Imba

or why React is a bad React

WTF is Imba

React in Coffeescript (*Kinda*)

- Except 10x Better
- And 100x Faster

Before React - Javascript

Javascript was 💩

- Most people coded in "jQuery Language"

Coffeescript took kitchen sink approach

- Big improvement
- At what cost?

Before React - Frameworks

- New framework every week
- Did you know there are two js frameworks called svelte?
  - svelte.dev
  - sveltejs.com
- 50 Shades of Model View Controller
  - Real Models require OOP
  - Real Controller lives on the server
  - View is the only part that makes sense

React Won Framework Wars

- Dropped MVC
- Very simple with minimal learning curve
- Focused on simpler problem of doing Views well
- Embraced transpilation
  - Without going full Coffeescript
  - ES6 took half of Coffeescript and made JS tolerable
- Endorsed by Facebook
- Angular committed suicide on camera
  - It was absolutely glorious and unprecedented in history of tech

Reactive Paradigm

- Tree of Components
  - Some HTML elements
  - Some application specific
- Each component declaratively describes its children
- Props go down
- Events go up
- Some Components have State
- Nothing about Models and Controllers here
- Sprinkling of transpilation to make `render` functions look nice

React is a bad React

If this is what React is trying to do, it's failing hard.

Events go up

Good way:

```
this.trigger("setTheme", {theme: "dark"})
```

Browsers have perfectly good event bubbling.

```
onSetTheme(event, payload) { ... }
```

Events go up

React way:

- React was created during JS callback insanity period
- Embraced callback hell
- JS moved to promises, `async`/`await` etc.

It fails so hard:

- Extremely verbose
- Any element in between needs a lot of `onfoo=this.props.onfoo`
- Many people give up and just use global state with Redux/MobX etc.
  - `.bind(this)` for extra 💩 on top

Angularization of React

Good way:

- Very simple with minimal learning curve
- *If you have to write React, I recommend sticking to classic style with classes*

React way:

- 100+ ways to write components
- New ways added every release
- Latest are hooks, but more coming for sure
- Initial simplicity is all gone
- Redux ecosystem makes it order of magnitude worse

Components have State

Good way:

```
this.state.todos[index].done = true
```

React way:

```
this.setState(prevState => ({
  todos: [
    ...prevState.todos.slice(0, index),
    {...prevState.todos[index], done: true},
    ...prevState.todos.slice(index + 1),
  ]
}))
```

This is not even standard JS.

`immerjs` makes it somewhat tolerable

Components have State

Good way:

- Local state on relevant components

Redux way:

- Ridiculous amount of extra complexity
- To hide that it's all basically global variables

`render` functions

Good way:

```
if (todos.length === 0) {
  <div>Nothing to do</div>
} else {
  <ul>
    for(let item in todos) {
      <li>{ item }</li>
    }
  </ul>
}
```

React way:

```
{ (todos.length === 0) ? (
  <div>Nothing to do</div>
  ) : (
  <ul>{ todos.map(item => (
    <li>{ item }</li>
  ))}</ul>)
}
```

`render` functions

Beyond very simple cases, turns into hellscape of

- `{{}}`
- `(())`
- `?:`
- `=>`
- `.bind(this)`

Contender for the worst looking templating language ever.

React Performance

React is "fast enough", i.e. fairly slow.

Two DOM trees and careful synchronization between them.

Fake immutability makes it a lot slower.

Redux makes it a lot slower.

Imba

Sort of:

- React-style framework
- on top of Coffeescript-like language

Solves all problems mentioned.

At what cost?

Imba Hello World

```
tag App
  def render
    <self>
      <header>
        "Hello, world!"

Imba.mount <App>
```

Imba Performance

Imba is *hilariously* fast.

Imba doesn't bother tracking state.

Imba just rerenders the whole page on *every browser event*.

Single tree of memoized DOM.

All video games render this way.

Imba Events

- Bubble up like normal browser events
- Zero callback hell
- `trigger` emits event
- `onX` catches them or they keep bubbling up
- Or handle inline inside `render` function
- Works for builtin and custom events

```
tag CellTag < svg:g
  def onclick
    data:state = !data:state
    trigger("pause")
```

```
<button :tap.pause>
  "Pause"
```

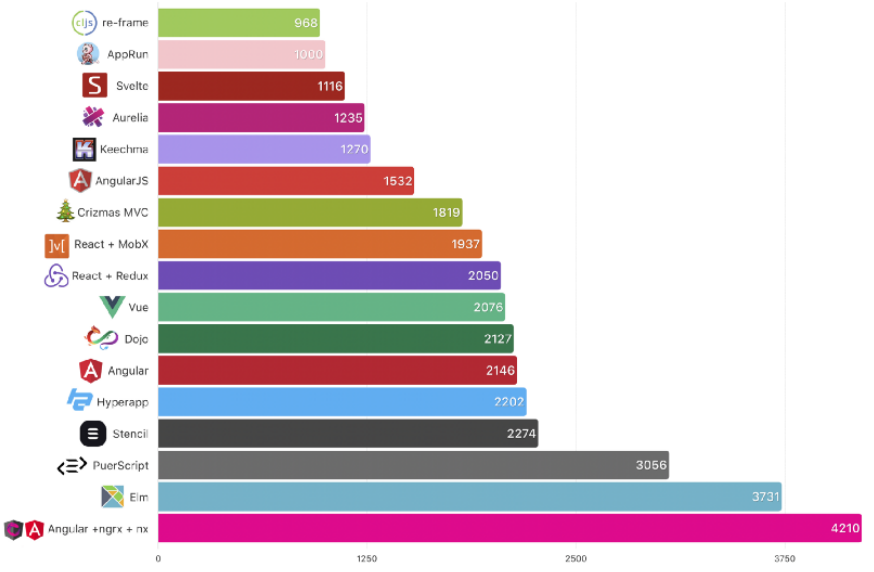
Imba `render` function

```
def render
  <self.todoList>
    <h2>
      title
    <form.todo :submit.addTodo>
      <input[@newTodo] placeholder="What's next?">
      <button type="submit">
        "ADD"
    <ul>
      for item in data
        <TodoItem[item]>
    if anyDoneItems
      <button :tap.cleanupDone>
        "Clean up DONE"
```

Comparison

From "A RealWorld Comparison of Front-End Frameworks with Benchmarks (2019 update)"

Imba 627 LOC.



But at what cost?

Coffeescript variant, not even plain Coffeescript.

Own class system at its core.

Going very far with things like : vs . etc.

Lacks many ES5/ES6 features like getters/setters, destructuring etc.

A lot of minor issues.

Future

My perfect scenario would be someone creating something Imba-like on top of ES6 class system.

Alternatives

Coffeescript 2 + React

- A bit more mainstream language
- `render()` looks nicer
- All other React problems

Opal + Hyperstack (basically React)

- Opal is 90% like regular Ruby
- All other React problems
- Ruby - JS interop is hard

Other Memoized DOM (glimmerjs)

- Separate templating language for views
- ES6 for everything else

Imba 2

- experimental branch
- a few steps towards ES6

Imba Examples

https://github.com/taw/imba-examples