Redux

```
(previousState, action) => newState
```

Who Am I?

Brian Egizi

Developer at MojoTech

React

Virtual DOM

Why is the DOM hard?

- Slow
- Difficult to maintain state
- Difficult to test

Virtual DOM

- Decoupled
- Optimized
- In-Memory
- Consistent API

JSX

JSX

- Virtual DOM API Preprocessor
- HTML like syntax for building components
- Keeps developers sane by giving them a familiar view paradigm

JSX vs HTML

```
<div className=”my-klass”>
  <h1>Hello, world</h1>
</div>
```

```
<div class=”my-klass”>
  <h1>Hello, world</h1>
</div>
```

Components

Components are awesome because...

- Reusable
- Composable
- Testable
- Easily maintained

<HelloWorld />

```
class HelloWorld extends React.Component {
  render() {
    return (
      <h1>Hello, world!</h1>
    );
  }
}
```

One Way Data Flow

Data passed from parent as props

```
<Counter count={0} />
```

Using a prop

```
class Counter extends React.Component {
  render() {
    return (
      <div>
        Counter Value {this.props.count}  
        <button>+</button>
        <button>-</button>
      </div>
    );
  }
}
```

One Way Data Flow (cont.)

Events bubble up to parents via callbacks

```
<Counter count={0} onIncrement={this.handleIncrement} />
```

Using a callback

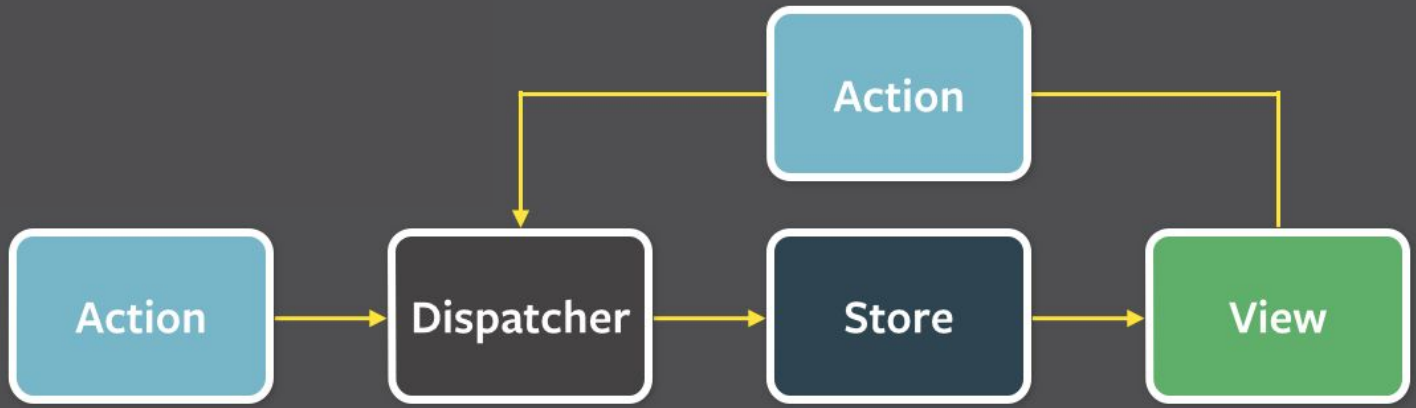
```
class Counter extends React.Component {
  render() {
    return (
      <div>
        Counter Value {this.props.count}  
        <button onClick={this.props.onIncrement}>+</button>
      </div>
    );
  }
}
```

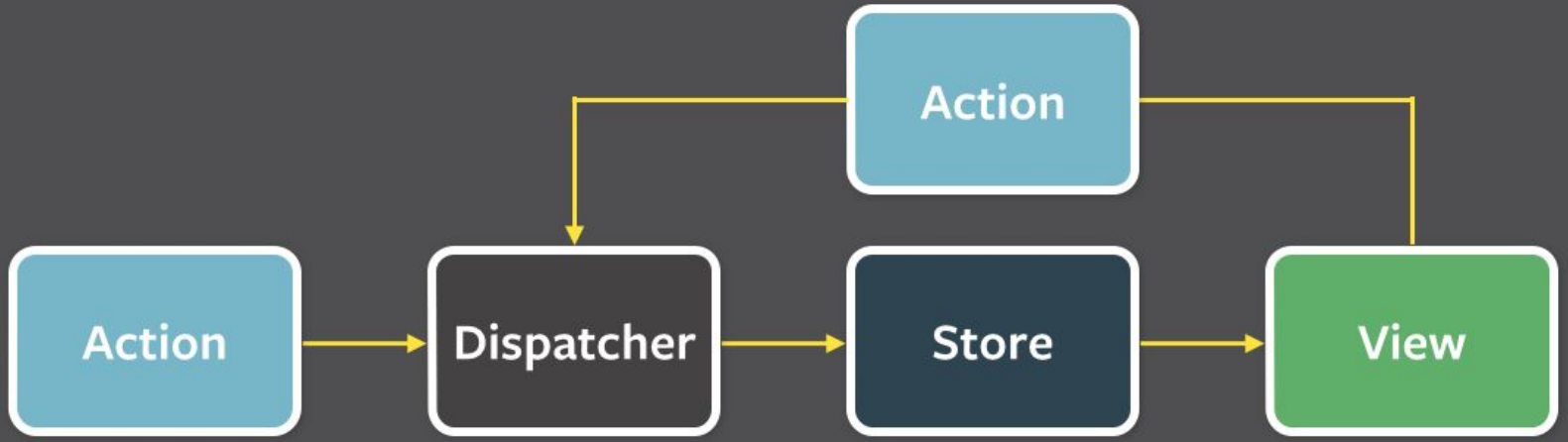
Redux

Redux Principles

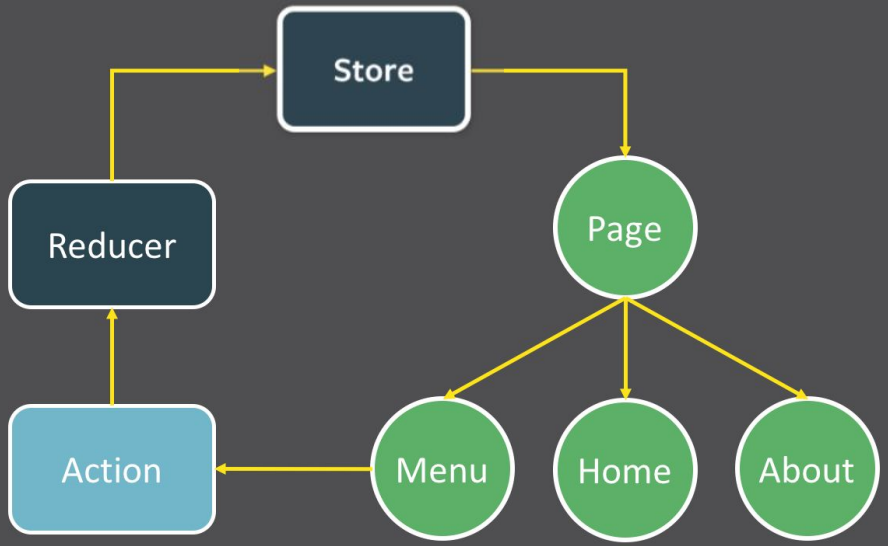
- The state is the single source of truth
- The state is read only
- State changes are made with pure functions

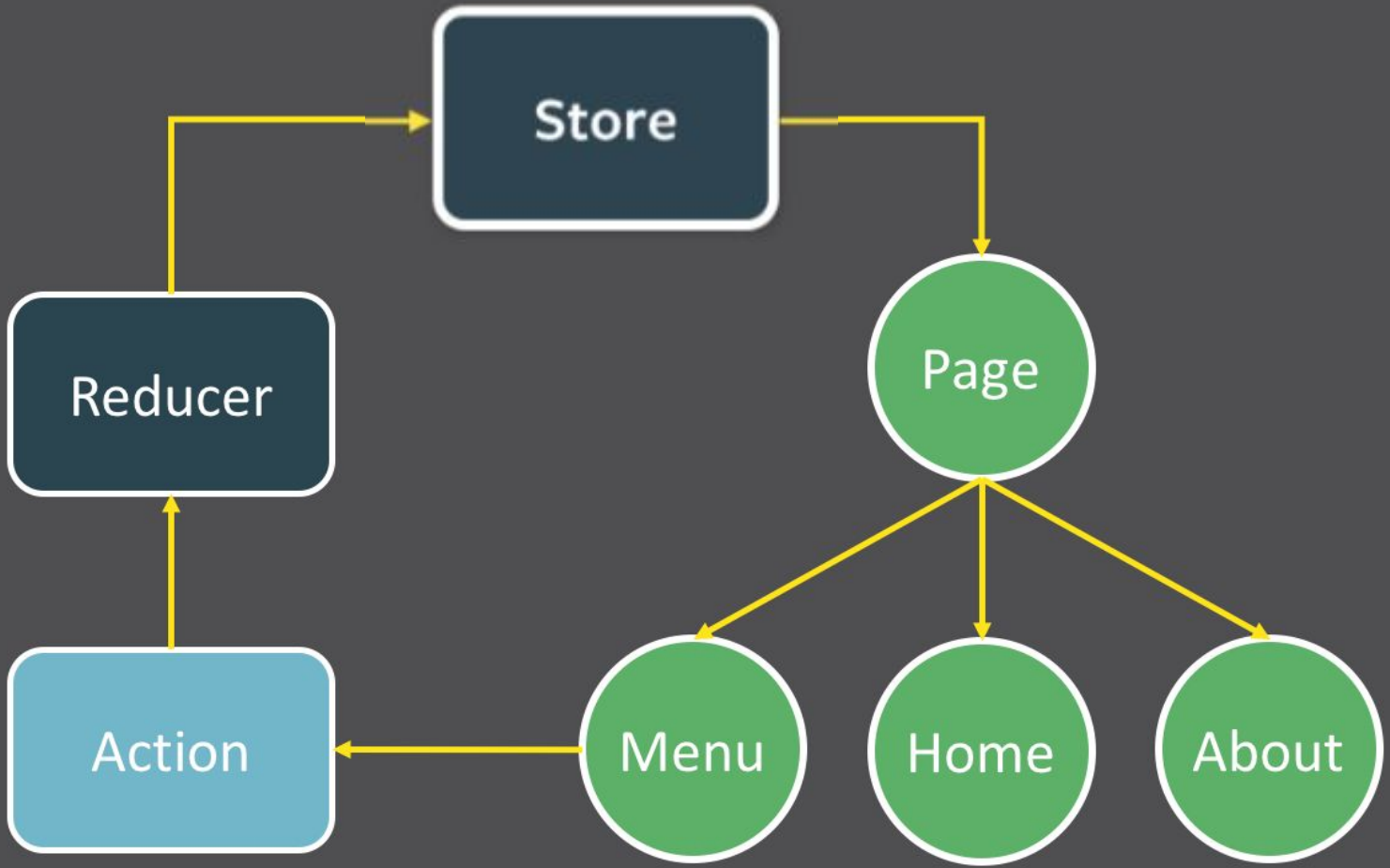
Flux




Action
Action
Dispatcher
Store
View

Redux




Store
Reducer
Page
Action
Menu
Home
About

Store

```
connect(mapStateToProps)(Component)
```

Benefits of a single application store

- Single Source of Truth
- State can easily be reverted to a previous or predetermined state and the full application reflects the change (Read as “Dead simple UNDO/REDO”)
- Testing

Actions

```
{
  type: “LOAD_NEXT_SLIDE”,
  transition: “move_up”
}
```

Anatomy of an Action

```
{ // Actions are just objects

  type: “SOME_ACTION_TYPE”, // Some indicator of type

  payload: 3 // Anything else to include with the action

}
```

Reducers

```
(previousState, action) => newState
```

Anatomy of a Reducer

```
function applicationReducer(state, action) {
  switch (action.type) {
    case “SOME_ACTION_TYPE”:
      return { myNewNumber: action.payload };
    default:
      return state;
  }
}
```

Putting it all together (demo)

Helpful Tools and Plugins

- Redux Dev Tools https://github.com/gaearon/redux-devtools
- DevTools Chrome Extension https://github.com/zalmoxisus/redux-devtools-extension
- redux-thunk https://github.com/gaearon/redux-thunk
- redux-api-middleware https://github.com/agraboso/redux-api-middleware

Questions?

Thanks!

Demos available at github.com/begizi/react-symposium