React Components

should (almost) always be stateful

Aaron Craig • 08.31.2017 • js.la

Who am I?

Staff Engineer at Rival
(don't ask)

Head up front end client dev

- React
- React Native
- + React Native Web

Metric driven development for code that is performant and easily maintained.

PS. we're hiring!

Metrics

User Experience

- Speed (actual & perceived)
- Consistent experience
- Reliable, etc

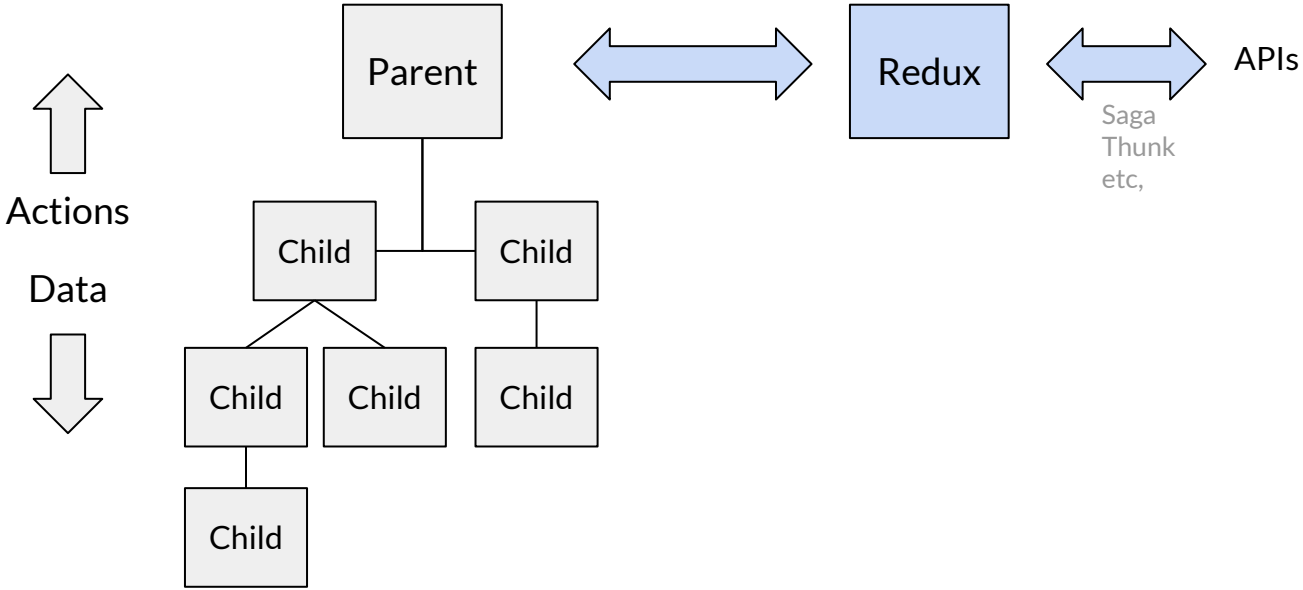
Code maintenance

- Ease of testing
- Ease of understanding
- Ease of extending
- Ease of composing

Are stateless/function components an anti-pattern?

```
const MyComponent = (props) => ( ... some jsx ... );
```

A typical React + redux flow



Issues with stateless

- It’s still a Component
- Useless rerenders of tree
- Brittle
- Violates separation of concerns
- Can be hard(er) to debug

What does Facebook say about them?

“The above two components are equivalent from React's point of view.”

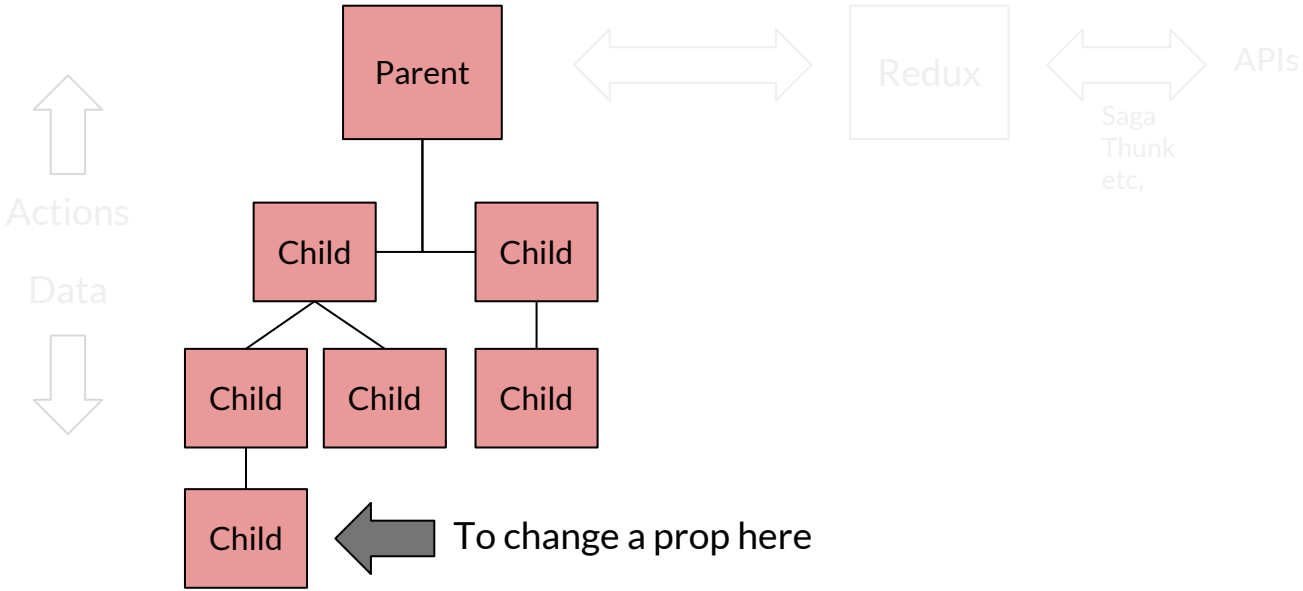
https://facebook.github.io/react/docs/components-and-props.html

Are stateless/function components an anti-pattern?

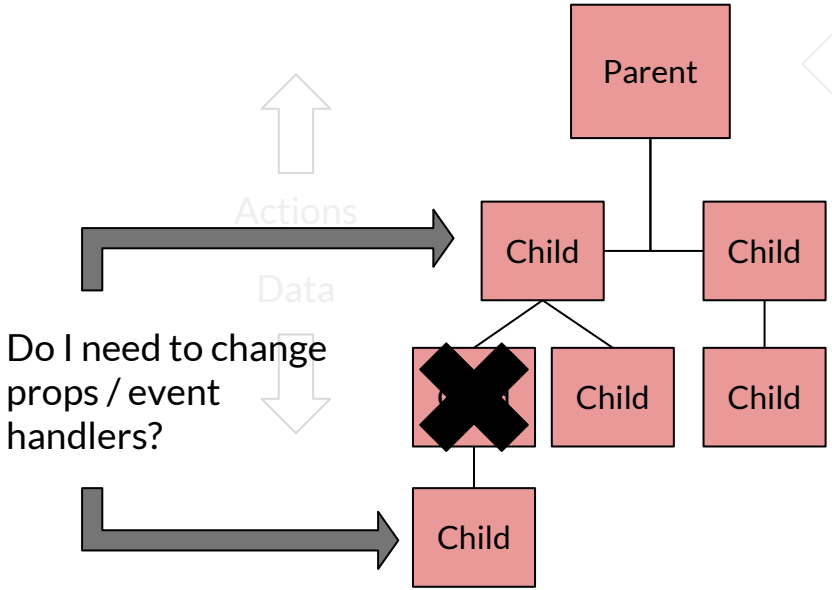
```
const MyComponent = (props) => ( ... some jsx ... );
                          =~
class MyComponent extends Component { ... }
```

Aside

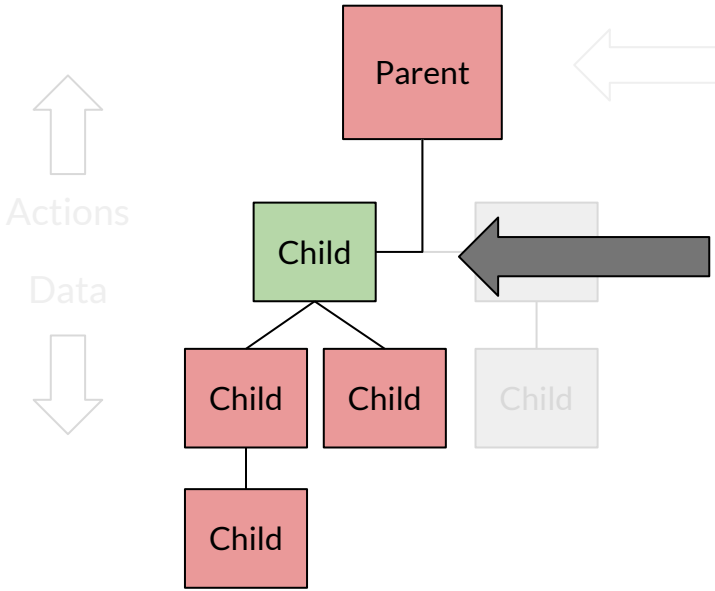
The stateless component render tree



The stateless component render tree



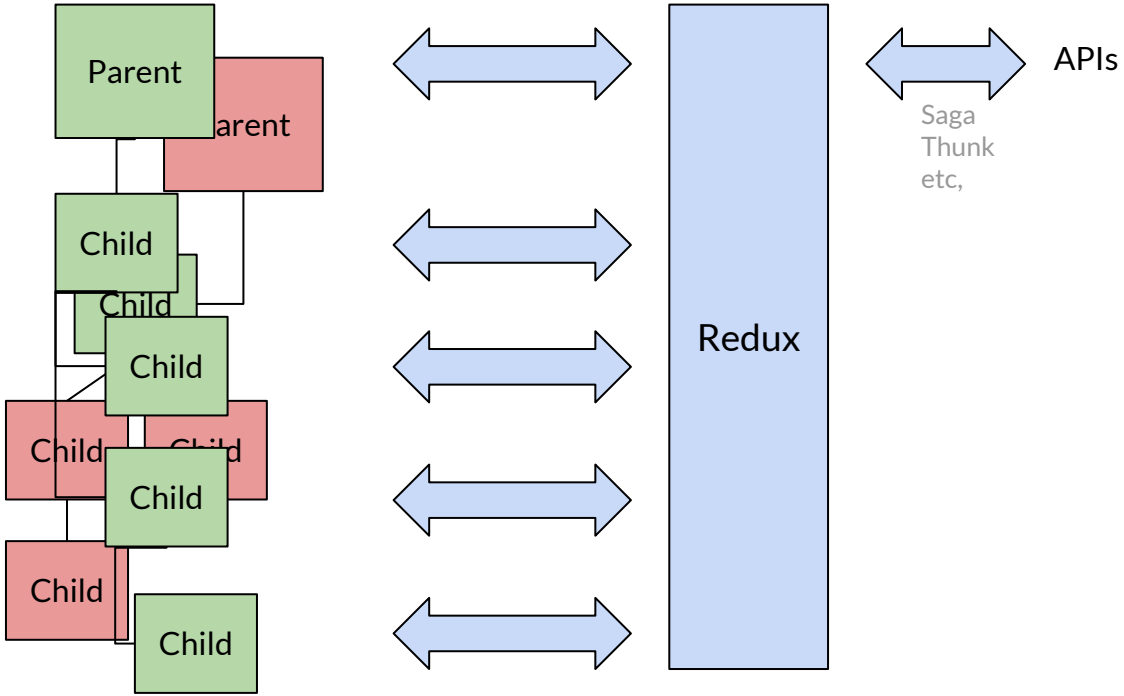
The stateless component render tree



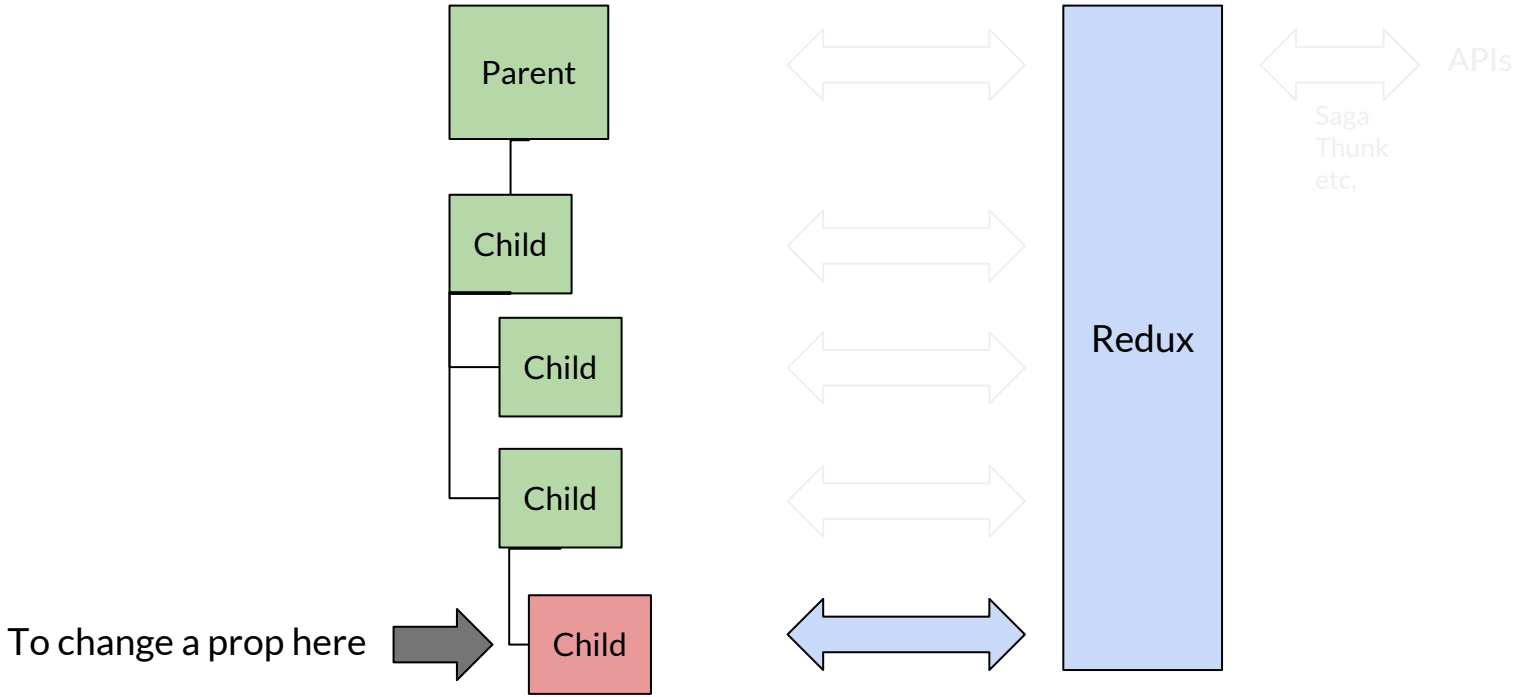
Must know about up/down stream

- data requirements
- event handlers

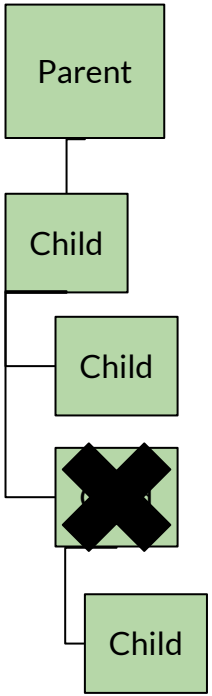
What if all the components were stateful?



What if all the components were stateful?

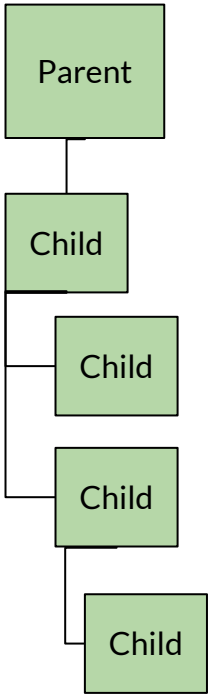


What if all the components were stateful?



No need to change props / event handlers

What if all the components were stateful?



Each component is an isolated concern

- Owns its data
- Can dispatch changes
- Autonomous, flexible

Benefits of stateful

- It’s a Component
- Access to shouldComponentUpdate
- Flexible
- Separation of concerns
- Easier to debug
- Easier to test

When to use stateless

End of branch components that are not dependent on app state

- Buttons
- Form elements
- Styled containers
- As the representational layer of an HOC

Final best practices

No heavy lifting in mapStateToProps or render()

Pre-process data in reducers (or even better, web services) so that mapStateToProps only grabs data from state (or maybe RxJS)

Only pass primitives in props

Use primitives to furnish components with keys to grab larger payloads from state.

Use shouldComponentUpdate or PureComponent

https://github.com/aaroncraigongithub/jsla-talk

Thanks for listening

- @aaroncraig
- aaroncraigongithub
- aaron@riv.al