Introduction to the React Ecosystem

Flux, Redux, React Native, Relay, Dev Tools

Rami Sayar - @ramisayar

Senior Technical Evangelist

Microsoft Canada

@ramisayar

Agenda

- Introduction to React & the Ecosystem
- What is Flux
- What is Redux
- What is Relay (and other Reactive Databases/APIs)
- What is React Native
- Useful React Dev Tools

Image from [Howard Lake](), licensed under [CC Attribution-ShareAlike 2.0 Generic]()

ES6

webpack
MODULE BUNDLER

Relay

Objective – Navigate the React Ecosystem

Level: Intermediate.

Assumption: Beginner React Developer

Quick 5 Minute Introduction to React

Introduction to React

- React is a UI *library* developed at Facebook.
- Lets you create interactive, stateful & reusable UI components.
- React can be used on both the client and server side.
- Uses a Virtual DOM to selectively render subtrees of components on state change.

Introduction to React

- Adds this weird thing to your HTML called JSX.
- Let’s you write HTML-ish tags in JavaScript to simplify creating components.

```
var HelloWorldComponent = React.createClass({
  render: function(){
    return ( <h1>Hello, world!</h1> );
  }
});
```

Introduction to React

- Added attributes are called props and can be used to render dynamic data.

```
var HelloNameComponent = React.createClass({
  render: function(){
    return ( <h1>Hello, {this.props.name}!</h1> );
  }
});
ReactDOM.render(<HelloNameComponent name="Rami"
/>, document.getElementById('app'));
```

Introduction to React

- Render method is the only required spec for creating a component.
- React has lifecycle methods:
  - componentWillMount – Invoked once, on both client & server before rendering occurs.
  - componentDidMount – Invoked once, only on the client, after rendering occurs.
  - shouldComponentUpdate – Return value determines whether component should update.
  - componentWillUnmount – Invoked prior to unmounting component.

Introduction to React

- Every component has a state object and a props object.
- Functions & Objects:
  - getInitialState – Return value is the initial value for state.
  - setState – Sets the state and triggers UI updates.
  - getDefaultProps – Sets fallback props values if props aren’t supplied.

Final Notes – Introduction to React

- React events are attached as properties and can trigger methods.
- Data flows unidirectionally via the state and props objects.
- React seams to rerender the whole app on every data change but really it only ends up rerendering the parts that changed.

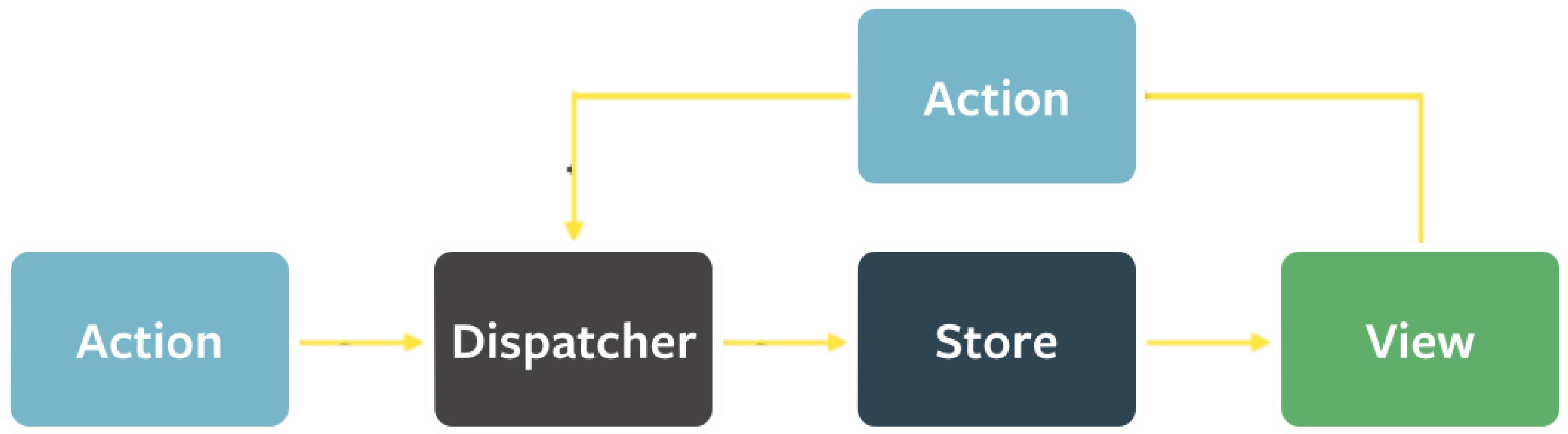
- To assign CSS classes you have to use className.
- You can use ES6 Classes instead of the createClass function.

React == UI Library. What About App State?

Flux Architecture

- Flux is the architecture that Facebook uses for building client-side web apps.
- More of a pattern rather than a framework.
- Flux doesn’t follow MVC in favor of a unidirectional data flow.
- Flux architecture is composed of four major parts: Dispatchers, Stores, Views and Actions.

Flux Architecture



Source: https://facebook.github.io/flux/docs/overview.html

Flux Architecture

- Dispatchers are the central hubs that manage data flow.
  - Essentially registry of callbacks into Stores.
  - When Actions passed into the central Dispatcher, they are redistributed to the Stores.
  - Dispatchers manage dependencies between Stores.
- Stores contain the application state and logic (sort of like the Model in MVC)
  - Stores register themselves with the Dispatcher to receive Actions via a callback.
  - Actions describe state changes in the Store.
  - Stores broadcast an event saying they have changed so Views can update.

Flux Architecture

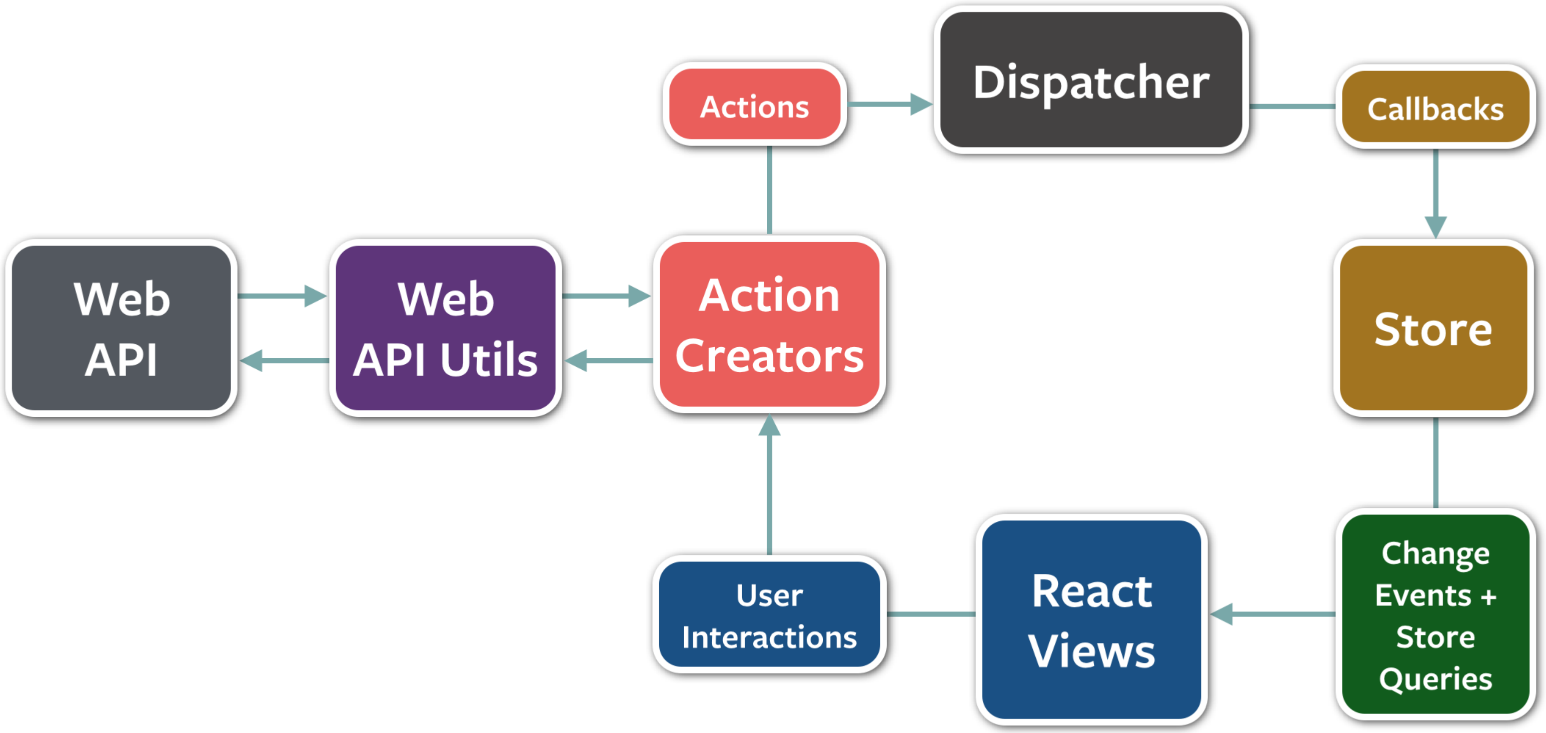
- Views are the React Components.
  - React Components are composable and are typically nested in a tree hierarchy.
  - A special “App View” behaves like a controller-view and provides glue code to propagate states down the chain.
  - Events cause Views to request the State from a Store to setState() so that render() will be executed.
- Actions describe a change and include a payload of data.

Flux Implementations

- Several Different Implementations of Flux:
  - Flux by Facebook
  - Redux by Dan Abramov
  - Alt by Josh Perez
  - Reflux by Mikael Brassman
  - Fluxxor by Michelle Tilley

Flux Implementations

- Flux by Facebook



Flux Implementations - Redux

- "Redux evolves the ideas of Flux, but avoids its complexity by taking cues from Elm."
- "The Gist of Redux
  - The whole state of your app is stored in an object tree inside a single *store*.
  - The only way to change the state tree is to emit an *action*, an object describing what happened.
  - To specify how the actions transform the state tree, you write pure *reducers*."

Source: https://github.com/reactjs/redux

React Libraries

react-router

- React Router keeps your UI in sync with the URL.
- Features like lazy code loading, dynamic route matching, and location transition handling are built in.
- Use react-router-redux to sync routing state with your Redux stores.

```
render((
  <Router history={browserHistory}>
    <Route path="/" component={App}>
      <Route path="about" component={About}/>
      <Route path="*" component={NoMatch}/>
    </Route>
  </Router>
), document.body)
```

reselect

- reselect is a selector library for Redux
  - Compute derived data => reduces size of the state object in Redux
  - Efficient way to handle computing derived data => don't recompute state if arguments didn't change. Selectors are composable.

```
const shopItemsSelector = state => state.shop.items;
const taxSelector = state => state.shop.taxPercent;
const subtotalSelector = createSelector( shopItemsSelector,
  items => items.reduce((acc, item) => acc + item.value, 0)
)
const taxSelector = createSelector( subtotalSelector,
 taxSelector, (subtotal, taxPercent) => subtotal * (taxPercent /
100)
)
```

Immutable.js

- Immutable.js provides immutable collections and data structures.
  - Immutable: Once created, cannot be altered at another point.
  - Persistent: Both original and mutated collections are valid.
  - Structural Sharing: New collections are created using the same structure as the original collection to reduce copying and achieve space/performance efficiencies.
- List, Stack, Map, OrderedMap, Set, OrderedSet and Record.
- Use in combination with Redux.

Flux-Standard-Action

- Flux Standard Action is a human-friendly standard for Flux action objects.
- Action objects must be plain JavaScript objects and have a type property.
- They can also have an error, payload and a meta property.
- Use with:
  - redux-actions - a set of helpers for creating and handling FSA actions in Redux.
  - redux-promise - Redux promise middleware that supports FSA actions.

React UI Component Libraries

react-bootstrap

- react-bootstrap wraps Bootstrap into React Components.

```
var buttonGroupInstance = (
  <ButtonGroup>
    <DropdownButton bsStyle="success" title="Dropdown">
      <MenuItem key="1">Dropdown</MenuItem>
    </DropdownButton>
    <Button bsStyle="info">Hello</Button>
  </ButtonGroup>
);
```

React Foundation

“Foundation Apps is a new framework for building web apps. It has awesome new features like flexbox based grid, motion-ui, and several core components for building web apps.” https://github.com/akiran/react-foundation-apps

Khan Academy’s React Component Library

“This is a collection of some of the most reusable React components built at Khan Academy. […] We're trying to make it just as easy to jumpstart React applications with a well-tested, thoughtful, and beautiful library of components.” http://khan.github.io/react-components/

Handling Data & Side Effects in React

Relay & GraphQL will be covered by Marc-Andre Giroux.

Falcor

- Falcor is a JavaScript library for efficient data fetching by Netflix.
  - One Model Everywhere: Represent remote data as a JSON graph. Treat data the same everywhere (in memory, client, network, etc).
  - Data is the API: JavaScript-like path syntax to access data. Retrieve data using JavaScript operations like get, set, and call.
  - Bind to the Cloud: Falcor automatically traverses references in your graph and makes requests as needed. Falcor transparently handles and aggregates requests for network efficiencies.

react-resolver

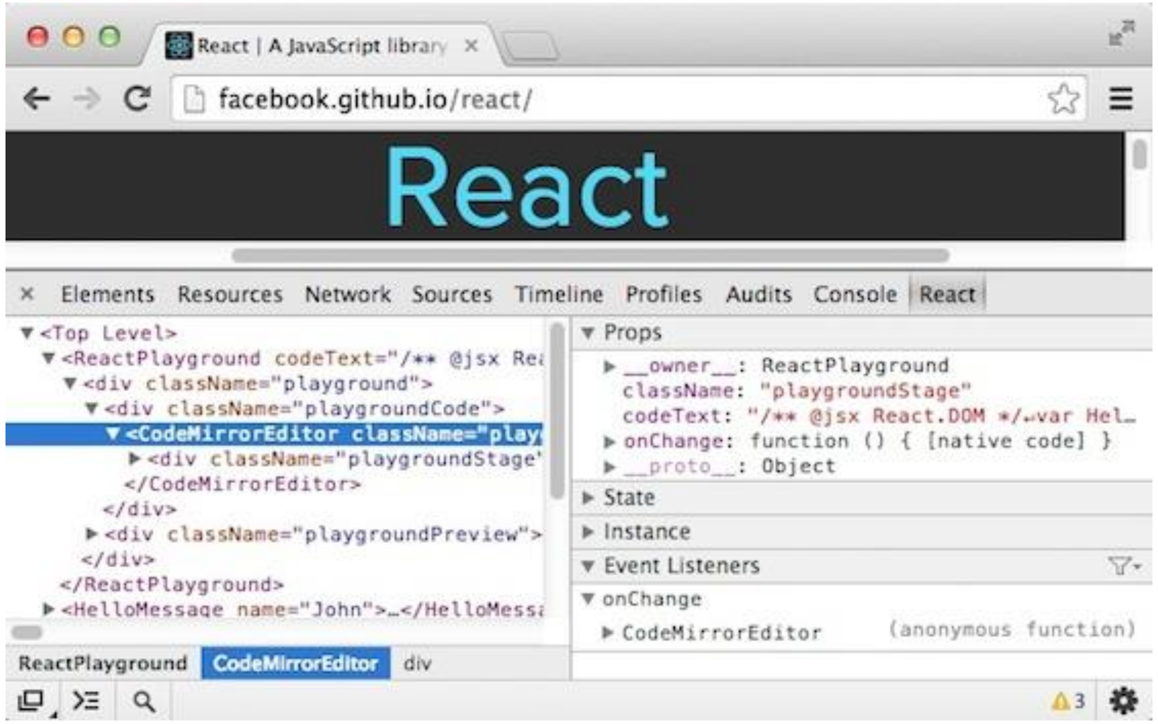
- React Resolver lets you **define data requirements** ***per-component*** and will handle the nested, async rendering on both the server & client for you.

```
@resolve("user", function(props) {
  return http.get('/api/users/${props.params.userId}');
})
class UserProfile extends React.Component {
  render() { ...
```

Tools for the React Ecosystem

React Chrome Developer Tools

- Adds a new tab titled "React" in your Chrome DevTools.
- Shows list of the React Component hierarchy.



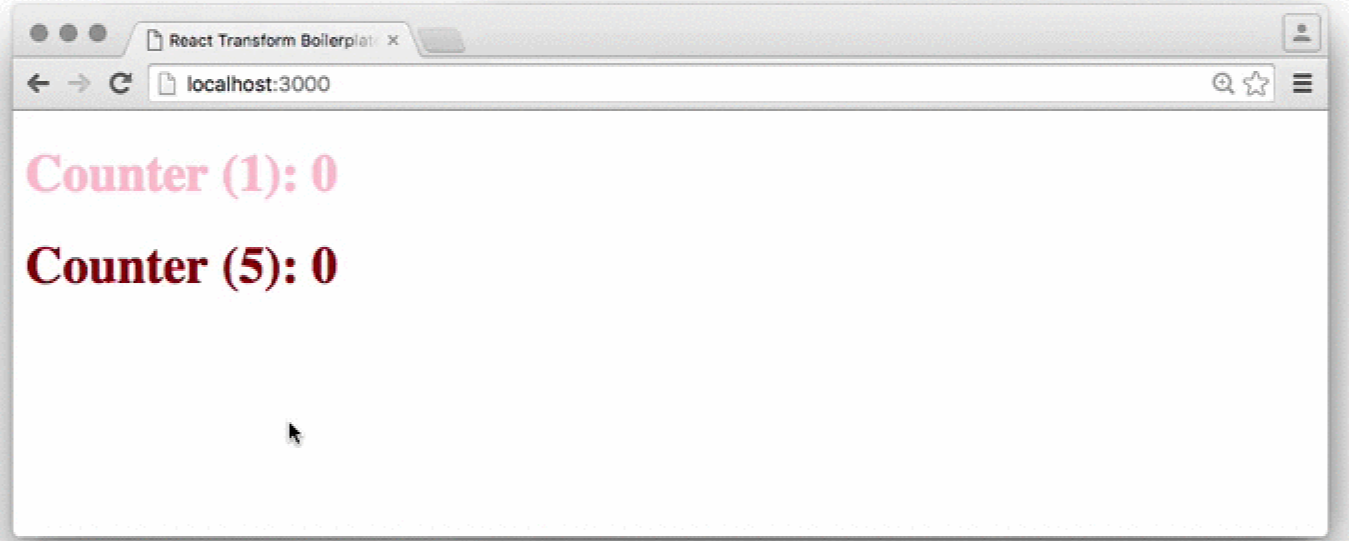
```
▼<Top Level>
  ▼<ReactPlayground codeText="/** @jsx Rea
    ▼<div className="playground">
      ▼<div className="playgroundCode">
        ▼<CodeMirrorEditor className="play
          ▶<div className="playgroundStage'
        </CodeMirrorEditor>
      </div>
      ▶<div className="playgroundPreview">
    </div>
  </ReactPlayground>
  ▶<HelloMessage name="John">…</HelloMessa
```

```
▼ Props
  ▶ __owner__: ReactPlayground
    className: "playgroundStage"
    codeText: "/** @jsx React.DOM */↵var Hel…
  ▶ onChange: function () { [native code] }
  ▶ __proto__: Object
▶ State
▶ Instance
▼ Event Listeners
▼ onChange
  ▶ CodeMirrorEditor        (anonymous function)
```

React Transform

- babel-plugin-react-transform wraps React components with arbitrary transforms. In other words, **it allows you to instrument React components** in any way—limited only by your imagination.
  - **react-transform-hmr** - enables hot reloading using HMR API
  - **react-transform-catch-errors** - catches errors inside render()
  - **react-transform-debug-inspector** - renders an inline prop inspector
  - **react-transform-render-visualizer** - highlight components when updated

React Transform Demo



Counter (1): 0

Counter (5): 0

Putting Everything Together

React Static Boilerplate

- A static website starter kit powered by React.js and Webpack.
    - ✓ Generates static .html pages from React components
    - ✓ Generates routes based on the list of files in the /pages folder
    - ✓ Next generation JavaScript with Babel
    - ✓ Sass syntax for CSS via postCSS and precss
    - ✓ Development web server with BrowserSync and React Transform
    - ✓ Bundling and optimization with Webpack
    - ✓ Yeoman generator (generator-react-static)

React Starter Kit – “isomorphic” boilerplate

- React Starter Kit is an opinionated boilerplate for web development built on top of Facebook's React library, Node.js / Express server and Flux architecture.
- Containing modern web development tools such as Webpack, Babel and Browser Sync.

MERN – "isomorphic framework"

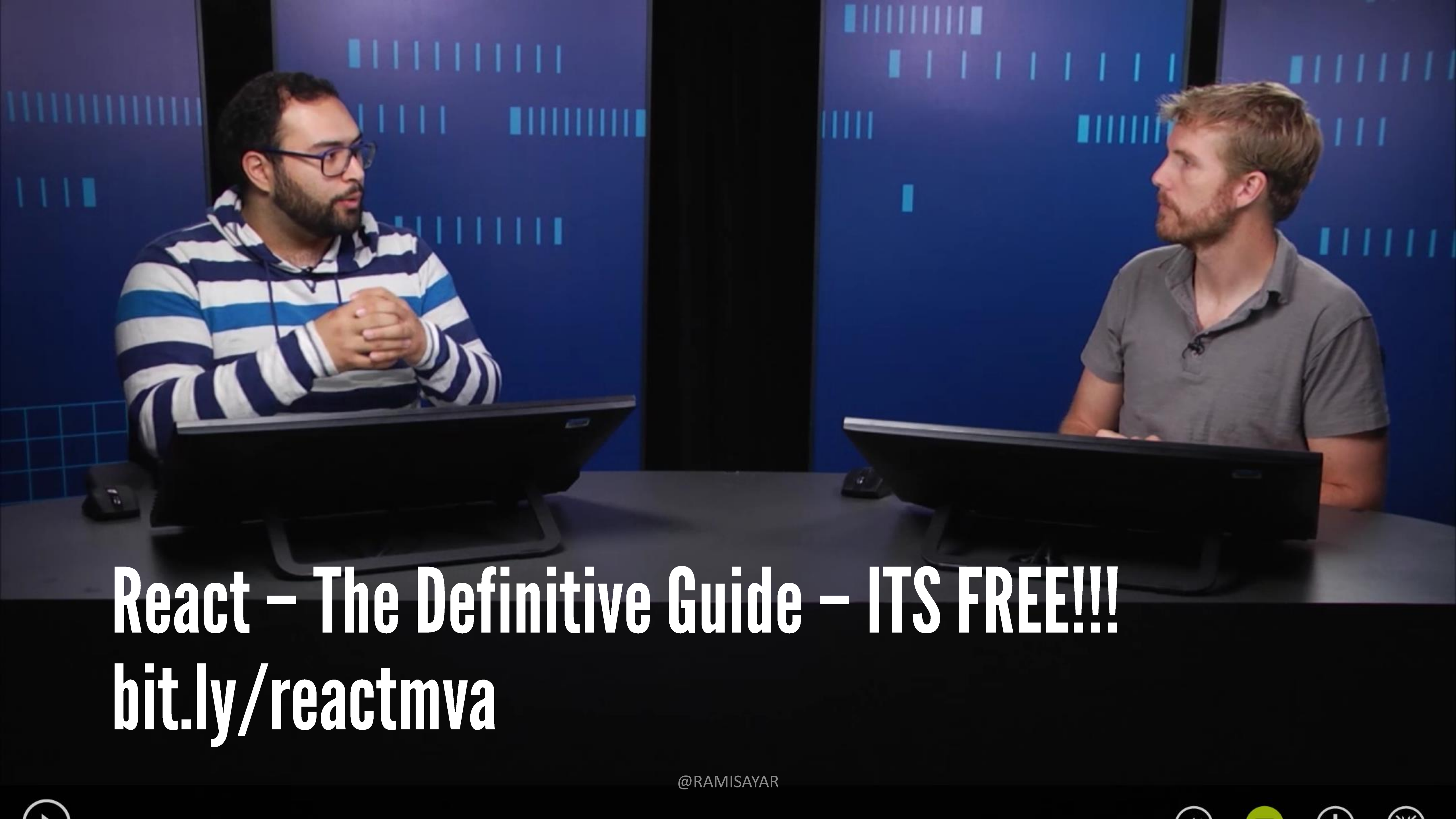
- MERN is the easiest way to build isomorphic JavaScript apps using React and Redux.
- MERN is a scaffolding tool which makes it easy to build isomorphic apps using Mongo, Express, React and NodeJS.

I’ve seen the

FUTURE

It’s in my

BROWSER



React – The Definitive Guide – ITS FREE!!!
bit.ly/reactmva

What did we learn?

- Introduced React & the Ecosystem
- What is Flux Architecture
  - Flux, Redux, Alt.js, etc.
- Introduced React Libraries and UI Component Libraries
- Introduced Falcor & React Resolver
- Useful React Dev Tools

Thank You! Questions?

tw: @ramisayar | gh: @sayar

Resources, References, Links

- https://www.toptal.com/react/navigating-the-react-ecosystem
- https://github.com/enaqx/awesome-react
- https://github.com/facebook/react/wiki/Complementary-Tools
- http://slides.com/cguedes/a-tour-on-react-ecosystem

Microsoft