DrupalCon
NASHVILLE 2018
APRIL 9-13

Introduction to React Design Patterns

Brandon Williams
Amazee Labs

Before We Begin

React

- Declarative
- Flexible
- ***Library***

Design Patterns

Design patterns are **formalized** best practices that the programmer can use to solve common problems when designing an application or system.

But these are just my **opinions**.

Brandon Williams
Lead Developer

Amazee Labs

- Drupal 8 sites since Alpha
- React projects since 2016
- SPAs, Interactive Touch Displays, Embeddable Widgets, Improved Filtered Search
- GraphQL Module

Design Patterns

Presentational/Container Components

Presentational

- How things look
- Little to no state
- Receive all data via props
- Functional component
- Same use case as Twig templates

Container

- How things work
- Render presentational and container components
- Provide data and behavior
- Stateful

Presentational/Container Components

Presentational

```
const Commentlist = comments => (
  <ul>
    {comments.map(
      ({ body, author }) =>
      <li>{body}-{author}</li>
    )}
  </ul>
)
```

Container

```
class CommentListContainer extends
React.Component {
  componentDidMount() {
    loadComments("/my-comments.json")
  }

  render() {
    return <CommentList
      comments={this.state.comments}
    />;
  }
}
```

Presentational/Container Components

Why?

- Separation of concerns
- Reusable
- Testable
- Style guides

Difficulty

★

Stateless Components + Functional Programming

- Pure functions
- No side-effects or shared state
- Declarative
- Composition over inheritance
- Immutable state

```
const SubmitButton = ({
  label,
  onSubmit,
  disabled,
}) => (
  <input
    type="submit"
    disabled={disabled}
    value={label}
    onClick={onSubmit}
  />
);
```

Stateless Components + Functional Programming

Why?

- Declarative programming for declarative library
- Immutability helps avoid render errors
- Easier to reason about component dependencies
- Fun vocabulary: map, reduce, functor, monad, currying

Difficulty

★★★

Higher Order Components/Functions

- A function that returns a component/function
- Wraps functional components
- Recompose library for React integration
- Same use case as PHP Traits or OO Decorator pattern

```
const withHideLoading = (BaseComponent) =>
  (props) => (
    {props.loading &&
      <BaseComponent />}
  )

const ButtonWithLoading =
withHideLoading(<SubmitButton
  label="Submit"
/>)

<ButtonWithLoading loading={true} />
```

Higher Order Components/Functions

```
const withDisabledState = withState(
  'disabled',
  'setDisabled',
  false);

const withDisabledOnSubmit =
withHandlers({
  onSubmit: ({ setDisabled }) =>
    () => setDisabled(true),
})
```

```
const SubmitButton = ({
  label,
  onSubmit,
  disabled,
}) => (<input
    type="submit"
    disabled={disabled}
    value={label}
    onClick={onSubmit}
  />);


const ButtonWithDisable = compose(
  withDisabledState,
  withDisabledOnSubmit,
)(<SubmitButton label="submit" />)
```

Higher Order Components/Functions

Why?

- Keep code DRY
- Keep functional components pure
- Separation of concerns

Difficulty

★★

Render Props/Function as Children

Alternative to HoC

```
const HideLoading = (props) => {
  if (props.loading) {
   return null;
  }

  return this.props.render();
}

<HideLoading
  loading{true}
  render={() => (
   <SubmitButton label="submit" />
  )}
/>
```

Difficulty

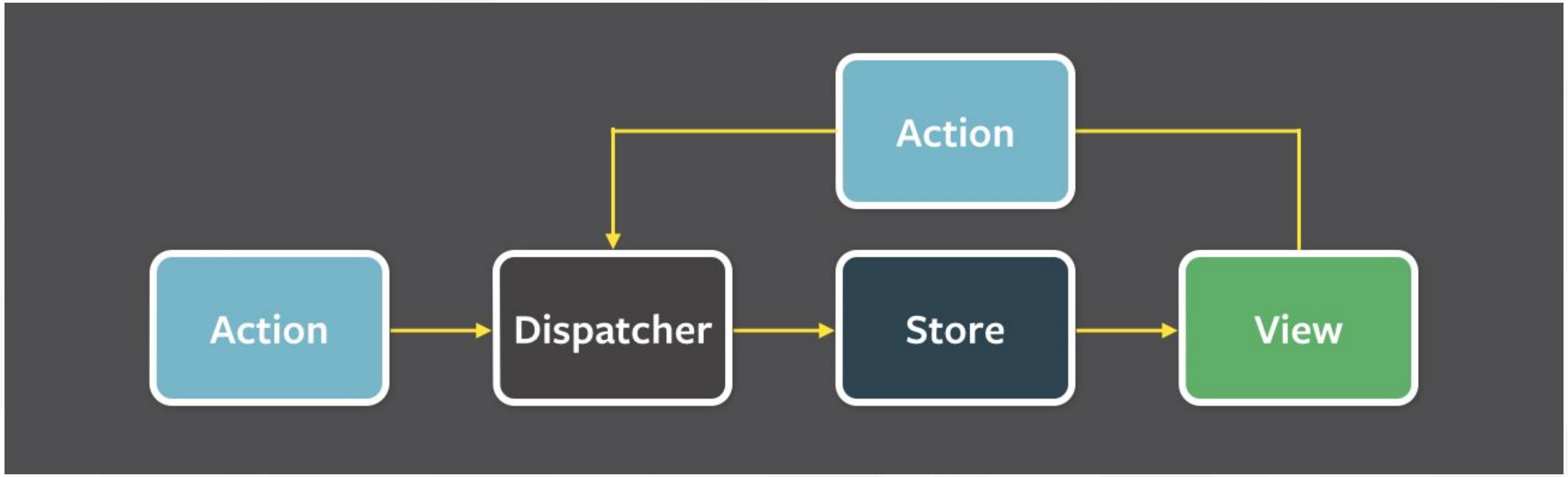
★

Flux

- Pattern for managing data
- All state flows in one direction
- All state managed in the store
- State changes cause re-renders
- Redux

Difficulty

★★



TEXAS CAMP

ATX
MAY 31
JUNE 2
2018

CFP Open! 2018.texascamp.org

DrupalCon
NASHVILLE 2018
APRIL 9-13

Join us for contribution sprints

Friday, April 13, 2018

Mentored Core sprint

9:00-18:00
Room: 103

First time sprinter workshop

9:00-12:00
Room: 101

General sprint

9:00-18:00
Room: 104

#drupalsprint

Questions?
Brandon Williams

@rocketeerbkw