

# REACT AND FRIENDS

---

- James Kent
- Matt  
Gilbride

# AGENDA

---

- What is React?
- Intro to Create React App
- What React isn't
- Common companion libraries
- What makes React a good choice for your team

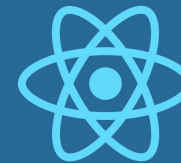
# WHAT IS REACT?

---

- A view library for building component-based UIs
- Created by Facebook in 2013
- Includes a declarative DSL for describing components called JSX
  - "html in your javascript"
- Components are a function of two parameters
  - "props" = read-only
  - "state" = read/write
- Components have a specific lifecycle that can be used to influence behavior
  - <http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

# CREATE REACT APP

---



- <https://github.com/facebook/create-react-app>

# WHAT REACT ISN'T

---

- A "framework"
- The following is up to you:
  - State Management
  - Routing
  - Forms
  - Styling
  - Testing

# OUR STACK

---

# STATE MANAGEMENT

---



Flux



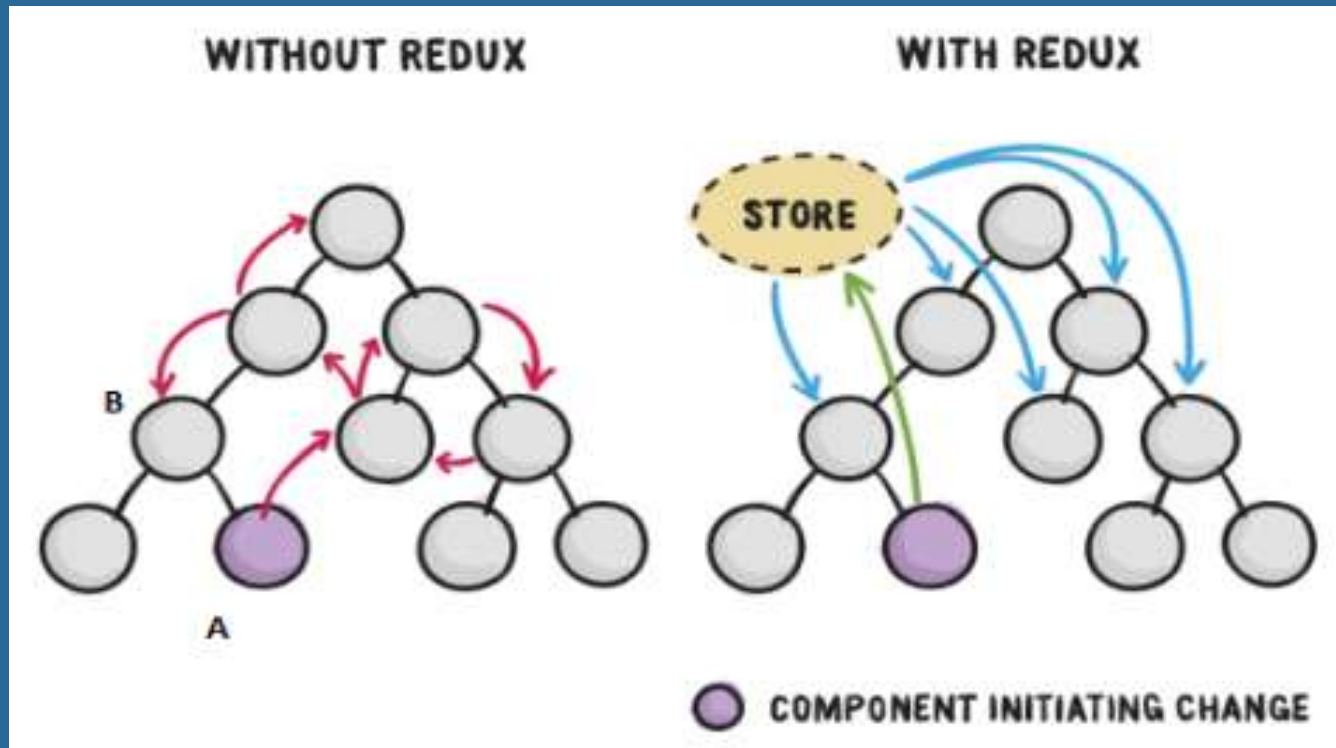
Redux



Mobx

- It is totally possible to use React without any state management library!

# REDUX - DATA FLOW



Source: <https://www.slideshare.net/binhqdgmail/006-react-redux-framework>



# REDUX - VOCABULARY

---

- STORE - a single, global data repository and an API to change said data
- STATE - the data
  - Important: this is NOT the same as the "state" of a component : - (
- ACTION - an event
  - Usually, causes some update to STATE
- REDUCER - function which responds to ACTION(s) and decides how to update STATE
- DISPATCH - function that passes actions to reducers
  - you "dispatch" an "action"

# REACT-REDUX

---

- (yet another) a companion library
- provides a nice API for using redux with your React components
- subscribe to pieces of the data, receiving updates when they change
- make changes to the data from your components

# REACT-REDUX IN PRACTICE

```

9  interface StateProps {
10     inFlight: boolean;
11     schedule?: Session[];
12 }
13
14 interface DispatchProps {
15     scheduleRequest: () => void;
16 }
17
18 type ScheduleProps = StateProps & DispatchProps;
19
20 class ScheduleComponent extends Component<ScheduleProps> {
21
22     public componentDidMount() {
23         this.props.scheduleRequest();
24     }
25
26     public render() {
27         return (
28             <Fragment>
29                 <h3>Choose an appointment time:</h3>
30
31                 <div className={styles.schedule}>
32                     {(this.props.inFlight || !this.props.schedule)
33                     ? <SpinnerComponent />
34                     : <Fragment>
35                         {this.props.schedule.map((session: Session) => (
36                             <SessionLink
37                                 key={session.id}
38                                 {...session}
39                             )>
40                         )}}
41                     </Fragment>
42                 </div>
43             </Fragment>
44         );
45     }
46 }
47
48
49 const mapStateToProps: MapStateToProps<StateProps, ScheduleProps, { spa: SpaState }> = state => ({
50     inFlight: state.spa.inFlight,
51     schedule: state.spa.schedule,
52 });
53
54 const mapDispatchToProps: MapDispatchToProps<DispatchProps, ScheduleProps> = dispatch => ({
55     scheduleRequest: () => dispatch(ActionCreators.scheduleRequest()),
56 });
57
58 export const ScheduleContainer = connect(mapStateToProps, mapDispatchToProps)(ScheduleComponent);
59

```

# REACT-REDUX IN PRACTICE

---

- tooling matters

# ASYNC ACTIONS



redux-thunk



redux-saga

redux-observable

```
function makeASandwichWithSecretSauce(forPerson) {  
  return function (dispatch) {  
    return fetchSecretSauce().then(  
      sauce => dispatch(makeASandwich(forPerson, sauce)),  
      error => dispatch(apologize('The Sandwich Shop', forPerson, error))  
    );  
  };  
};
```

- a.k.a. but I need to talk to a server!

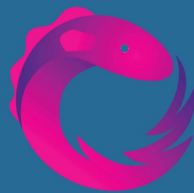
# REDUX-OBSERVABLE

---

- turns redux actions into observables
- "Epics": actions in, actions out


# RXJS

---




- "Observable" - think of it as a stream
- resources
  - <http://rxmarbles.com/>
  - <http://reactivex.io/documentation/operators/merge.html>
  - <http://reactivex.io/documentation/operators/zip.html>
  - Video: Marble diagrams in Rx

# REDUX-OBSERVABLE IN PRACTICE



Chariot SPA Day



CHARIOT  
SOLUTIONS

### Welcome to the SPA

Just float and wait for the wind to blow you around. Here's another **little happy bush**. Let all these little things happen. Don't fight them. Learn to use them. No worries. No cares.

How do you make a round circle with a square knife? That's your challenge for the day. Anytime you learn something your time and energy are not wasted. **Let's get crazy**. At home you have unlimited time. We need a shadow side and a highlight side. Nothing's gonna make your husband or wife madder than coming home and having a snow-covered dinner.

This is your world, **whatever makes you happy** you can put in it. Go crazy. Everybody needs a friend. It's important to me that you're happy. Isn't that fantastic? You can just push a little tree out of your brush like that. They say everything looks better with odd numbers of things. But sometimes I put even numbers —just to upset the critics. Maybe we got a few little happy bushes here, just covered with snow.

Didn't you know you had that much power? **You can move mountains**. You can do anything. Don't kill all your dark areas - you need them to show the light. It just happens - whether or not you worried about it or tried to plan it.


### Success! Your registration is complete

Appointment: Nov. 28th - 10 am - 11 am  
Name: **Matthew T Gilbride**  
Email: **mtg5014@gmail.com**  
Treatment: **Chemical peel**

#### Complete Registrant List

Name	Email	Treatment
Matthew T Gilbride	mtg5014@gmail.com	Exfoliation
Matthew T Gilbride	mtg5014@gmail.com	Exfoliation
Matthew T Gilbride	mtg5014@gmail.com	Exfoliation
Matthew T Gilbride	mtg5014@gmail.com	Chemical peel
Matthew T Gilbride	mtg5014@gmail.com	Chemical peel

[Back to Schedule](#)



Inspector

React App

filter... Commit

@@@INIT 7:51:34.65

spa/register/REQUEST +00:12.42

spa/register/SUCCESS +00:00.01

spa/schedule/REQUEST +00:00.01

@@router/LOCATION\_CHAN... +00:00.01

spa/schedule/SUCCESS +00:01.04

Diff

Action State Diff Test

Tree Raw

spa (pin)

inflight (pin): ~~true~~ => false

schedule (pin): { 0: {...} }

▶

1x

{ type: '' }

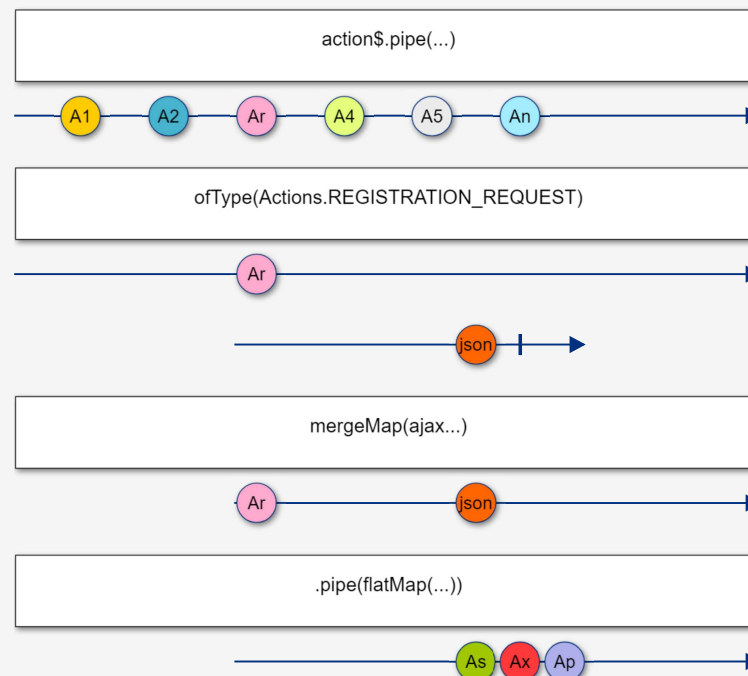
Custom action Dispatch

Pause Lock



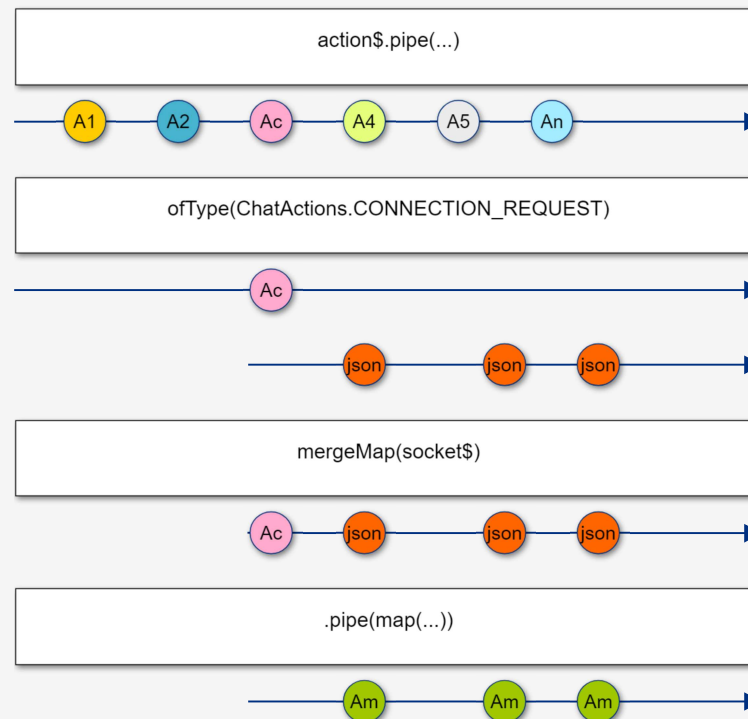
# REDUX-OBSERVABLE IN PRACTICE

## registrationRequestEpic (spaEpic.ts)



# REDUX-OBSERVABLE IN PRACTICE

## connectEpic (chatEpic.ts)



# ROUTING



React Router



Aviator



Backbone Router

**i** React's website has a more comprehensive list of available routers: [Link](#)

# FORMS

---



Redux Forms



Formik

Formsy

**i** React's basic documentation on controlled and uncontrolled forms: [Link](#)

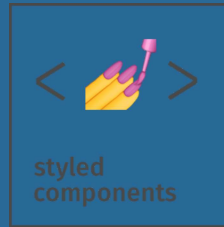
# FORMS

The image is a composite of two parts. The left part shows a web application for "Chariot SPA Day". It has a dark blue header with a logo and the text "Chariot SPA Day" and "CHARIOT SOLUTIONS". Below the header, there's a "Welcome to the SPA" section with a paragraph about the day. Then, there's a "Register for your appointment:" section with a form containing fields for "Name" (filled with "Matthew T Gilbride"), "Email" (filled with "mtg5014@gmail.com"), and "Treatment" (a dropdown menu with "Choose a treatr" selected). A green "Register" button is at the bottom of the form. Below the form, there's a paragraph about the power of the user. At the bottom right, there's a green button with a speech bubble icon and the word "CHAT".

The right part shows the React DevTools interface. The "Elements" panel displays the component tree, starting with "<Provider>", followed by "<Connect(ConnectedRouter)>", "<ConnectedRouter action='PUSH'>", "<Router>", "<App>", "<Header>", "<Home>", and a "<div className='Home\_home\_Msffs'>" containing a "<h1>Welcome to the SPA</h1>", a "<div className='Home\_row\_3RM-M'>" containing a "<div className='Home\_column\_3MWvA'>" with "<Welcome>", and another "<div className='Home\_column\_3MWvA'>" containing "<Routes>", "<Switch>", "<Route exact={true} path=' /schedule/:id/register'>", and "<Connect(RegistrationComponent)>". The selected component is "<RegistrationComponent inFlight={false}> == \$r", which contains "<h3>Register for your appointment:</h3>", "<form className='Registration\_form\_CctT5'>", "<h4>Appointment: ">Nov. 28th - 11 am - "12 pm"</h4>", and "<div><label>Name</label><input type='text'". The "Props" panel on the right shows the state of the selected component: "history: {...}", "inFlight: false", "location: {...}", "match: {...}", "registrationRequest: regi", and "session: {...}". The "State" panel shows "email: 'mtg5014@gmail.co m'", "name: 'Matthew T Gilbride'", and "treatment: ''".

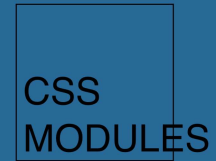
# STYLING

---



Global Styles or  
Imported Component Styles

React Specific Approaches



CSS Modules

# TESTING

---



Jest



Enzyme

# WHO IS REACT GOOD FOR?

---

- Existing projects
- Strong development teams which want flexibility
- Small projects



# THANKS!

---

- Questions?