Higher Order Components

What are Higher Order Components? 🕴

- Functions that take a component as an argument and return another component
- Components that wrap other components

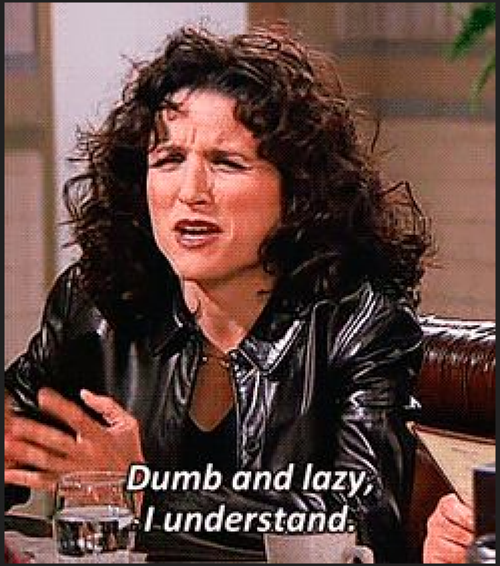
```
const FinalComponent = higherOrderComponent(WrappedComponent);
```

Why should I use HOCs?

Do your components look like this?

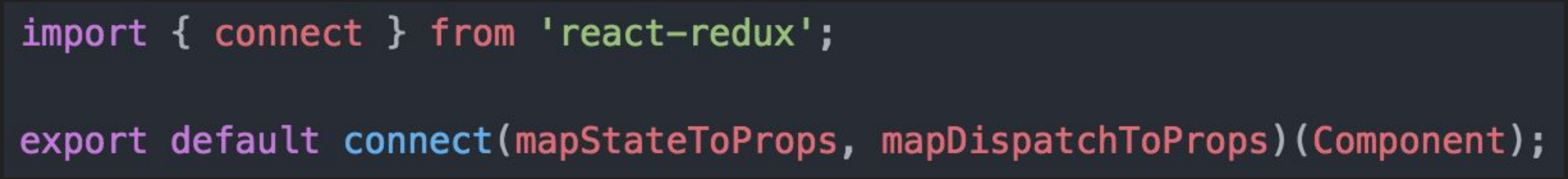
Why should I use HOCs?

- Composition > Inheritance
- Keep code DRY
- Define abstractions
- Extend the wrapped component
- Modify props
- Modify child components
- Let your components be dumb and lazy!

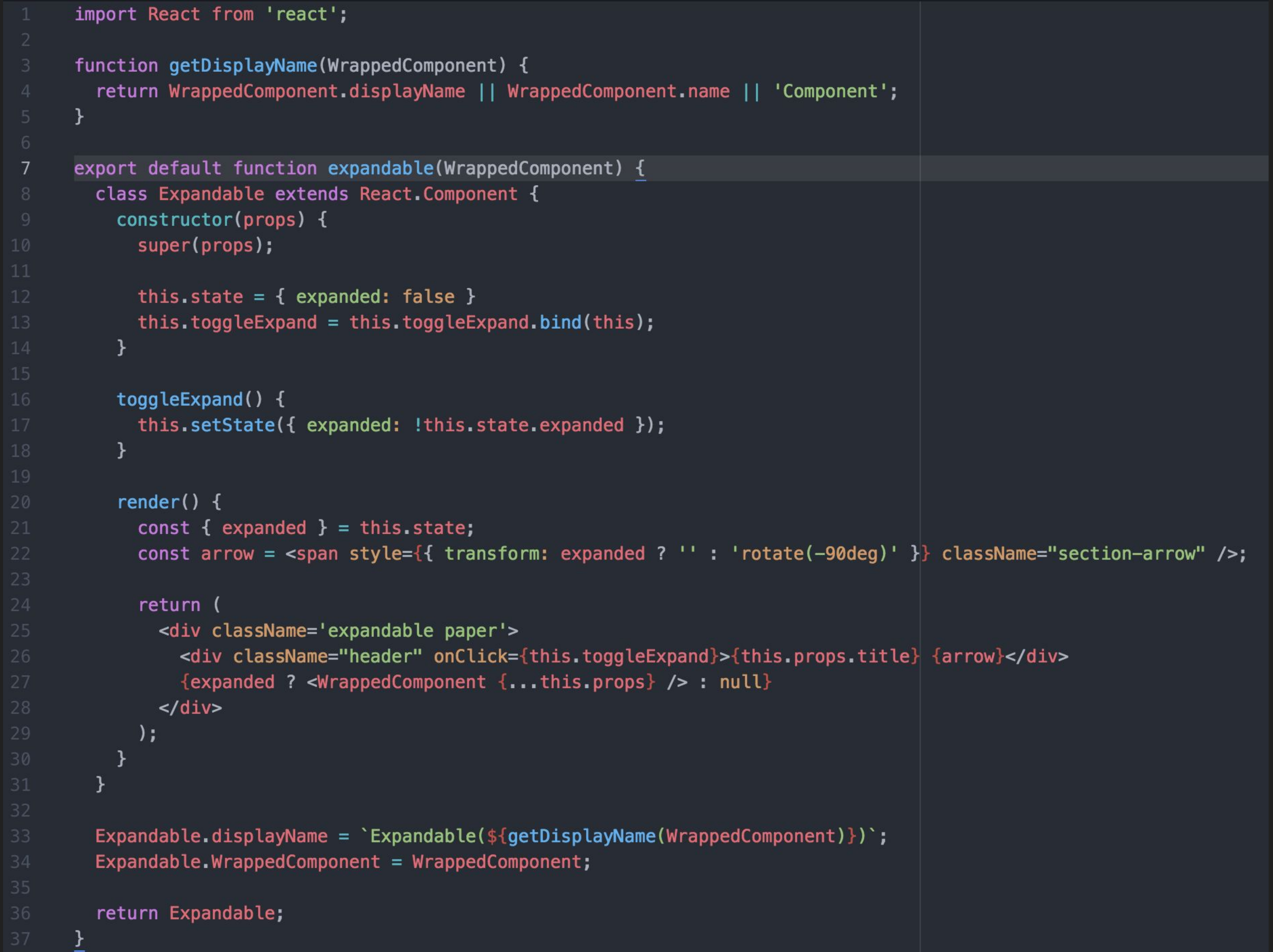


You already know one!

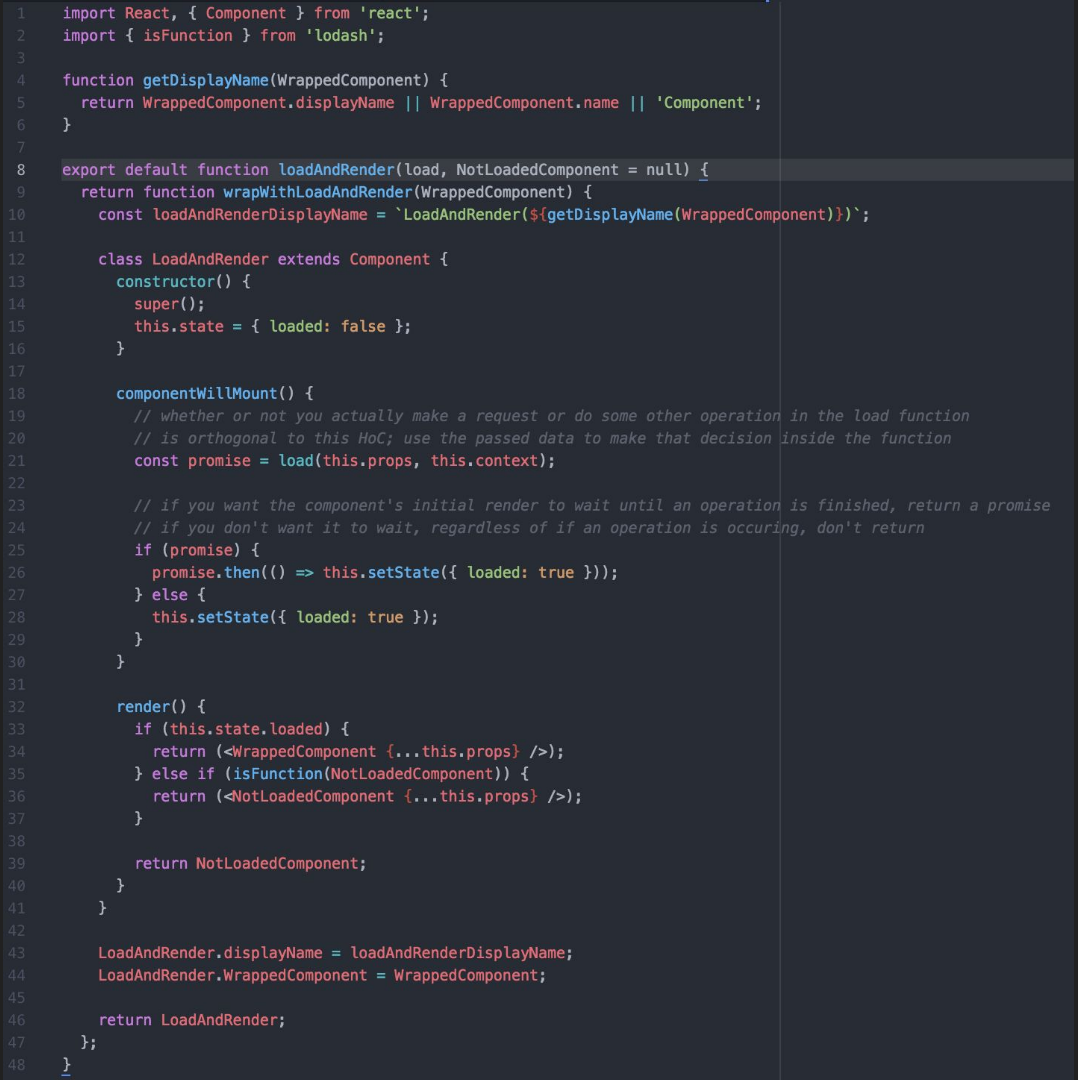
- connect in react-redux takes a component and returns a new component that is connected to the Redux store



```
import { connect } from 'react-redux';

export default connect(mapStateToProps, mapDispatchToProps)(Component);
```



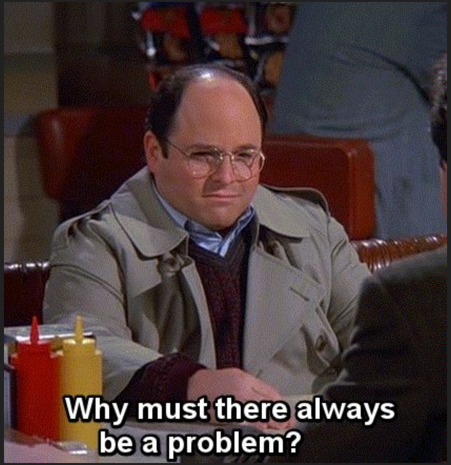
```
import React from 'react';

function getDisplayName(WrappedComponent) {
  return WrappedComponent.displayName || WrappedComponent.name || 'Component';
}

export default function expandable(WrappedComponent) {
  class Expandable extends React.Component {
    constructor(props) {
      super(props);

      this.state = { expanded: false }
      this.toggleExpand = this.toggleExpand.bind(this);
    }

    toggleExpand() {
      this.setState({ expanded: !this.state.expanded });
    }

    render() {
      const { expanded } = this.state;
      const arrow = <span style={{ transform: expanded ? '' : 'rotate(-90deg)' }} className="section-arrow" />;

      return (
        <div className='expandable paper'>
          <div className="header" onClick={this.toggleExpand}>{this.props.title} {arrow}</div>
          {expanded ? <WrappedComponent {...this.props} /> : null}
        </div>
      );
    }
  }

  Expandable.displayName = `Expandable(${getDisplayName(WrappedComponent)})`;
  Expandable.WrappedComponent = WrappedComponent;

  return Expandable;
}
```



```
import React, { Component } from 'react';
import { isFunction } from 'lodash';

function getDisplayName(WrappedComponent) {
  return WrappedComponent.displayName || WrappedComponent.name || 'Component';
}

export default function loadAndRender(load, NotLoadedComponent = null) {
  return function wrapWithLoadAndRender(WrappedComponent) {
    const loadAndRenderDisplayName = `LoadAndRender(${getDisplayName(WrappedComponent)})`;

    class LoadAndRender extends Component {
      constructor() {
        super();
        this.state = { loaded: false };
      }

      componentWillMount() {
        // whether or not you actually make a request or do some other operation in the load function
        // is orthogonal to this HoC; use the passed data to make that decision inside the function
        const promise = load(this.props, this.context);

        // if you want the component's initial render to wait until an operation is finished, return a promise
        // if you don't want it to wait, regardless of if an operation is occuring, don't return
        if (promise) {
          promise.then(() => this.setState({ loaded: true }));
        } else {
          this.setState({ loaded: true });
        }
      }

      render() {
        if (this.state.loaded) {
          return (<WrappedComponent {...this.props} />);
        } else if (isFunction(NotLoadedComponent)) {
          return (<NotLoadedComponent {...this.props} />);
        }

        return NotLoadedComponent;
      }
    }

    LoadAndRender.displayName = loadAndRenderDisplayName;
    LoadAndRender.WrappedComponent = WrappedComponent;

    return LoadAndRender;
  };
}
```

Tips & Tricks

- Manually set the displayName to make debugging easier
- Don't mutate the wrapped component!
- HOCs & "ref"s: ref will refer to the wrapper component rather than the wrapped component
    - You can use a callback to get at the inner ref if necessary
- Pass through all unrelated props to the wrapped component



debugging without custom displayNames

debugging with custom displayNames

More resources

- https://facebook.github.io/react/docs/higher-order-components.html
- https://medium.com/@franleplant/react-higher-order-components-in-depth-cf9032ee6c3e

Seinfeld References

S9E3: The Serenity Now

S9E12: The Reverse Peephole

S9E16: The Burning

S9E18: The Frogger

S9E19: The Maid

(me when y'all don't understand my references)