ClojureScript: I can't believe this is JavaScript

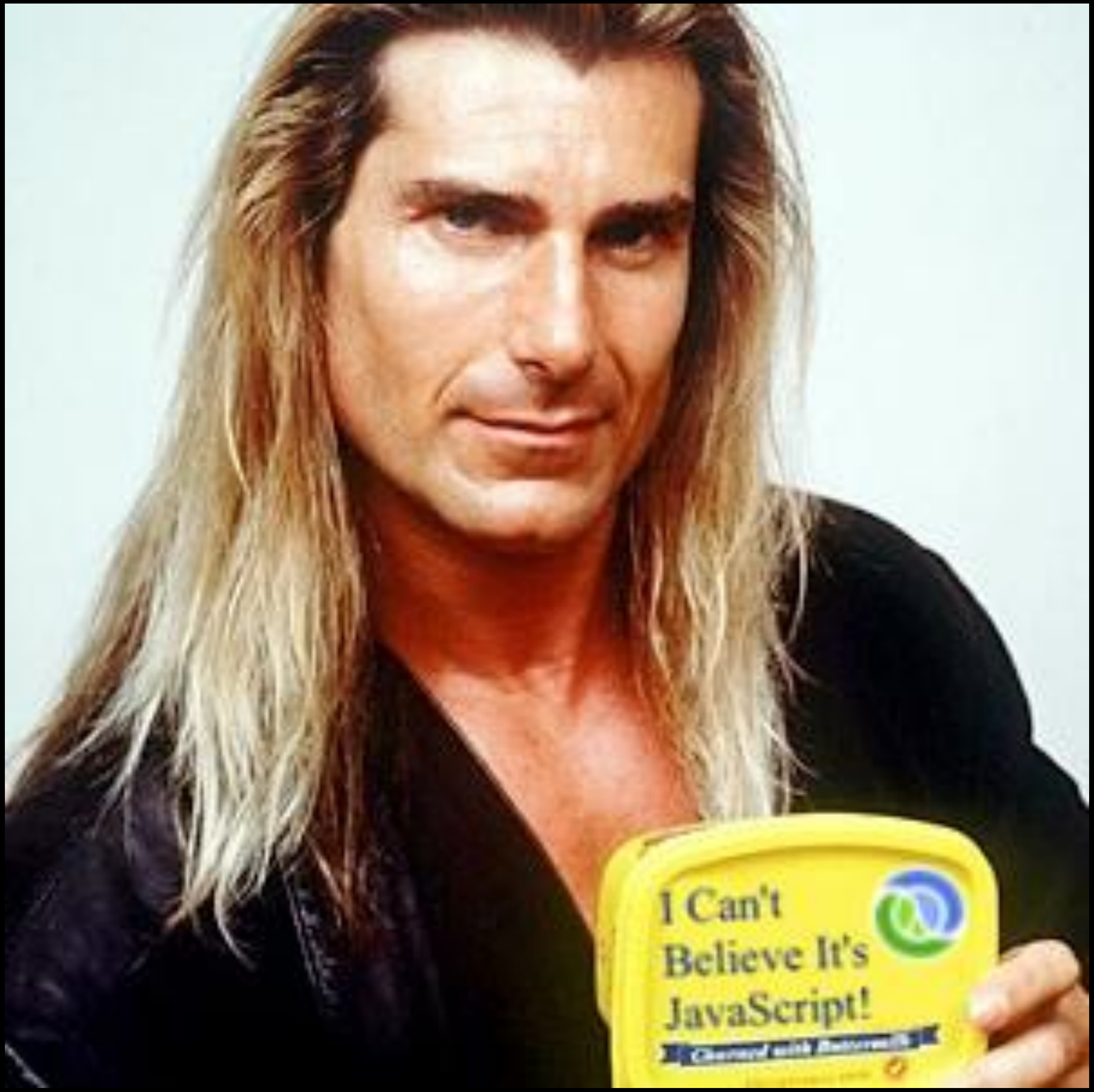
Eric Normand

PurelyFunctional.tv

Clojure transpiled to JavaScript

ClojureScript

I can't believe this is JavaScript!



For the video and transcript of this presentation, click here:

https://lispcast.com/clojurescript-cant-believe-javascript/

Callback Hell

Callback Hell

- JavaScript uses callback-style for async
- Not just about indentation
- About inversion of control
    - I don't know when or in what order *or even if* my code will be called!

JavaScript vs Callback hell

- Promises
  - Helps a lot!
  - Still have promises within promises within promises
- async/await
  - nice!
- Together: wow, a really good solution!

ClojureScript vs Callback Hell

- core.async
  - A library that turns sequential code into callbacks
  - Never give up control
  - Based on Communicating Sequential Processes
    - (the same thing Go is based on)

Code Optimization

Code Optimization

- Milliseconds === $$$$
- Minify code
- Remove dead code
- Split code

JavaScript vs Optimization

- webpack
  - minification
  - dead code elimination
  - splitting
- Rollup
  - tree shaking

ClojureScript vs Optimization

- Google Closure Compiler
  - heavily tested
  - optimizes Gmail
  - "just works"

Stateful DOM

Stateful DOM

- Dynamic apps need to modify the DOM
- DOM and your data get out of sync

JavaScript vs DOM

- JavaScript sprinkles
  - Just use JS for making small improvements to a static page
- Angular/Ember
  - Make a new HTML that is responsive to changes to state
- React
  - Build new components that manage their view state
- Vue

ClojureScript vs DOM

- React
  - Re-frame, Om, Rum — Basically just the Virtual DOM
  - Manage app state in one place, View in another
  - Really fast
  - Very functional
- Hoplon
  - Spreadsheet cells for changes

Application State Management

Application State

- Applications have a lot data to keep track of
- It's hard to tell what has changed and when
- It's hard to keep in synch with the server

JavaScript vs State

- Mutable objects and variables
- Component state
- Redux
  - Immutable.js
- GraphQL
- Falcor

ClojureScript vs State

- One mutable variable with immutable data
- Re-frame + Om Next
  - Redux was inspired by these
  - Manage their own “databases”
  - Encourage a “pull model” with the server

Build tool fatigue

Build tool fatigue

- There’s a lot of stuff a modern project needs
  - linting
  - transpiling
  - dependency management
  - bundling

JavaScript vs Tools

- Many options
  - Webpack
  - Grunt
  - Gulp
  - Browserify
  - NPM
  - Rollup
- These are just the bundlers!!
- No clear winner
- Broken promises
- Bad documentation

ClojureScript vs Tools

- Leiningen
    - Configuration / declarative
    - Plugins
- Boot
    - Procedural / task-based
    - Libraries

Development flow

Development flow

- Web development is very visual
- We want to see changes right away

JavaScript vs Flow

- “Plain” flow
  - Save changes to JS + CSS.
  - Watcher recompiles on change.
  - Reload the browser.
  - Click around to where you were in your app.
- Hot Module Reloading
  - Save changes to JS + CSS.
  - Watcher recompiles on change.
  - Watcher sends new code to browser.
  - Works for some applications.

ClojureScript vs Flow

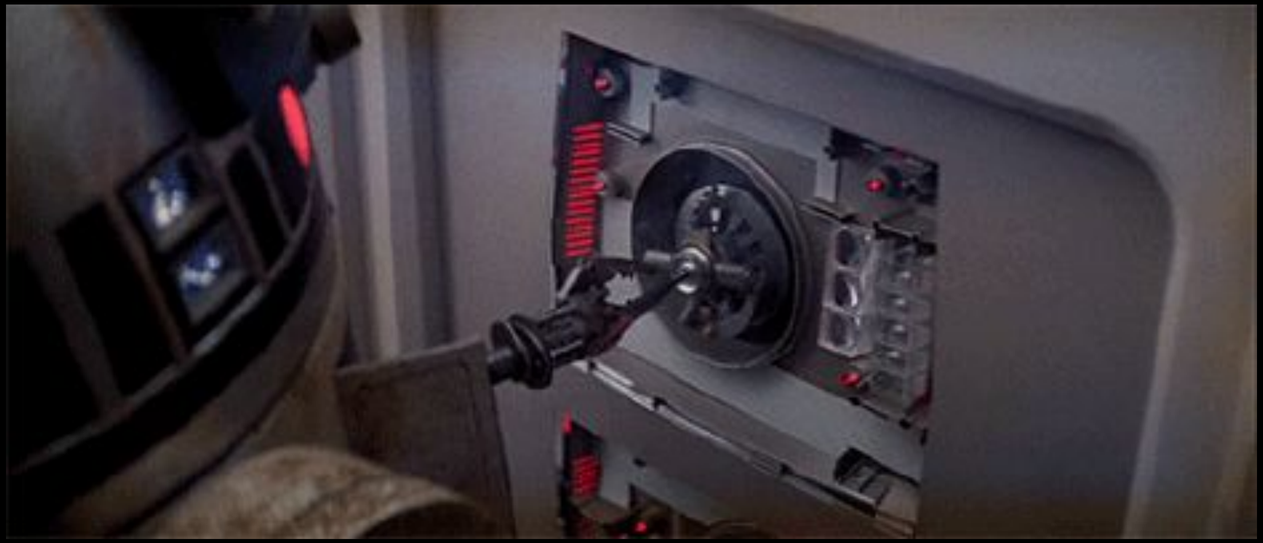
- Figwheel
  - Compiles ClojureScript and sends it to browser
  - See changes in less than a second
  - Application state is untouched

Extra slides

Embrace the host

Built-in

Platforms



React

Other Goodies

Module Systems

core.async

EDN

Google Closure Compiler

Eric Normand

LispCast

Follow Eric on:

Eric Normand @EricNormand

lispcast.com eric@lispcast.com