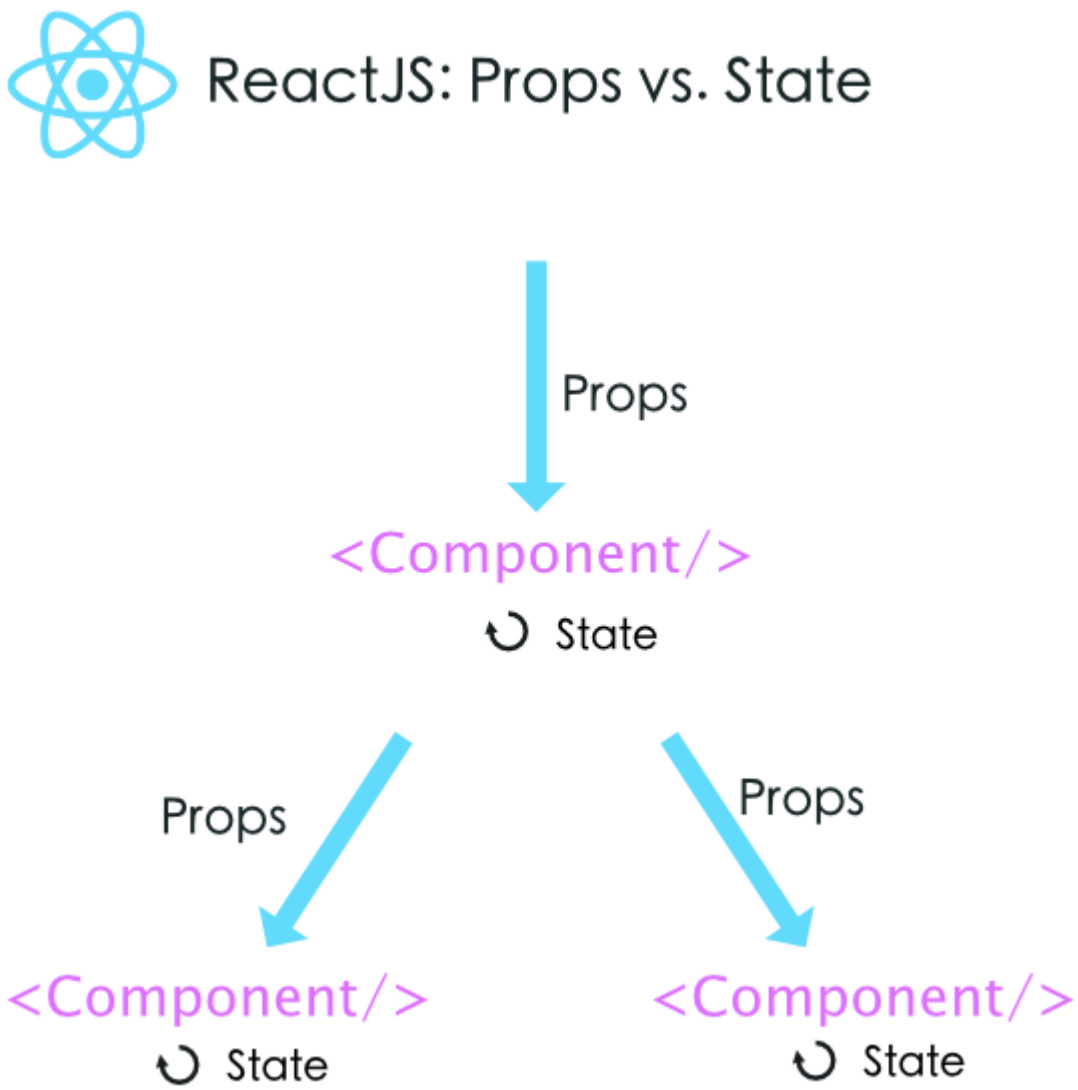
3 - Welcome

Props

Props are used to pass data from parent to child or by the component itself. They are immutable and thus will not be changed.

Funtional Component

Function that returns a JSX object. Can receive props as argumesnts. Export the Funtional Component to use it on other parts of your app. Outside of the return object it works the same as a normal JS function.



In our app:

```
// Functional Component with prop object
const Welcome = (props) => {
  return (
    <section>
      <h1>{`Welcome to ${props.name} !`}</h1>
```

```
        <img alt={props.name} src={props.image}></img>
      </section>
    );
  }
```

Import and use components

```
import Welcome from './Welcome';

class App extends Component {
  constructor() {
    super(props);
    this.state = {
      name: "React Academy",
      image:
"https://media1.tenor.com/images/fe250a86e1dfa2648481e7da5ebd441b/tenor.gif?
itemid=5510026"
    }
  }

  render() {
    return (
      <section>
        <Welcome name={this.state.name} image={this.state.image}/>
      </section>
    );
  }
}
```

Read More:

- Components and Props

Bonus: Fuctional Components: The key thing that makes this type of component different from a class component is the **lack of state and lifecycle methods**. This is why functional components are also commonly referred to as stateless components.

Why Functional Components

- Functional components are easy to read
- Functional components are easy to test
- Functional components can potentially have a better performance
- Functional components are easy to debug
- Functional components are more reusable

<< return