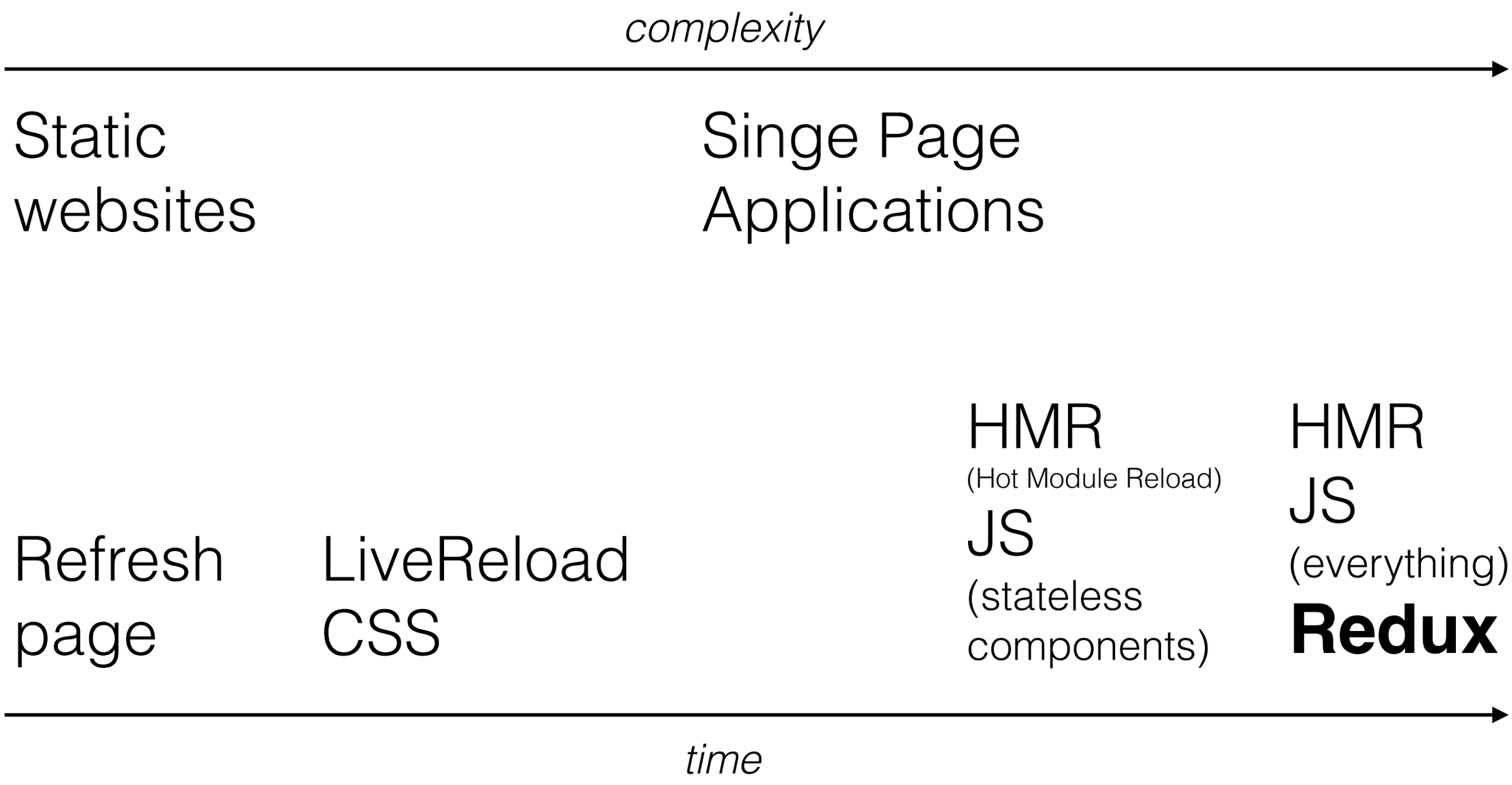
Using Redux

Maximising development efficiency

Why Redux is natural



Demo: Redux in use

criticalcss.com

About me

Jonas Ohlsson - front end developer

@pocketjoso pocketjoso

https://jonassebastianohlsson.com

Performance, tooling, React

Penthouse - critical css generator

This talk

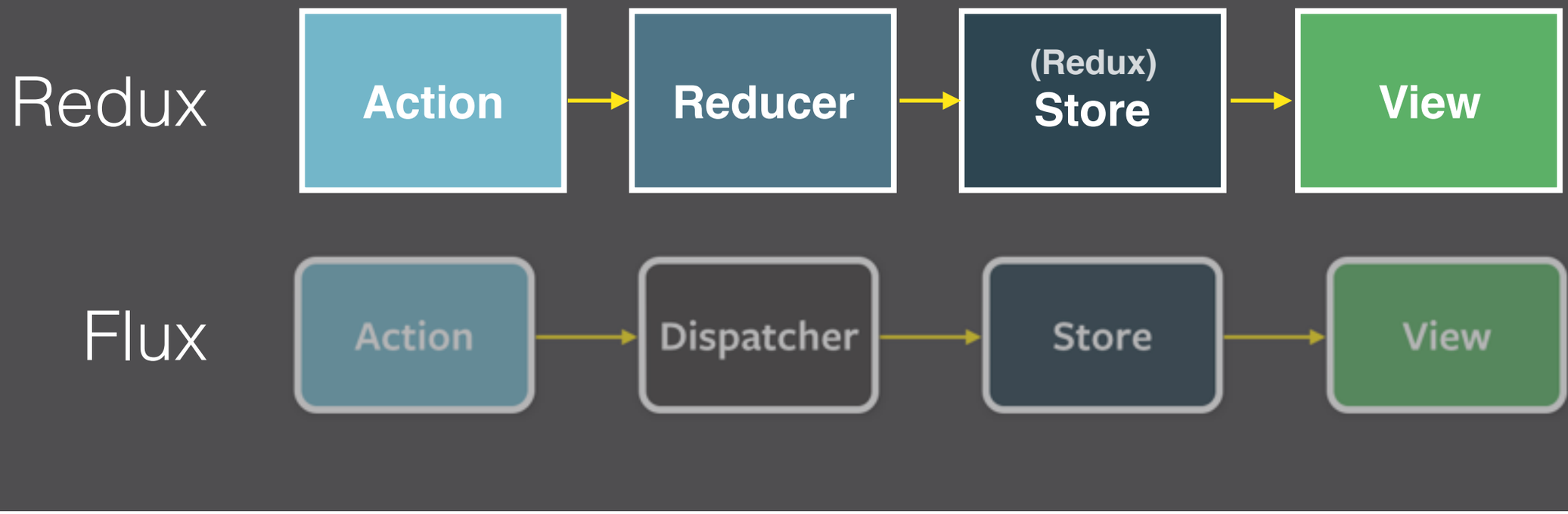
- Redux - overview

- criticalcss.com redux usage in practice

What is Redux

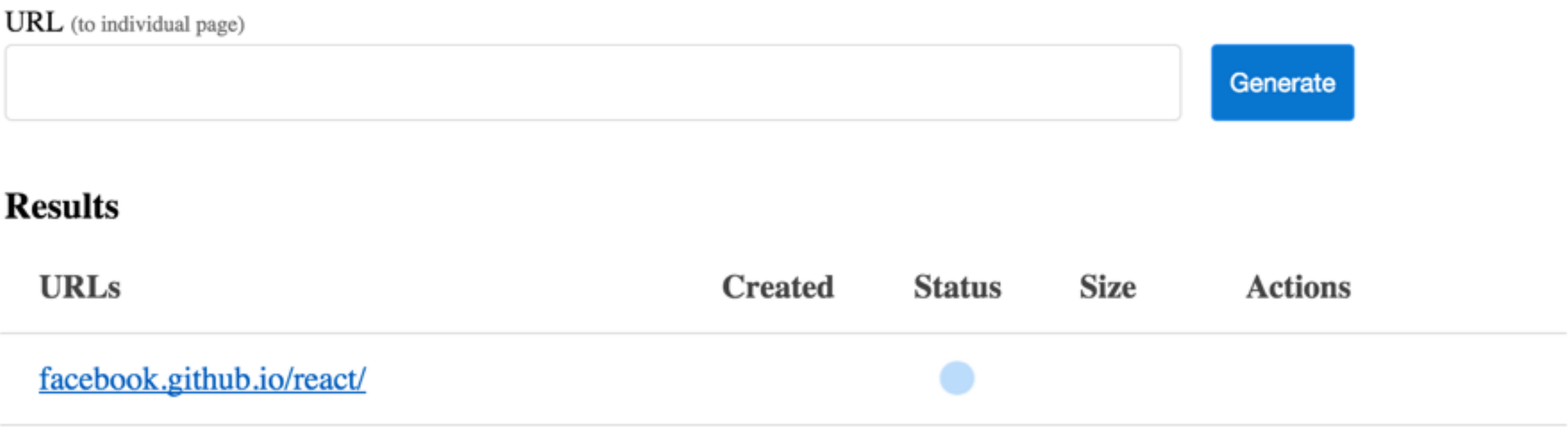
- Predictable State container (evolution of Flux)
- One single *immutable* State
- Reducers instead of Stores
- Hot Module Reloading and Time travel
- Modular, tiny (2kb!)
- Follows Best practice

Data flow

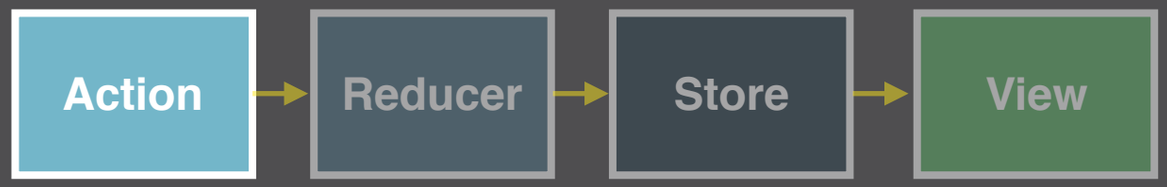


Redux vs traditional Flux

Today's example



- Dispatch action for new generate job
- Update State via reducer
- Receive State in Component



Action creator

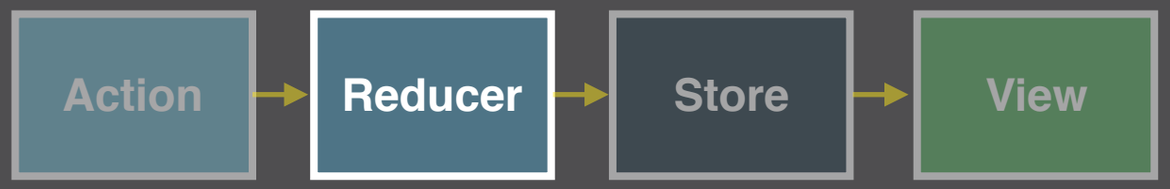
```
function requestGenerate (url, generateId) {
  return {
    type: REQUEST_GENERATE_CRITICAL_CSS,
    url,
    generateId
  }
}
```

- Returns actions; does **not** dispatch them

Reducers

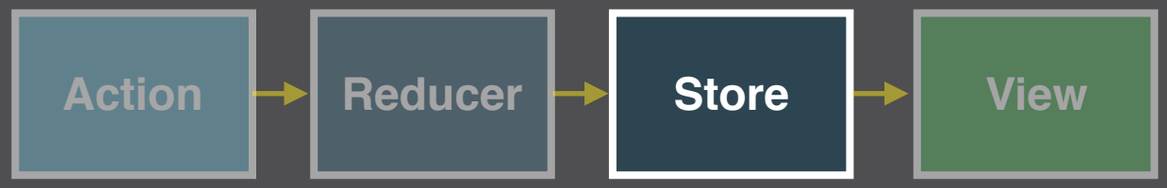
- Write reducers *instead* of Stores
- Pure **functions**
- (state, action) => newState

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce



Reducer

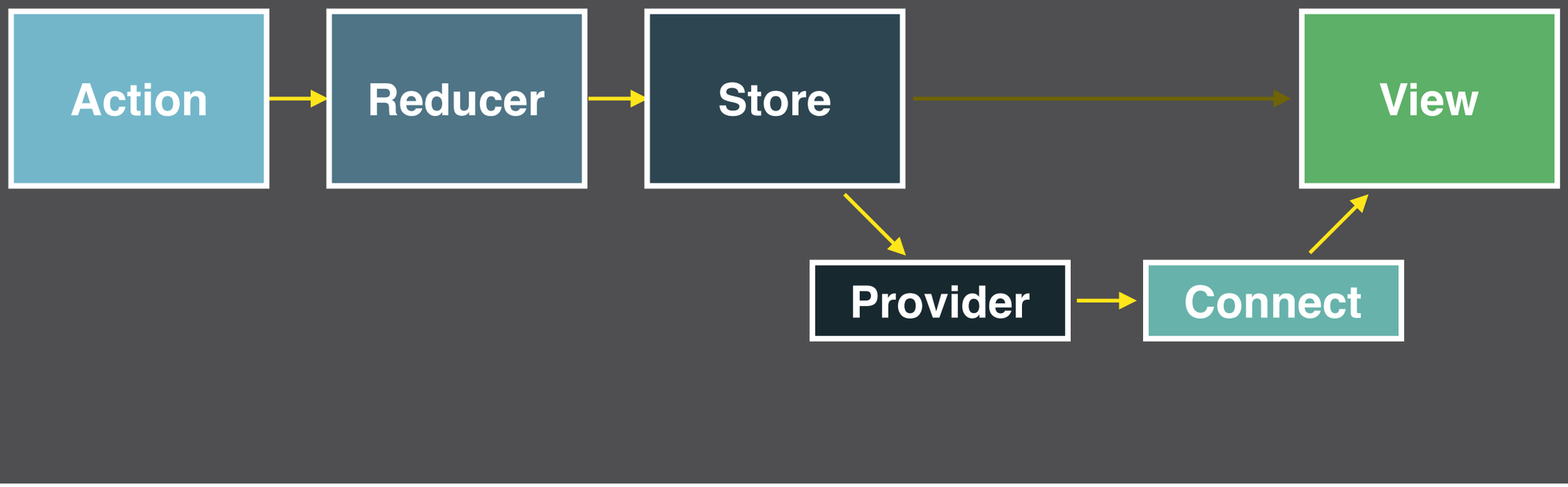
```
function resultsReducer (state = [], action) {
  if (action.type === REQUEST_GENERATE_CRITICAL_CSS) {
    return [{
      id: action.generateId,
      url: action.url,
      status: STATUS_JOB_ONGOING
    },
    ...state]
  }
  return state
}
```



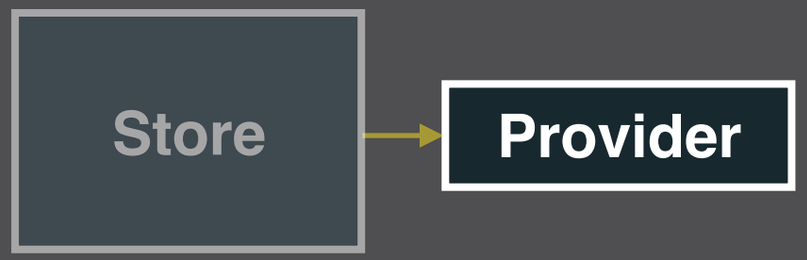
Redux Store

```
import { combineReducers, createStore } from 'redux'
import results from '../reducers/results'

// 1. create one rootReducer out of all your reducers
const rootReducer = combineReducers({ results, loggedInUser })
// 2. create the Redux Store
const store = createStore(rootReducer, initialState)

// 3. use Store in your Views
store.getState()
store.subscribe(() => {
  /* do something with store.getState() */
})
store.dispatch(requestGenerate())
```

Data flow with react-redux



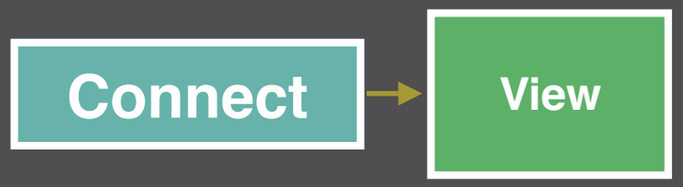
Redux using react-redux



(Store) Provider

```
import { Provider } from 'react-redux'
import App from 'app'
...
<Provider store={this.props.store}>
  <App/>
</Provider>
```

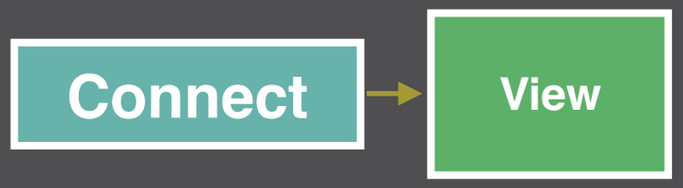
- Wraps whole App
- Makes Store available to components



Connect

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

const ResultsTable = class ResultsTable extends Component {
  render () {
    return <ul>{this.props.results.map(renderResult)}</ul>
  }
}

function mapStateToProps ({ results }) {
  return { results }
}

export default connect(mapStateToProps)(ResultsTable)
```

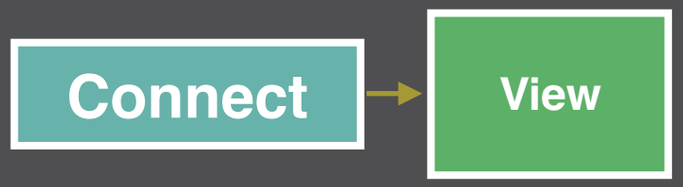
- Injects props, subscribes to Store
- Keeps components Dumb



Connect

```
import { requestGenerate } from '../actions/result'

const GenerateModule = class GenerateModule extends Component {
  render () {
    return <div>
      <label>URL <input type='url'/></label>
      <Button onClick={() => requestGenerate(...)}
        type='button'>Generate</Button>
    </div>
  }
}
```

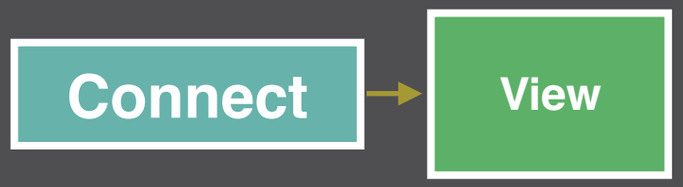
- **won’t** work - actionCreator does **not** dispatch action



Connect

```
const GenerateModule = class GenerateModule extends Component {
  render () {
    return <div>
      <label>URL <input type='url'/></label>
      <Button
        onClick={() => this.props.dispatch(requestGenerate(...))}
        type='button'>Generate</Button>
    </div>
  }
}

export default connect()(GenerateModule)
```

- Connect auto-injects dispatch to props



Connect

```
const GenerateModule = class GenerateModule extends Component {
  render () {
    return <div>
      <label>URL <input type='url'/></label>
      <Button
        onClick={() => this.props.requestGenerate(...)}
        type='button'>Generate</Button>
    </div>
  }
}

export default connect(null, { requestGenerate })(GenerateModule)
```

- 2nd param creates self-dispatching action functions

Summary

- *write* Reducers instead of Stores
- *keep* Components dumb using Connect
- *benefit* from:
  - Hot Module Reloading
  - Powerful DevTools
  - Components easier to understand and test

That’s all!

Resources

Recommended packages

- Redux - https://github.com/rackt/redux
- React transform boilerplate (for Hot Module Reloading) - https://github.com/gaearon/react-transform-boilerplate
- Redux DevTools - https://github.com/gaearon/redux-devtools

Redux resources

- Redux release and intro to Time travel - https://www.youtube.com/watch?v=xsSnOQynTHs
- Best starting point for understanding redux - Free video tutorial on Redux by Dan Abramov (creator) - https://egghead.io/lessons/javascript-redux-the-single-immutable-state-tree
- Written notes for above video tutorial - https://gist.github.com/diegoconcha/8918294bb9df69876b22
- Official docs (great) - http://rackt.org/redux/index.html
- Redux slim in a gist - https://gist.github.com/gaearon/ffd88b0e4f00b22c3159

Resources

Redux related resources

- Full stack Redux tutorial - http://teropa.info/blog/2015/09/10/full-stack-redux-tutorial.html
- Redux for state management - http://konkle.us/state-management-with-redux/
- Full stack Redux boilerplate - https://github.com/erikras/react-redux-universal-hot-example
- Support for Functional Components in React Transform -https://github.com/gaearon/babel-plugin-react-transform/issues/57
- Beyond just Redux for async actions - riadbenguella.com/from-actions-creators-to-sagas-redux-upgraded

Other related resources

- Webpack (react docs) - https://christianalfoni.github.io/react-webpack-cookbook/index.html
- Functional Stateless Components - http://tylermcginnis.com/functional-components-vs-stateless-functional-components-vs-stateless-components/
- https://medium.com/@esamatti/react-js-pure-render-performance-anti-pattern-fb88c101332f