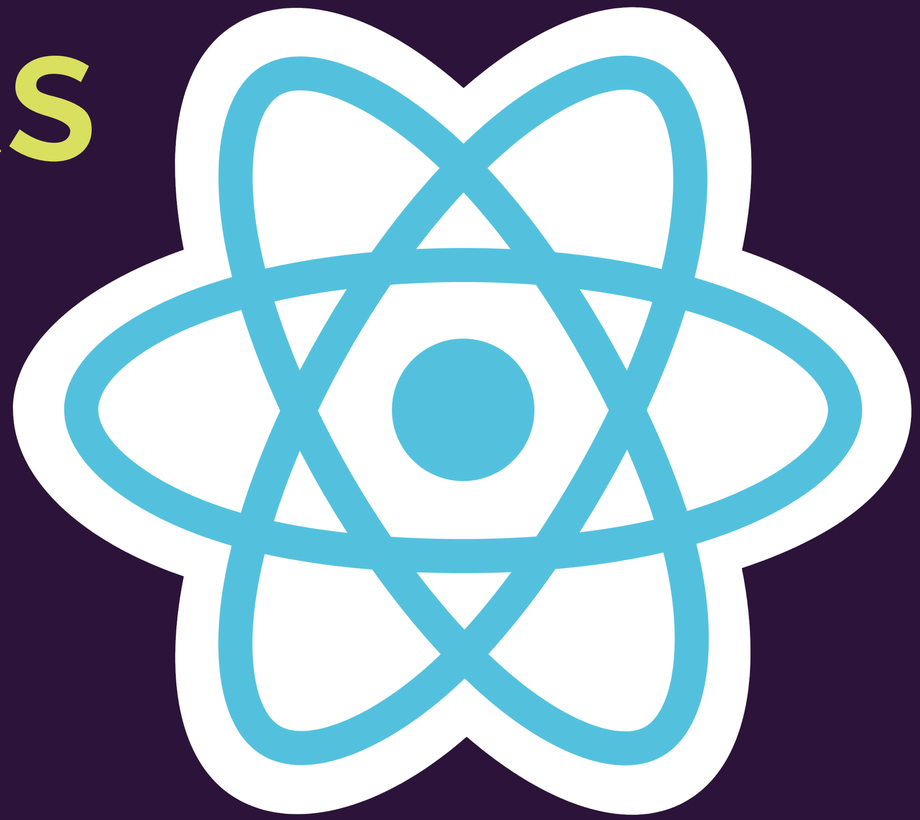
React Class Components
& Multi-room Real-time Chat



Hooks motive

Hooks let you use more of
React's features without classes.

Hooks motive

It’s hard to reuse stateful logic between components.

Complex components become hard to understand.

Classes confuse both people and machines.

Hooks motive

It’s hard to reuse stateful logic between components.

Complex components become hard to understand.

Classes confuse both people and machines.

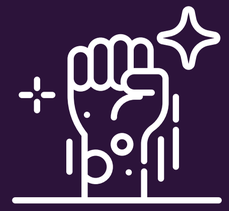
Hooks motive

It’s hard to reuse stateful logic between components.

Complex components become hard to understand.

Classes confuse both people and machines.

ES4
2003



ES6
2016

classes

A Component before ES6.

```
var React = require("react");

var Counter = React.createClass({
  getInitialState: function () {
    return { count: this.props.initialCount };
  },
  handleClick: function () {
    this.setState({
      count: this.state.count + 1,
    });
  },
  render: function () {
    return (
      <div>
        {count}
        <button onClick={this.handleClick}>Increment</button>
      </div>
    );
  },
});
```

Create a new instance of a function object.

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

Point.prototype.getPosition = function getPosition() {
  return [this.x, this.y];
};

const cursor = new Point(0, 0);
console.log(cursor.getPosition()); // [0, 0]
```

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  getPosition() {
    return [this.x, this.y];
  }
}

const cursor = new Point(0, 0);
console.log(cursor.getPosition()); // [0, 0]
```

ES6 Class Syntax

```
import React, { Component } from "react";

class Point extends Component {
  render() {
    return (
      <div>
        {this.props.x}, {this.props.y}
      </div>
    );
  }
}
```

ES6 Class Component

```jsx
const Point = ({ x, y }) => (
  <div>
    {x}, {y}
  </div>
);
```

Pure

Stateful

```
class Counter extends Component {
  state = {
    count: 0,
  };

  render() {
    return <div>{this.state.count}</div>;
  }
}
```

Constructor Usage

```
class Counter extends Component {
  constructor(props) {
    super(props);

    this.state = {
      count: 0,
    };
  }

  render() {
    return <div>{this.state.count}</div>;
  }
}
```

Changing State

```
class Counter extends Component {
  state = {
    count: 0,
  };

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

Handling Events

```
class Counter extends Component {
  state = {
    count: 0,
  };

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

  render() {
    return (
      <div>
        {this.state.count}
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

`this` context

```
class Counter extends Component {
  state = {
    count: 0,
  };

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

  render() {
    return (
      <div>
        {this.state.count}
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

`this` context

Component LifeCycle

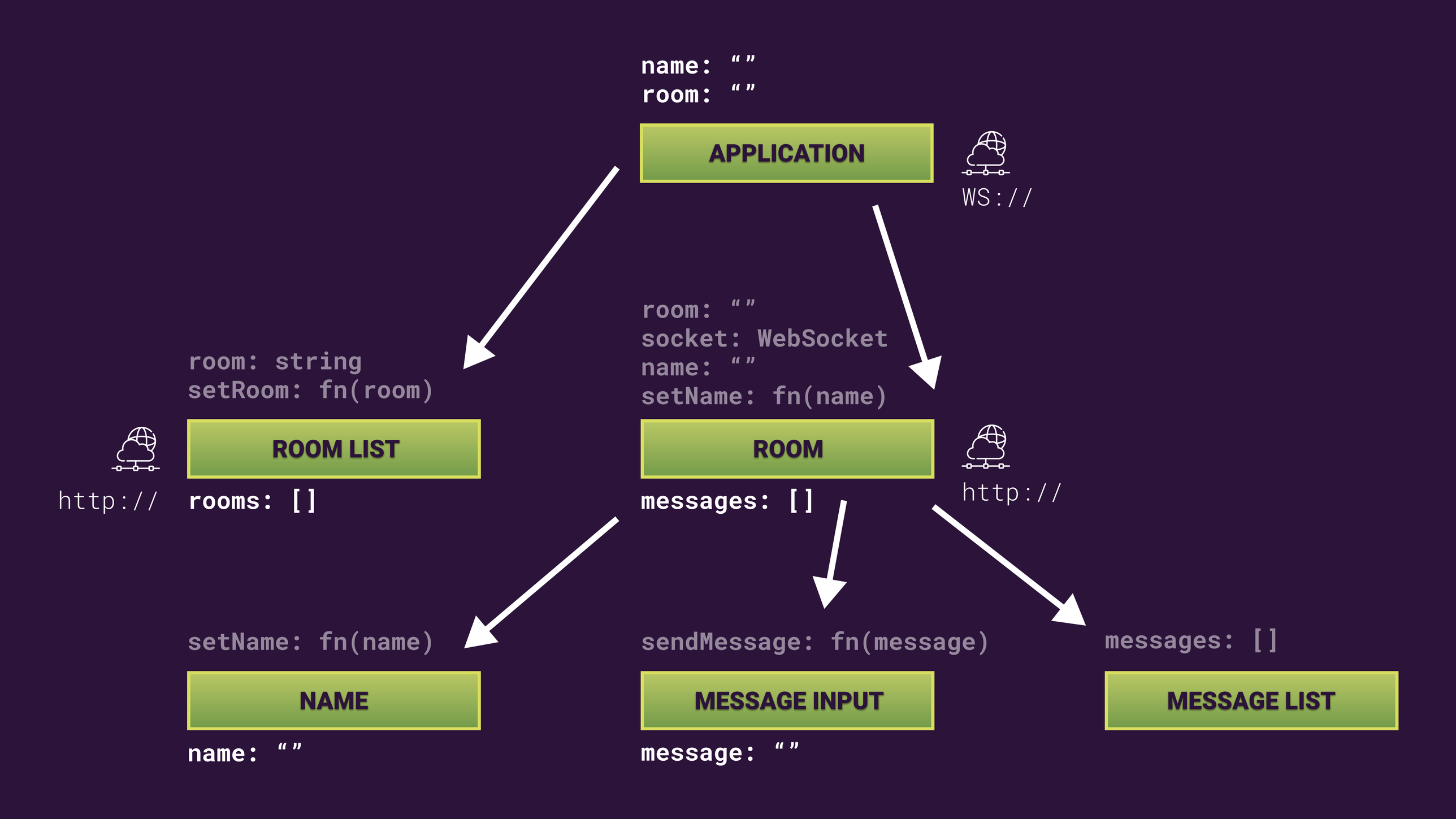
```
class MountingExample extends Component {
  constructor(props) {
    super(props);
    console.log("First the constructor");
  }

  componentDidMount() {
    console.log("Third the componentDidMount");
  }

  render() {
    console.log("Second the render");
    return <div />;
  }
}
```

```
class UnmountingExample extends Component {
  componentWillUnmount() {
    console.log("Called when the component is removed from the dom");
  }

  render() {
    return <div />;
  }
}
```

```
class Counter extends Component {
  state = {
    count: 0,
  };

  componentDidUpdate(prevProps, prevState) {
    console.log("Either props or state has changed.");

    if (prevState.count !== this.state.count) {
      console.log("The count state has changed.");
    }
  }

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

  render() {
    return (
      <div>
        {this.state.count}
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

Questions

Let's Code


name: “”
room: “”
APPLICATION
WS://
room: string
setRoom: fn(room)
ROOM LIST
http://
rooms: []
room: “”
socket: WebSocket
name: “”
setName: fn(name)
ROOM
http://
messages: []
setName: fn(name)
NAME
name: “”
sendMessage: fn(message)
MESSAGE INPUT
message: “”
messages: []
MESSAGE LIST