Universal Web Applications with React & NodeJS

Reducing Complexity

Building Scalable Web Applications

Things that Sam likes

Who's this clown?

- Samuel Reed - STRML.net
- In Milwaukee, WI, previously Hong Kong & Washington DC
- Frontend developer for 10 years
- CTO and Co-Founder of BitMEX, the Bitcoin Mercantile Exchange
- Maintainer of ~10 React libraries

Isomorphic / Universal JS

- NodeJS brought us JS on the server
- Browserify brought us shared libraries between client and server
  - Code transformers switch server libraries with browser libraries with identical APIs
- Webpack helps bring browser modules to the server
- React brings us an entire shared application

The Problem

Traditional webapps have complex state on a platform that was never designed for applications.

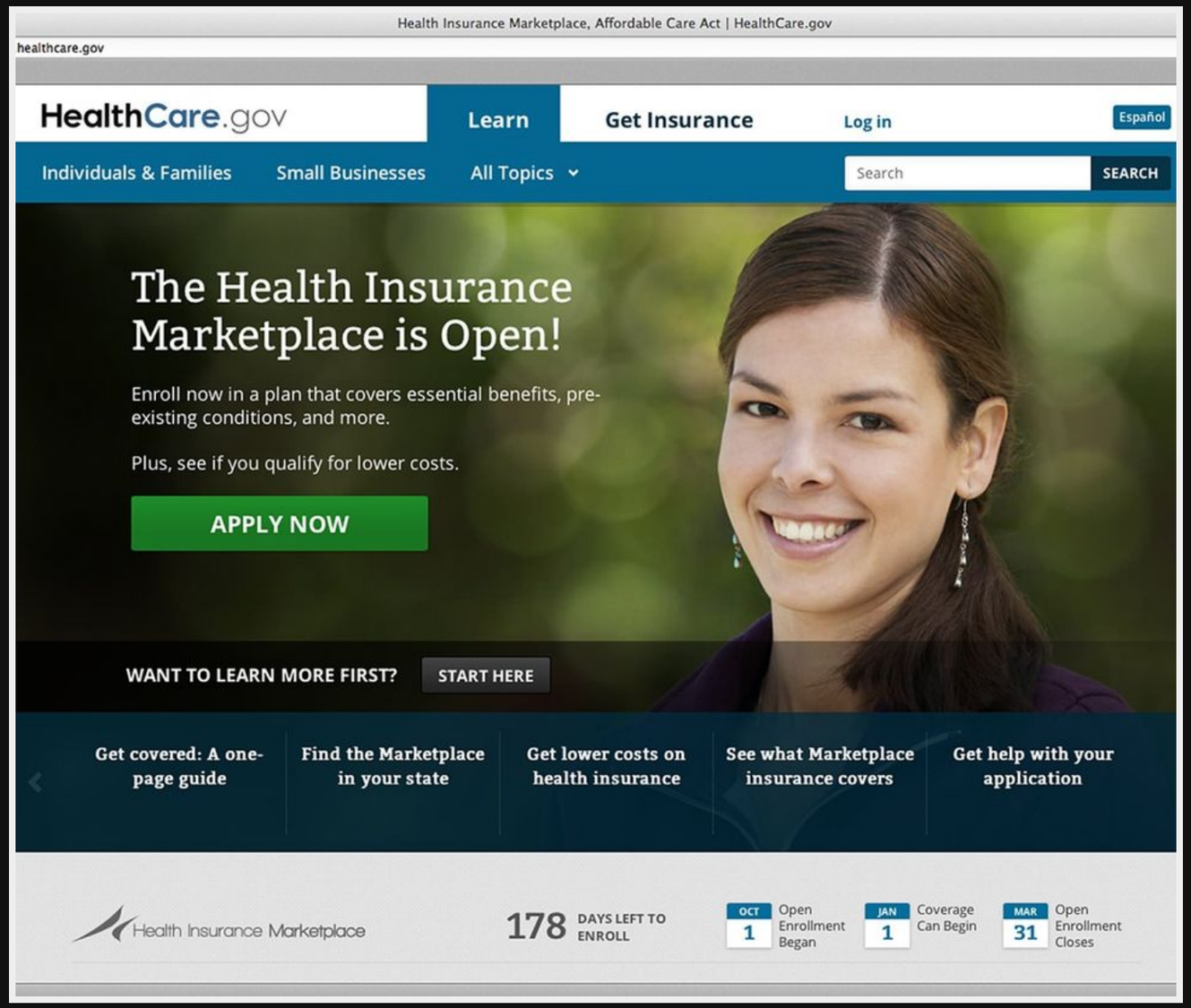
> *...Remember the days before React, when you had to write one piece of code to render your application and another to update it? HAHAHAHA. That was awful. React showed us that expressing our views declaratively leads to clearer, more predictable, and less error-prone applications.*

Andrew Clark, author of Flummox (Flux implementation)

Typical issues

- **Out-of-date data**
- **Complex Local State**
- **Cascading Updates**
- **Difficult testing**

OH GOD WHY


Health Insurance Marketplace, Affordable Care Act | HealthCare.gov
healthcare.gov
HealthCare.gov
Learn
Get Insurance
Log in
Español
Individuals & Families
Small Businesses
All Topics
Search
SEARCH
The Health Insurance Marketplace is Open!
Enroll now in a plan that covers essential benefits, pre-existing conditions, and more.
Plus, see if you qualify for lower costs.
APPLY NOW
WANT TO LEARN MORE FIRST?
START HERE
Get covered: A one-page guide
Find the Marketplace in your state
Get lower costs on health insurance
See what Marketplace insurance covers
Get help with your application
Health Insurance Marketplace
178 DAYS LEFT TO ENROLL
OCT 1 Open Enrollment Began
JAN 1 Coverage Can Begin
MAR 31 Open Enrollment Closes

Reducing Complexity

- Simple data flow
- Shared code is less code
- Small modules mean smaller tests

Data changing over time is the root of all evil.

To change the DOM, you need to erase or read what was there before, and make changes.

You have to think about every possible transition between states.

Why React

Simple, declarative syntax:

- Declare what you want your views to look like, as functions, on every frame. (Similar to graphics programming)
- Virtually rerender the entire app on every frame!
- Updates use an efficient tree-diffing function to determine needed DOM mutations.
  - Entire tree branches can be skipped efficiently.
  - The simplicity of static rendering, even better speed than two-way binding

React

Keeps state sane:

- Intermediate state (in the DOM, not in your data) is impossible.
- Rendering is a pure function. Can be run on the server and for non-DOM targets
  - Prerender views for speed or SEO
  - Run similar code on mobile with React Mobile

Virtual DOM


Vyacheslav Egorov
@mraleph
Follow
often devs still approach performance of JS code as if they are riding a horse cart but the horse had long been replaced with fusion reactor
12:31 PM - 13 Dec 2013
140
99

Examples

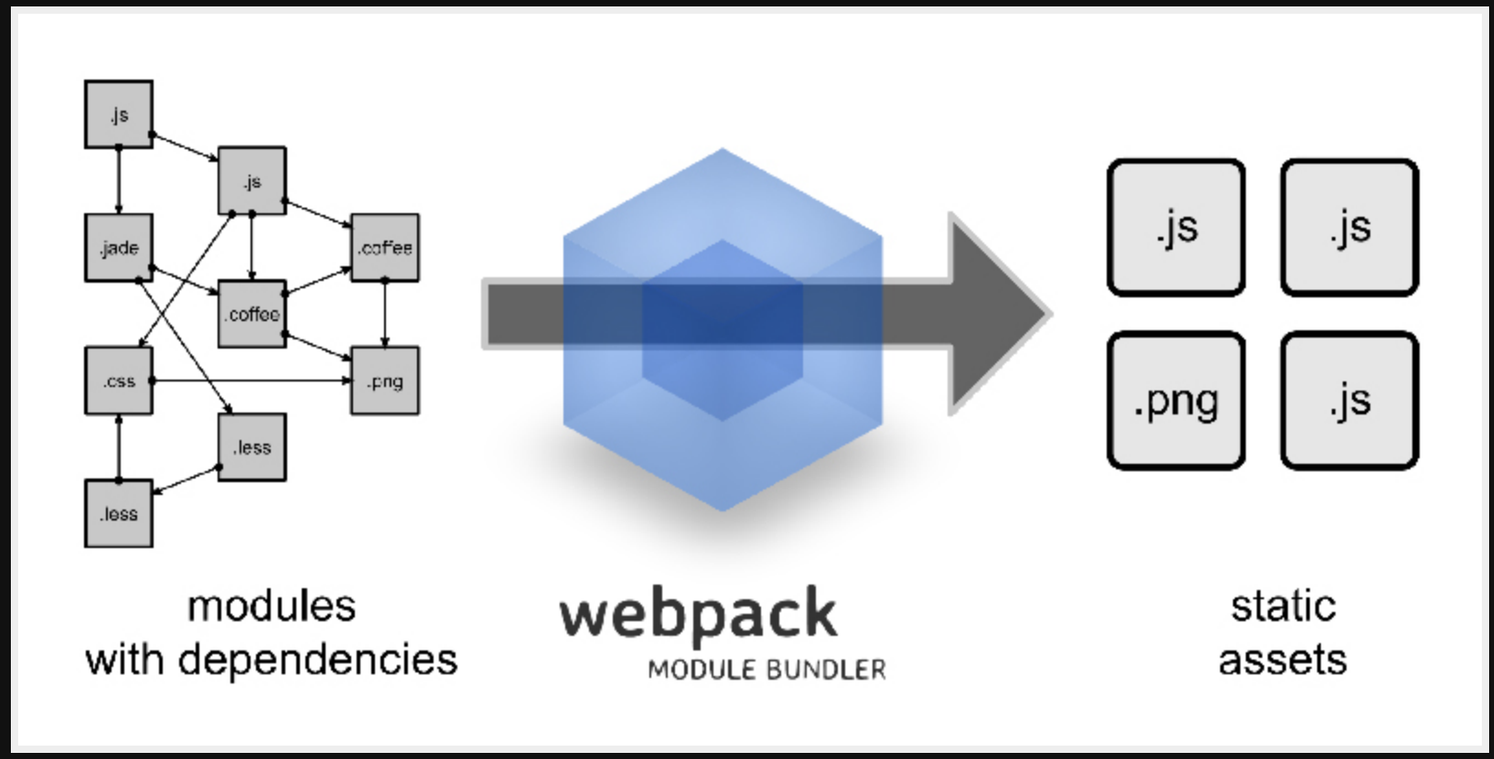
0: Building a basic component.

- To build a component, describe what you want it to look like in terms of functions.

1: Components are composable.

- You can nest components inside each other.
- Teams can share common components (like `<Table>` or `<Button>`) across projects.

What is Webpack?



- `require()` anything
  - `require('./styles.sass')`
  - `var data = require('./data.json')`
  - `var tpl = require('./index.jade')`
- Target browsers and servers
  - Targets: `web`, `webworker`, `node`, `async-node`, `node-webkit`, `electron`

Webpack is Extensible

> *Webpack is fed a configuration object. Depending on your usage of webpack there are two ways to pass this configuration object:*

CLI

If you use the CLI it will read a file `webpack.config.js` (or the file passed by the `--config` option). This file should export the configuration object:

```
module.exports = {
    // configuration
};
```

node.js API

If you use the node.js API you need to pass the configuration object as parameter:

```
webpack({
    // configuration
}, callback);
```

CONFIGURATION OBJECT CONTENT

> *Hint: Keep in mind that you don't need to write pure JSON into the configuration. Use any JavaScript you want. It's just a node.js module…*

Very simple configuration object example:

```
{
    context: __dirname + "/app",
    entry: "./entry",
    output: {
        path: __dirname + "/dist",
        filename: "bundle.js"
    }
}
```

`context`

The base directory (absolute path!) for resolving the `entry` option. If `output.pathinfo` is set, the included pathinfo is shortened to this directory.

> *Default:* `process.cwd()`

`entry`

The entry point for the bundle.

If you pass a string: The string is resolved to a module which is loaded upon startup.

If you pass an array: All modules are loaded upon startup. The last one is exported.

configuration object content

- `context`
- `entry`
- `output`
  - `output.path`
  - `output.filename`
  - `output.chunkFilename`
  - `output.sourceMapFilename`
  - `output.devtoolModuleFilenameTemplate`
  - `output.devtoolFallbackModuleFilenameTemplate`
  - `output.devtoolLineToLine`
  - `output.hotUpdateChunkFilename`
  - `output.hotUpdateMainFilename`
  - `output.publicPath`
  - `output.jsonpFunction`
  - `output.hotUpdateFunction`
  - `output.pathinfo`
  - `output.library`
  - `output.libraryTarget`
  - `output.umdNamedDefine`
  - `output.sourcePrefix`
  - `output.crossOriginLoading`
- `module`
  - `module.loaders`
  - `module.preLoaders` , `module.postLoaders`
  - `module.noParse`
  - automatically created contexts defaults `module.xxxContextXxx`
- `resolve`
  - `resolve.alias`
  - `resolve.root`
  - `resolve.modulesDirectories`
  - `resolve.fallback`
  - `resolve.extensions`
  - `resolve.packageMains`
  - `resolve.packageAlias`
  - `resolve.unsafeCache`
- `resolveLoader`
  - `resolveLoader.moduleTemplates`
- `externals`
- `target`
- `bail`
- `profile`

Webpack Hot Module Replacement (HMR)

- Like livereload, but works on JS too.
- Idempotent functions can be replaced in the running VM.
- Plugins available for styles, React components, action and store implementions, even request handlers.

2. Hot reloading & DevTools

- React components are pure functions, so they can be replaced at will without a refresh.
- Uses Webpack HMR
- Start
- Can use React-DevTools

Use Webpack with Babel

- Babel offers ES6+ syntax on older runtimes
  - Has support for ES7 features:
    - `async/await`
    - Static class properties
    - Comprehensions
- Babel plugins offer unique syntax transforms:
  - React optimizations (constants, inline objects)
  - Typechecking
  - Remove console/debugger
  - Inline arguments slice (slicing arguments causes deopt)
  - Dead code elimination
  - Lots more coming...

3a. ES6

- React is ready for ES6 and has nice syntax shortcuts.
- React components can be raw class objects.

3b. Universal App with Routing

- This app actually runs server-side.
- Even routing is possible at the server.
- Note the separate entry points for server prerender and client.

How Does This Work?

- Server grabs all needed data to create the app.
- Server runs app with data and route and renders to string.
- Server sends rendered page to client.
- Client's browser renders page immediately and starts to load JS.
- JS loads initial data payload and rebuilds app.
- When finished, checksums virtual DOM. If checksums match, do nothing.

```
<body>
    <div id="initialData" data-data="{&quot;conference&quot;:&quot;Web Unleashed&quot;}"></div>
    <div id="content">
        <div class="appContainer" data-reactid=".vnc3rdbpq8" data-react-checksum="1791062010">
```

React Moves Fast

React is not just about the DOM

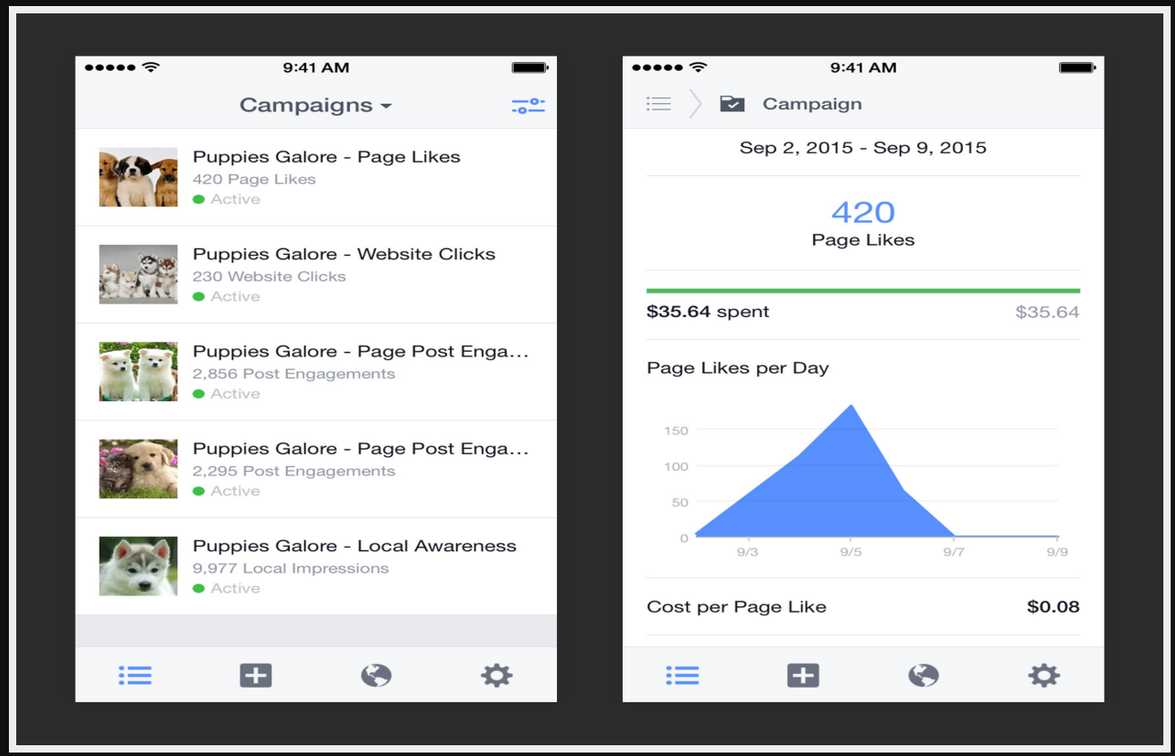
React is a general application platform. The DOM is one possible target (via `react-dom`).

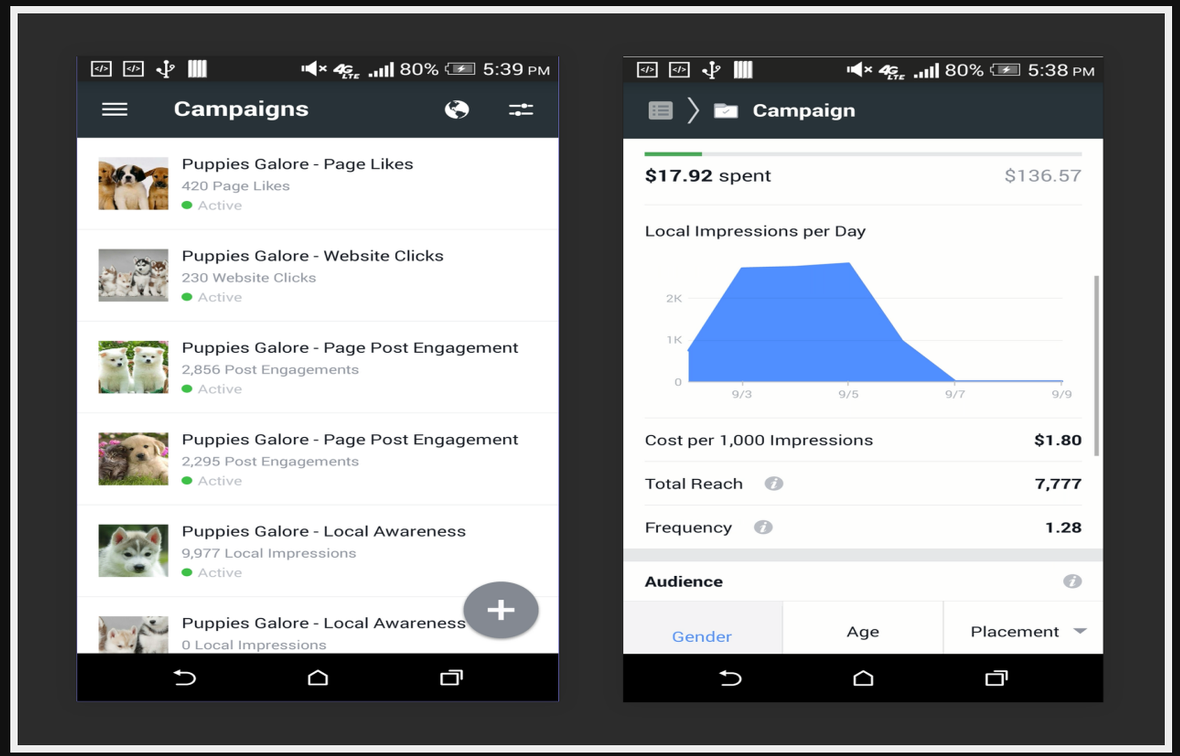
Other targets:

- iOS
- Android (just released Sep. 15)
- Canvas (react-canvas, Flipboard)
- D3 (react-d3)
- Three.js (react-three)
- Terminal (react-blessed)
- Let's take a look...

React-Native, iOS vs Android

- 85%+ code reuse, but completely native widgets with JS core



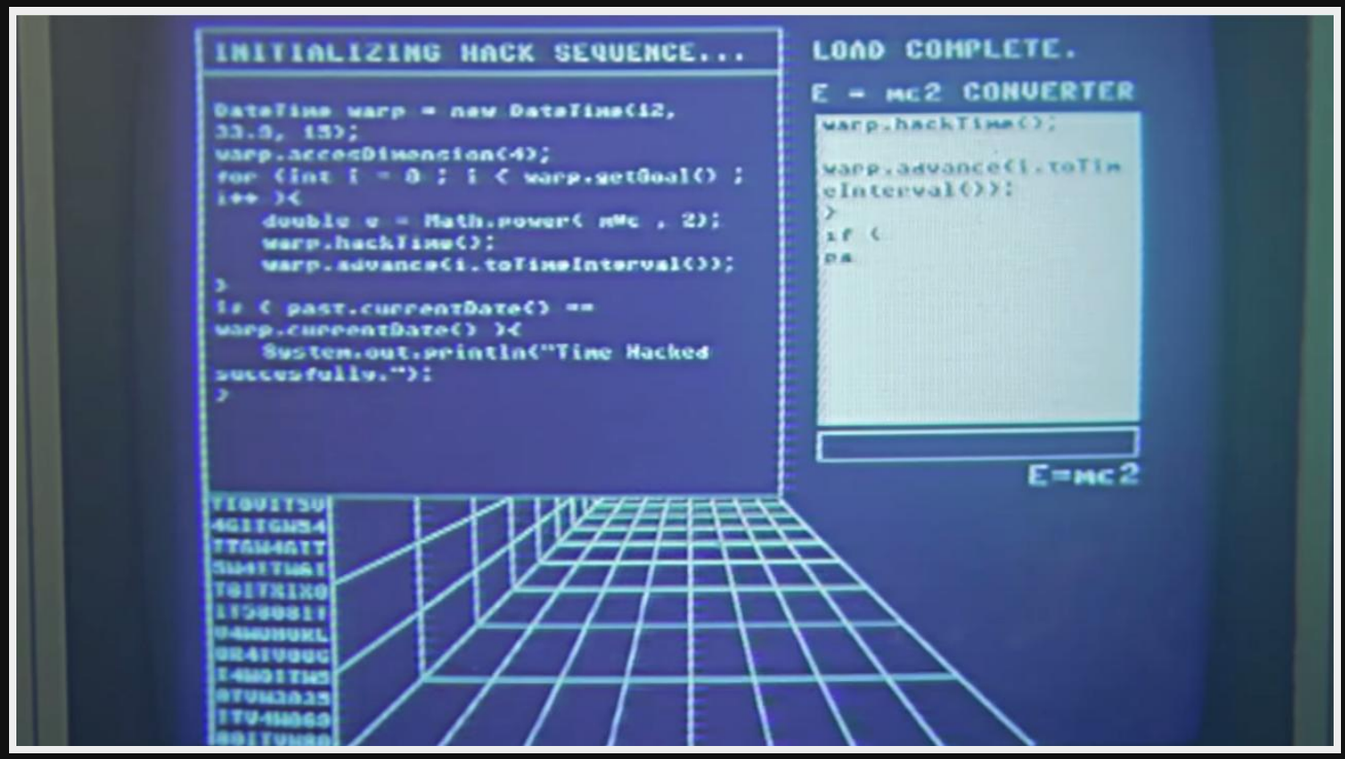



```
var MovieScreen = React.createClass({
  render: function() {
    return (
      <ScrollView contentContainerStyle={styles.contentContainer}>
        <View style={styles.mainSection}>
          <Image
            source={getImageSource(this.props.movie, 'det')}
            style={styles.detailsImage} />
          <View style={styles.rightPane}>
```

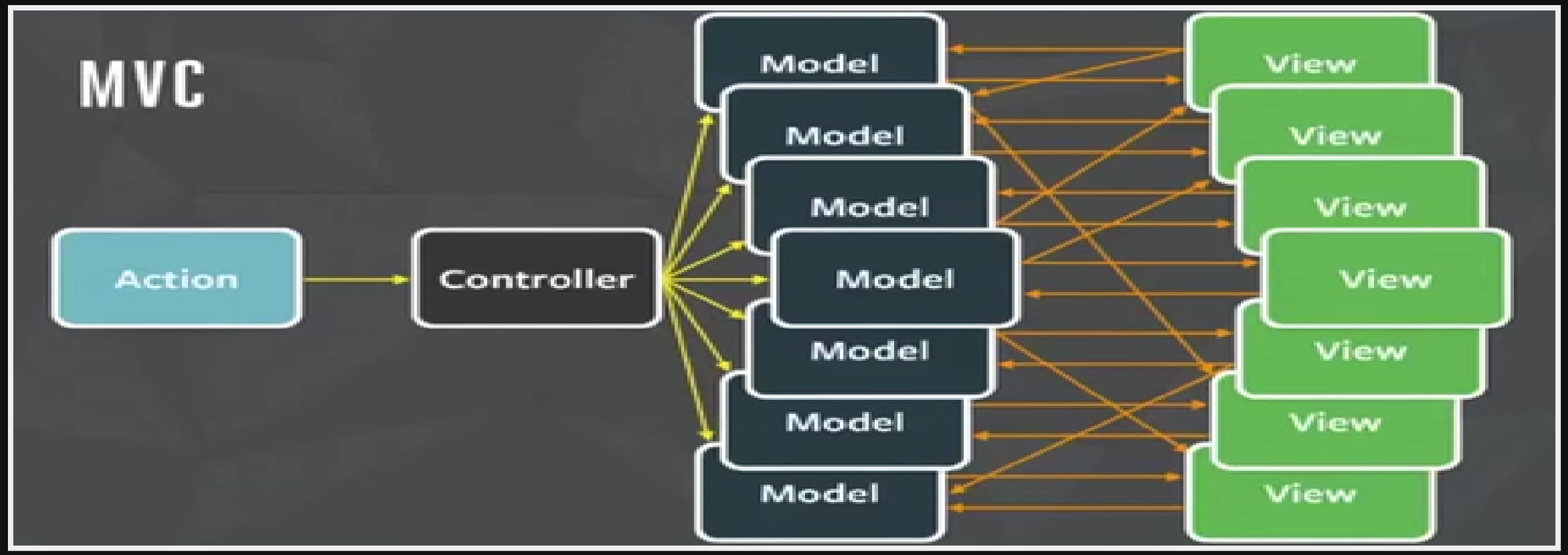
Three.js, really?

Terminal, really?

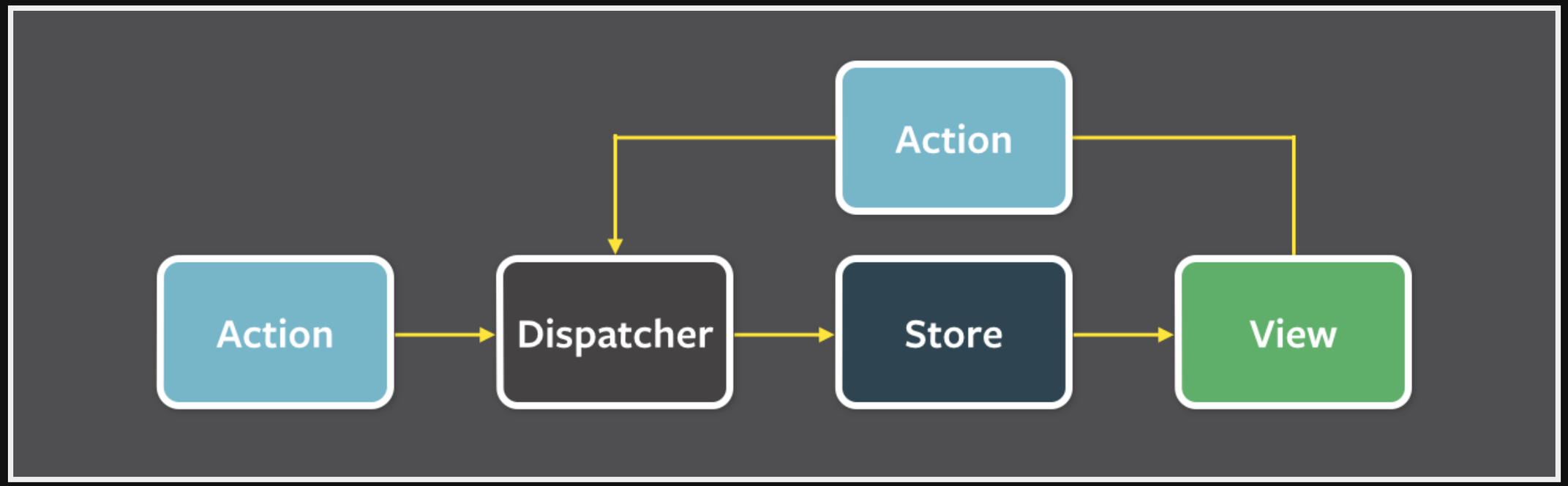
- YEAHHHH!!!



Traditional MVC



Flux / Redux



4. Flux / Redux

(`counter` example)

- All state of all components lives in a single JSON object.
- Views can trigger actions that create a new root state.
- Just like React components, Redux actions are pure functions.
- Pure functions can be reversed and re-applied at will.
- Start

Questions?

- @STRML_
- github.com/STRML