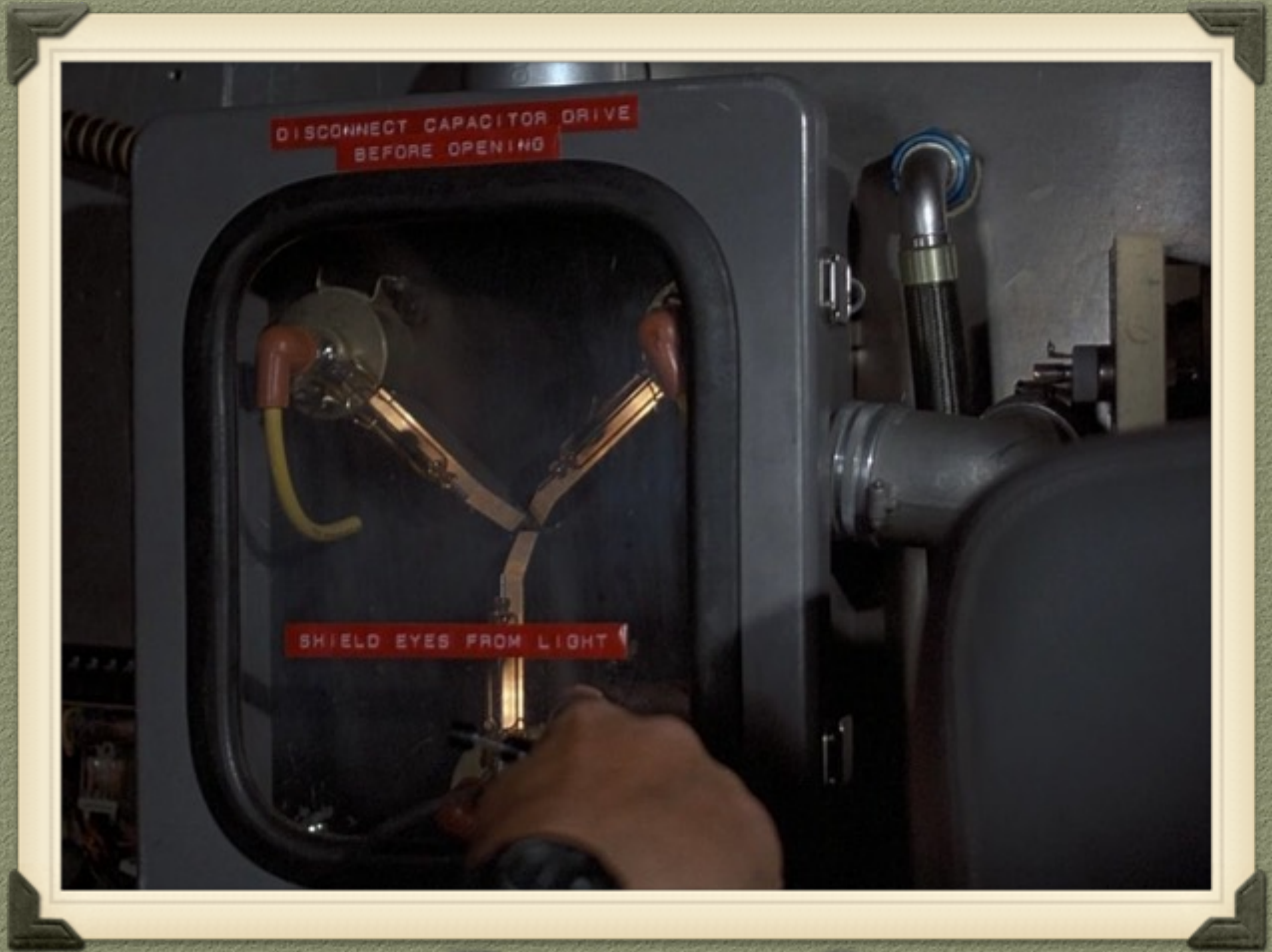


Flux and Redux Storytime

Ohana.js

@keokilee

github.com/keokilee

Lots of Flux Resources

Not as many resources on why

React + (Reflux || Redux)

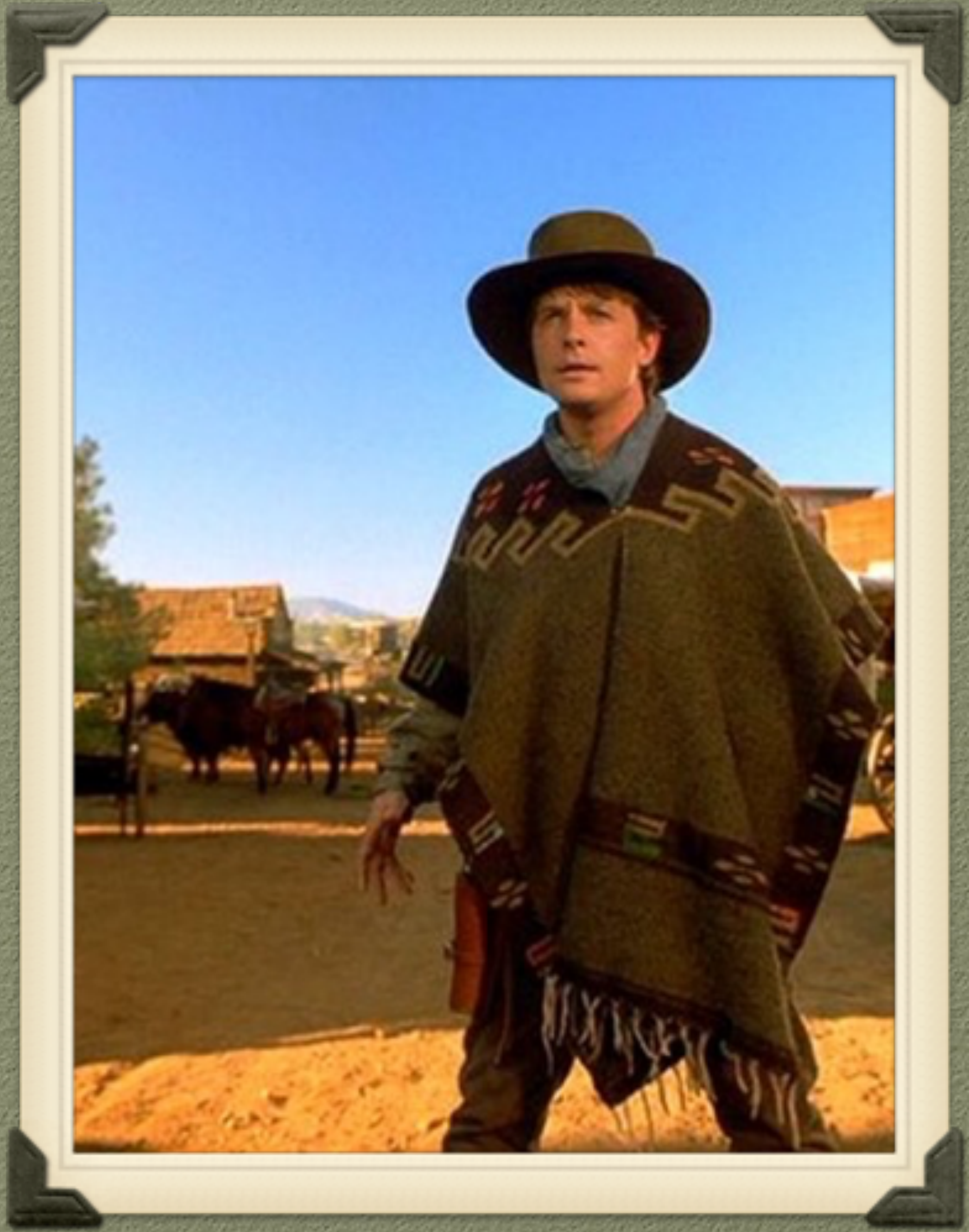
Evolution of Libraries

CALIFORNIA
OUTATIME

Year 1885

JS time, circa 2005

The Wild West



prototype
yui
script.aculo.us
LIPS.COM

Year 1955

JS Time, circa 2009

The Golden Years

ENCHANTMENT
UN THE SEA
jQuery
LIPS.COM

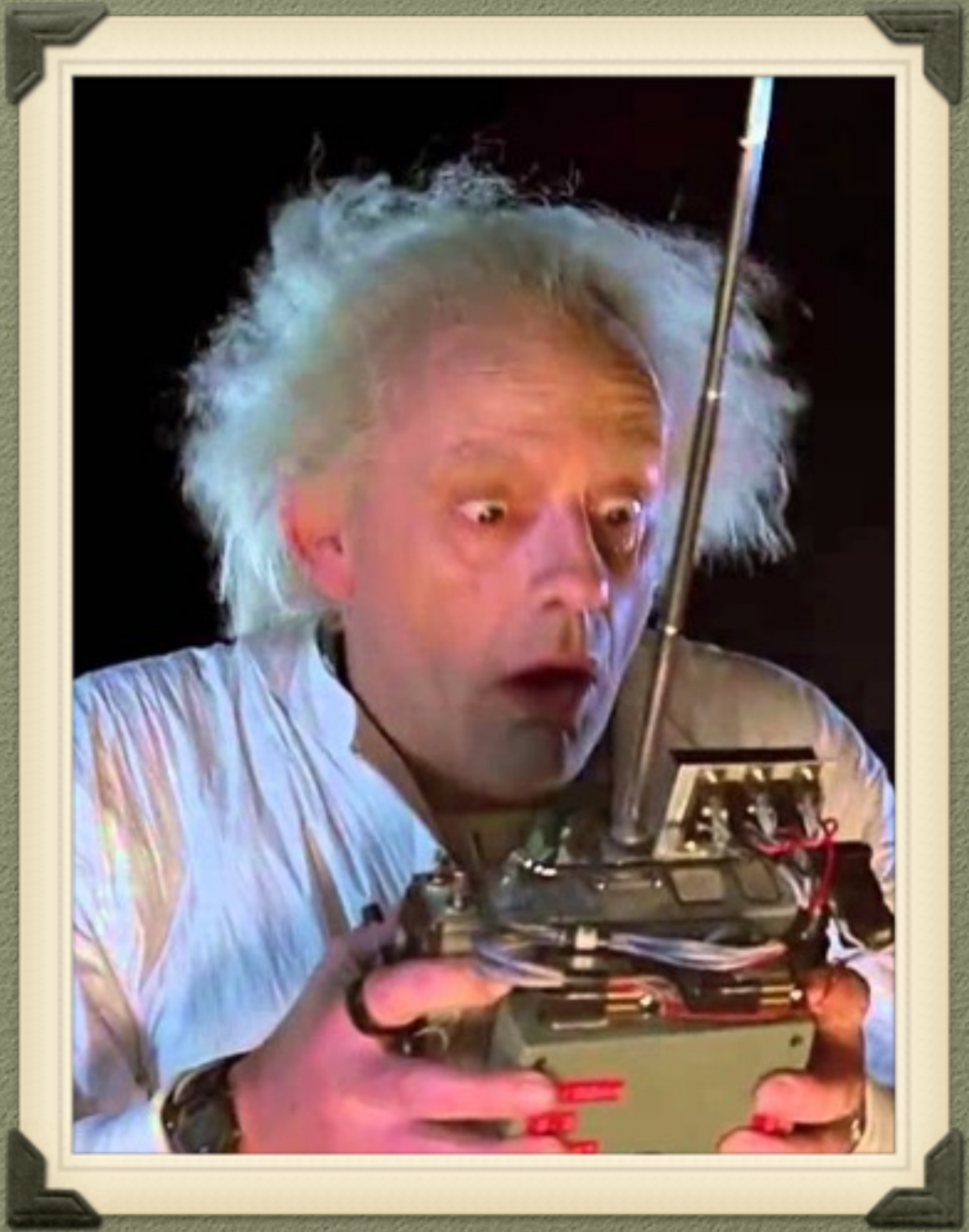
```
$(document).ready(function() {
  // All code goes here
  // ...
  $('.manure').on('click', function() {
    // body
  });
  // ...
});
```

Scalability? Testing?

Year 1985

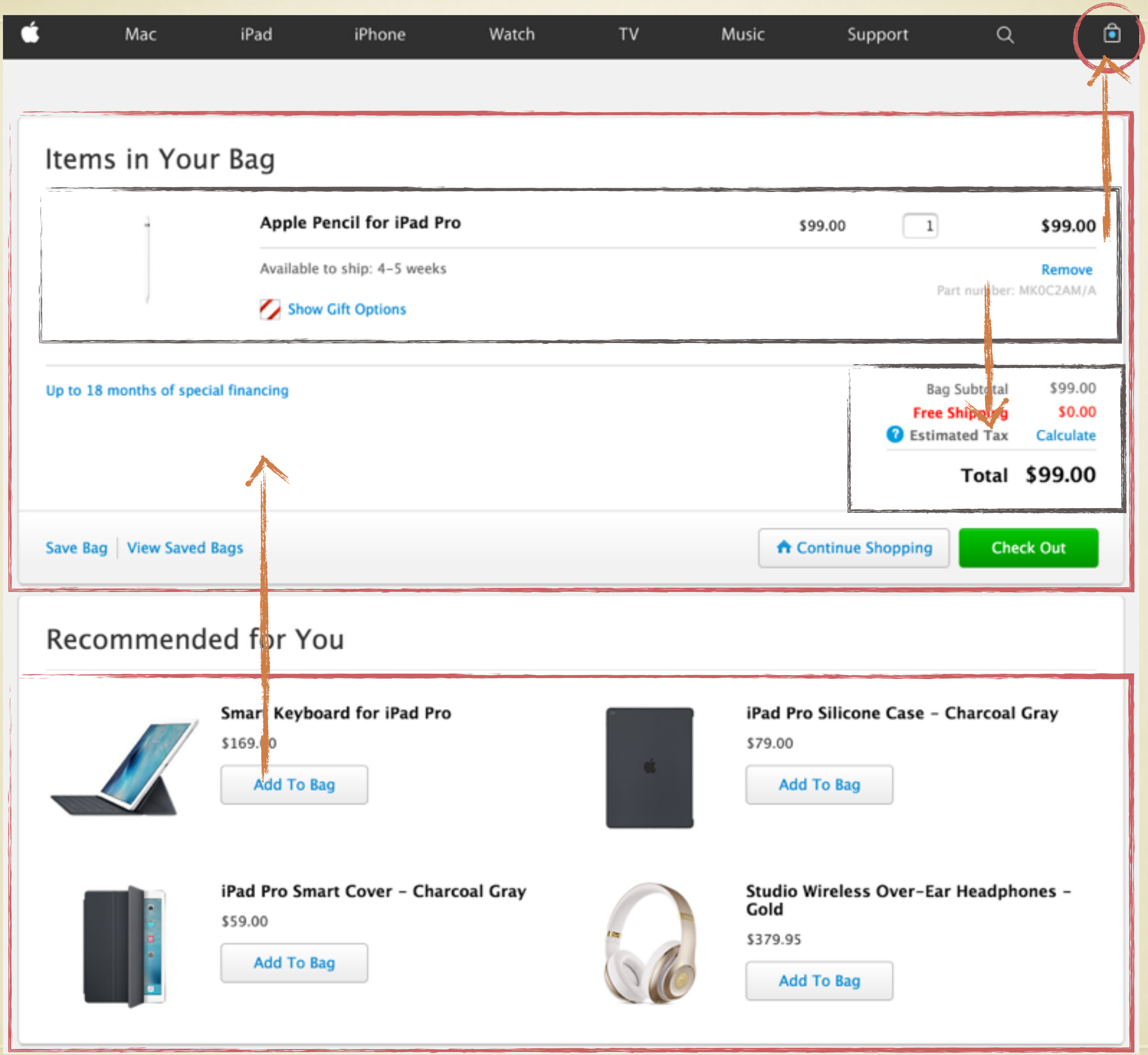
JS Time, Circa 2013

The Birth of Modern Frameworks



MV* Era

BACKBONE.JS



Items in Your Bag

Apple Pencil for iPad Pro	$99.00	1	**$99.00**
Available to ship: 4-5 weeks			Remove
Show Gift Options			Part number: MK0C2AM/A

Up to 18 months of special financing

Bag Subtotal	$99.00
Free Shipping	$0.00
? Estimated Tax	Calculate
Total	**$99.00**

Recommended for You

Smart Keyboard for iPad Pro
$169.00

iPad Pro Silicone Case - Charcoal Gray
$79.00

iPad Pro Smart Cover - Charcoal Gray
$59.00

Studio Wireless Over-Ear Headphones - Gold
$379.95

Parent Controller?

- ControllerAs is a must
- State is distributed. Who has what?

Events?

- Where do events come from?
- Need to document
- May need parent controller
- Fear of getting things out of sync

$rootScope?

- What's in $rootScope? ¯_(ツ)_/¯
- Who can change things in $rootScope?

Services/Factories

- Can be the most Flux-like (with a proper API)
- See $rootScope if we can make arbitrary changes to the internal state.
- State is still distributed

Managing State

You: What's your emergency?

Client: Something is broken on my app.

You: Can you tell me what happened?

Client: I don't know, I clicked on a bunch of things and it broke. Fix it.

You after trying to reproduce it:
(╯°□°）╯︵ ┻━┻

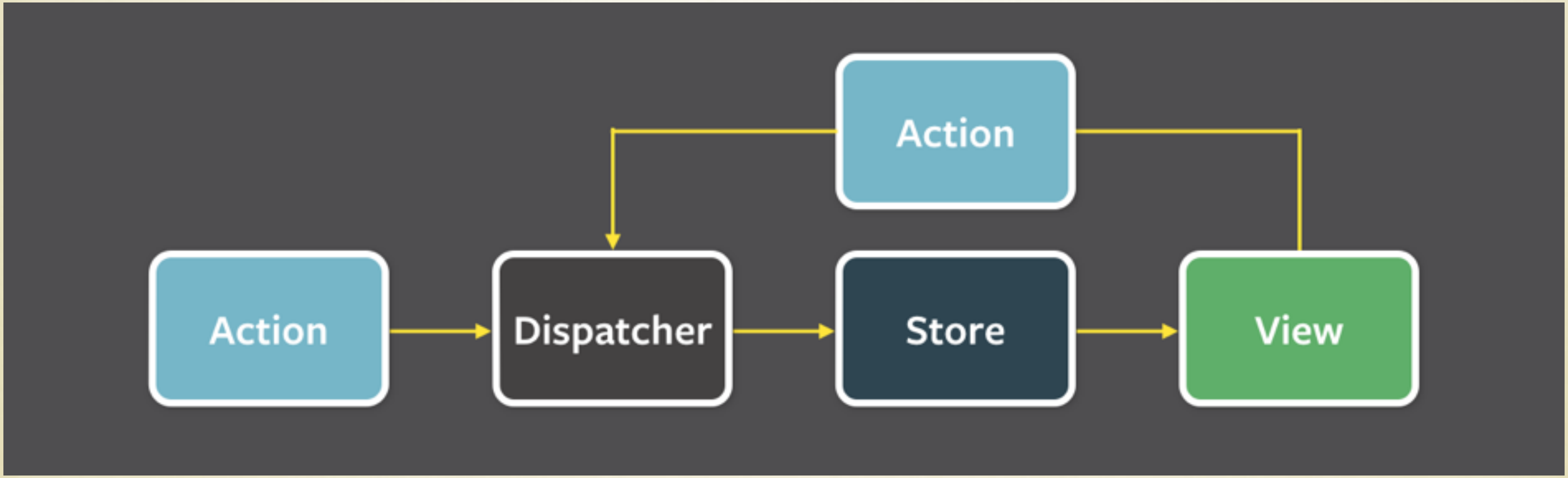
Year 2015

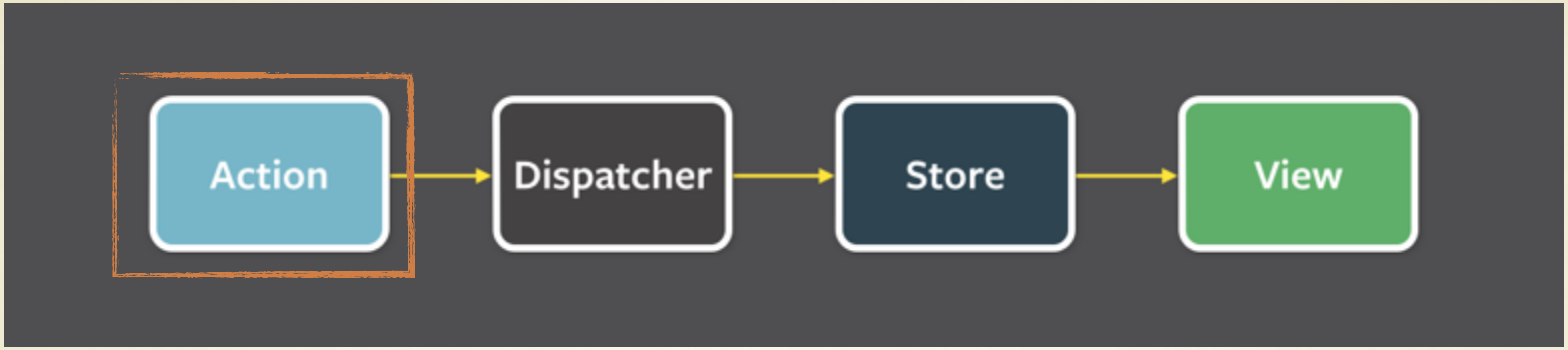
JS Time, Circa 2015
The Future

Flux Architecture

- Pattern used at Facebook for building out client web applications
- Wanted their state to be "predictable"
- Technically not dependent on React
- Not MV*, so don't try to make that comparison.

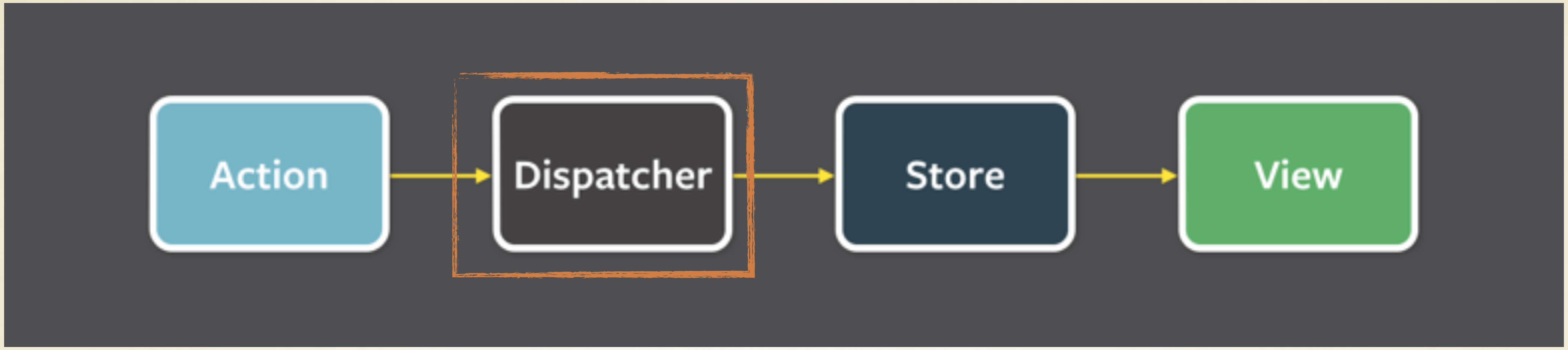
Unidirectional Data Flow



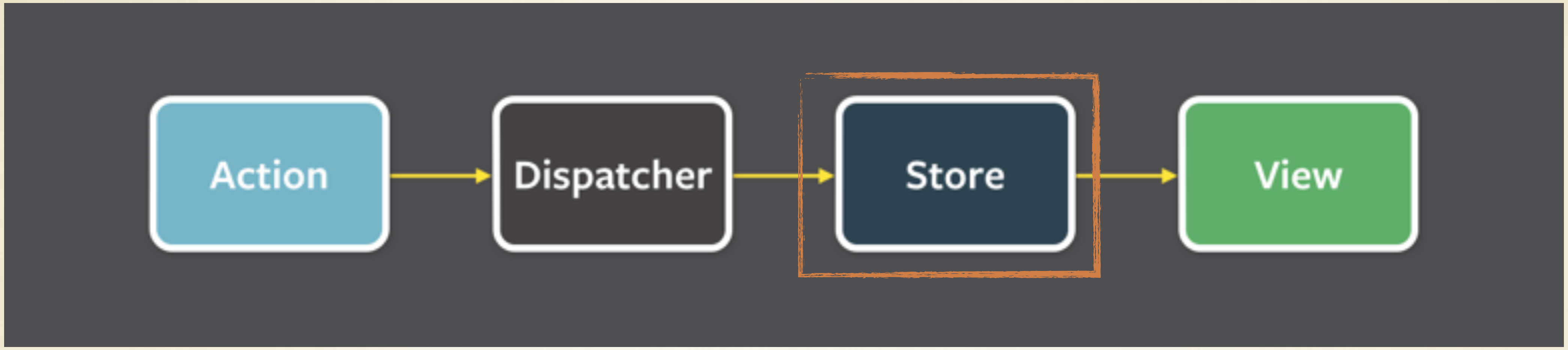


An action is an object/event containing the type and any arguments.

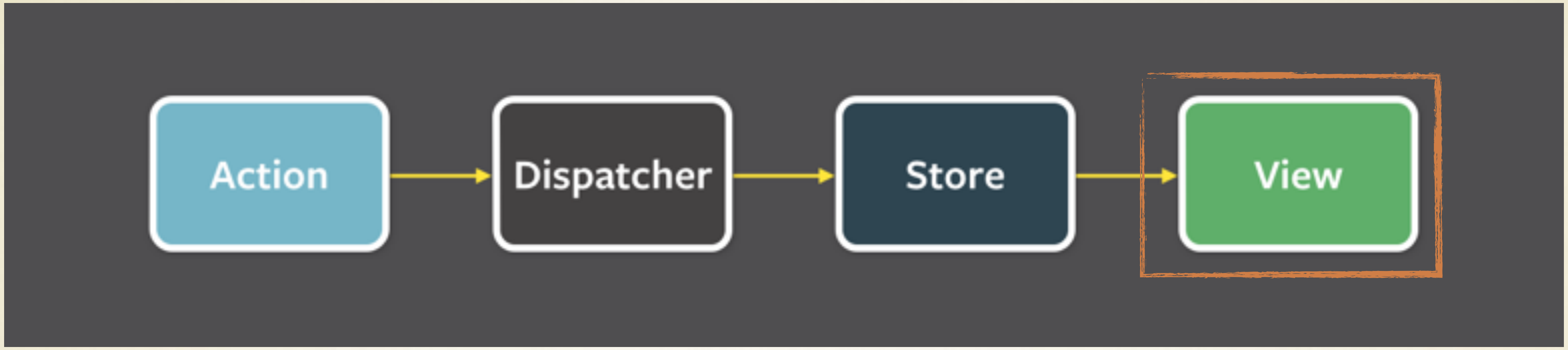
Action creators are invoked by views or other actions (in async actions)



Stores register with the dispatcher. The dispatcher then passes the actions to every store via callbacks.



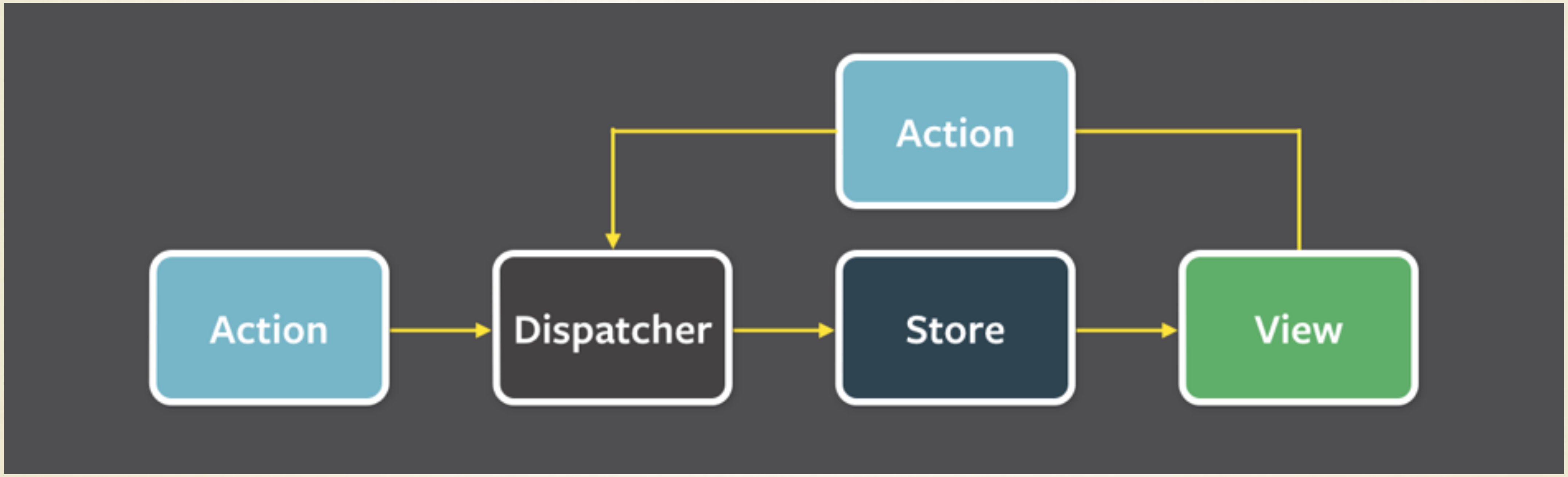
Central storage of state in the application. Checks the action type and makes changes to internal state based on the action. Emits a "changed" event if something changed.



Mostly self-explanatory. There is a main view that listens for changes in the store and then passes those changes down to child views.

In React, the Virtual DOM determines if a component should re-render.

Async Actions in Flux



- An action creator may run an async function. During this, it may fire multiple actions.
- If async actions are chained, we can do this loop multiple times.

What Did We Accomplish?

- State is centralized.
- We know what's in the store.
- We know what actions can be performed
- Testing is easy

Aside: React Components

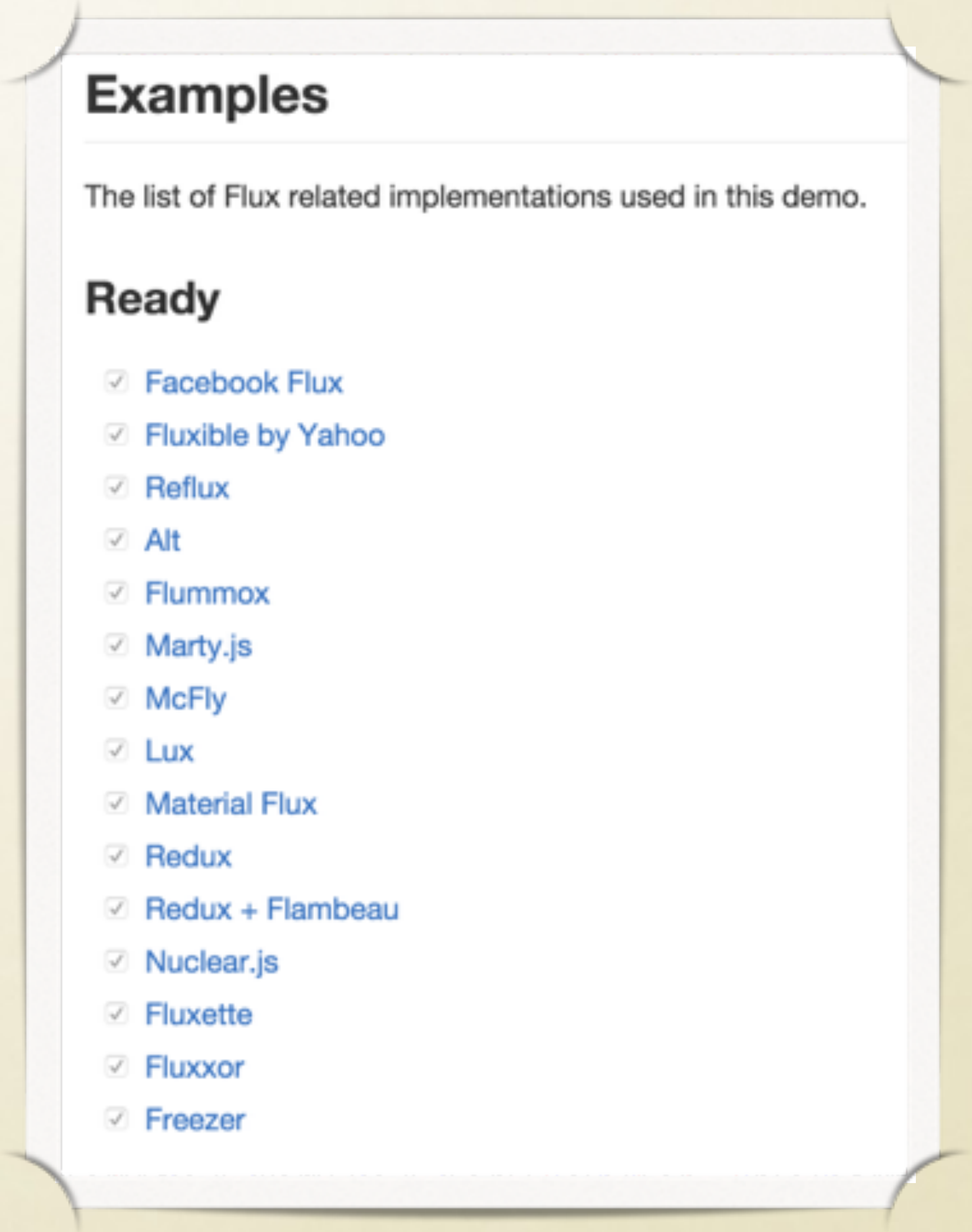
- Smart components
  - Listen to changes in state.
  - Typically do not render DOM of its own.
- Dumb components
  - Do not contain state of their own.
  - Purely a function of the properties passed in.

```
// Smart Component
export class App extends React.Component {
  render() {
    return (
      <Header title={this.state.title} />
    );
  }
}

// Dumb Component
var Header = (props) => {
  <div>
    <h1>{props.title}</h1>
  </div>
}
```

So we're good?

- https://github.com/voronianski/flux-comparison
- Varying opinions on Flux



Examples

The list of Flux related implementations used in this demo.

Ready

- ☑ Facebook Flux
- ☑ Fluxible by Yahoo
- ☑ Reflux
- ☑ Alt
- ☑ Flummox
- ☑ Marty.js
- ☑ McFly
- ☑ Lux
- ☑ Material Flux
- ☑ Redux
- ☑ Redux + Flambeau
- ☑ Nuclear.js
- ☑ Fluxette
- ☑ Fluxxor
- ☑ Freezer

Our Hero

Dan Abramov (@dan_abramov)

github.com/gaearon

Redux

- Dan Abramov didn't intend to create Redux
- Intrigued by Webpack's hot module replacement and time traveling debugging.
- Needed a way to maintain state during Webpack's hot module replacement.

Hot Modules

- In hot module replacement, replacing a module would blow away any associated state.
- Flux does most of the work already, except in one place: the store.

Redux Reducers

- Idea: Separate the actual state from the functions that handle actions and manipulate state.
- The functions get passed some subset of the current state from the store and the action.
- Apply the action and return the result.
- These functions are then combined to represent the final state.

Redux Store

- Single source of truth for your entire app.
- Only one store. App state is in one object tree.
- Assumption is that state never changes. Thus, reducers must return a new object.

Time Travel Debugging

- What if you tracked every action?
- Then, you can go back by replaying the actions

Other Differences

- No dispatcher. Store exposes a "dispatch" function that takes an action.
- Middleware - Can perform functions before and after an action gets applied to the reducer.
- Redux Thunk - Middleware that allows actions to return a function. This function gets the dispatch method so it can dispatch multiple actions.

Our Redux Experience

- Our Node Knockout app "Simmer" was built using React, Redux, and Typescript.
- I handled most of the Redux code.
- People creating "dumb components" did not need to touch Redux.

Resources

- facebook.github.io/flux
- redux.js.org
- http://redux.js.org/docs/introduction/Ecosystem.html
- https://code-cartoons.com/ - By Lin Clark (formerly of NPM, now at Mozilla)