Hello!

I’m Emily.

Front-end at Medean
(https://beta.medean.com)
Full-stack web
Mobile iOS
https://emkolar.ninja

MIGRATING TO REACT 16

- Why we decided to do it at Medean
- Getting started and necessary changes
- Gotchas you may run into
- Upgrading Enzyme and updating tests

What did we consider before updating?

👍 Pros

- Performance boosts
- More robust error handling
- Rad new and improved features, such as:
  - Portals
  - Render can return multiple elements
  - Async rendering (in the future)
- We can all sue Facebook now

👎 Cons

- What's the actual time to implement?
- Concerns with using any new library version
- At least 1 dependency was using React.PropTypes
- How to incorporate those rad new features?
- Probably we don't need to sue Facebook yet

❤ Biggest Actual Payoffs

Debugging with error boundaries

Smoother JS animations/transitions

Better knowledge of the library overall

Biggest surprise payoff? **Portals**.

(We knew portals would be available, but never thought we'd have an immediate, urgent need for them.)

THE BEAR NECESSITIES

Breaking Changes

Handling of runtime errors

Calling setState with null

Support is gone for:

- `React.PropTypes`
- `React.createClass`
- `React lib internals`

React 16 depends on:

- Collection types `Map` and `Set`
- `requestAnimationFrame`

Handling errors

Previously:

Runtime errors broke React

Strange error messages in this state

Now:

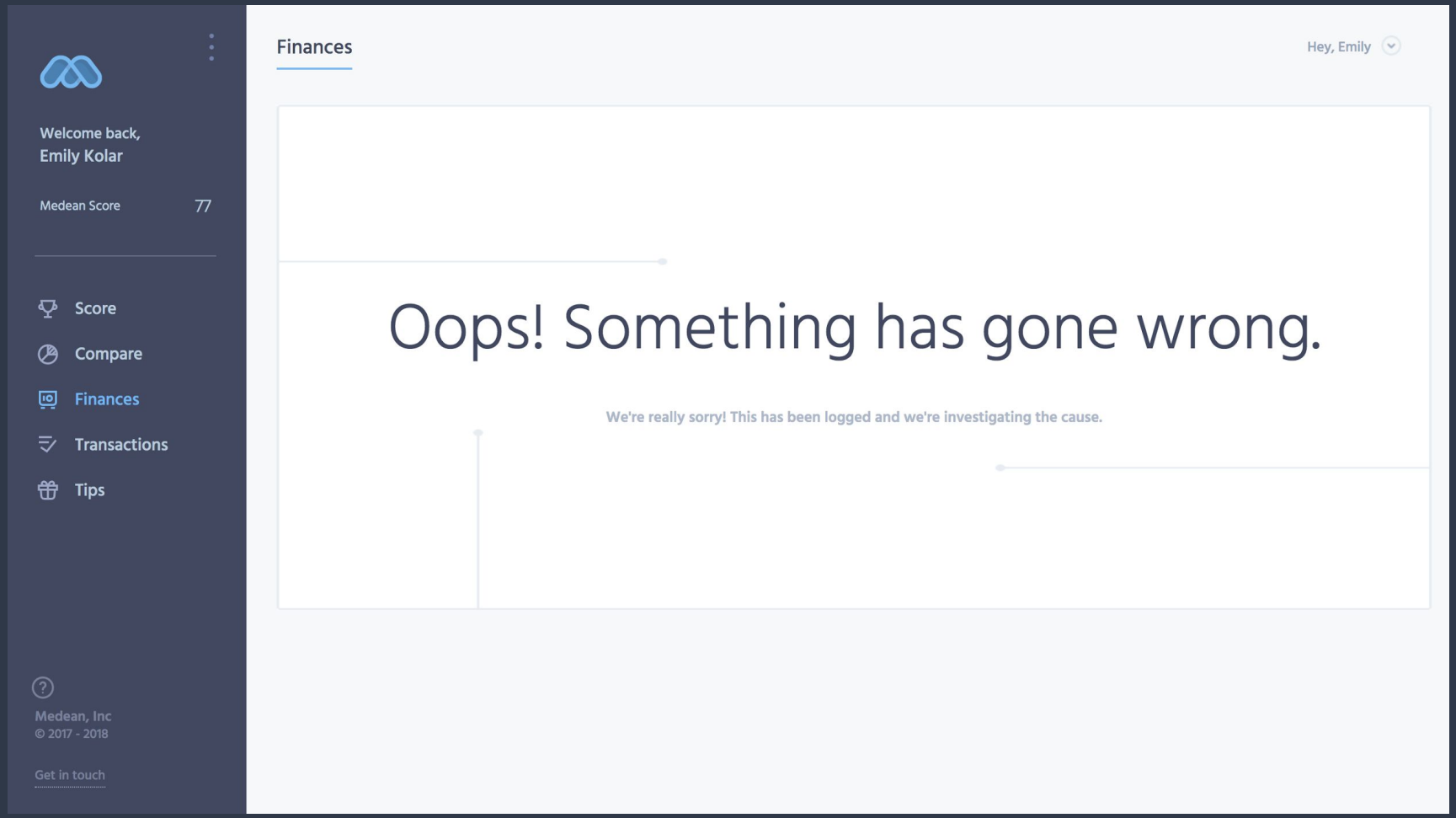
React will fallback to Error Boundaries

If none exist, unmounts the whole component tree

```
componentDidCatch(error, errorInfo) {
    this.setState({
        error,
        errorInfo,
    });
}

render() {
    if (this.state.error) return <MyErrorComponent />;

    return (
        <div>{this.props.children}</div>
    );
}
```


Welcome back,
Emily Kolar
Medean Score
77
Score
Compare
Finances
Transactions
Tips
Medean, Inc
© 2017 - 2018
Get in touch
Finances
Hey, Emily
Oops! Something has gone wrong.
We're really sorry! This has been logged and we're investigating the cause.

`setState()`

Previously:

Calling `setState({ foo: null })` would trigger an update

Calling `setState` from render didn't always trigger updates

Now:

Calling `setState({ foo: null })` will NOT trigger updates

Calling `setState` from render always triggers an update

PropTypes and createClass

React.PropTypes and React.createClass have moved to external packages, `prop-types` and `create-react-class`

Change these by hand, or use the codemod!

(Word on the street is you'll still have to resort to find/replace for a handful of things afterward)

To use the codemod, install:

https://github.com/facebook/jscodeshift

https://github.com/reactjs/react-codemod

```
# move to prop-types

jscodeshift -t
./react-codemod/transforms/React-PropTypes-to-prop-types.js
/path/to/your/repo

# move to Component, PureComponent, createReactClass

jscodeshift -t ./react-codemod/transforms/class.js
--mixin-module-name=react-addons-pure-render-mixin
--flow=true --pure-component=true
--remove-runtime-proptypes=false /path/to/your/repo
```

Credit and explanation:
https://blog.discordapp.com/lessons-from-migrating-a-large-codebase-to-react-16-e60e49102aa6

Map, Set, requestAnimationFrame

Older browsers (IE <11, Safari <8, etc) will require polyfills.

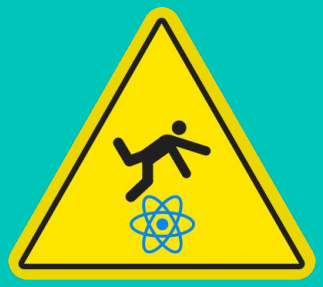
Two good options for this:

https://github.com/zloirock/core-js
https://babeljs.io/docs/usage/polyfill/

```
// app.js polyfills (we use babel)
import 'babel-polyfill';


// simple requestAnimationFrame shim in test
global.requestAnimationFrame = (callback) => {
    setTimeout(callback, 0);
};


 // package.json
 "jest": {
     "setupFiles": [
         "path/to/your/raf/shim"
     ]
 }
```



POTENTIAL BLOCKERS

Dependencies

Dependent on packages that use deprecated code?

- For actively maintained packages, it may be worth it to wait for the maintainers to update before upgrading, or to open a PR if it's reasonable to spend your time on those changes.
- For less active or totally **un**maintained packages, replace the dependency if you can (easier said than done), or fork the repo

(We currently maintain our own fork of the sematable package, for this and other reasons.)

Older React Internals

Using undocumented internals or private APIs from `react/lib/*`, `react-dom/lib/*`?

- Some are accessible in other forms, such as:

  `react-addons-test-utils` under `react-dom/test-utils`, `React.DOM` as `react-dom-factories`, shallow renderer under `react-test-renderer/shallow`

- Others are **just gone**. 😓 For these, the React devs have often recommended inlining (literally copying and pasting) the needed functions into your own code.

ENZYME V3

Why switch to v3?

Other options: **0**

Enzyme v2 is not compatible with React 16.

What's new?

v3 is a near-total rewrite of Enzyme's internals.
We were impacted mainly by:

- Required configuration of an adapter
- Referential identity no longer preserved
- `children()` means something different
- Updates to `mount()`

Adapter System

Implemented to allow future compatibility with React-like libraries (Inferno, Preact, Riot, etc)

Installed as separate modules. For React 16: `enzyme-adapter-react-16`

If using with jest, the config script can be included in package.json as a setup file

```
// react 16 adapter config

import Enzyme from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

Enzyme.configure({ adapter: new Adapter() });
```

Referential Identity

Previously:

An imported constant could be seen as a duplicate when rendered twice into the tree. This resulted in its removal from elements matching a given selector.

Now:

Enzyme keeps a separate reference to each rendered instance.

```
const Buttons = {
    play: <Button className="play" />,
    stop: <Button className="stop" />,
};

const ActionButton = ({ code, label }) => (
    <div>
        {label}
        {Buttons[code]}
    </div>
);

const wrapper = mount(
    <div>
        <ActionButton code="play" label="Play" />
        <ActionButton code="stop" label="Stop" />
        <ActionButton code="play" label="Resume" />
    </div>
);

console.log(wrapper.find(Button).length); // => 2 (2.x)
console.log(wrapper.find(Button).length); // => 3 (3.x)
```

children() now has a slightly different meaning

https://github.com/airbnb/enzyme/blob/master/docs/guides/migration-from-2-to-3.md

```
const ActionLabel = ({ label }) => (
    <div>
        <span>{label}</span>
    </div>
);

const wrapper = mount(
    <div>
        <ActionLabel label="Play" />
    </div>
);

wrapper.find(ActionLabel).children().debug();
// (2.x) => <span>Play</span>

wrapper.find(ActionLabel).children().debug();
// (3.x) => <div> <span>Play</span> </div>
```

mount()

Enzyme no longer receives automatic updates to the rendered tree.

Result: You'll probably have to call `update()` more frequently on mounted wrappers.

`instance()` can be called at any level of the tree to return the component instance.

Result: Methods like `setState()` are now callable on nested children.

```
const wrapper = mount(<PlayButton />);

wrapper.text(); // => “Play”

wrapper.instance().togglePlaying();

wrapper.text(); // => “Play” (still)

wrapper.update();

wrapper.text(); // => “Pause”
```

Time until first meaningful app paint

1400ms => 1100ms

Debugging and component dev time saved over 4 weeks

~6 hours

Making front-end dev just a little less strange

Priceless?

Thanks! :)

Drop me a line anytime

at emily@medean.com