Design Patterns for ReactJS.

Higher Order Components and Protected Routes

Protected Routes

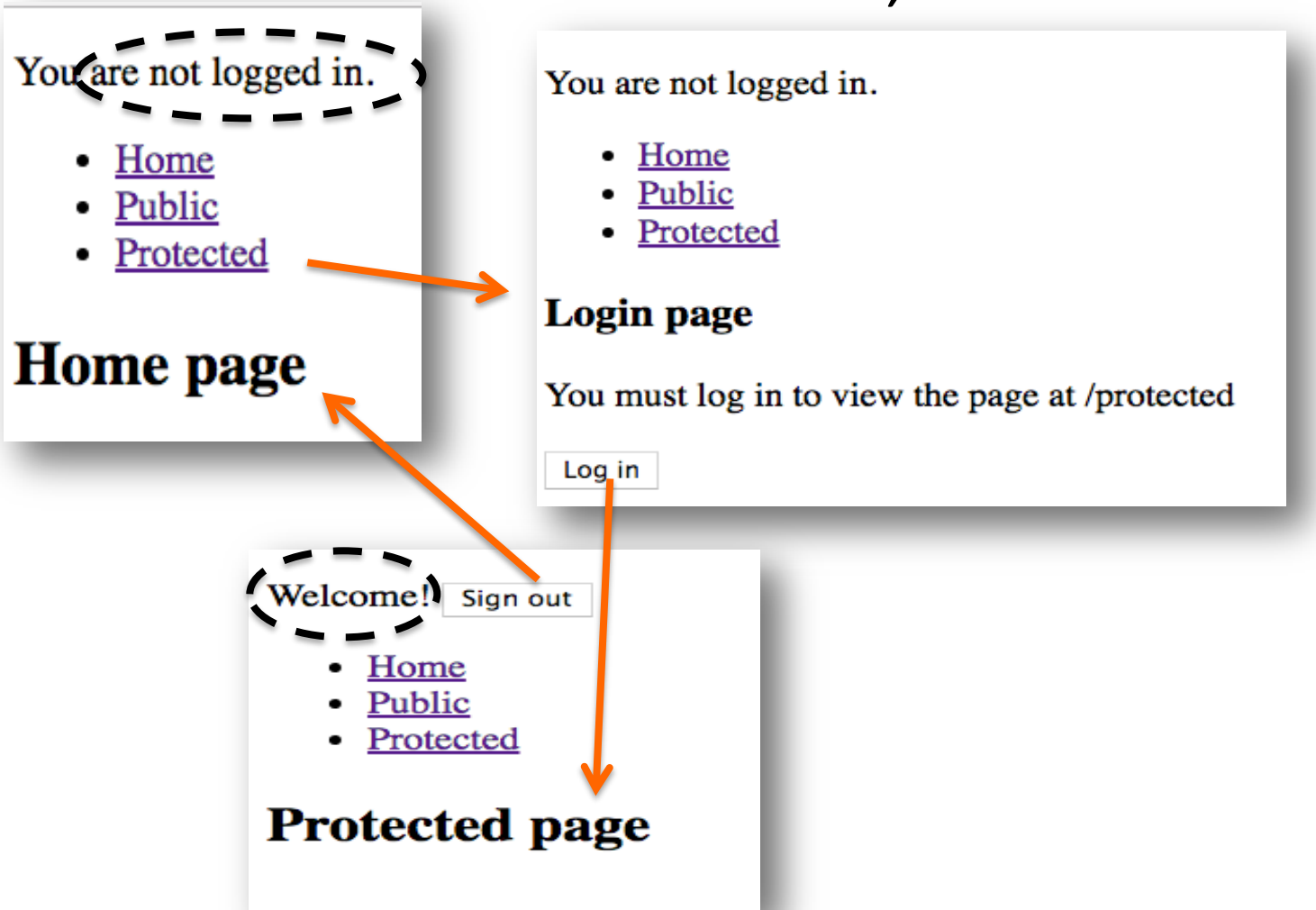
Protected Routes

- Objective:

```
<BrowserRouter>
  <Switch>
    <Route path=”/catalogue" component={Catalogue} />
    <PrivateRoute path="/accounts" component={Accounts} />
    <Route path=”/offers" component={Sales} /> {/* New route */}
    <PrivateRoute path=”/admin" component={Admin} />
    <Route exact path="/" component={App} />
    <Redirect from="*" to="/" />
  </Switch>
</BrowserRouter>
```

Protected Routes

- Not native to React Router; A custom solution.



Protected Routes

- **See archive for source code.**
- **Solution implemented incrementally:**
  - version1 – Skeleton.
  - version2 - basic <PrivateRoute> component, Redirects to skeleton Login page
  - version3 - Login supported but redirects to / (root) always; Protected page now accessible to authenticated user.
  - version4 - Login now redirects to selected protected page.
  - version5 – Signout supported.

- **To step through sample versions:**

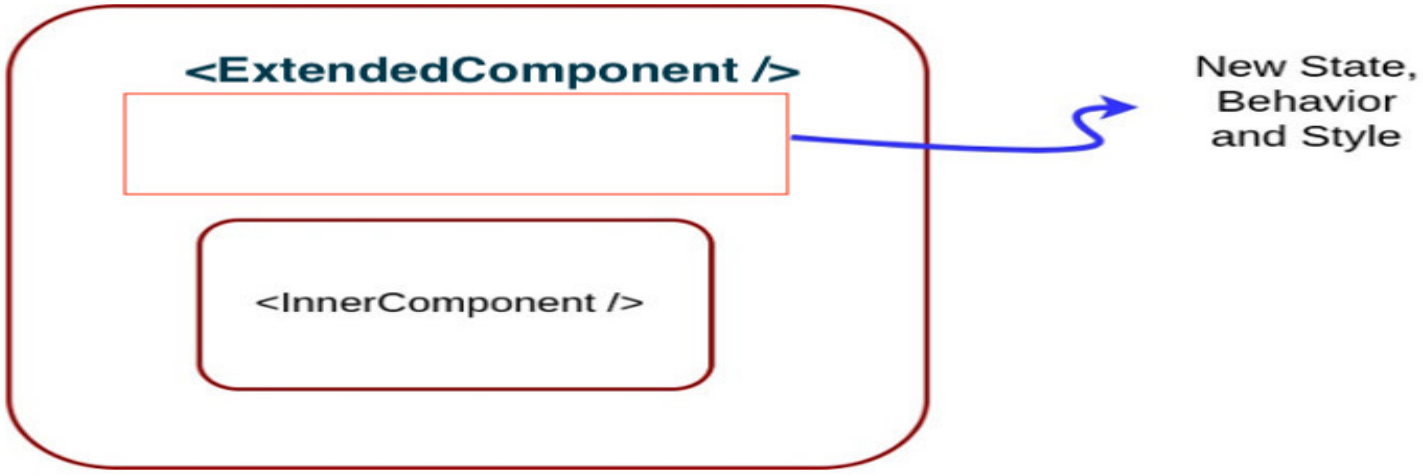
  $ git checkout versionX e.g. git checkout version3
- **To return to normal:**

  $ git checkout master

Higher Order Components.

Higher Order Components (HOC)

- **A higher-order component is a** function **that accepts a component as an argument and returns an** extended **version of that component**
- **Extended Component is like a** container**. It renders the Input Component, but because we're returning a new component, it adds an extra layer of abstraction.**



HOCs

- **A pattern; not part of React.**
- **Why HOC? The DRY principle. Code reuse.**
- **Many React third party libraries use the pattern:**
  - **react-router, e.g. withRouter(…..)**
  - **react-redux**
  - **react-google-maps**
- **Conventions:**
  - **The** `with` **prefix for the function name.**
  - **‘**WrappedComponent**’ for the input component.**

HOC – simple example

- **Assume this CSS -** .bgStyle { backgroundColor: 'grey'; }

```
const withGreyBg = (WrappedComponent) => {
    return class NewComponent extends Component {
      render() {
        return (
          <div className="bgStyle">
            <WrappedComponent {...this.props} />
          </div>
        );
      }
    };
```

HOC – simple example

- **Assume custom components: SmallBox (origin prop) and BigBox (border prop)**

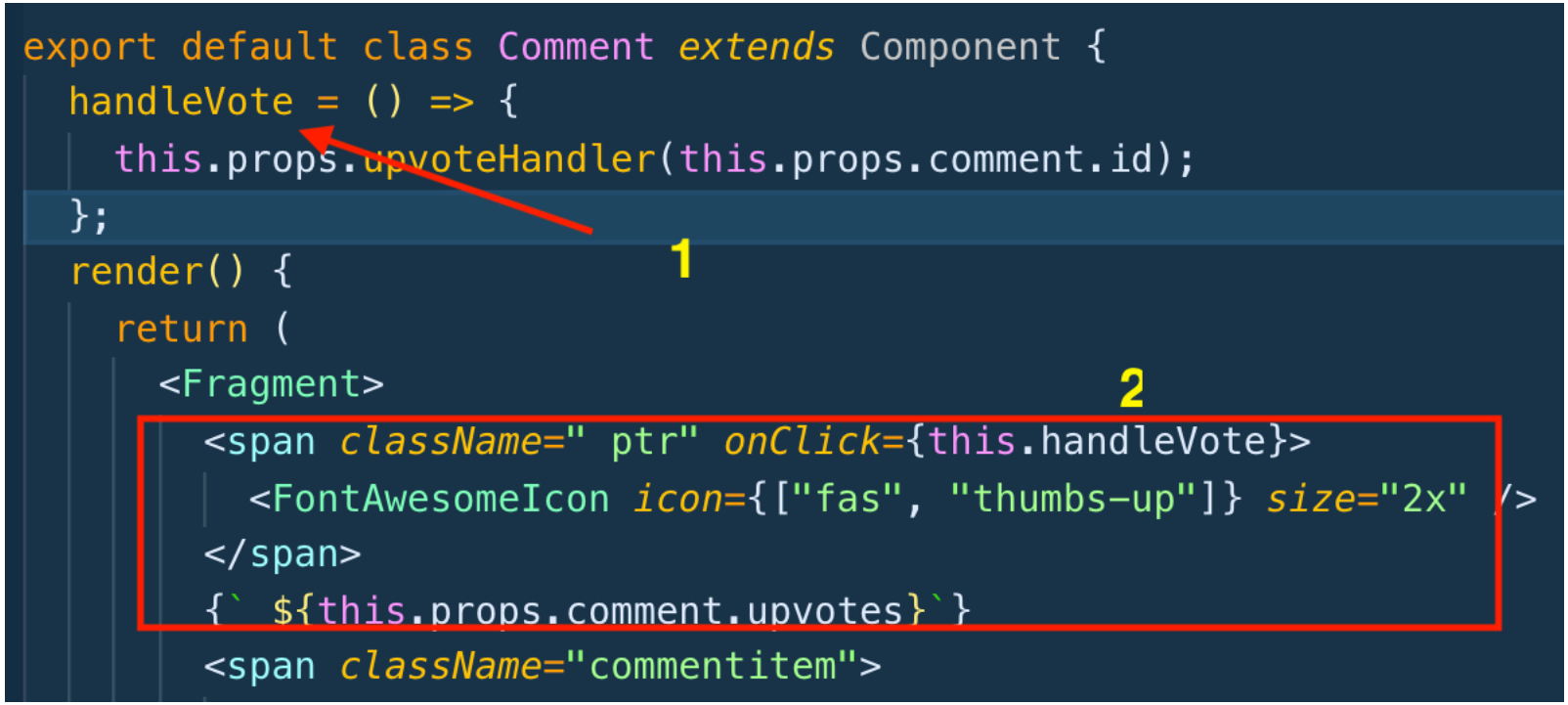
```
const SmallBoxWithGreyBg = withGreyBg(SmallBox);
const BigBoxWithGreyBg = withGreyBg(BigBox);

class BoxesApp extends Component {
    render() {
        <Fragment>
            <SmallBoxWithGreyBg origin={[12,32]} />
            <SmallBoxWithGreyBg origin={[50,32]} />
            <BigBoxWithGreyBg border={‘solid’} />
        <Fragment>
    }
}
```

HOC example - Hacker News App

- **EX.: Two custom components share a common feature: upvoting.**

HOC example - Hacker News App



- **Upvoting logic can be abstracted to a HOC.**

HOC example - Hacker News App

```
const withUpvoting = (WrappedComponent) =>
  class VotableComponent extends Component {
    handleVote = () =>  this.props.upvoteHandler(this.props.source.id);

    render() {
      return (
          <Fragment>
              <span className="ptr"  onClick={this.handleVote}>
                  <FontAwesomeIcon icon={["fas", "thumbs-up"]} size="2x" />
              {` ${this.props.source.upvotes}`}
              </span>
              <WrappedComponent {...this.props} />
          </Fragment>
      );
  }
}
```

Abstract name

Pass all props down to subordinate

(BTY: Curly braces **and** return **keyword are optional for arrow functions with a** single statement **body.)**

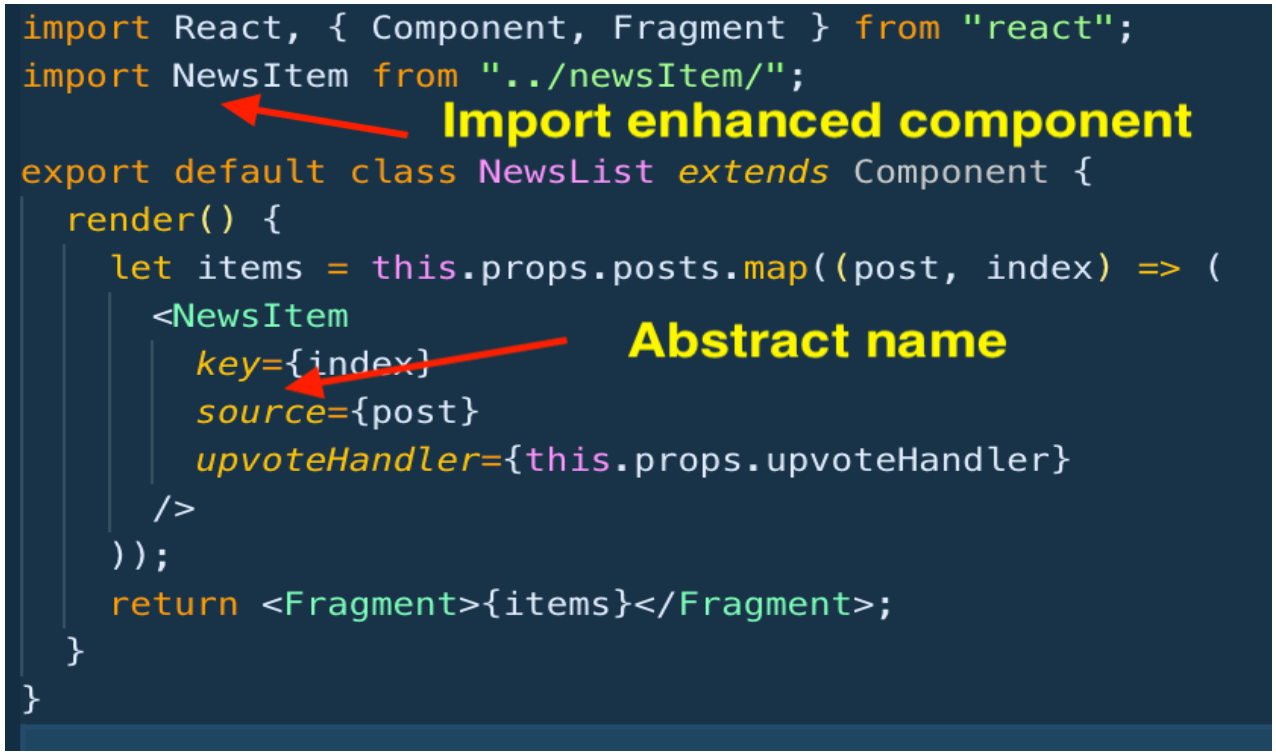
HOC example - Hacker News App

- **Remove code elevated to HOC from the 'wrapped' components.**

- **(Ditto for Comment component)**

HOC example - Hacker News App

- **Use enhanced component as normal in the app,**




The End
Metro-Goldwyn-Mayer
PICTURE