

React Fundamentals

React Core Concepts



Topics

- Understanding Components
 - Functional Components
 - Class Components
- Anatomy of a Component:
 - Render
 - Props
 - State
 - Methods



React Core Concepts

Understanding Components

Functional Components

- Bare minimum encapsulated React component
- Returns JSX / some other component(s)

```
function App() {  
  return (  
    <div>  
      Hello World!  
    </div>  
  )  
}
```

Class Components

- Extends React.Component class
- Allows for defining 'state' property
- Re-renders when state or props are updated

```
class App extends React.Component {  
  constructor(props) {  
    this.state = {  
      someValue: 'World'  
    }  
  }  
  
  render() {  
    return (  
      <div>  
        Hello {this.state.someValue}!  
      </div>  
    )  
  }  
}
```



React Core Concepts

Component Rendering

Component Rendering

- All components must include a return statement
- A component with a return represents the minimum possible component
- Typically, return is composed of JSX / child components
- In a class component, the return statement is included inside the render() method

```
function App() {  
  return (  
    <div>  
      This is the rendered div!  
    </div>  
  )  
}
```

```
class App extends React.Component {  
  
  render() {  
    return (  
      <div>  
        This is the rendered div!  
      </div>  
    )  
  }  
}
```



React Core Concepts

Component Props

Component Props

- Components may receive props from a parent component
- Props are received as an argument in function a functional component
- Components re-render when props are updated
- Props are READ-ONLY – do not try to change props

```
function Child(props) {  
  return (  
    <div>  
      <h1>Hello, {props.name}</h1>  
    </div>  
  )  
}  
  
function Parent() {  
  return (  
    <div>  
      <Child name='Tim' />  
    </div>  
  )  
}
```



React Core Concepts

Component State

Component State

- Values necessary for your application to function
- Can also be defined directly as class property (without constructor)
- Should only be updated by calling `this.setState()` method
- Components re-render when state is updated

```
class App extends React.Component {  
  constructor(props) {  
    this.state = {  
      someValue: 'World'  
    }  
  }  
  
  // state can also be defined directly  
  // without constructor  
  // state = {  
  //   someValue: 'World'  
  // }  
  
  render() {  
    return (  
      <div>  
        Hello {this.state.someValue}!  
      </div>  
    )  
  }  
}
```



React Core Concepts

Component Methods

Component Methods

- Used to handle events and update state value
- Updating state triggers component re-render
- If method is passed to another component or element, `this` must be bound inside constructor*

* arrow functions allow methods to automatically bind to class's `this` value

```
class App extends React.Component {
  constructor(props) {
    this.handleIncrement = this.handleIncrement.bind(this)

    this.state = {
      count: 0
    }
  }

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

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



Let's try it!

