }
}

class Header extends React.Component {
  render() {
    return <p>{this.props.headerText}</p>;
  }
}
```



```
class Root extends React.Component {
  render() {
    return <Header headerText="Me text" />;
  }
}

class Header extends React.Component {

  constructor(props){
    super(props);
    this.state = {
      headerText = props.headerText;
    }
  }

  render() {
    return <p>{this.state.headerText}</p>;
  }
}
```



```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      headerText: props.headerText
    };
  }

  componentDidMount() {
    if (this.props.userId == null) {
      this.setState({ headerText: 'Whaddup, Anonymous?' });
    }
  }

  render() {
    return <p>{this.state.headerText}</p>;
  }
}
```

```
componentDidMount() {
  // handle mounting logic here
  if (this.props.userId == null) {
    this.setState({ headerText: "Whaddup, Anonymous?" });
  }
}

componentWillUnmount() {
  // clean up after yourself
  clearTimeout(this.timeout);
}
```

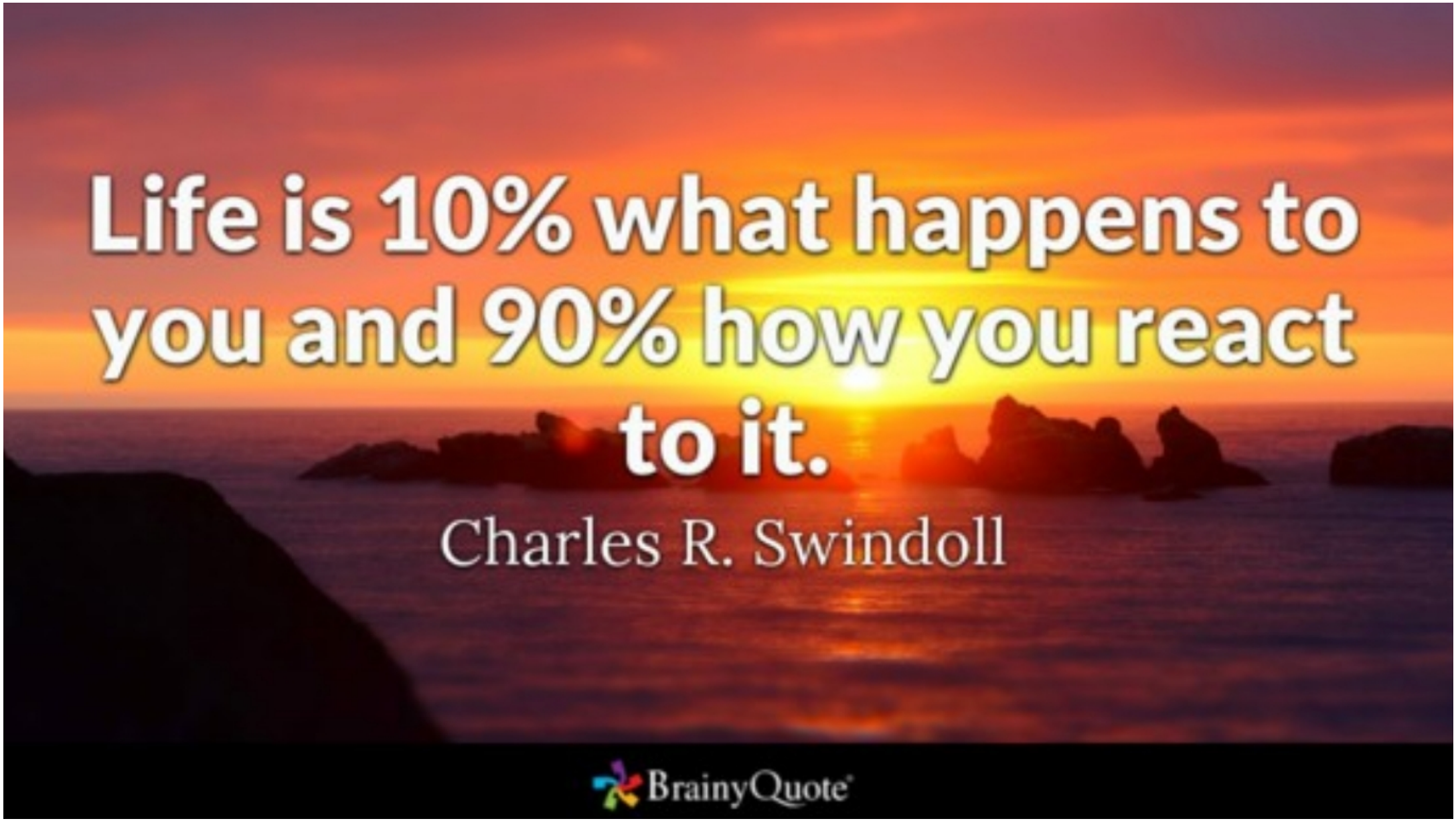
```
shouldComponentUpdate(nextProps, nextState) {
  // tell component to maybe skip an unnecessary update
  return !equals(nextProps, this.props) || !equals(nextState, this.state);
}

componentDidUpdate(prevProps, prevState) {
  // handle data change or component transition
  // DO NOT set state here
  if (this.props.fetchSuccess && !prevProps.fetchSuccess) {
    this.closeSpinner();
  }
}
```

Build encapsulated components that manage their own state, then compose them to make complex UIs

reactjs.org

Think about how the UI should look at any given moment rather than how to change it

reactjs.org


Life is 10% what happens to you and 90% how you react to it.
Charles R. Swindoll
BrainyQuote