React JSX & Components

React Session-2

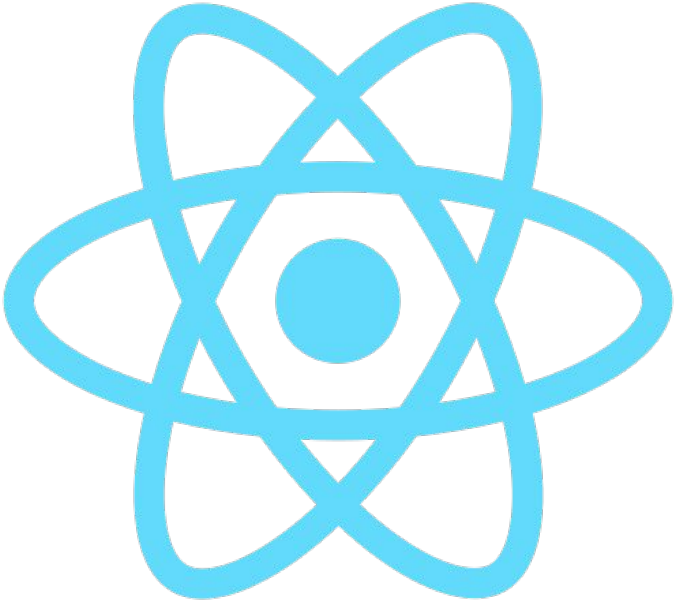
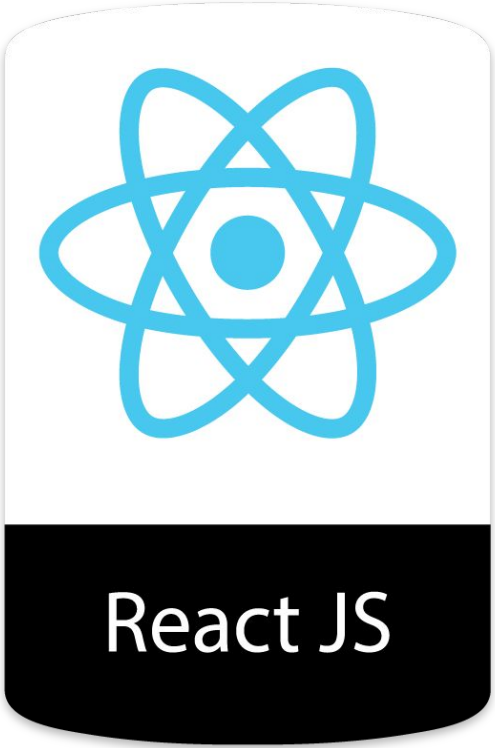


Table of Contents

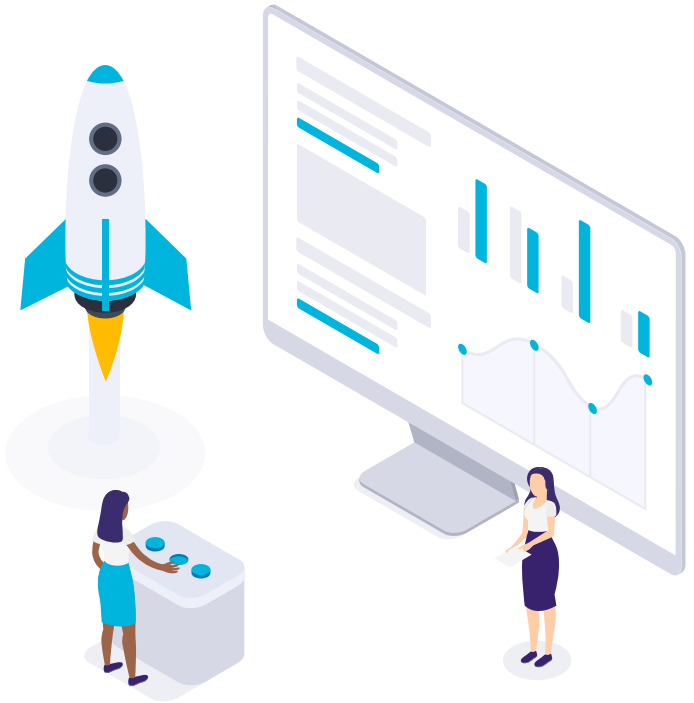
Did you finish
React
pre-class
material?



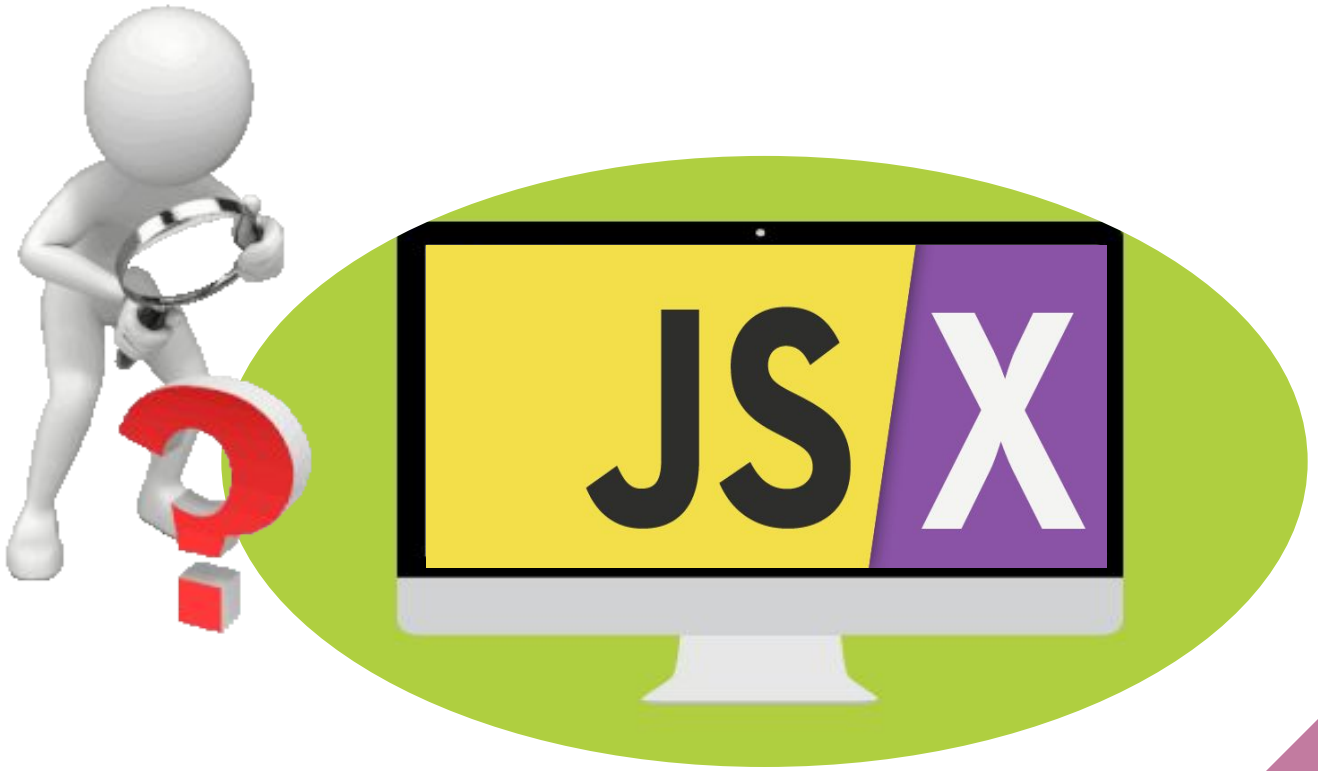
Students choose an option


Kahoot!

1 What is JSX?



What is JSX?



What is JSX?

- Arguably, one of the coolest things in React
- XML-like syntax for generating component's HTML
- Easier to read and understand large DOM trees
- Translates to plain JavaScript using react-tools

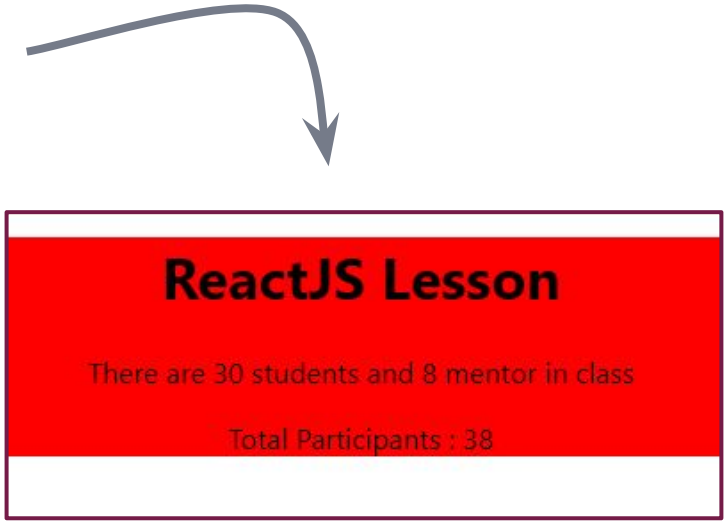
What is JSX?

```
/* @jsx React.DOM */
function App(){
  return (
     <div className=”test”>
          <h2>
               <strong>Example : </strong> React App
          </h2>
     </div>)
}
/* Regular DOM */
function App() {
  return React.createElement("div", {
    className: "test"
  }, React.createElement("h2", null, React.createElement("strong", null, "Example : "), " React App"));
}
```

JS =? JSX

What is JSX?

```
function App() {
  const header = "ReactJS Lesson";
  const numberOfStudent = 30;
  const numberOfMentor = 8;
  let bgCol = "red";
  return (
    <div className="App" style={{ backgroundColor: bgCol }}>
      <h1>{header}</h1>
      <p className="parg">
        There are {numberOfStudent} students and {numberOfMentor}
mentor in class</p>
      <p className="parg">
        Total Participants : {numberOfStudent + numberOfMentor}
      </p>
    </div>
  );
}
```



Comment in JSX

- You can add comments to JSX by using the normal JavaScript comments inside an expression:

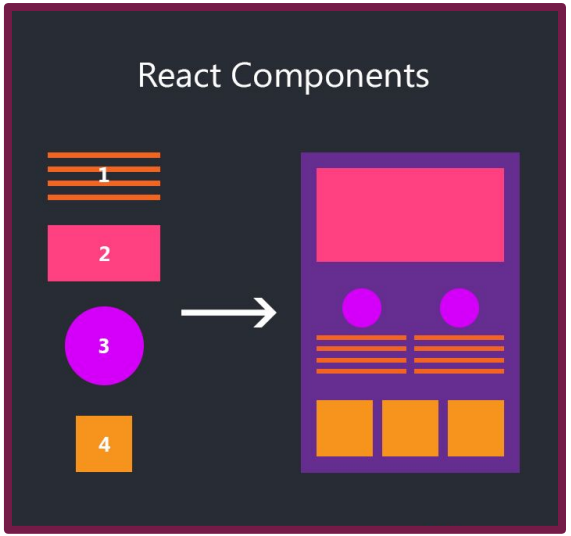
```
function App() {

  return (
    <div>
        <p>
          {/*
          multiline
          comment
          */}
          {
            //another comment
          }
        </p>
    </div>
  );
}
```

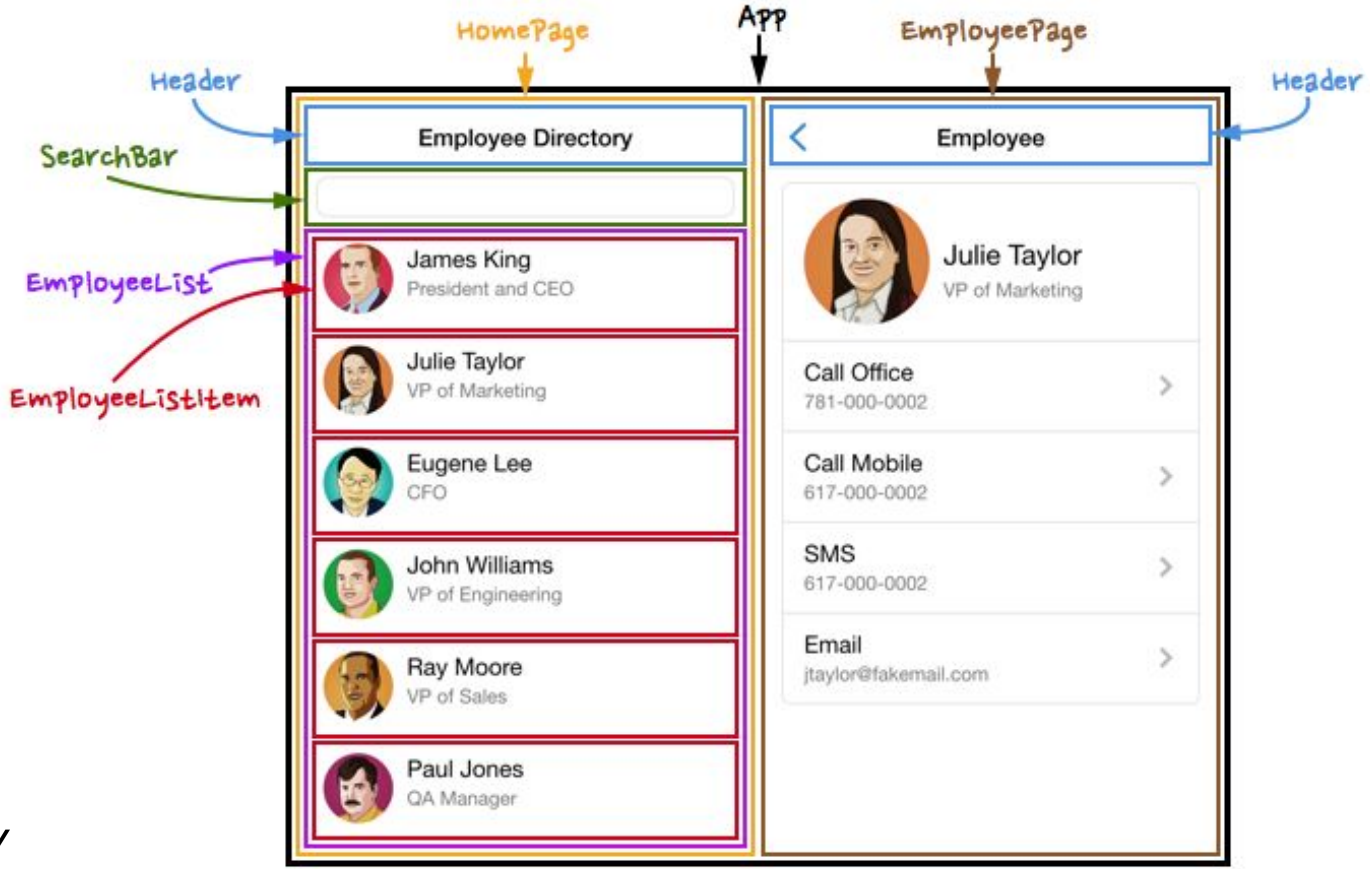
2 What is Component?

What is component?

- Components are self-contained reusable building blocks of web application.
- React components are basically just idempotent functions(same input produces same output).
- They describe your UI at any point in time, just like a server-rendered app.



What is component?



3

Class & Functional Component

```
class ClassComponent extends React.Component {
  render() {
    return <div>{this.props.name}</div>;
  }
}
```


```
const StatelessComponent = props => <div>{props.name}</div>;
```

Class component

- **Class components generally use in old projects.**
- **Should use `React.Component`**
- **Must contain `render()` method.**
- **Class components are relatively slow from function components.**
- **You need to use `this` keyword.**

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, Ed!</h1>;
  }
}
```

Functional component

- **Functional component just javascript function and simple**
- **Functional components generally used in new projects.**
- **Functional components are faster than class components.**
- **Can be used lifecycle methods and states in functional component (via hooks)**

```
function Welcome() {
  return <h1>Hello, Diana!</h1>;
}
```

Functional component

Functional Component

===

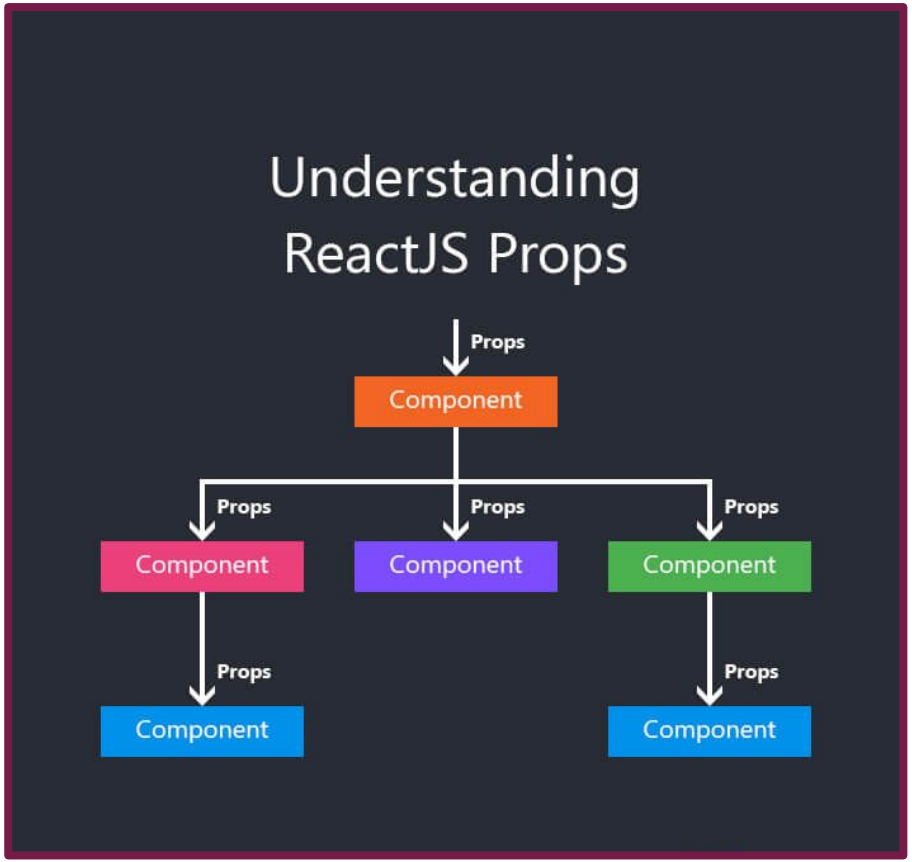
Stateless Component

Class Component

===

Stateful Component

4 Props


Understanding
ReactJS Props
Props
Component
Props
Component
Props
Component
Props
Component
Props
Component
Props
Component

Props

- **Passed down to component from parent component and represents data for the component**
- **Accessed via *this.props* in class components**

```
class Dog extends React.Component {
  render() {
    return <h1>My dog is a {this.props.breed}!</h1>
  }
}


const myComponent = <Dog breed="Poodle" />;
```

Props

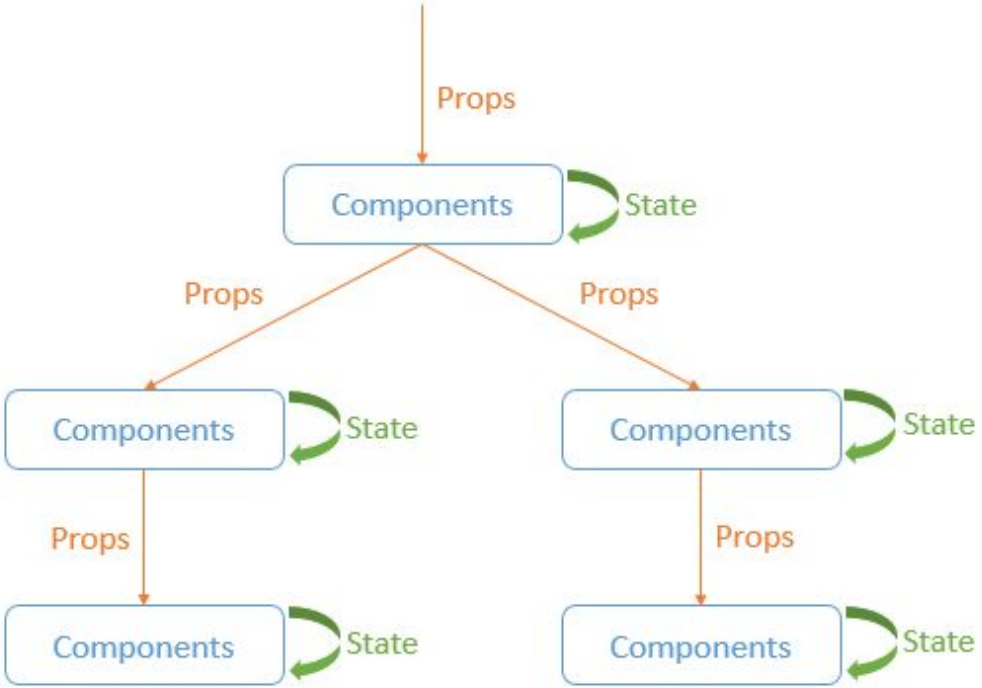
In functional components

- **You should put prop in arguments of function.**
- **You can get props destruct the props `({breed})`.**

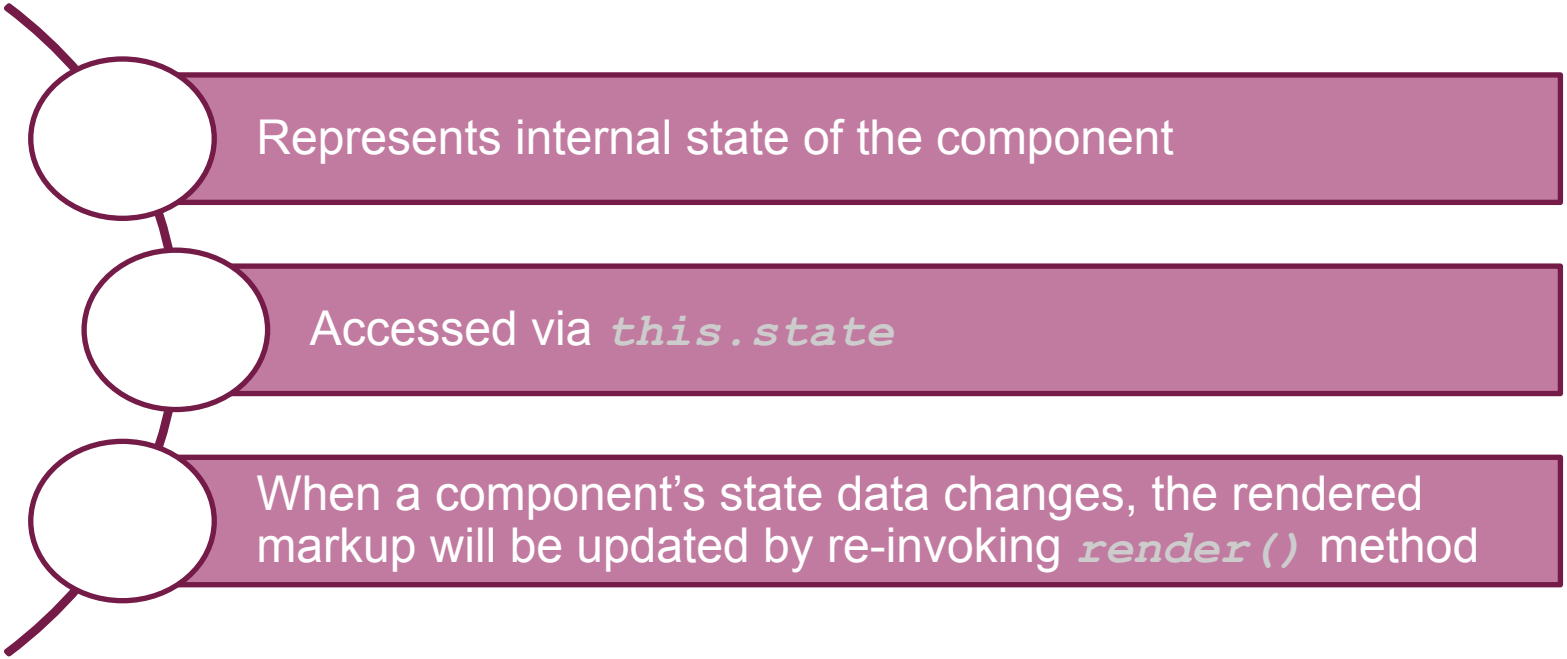
```
const Dog  = (props) => {
    return <h1>My dog is a {props.breed}!</h1>
}


const myComponent = <Dog breed="Poodle" />;
```

5 State


Props
Components
State
Props
Props
Components
State
Components
State
Props
Props
Components
State
Components
State

State


Represents internal state of the component
Accessed via this.state
When a component's state data changes, the rendered markup will be updated by re-invoking render() method

State

```
class Dog extends React.Component {
  constructor(props) {
    super(props);
    this.state = {breed: "Dalmatian"};
  }
  render() {
    return (
      <div>
        <h1>My Dog is {this.state.breed}</h1>
      </div>
    );
  }
 }
```

Changing State

```
class Counter extends React.Component {
 constructor(props) { super(props);
   this.state = { counter: 0 } }
 changeCounter = () => { this.setState({ counter: this.state.counter + 1 })};
 render() {
   return (
     <div>
       <h1>My counter: {this.state.counter}</h1>
       <button type="button" onClick={this.changeCounter}>Change Counter</button>
     </div>
   )}
}
```

Props vs State

props

- props get passed to the component
- Function Parameters
- props are immutable
- Functional Components : props
- Class Components : this.props

state

- state is managed within the component
- Variables declared in the function body
- state can be changed
- Functional Components: useState Hook
- Class Components: this.state

Destructuring State and Props

```
const Welcome = ({ imgUrl, name, description }) => {
  // const Welcome = (props) => {
  // const { imgUrl, name, description } = props
  return (
    <div>
      <img src={imgUrl} alt="dog" />
      <h1>{name}</h1>
      <p>{description}</p>
    </div>
  );
};
export default Welcome;
```

THANKS!

Any questions?