

Connecting React to Redux



Cory House

@housecor

reactjsconsulting.com



Here's The Plan



Container vs Presentation Components

React-Redux

- Provider
- Connect

A Chat with Redux



Two Component Types

Container

Focus on how things work

Aware of Redux

Subscribe to Redux State

Dispatch Redux actions

Generated by react-redux

Presentational

Focus on how things look

Unaware of Redux

Read data from props

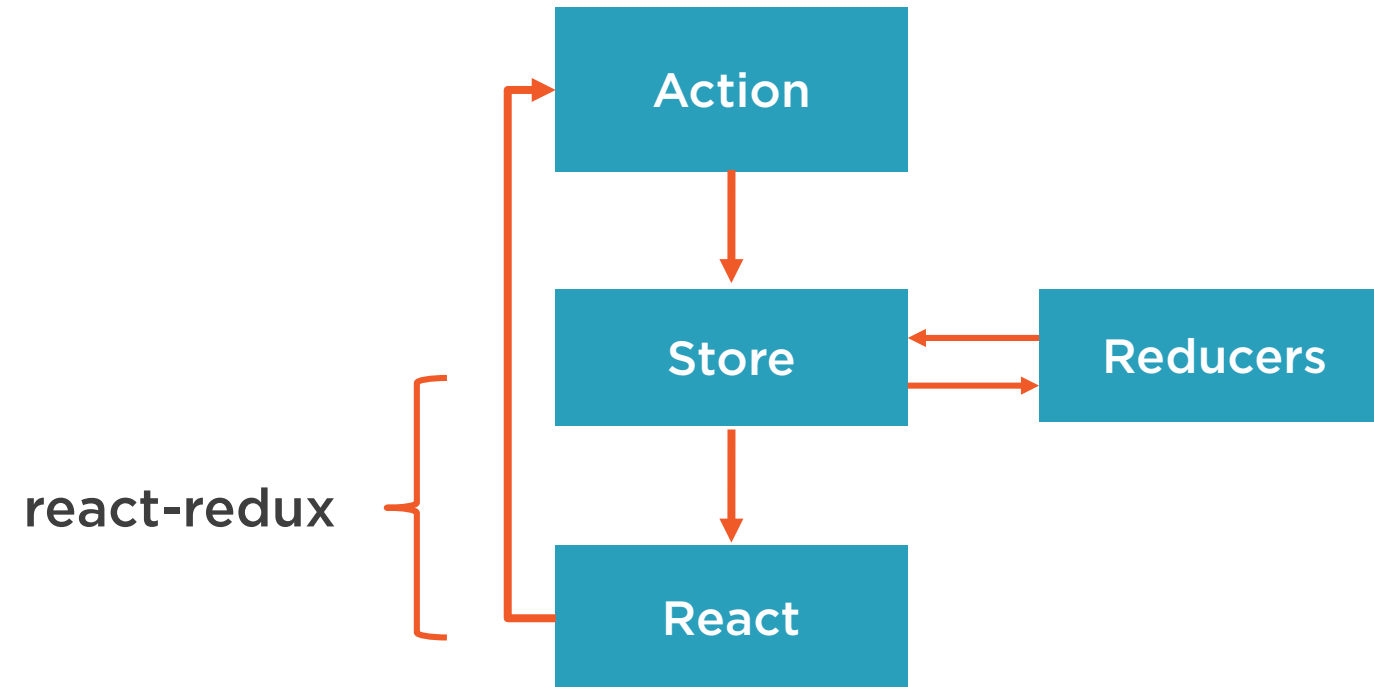
Invoke callbacks on props

Written by hand



Connecting React to Redux





React-Redux

Provider

Attaches app to store

Connect

Creates container components



React-Redux Provider

```
<Provider store={this.props.store}>  
  <App/>  
</Provider>
```



Connect

- Wraps our component so it's connected to the Redux store.

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



Flux

```
componentWillMount() {  
  AuthorStore.addChangeListener(this.onChange);  
}  
  
componentWillUnmount() {  
  AuthorStore.removeChangeListener(this.onChange);  
}  
  
onChange() {  
  this.setState({ authors: AuthorStore.getAll() });  
}
```



Redux

```
function mapStateToProps(state, ownProps) {  
  return {appState: state.authorReducer };  
}
```

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

Benefits:

1. No manual unsubscribe
2. No lifecycle methods required
3. Declare what subset of state you want
4. Enhanced performance for free



React-Redux Connect

`connect(mapStateToProps, mapDispatchToProps)`



What state should I expose as props?

```
function mapStateToProps(state) {  
  return {  
    appState: state  
  };  
}
```



Reselect

- Memoize for performance



React-Redux Connect

```
connect(mapStateToProps, mapDispatchToProps)
```

↑
What actions do I want on props?

```
function mapDispatchToProps(dispatch) {  
  return {  
    actions: bindActionCreators(actions, dispatch)  
  };  
}
```



3 Ways to Handle mapDispatchToProps

```
this.props.dispatch(loadCourses());
```

Ignore it. Use dispatch.

```
function mapDispatchToProps(dispatch) {  
  return {  
    loadCourses: () => {  
      dispatch(loadCourses());  
    }  
  };  
}
```

Manually wrap

```
function mapDispatchToProps(dispatch) {  
  return {  
    actions:  
      bindActionCreators(actions, dispatch)  
  };  
}
```

Use bindActionCreators



Option 1: Use Dispatch Directly

```
// In component...  
this.props.dispatch(loadCourses())
```

Two downsides

1. **Boilerplate**
2. **Redux concerns in child components**



Option 2: Wrap Manually

```
function mapDispatchToProps(dispatch) {  
  return {  
    loadCourses: () => {  
      dispatch(loadCourses());  
    },  
    createCourse: (course) => {  
      dispatch(createCourse(course));  
    },  
    updateCourse: (course) => {  
      dispatch(updateCourse(course));  
    }  
  };  
}
```

```
// In component...  
this.props.loadCourses()
```



Option 3: bindActionCreators

```
function mapDispatchToProps(dispatch) {  
  return {  
    actions: bindActionCreators(actions, dispatch)  
  };  
}
```

Wraps action creators in dispatch call for you!



// In component:

```
this.props.actions.loadCourses();
```



A Chat With Redux

- React** Hey CourseAction, someone clicked this “Save Course” button.
- Action** Thanks React! I will dispatch an action so reducers that care can update state.
- Reducer** Ah, thanks action. I see you passed me the current state and the action to perform. I’ll make a new copy of the state and return it.
- Store** Thanks for updating the state reducer. I’ll make sure that all connected components are aware.
- React-Redux** Woah, thanks for the new data Mr. Store. I’ll now intelligently determine if I should tell React about this change so that it only has to bother with updating the UI when necessary.
- React** Ooo! Shiny new data has been passed down via props from the store! I’ll update the UI to reflect this!



Time to code!



Next up: Let's code Redux!

