

Several Components are Rendering

Client Performance at Slack-Scale

Jenna Zeigen
NY UX Speed & Performance Meetup
8/13/2024

Or, how Slack's made the
app perform... Swiftly

Senior Staff Software Engineer on Slack's Client Performance Infrastructure Team

jenna.is/at-ny-perf

@zeigenvector

**Senior Staff Software Engineer
on Slack's Client Performance Infrastructure Team**

**(Future) Software Engineer
on Notion's Web Performance Team**

jenna.is/at-ny-perf

@zeigenvector

Performance?!

What?

Make the app go fast! Reduce latency! Buttery smooth interactions! Get rid of jank! No dropped frames!

How?

Doing less work!

Why?

✨ So our users have a great experience! ✨

**First some stuff
about Slack**

Slack, a React app on your Desktop

The screenshot shows the Slack desktop application interface. On the left is the sidebar with team members (Matt Brewer, slackbot) and channels (#announcements, #design-crit, #media-and-pr, #social-media). The main area shows the #social-media channel with a message from Acme Team about a meeting starting in 15 minutes. Below it is a message from Harry Boone with a quick note about @Liza joining the sync. Lee Hao's message follows, mentioning meeting notes from the sync with @Liza. A file named "1/9 Meeting Notes" is attached. A note indicates Zenith Marketing is in the channel. The bottom of the screen shows the message input field and various text formatting and media icons.

Acme Inc

Matt Brewer

All unreads

Threads

Mentions & reactions

Drafts

Show more

Channels

announcements

design-crit

media-and-pr

social-media

Direct messages

slackbot

Matt Brewer (you)

Lee Hao, Sara Parras

#social-media ★

21 | 1 | Track and coordinate social media

Acme Team APP 12:45 PM

Event starting in 15 minutes:

Team Status Meeting

Today from 1:00 PM to 1:30 PM

Harry Boone 12:58 PM

Quick note: today @Liza will join our team sync to provide updates on the launch. If you have questions, bring 'em. See you all later... er, in 2 minutes 😅

Lee Hao 12:58 PM

Meeting notes from our sync with @Liza

Post

1/9 Meeting Notes

Last edited just now

Z Zenith Marketing is in this channel

Message #social-media

B I S </> C ½≡ ... Aa @ ☺ 0

Details

#social-media

Add Find Call More

About

Topic

Track and coordinate social media

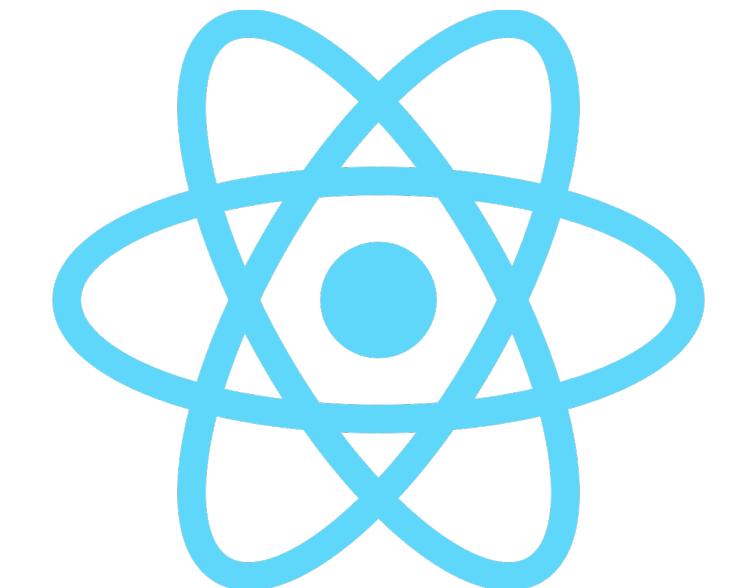
Description

Home of the social media team

Created on October 18th, 2019

Members 21 >

Organizations Z 2 >



Slack, a Long-Lived Single Page App

- ⚛️ Very long sessions
- ⚛️ No clear "page load" except at boot
- ⚛️ A lot of dynamic content
- ⚛️ Frequent updates via websocket

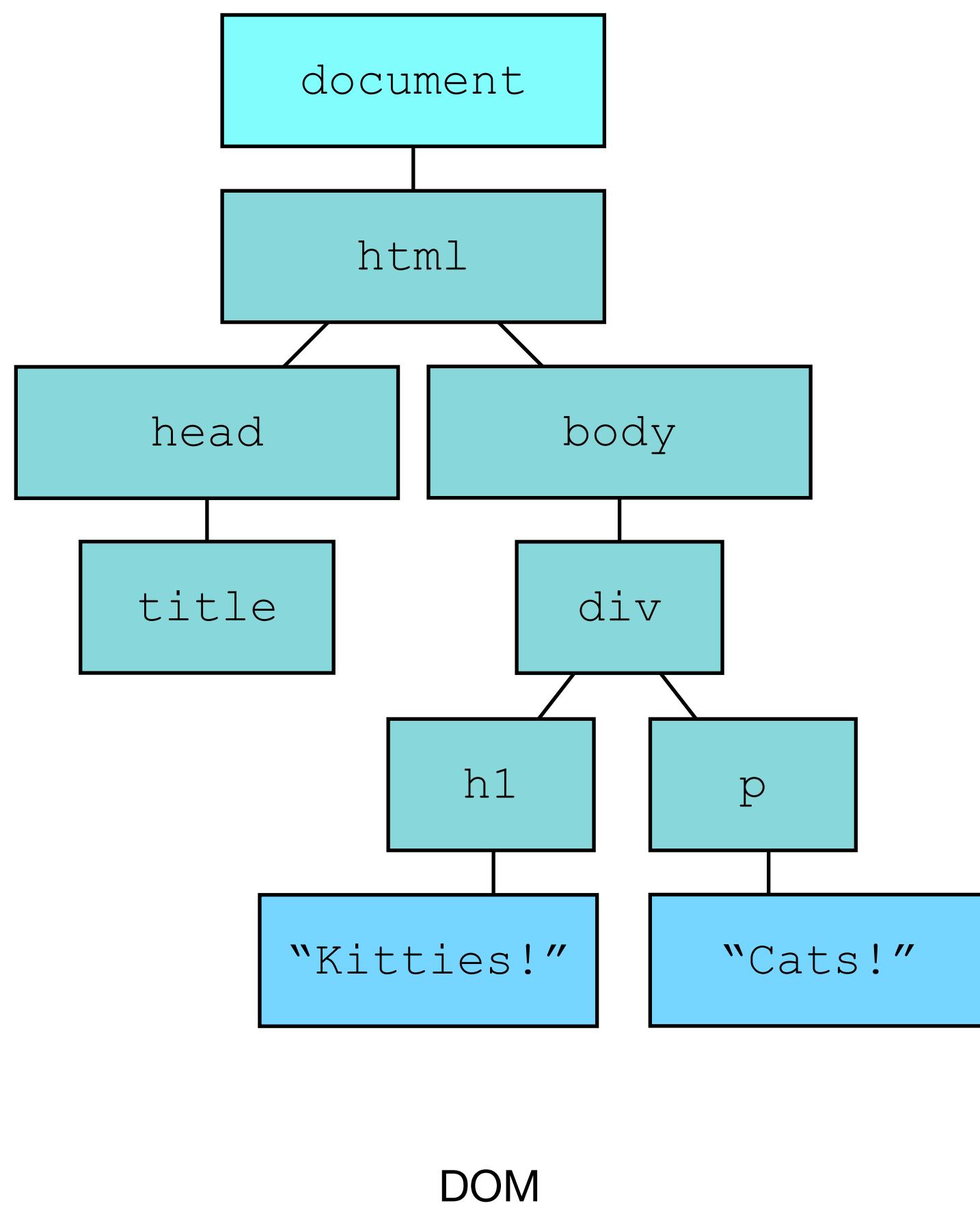
♪♪
And for a fortnight there, we were forever
Run JS sometimes, ask about the weather
Now you're in my channel, turned into coworkers
You want to ship the features
I want to measure
♪♪



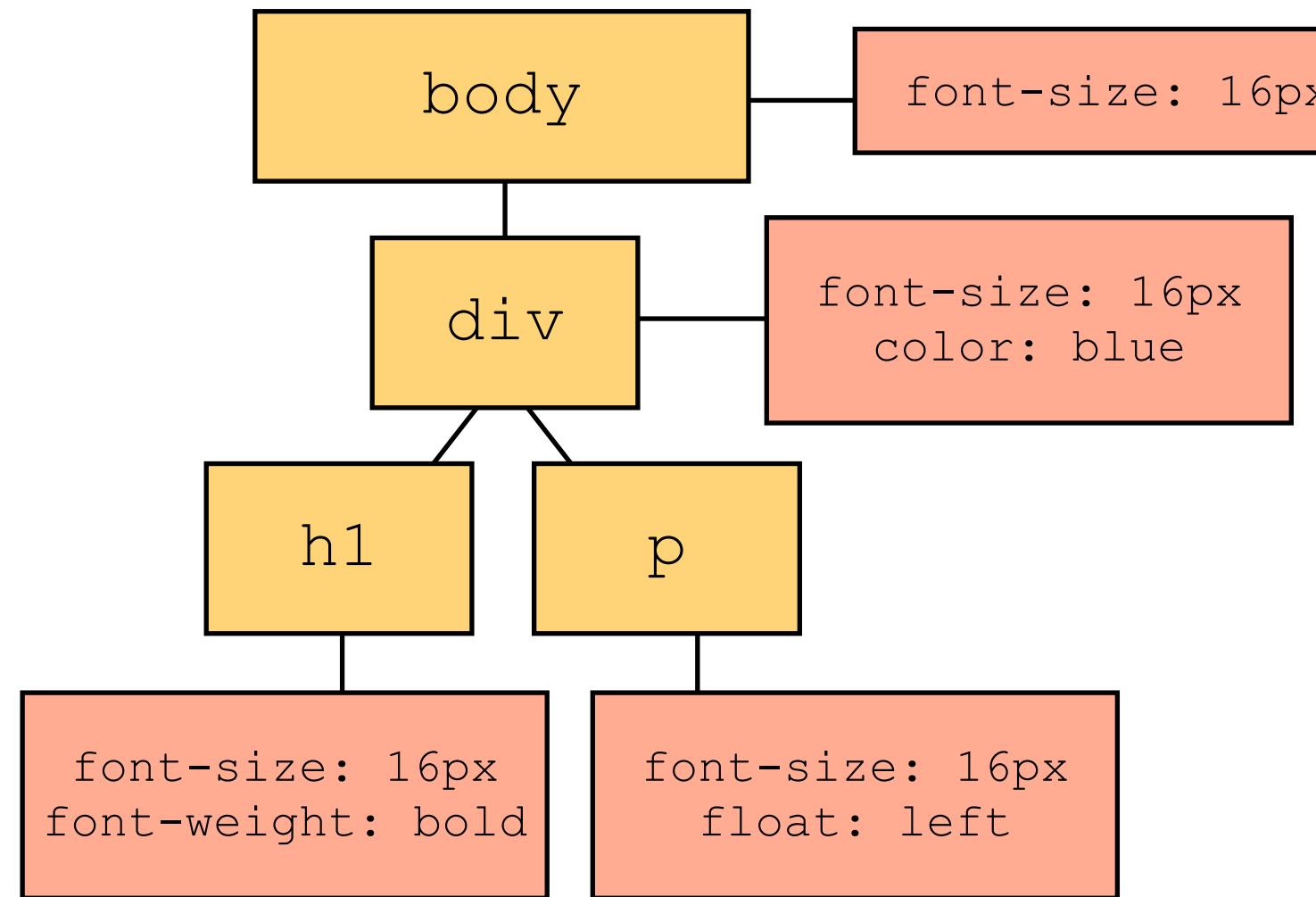
**Now, some stuff about
browsers**

How Do Browsers Even?

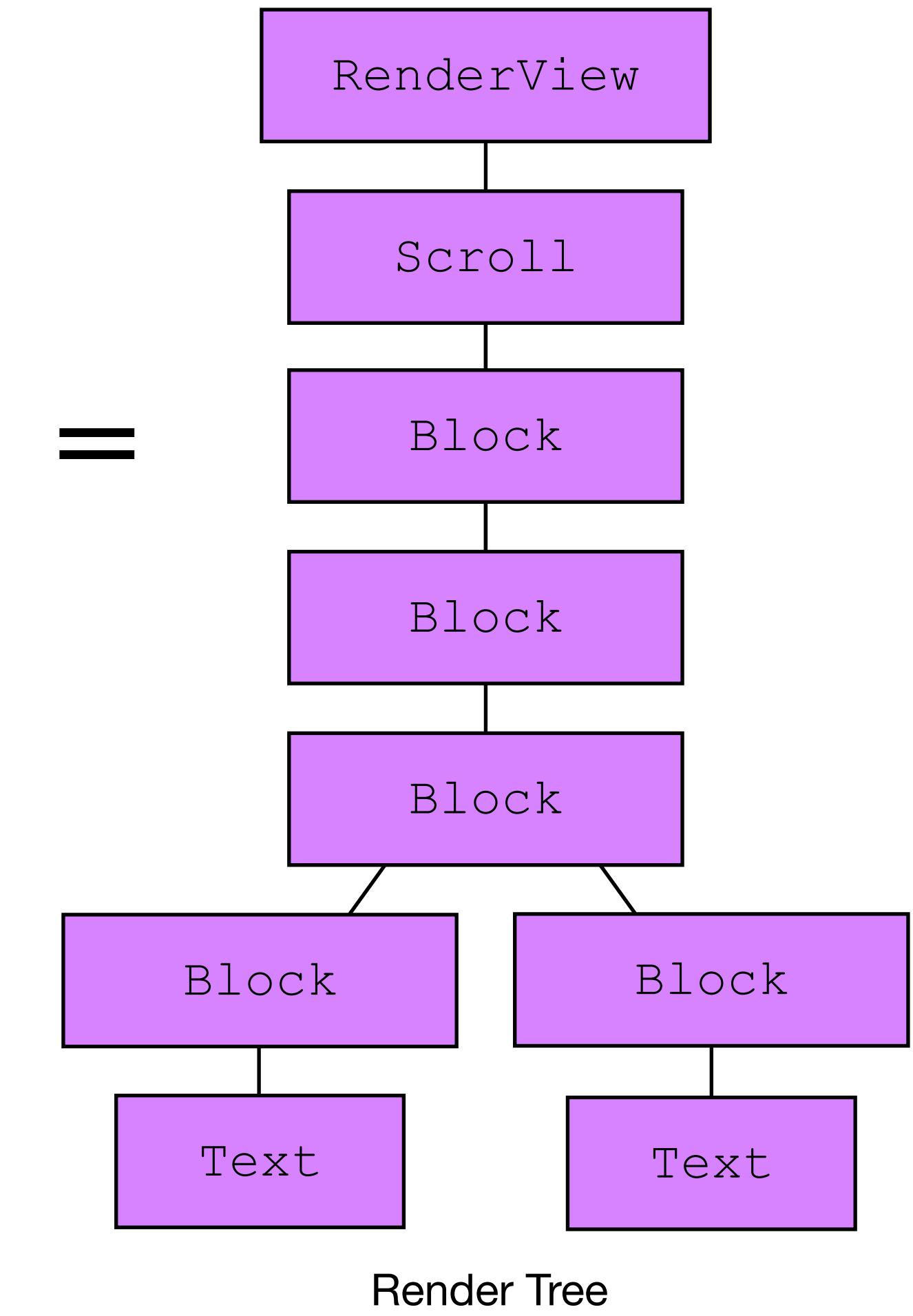
tl;dr you (might) have 16ms to do all your work before the next paint



+

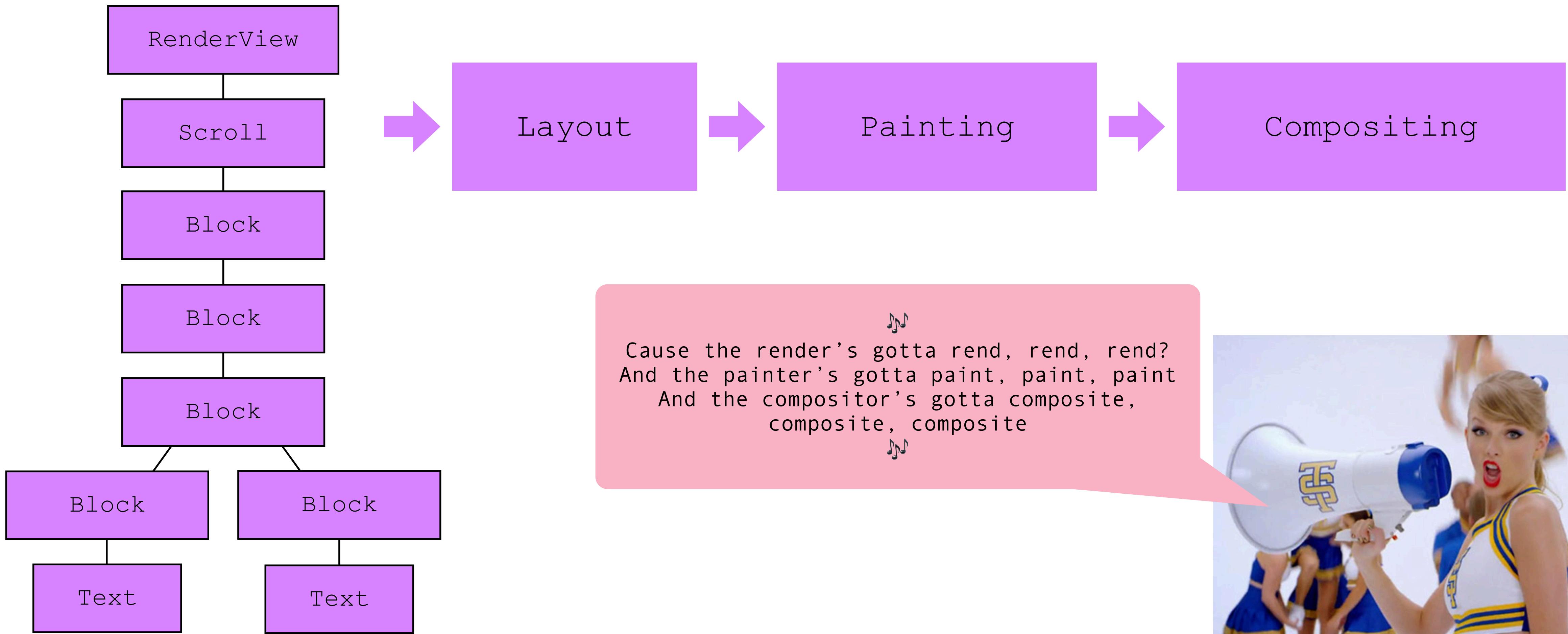


=



How Do Browsers Even?

tl;dr you (might) have 16ms to do all your work before the next paint



How Do Browsers Even?

tl;dr JavaScript is single threaded

- ⚛️ All JavaScript goes onto the call stack
 - ⚛️ Synchronous calls go right on
 - ⚛️ Async callbacks, i.e. event handlers, get thrown into a callback queue and are moved to the stack by the event loop once the stack is cleared
- ⚛️ The browser won't complete a repaint if there's anything on the JavaScript stack
- ⚛️ ⚡ **If your JavaScript takes longer than 16ms to run, you can end up with dropped frames and laggy inputs ⚡**

Performance, a UX Perspective

According to Google's RAIL model:

- ❖ Respond to user actions within 100ms, or they start feeling it
 - ❖ Make sure you process actions within 50ms to give time for other work
- ❖ Produce an animation frame in 16ms, or you drop frames and block the loop and animations start to feel choppy
 - ❖ Have the setup done in 10ms, since browsers need ~6ms to actually render the frame

♪
And I'm so furious
At you for making me feel this way
But, what can I say?
♪



Another Note About Frontend Performance

“In my experience the application is rarely reengineered unless the inefficiency is egregious and the fix is easy and obvious”

- Bob Wescott, *The Every Computer Performance Book*

✨ On the frontend, we're running code on other people's computers.

It's all re-engineering for us! ✨

♪
You don't know about me
But I'll bet you want to
Everything will be alright if
You just keep coding like I'm an M2 (jk)
♪

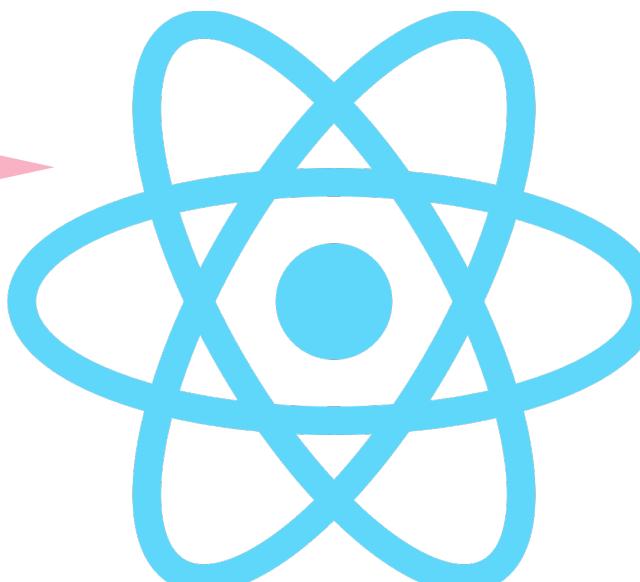


And now, a Primer on React and Redux

React and Redux 101

- ⚛ React is a popular, well-maintained, easy-to-use component-based UI framework that promotes modularity by letting engineers write their markup and JavaScript side-by-side
- ⚛ Components get data as “props” or store data in component state
- ⚛ Changes to props or component state cause components to re-render

Ask me what I learned from all those years
Ask me what I earned from all those tears
Ask me why so many fade, but I'm still here
(I'm still, I'm still here)



```
function Avatar({ person, size }) {  
  return (  
    <img  
      className="avatar"  
      src={getImageUrl(person)}  
      alt={person.name}  
      width={size}  
      height={size}  
    />  
  );  
}
```

```
<Avatar  
  size={100}  
  person={  
    name: 'Taylor Swift',  
    imageId: '1989'  
  }  
/>
```

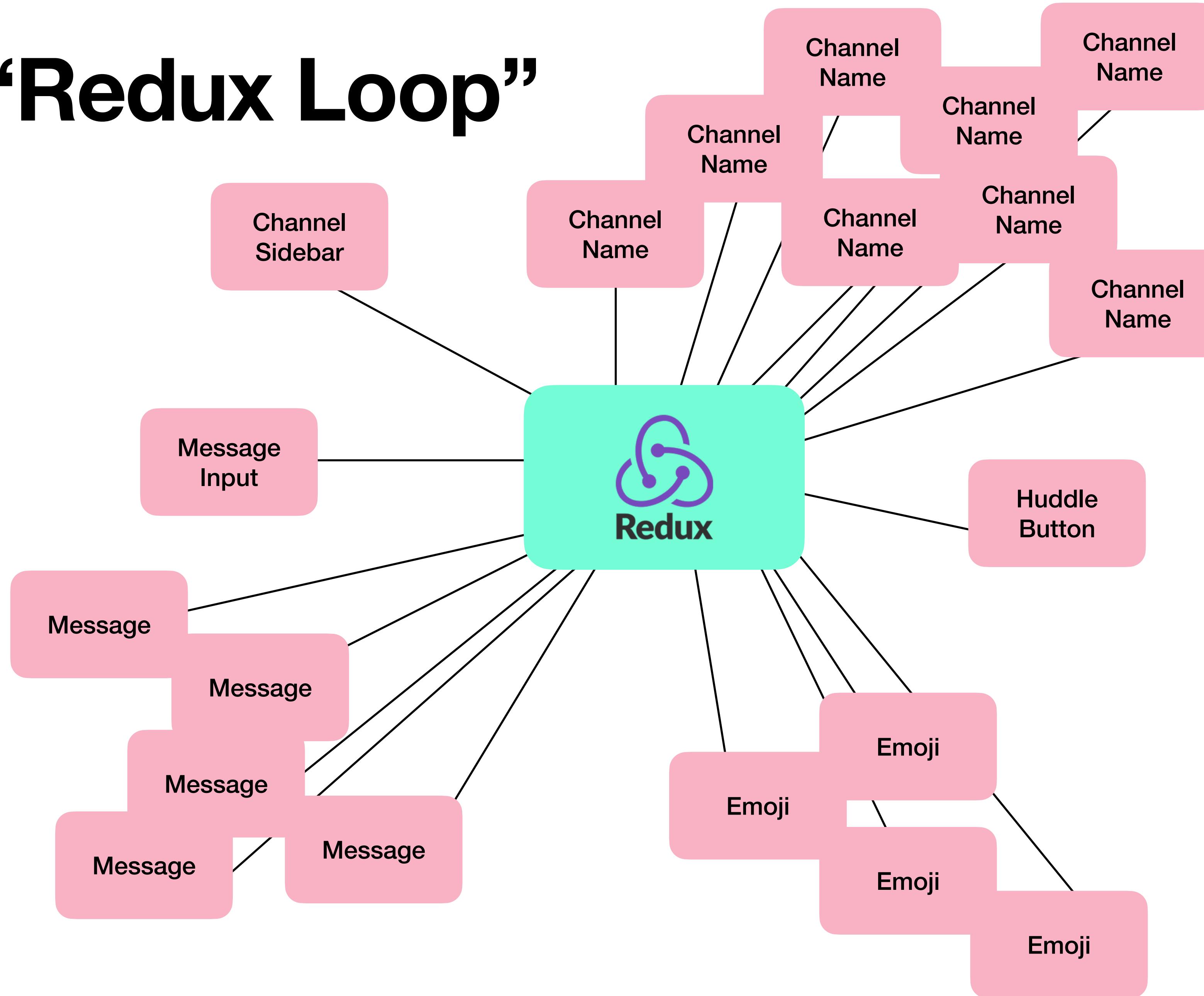
React and Redux 101

- ❖ **Redux** is a state-management library that can be used to supplement component state with a central store that components “connect” to
- ❖ Data is read from Redux via “selectors” which aide in computing connected props

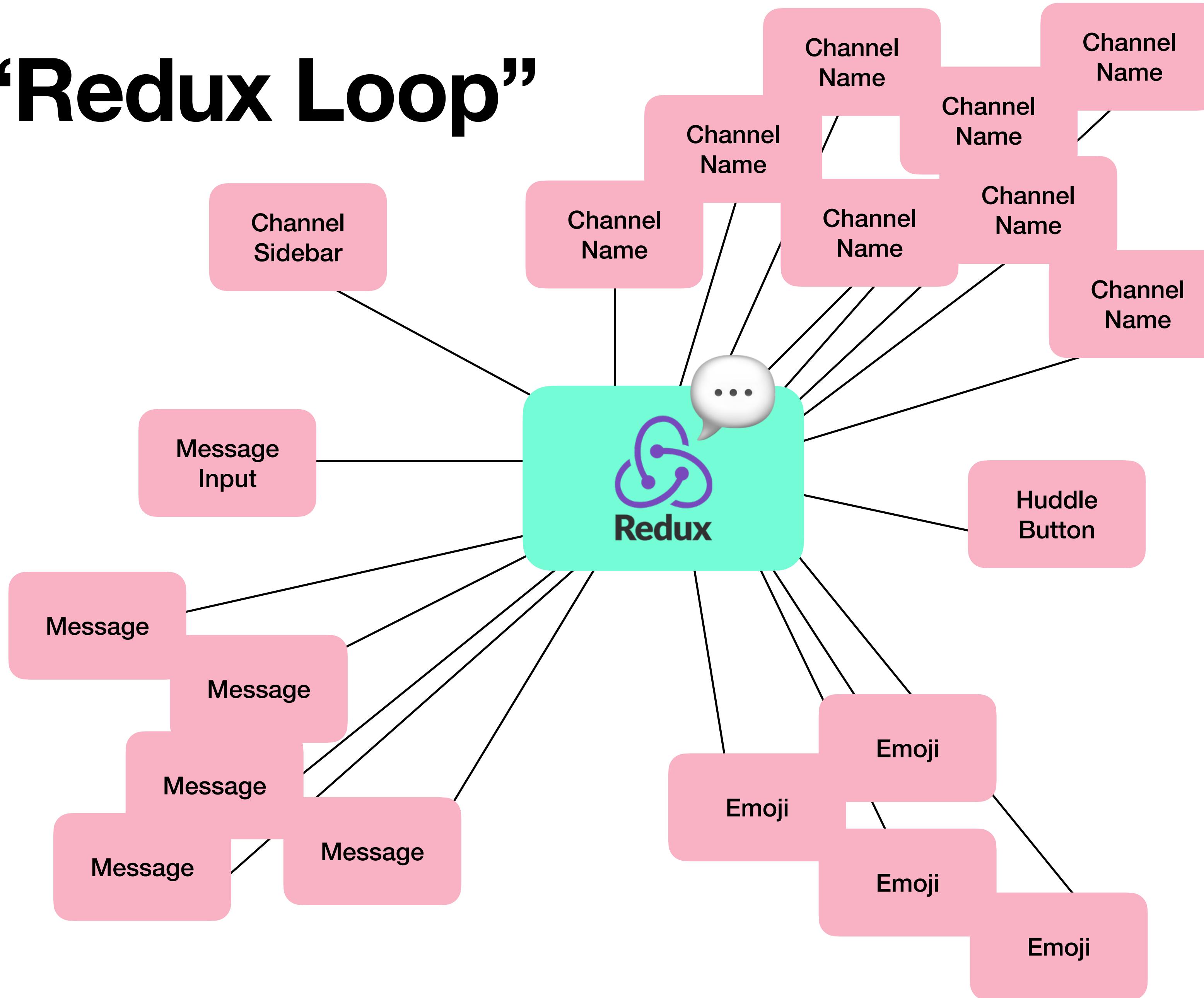
```
function Avatar({ id, size }) {  
  const person = useSelector((state) =>  
    getPersonById(state, id));  
  
  return (  
    <img  
      className="avatar"  
      src={getImageUrl(person)}  
      alt={person.name}  
      width={size}  
      height={size}  
    />  
  );  
}
```

```
<Avatar  
  size={100}  
  id={'1989'}  
/>
```

The “Redux Loop”

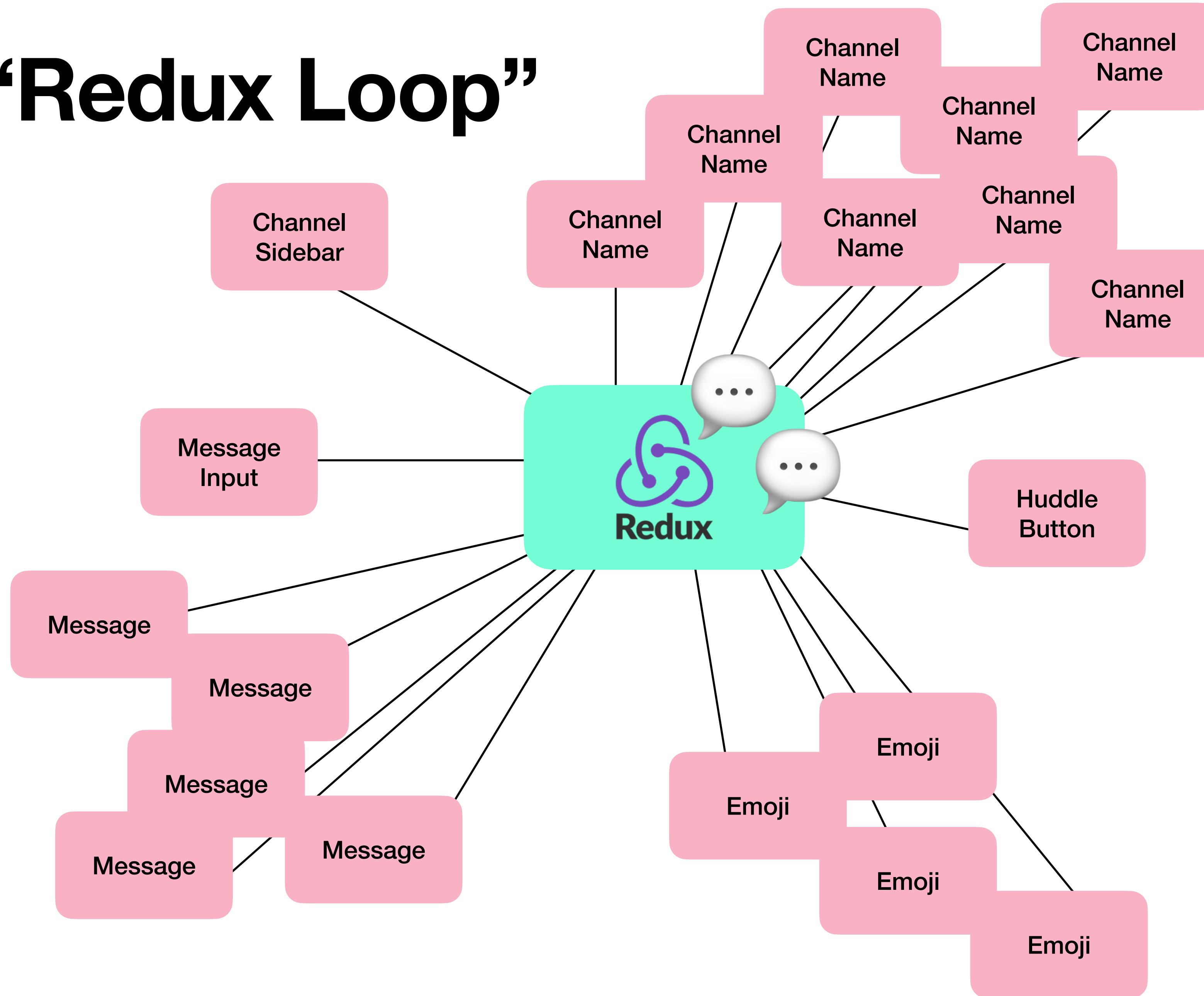


The “Redux Loop”



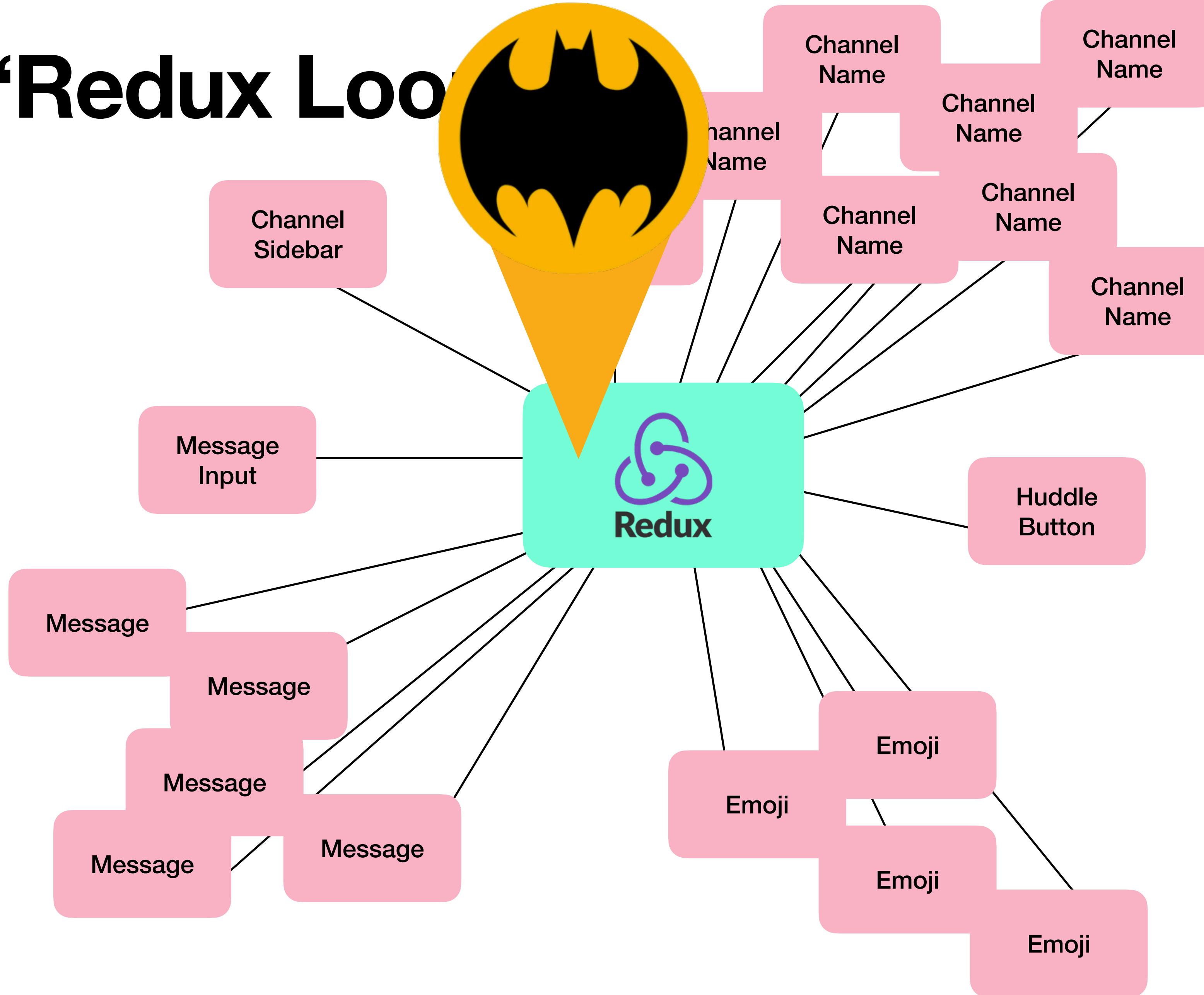
Actions are dispatched to Redux, causing “reducers” to run, which updates Redux state

The “Redux Loop”



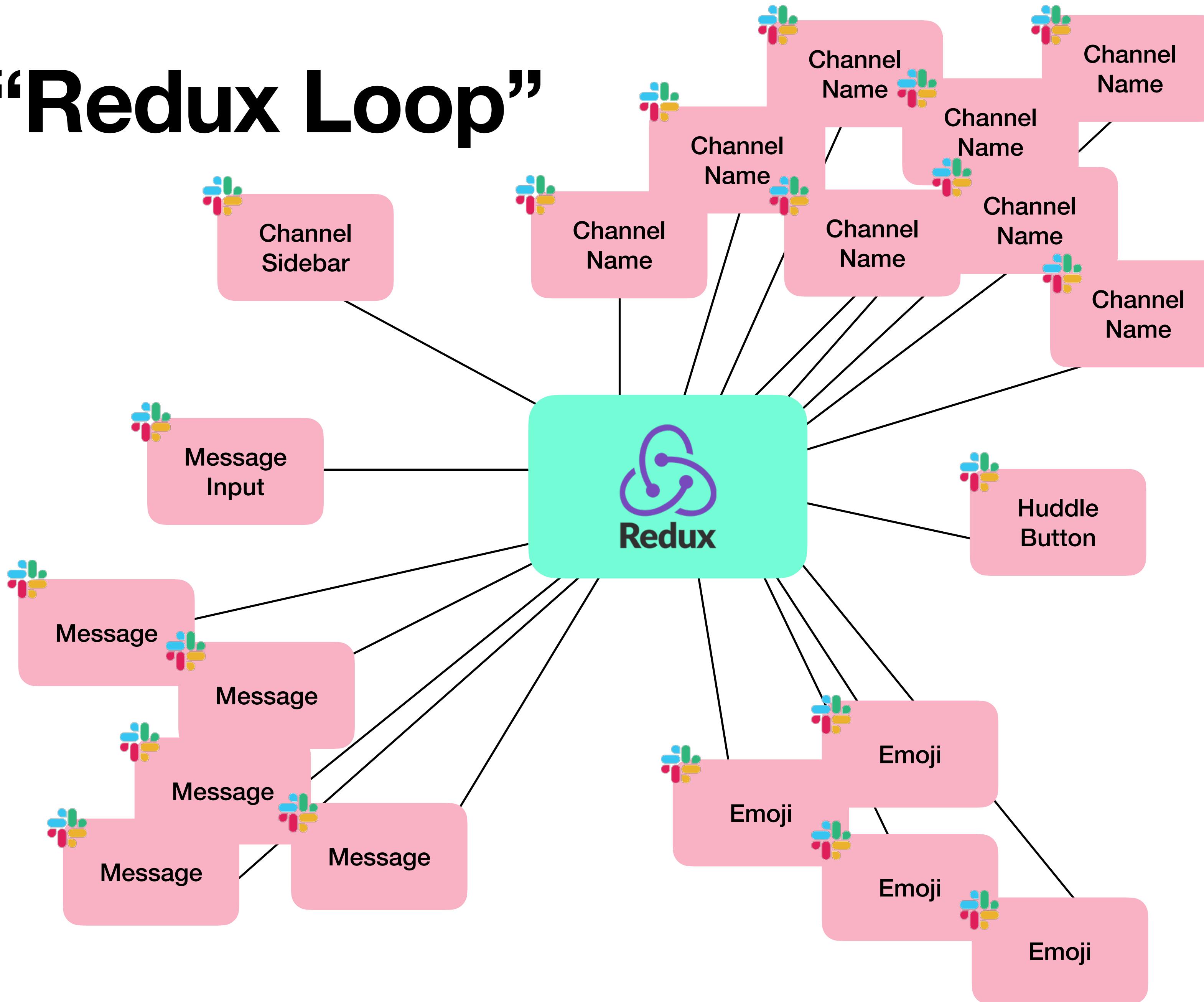
Actions are dispatched to Redux, causing “reducers” to run, which updates Redux state

The “Redux Loop”



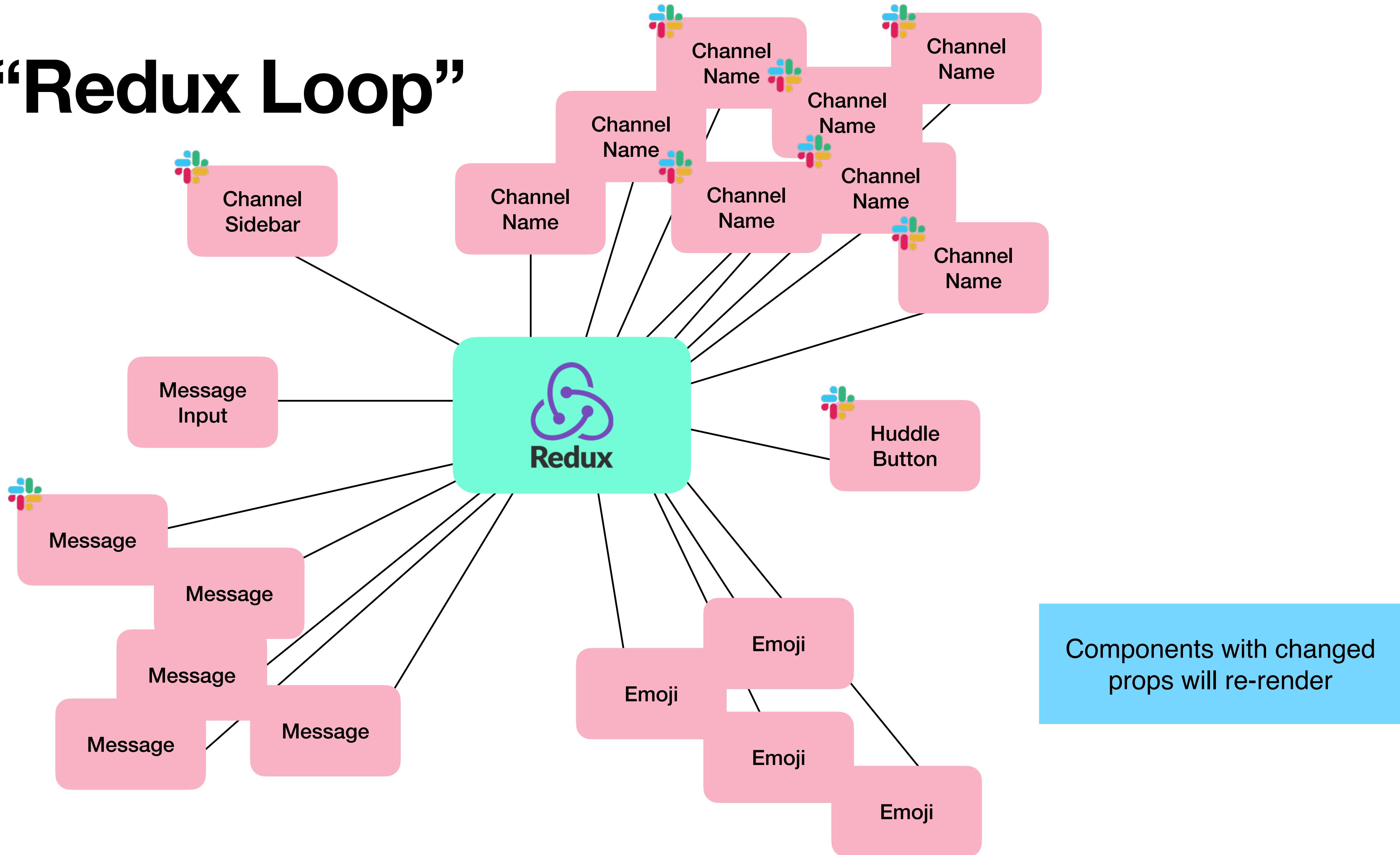
Redux sends out the notification to every connected component

The “Redux Loop”

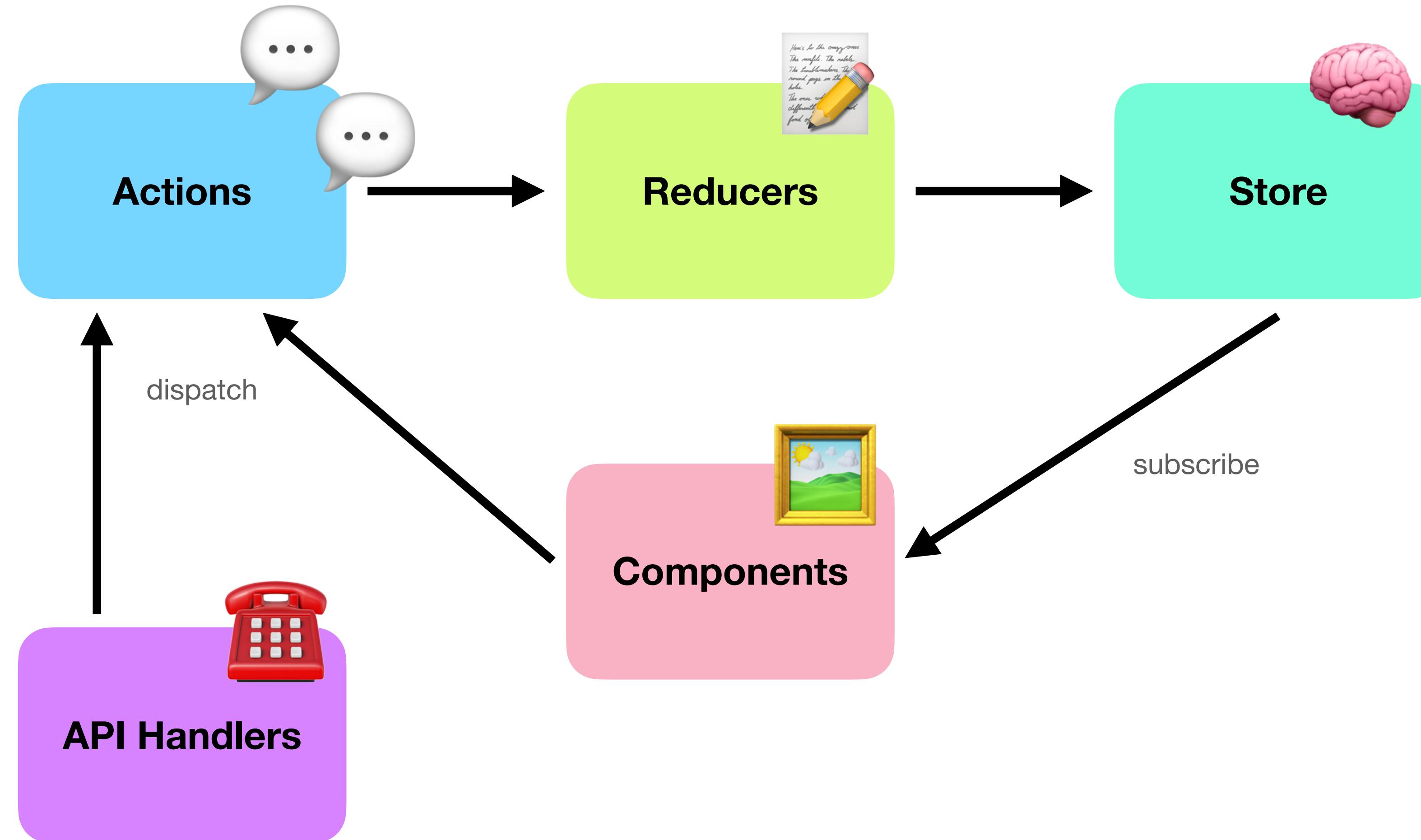


Everything connected to Redux checks if state has changed, and if so, recalculates connected props to see if any values have changed

The “Redux Loop”



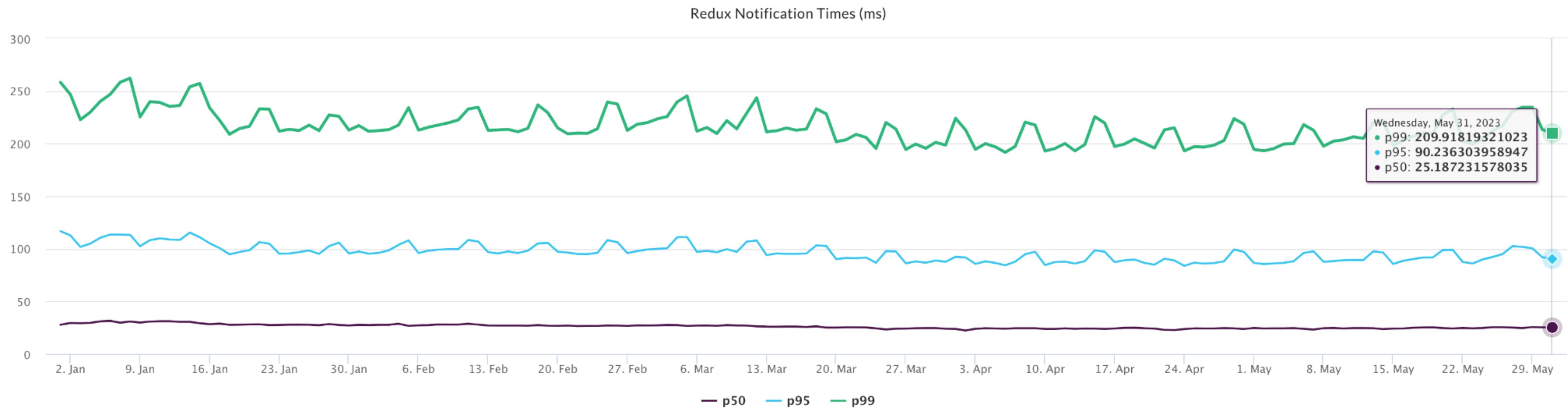
The “Redux Loop”



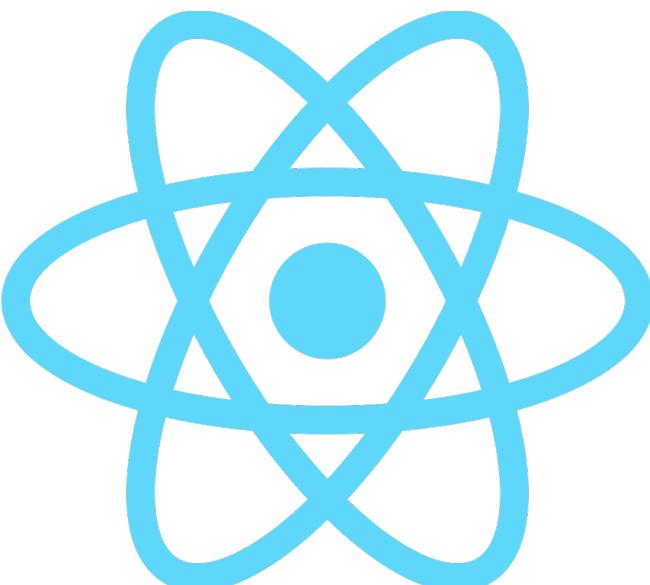
**Ok, how does this go
wrong and cause
performance issues?**

Redux Loop Time is Too Damn High

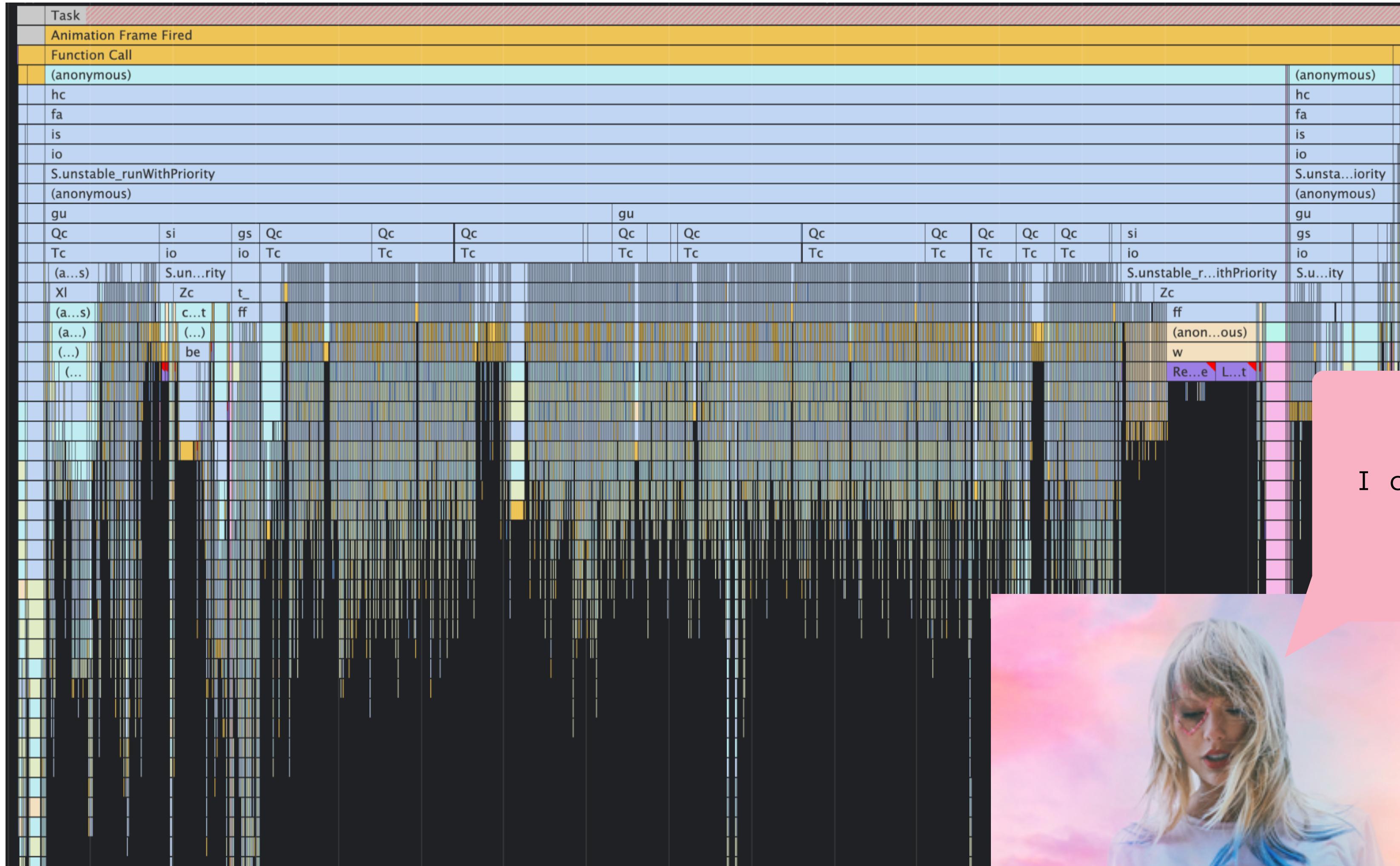
Ideally, all Redux work happens within an animation frame so we're not dropping frames and blocking inputs, but... we weren't there yet!



♪♪
I love you
It's ruining my life
♪♪



Papercuts?



I can't pretend it's okay when it's not
It's death by a thousand cuts



Where Does Performance Break Down

1. Every change to Redux results in a Redux notification firing
2. Redux notification means all selectors are running, which means spending too long running selectors
3. Spending too long re-rendering components (often, unnecessarily)



Shouty Redux

How this happens:

- ⚛ Slack stores most of its data in Redux, which means lots of updates
- ⚛ API calls, websocket events, user interactions all cause a subscriber notification
 - ⚛ Every time you switch channels
 - ⚛ Every time you send a message
 - ⚛ Every time you receive a message
 - ⚛ Every time someone reactjis on a message in a channel you're in
 - ⚛ Every time someone updates their custom status
- ⚛ ...

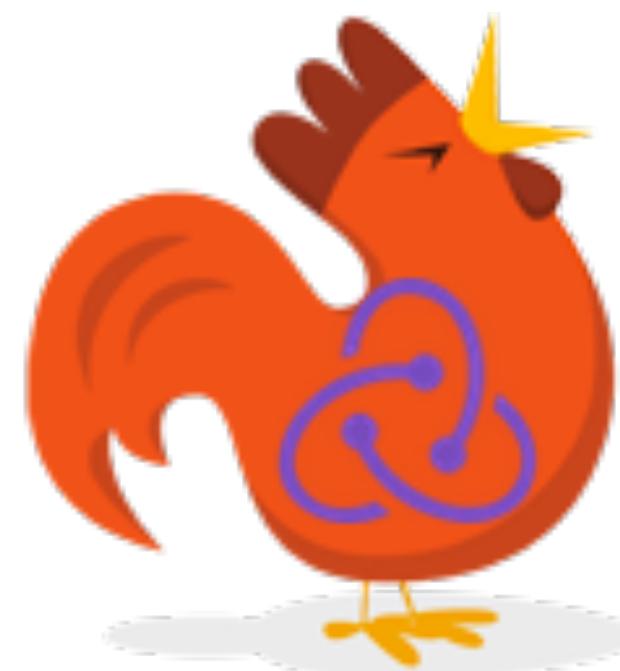
♪♪
Ooh, look what you made me do
Look what you made me do
Look what you just made me do
Look what you just made me...
♪♪



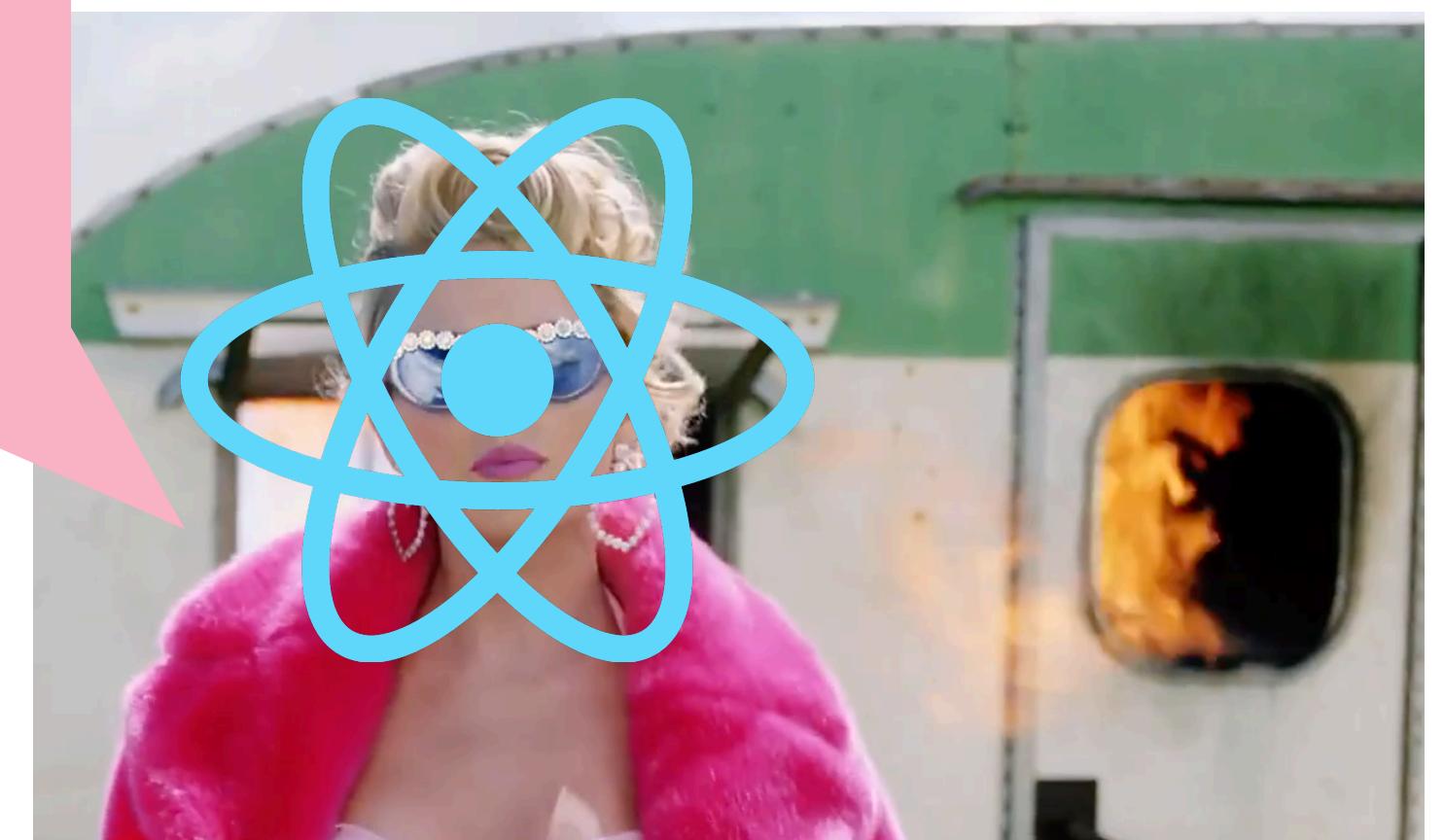
So Many Selectors, So Little Time!

How this happens:

- ⚛️ ⚡ Every connection runs every time Redux notifies ⚡
- ⚛️ Practically, 5,000 to 25,000 connected props being calculated per loop



You need to calm down
You're being too loud
And I'm just like oh-oh, oh-oh
You need to just stop
Like, can you just not send out that shout?
You need to calm down

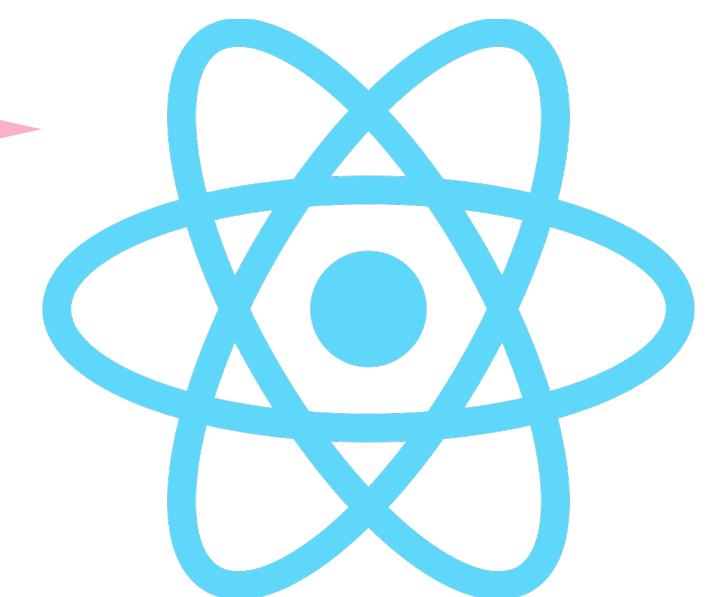


Unnecessary Re-Renders

Many components receive or calculate props that fail equality checks but are deep-equal (i.e. are “unstable”)

How this happens:

- ⌚ Calculating a prop via map/filter/reduce/etc
- ⌚ Returning [] or {} from a calculation as a default
- ⌚ Passing anonymous functions as callbacks
- ⌚ And more!



**Cool, how did we start
making it better?**

Doing Less Work!

Thanks!

Thanks! (lol jk 😂)

Doing Less Work!!!

- 1. Targeting Problem Components**
- 2. Broad-Spectrum Solutions**

1. Targeting Problem Components

Any Guesses? 🤔

Acme Inc

Matt Brewer

All unreads

Threads

Mentions & reactions

Drafts

Show more

Channels

announcements

design-crit

media-and-pr 1

social-media

Direct messages

slackbot

Matt Brewer (you)

Lee Hao, Sara Parras

#social-media ★

21 | 1 | Track and coordinate social media

Acme Team APP 12:45 PM
Event starting in 15 minutes:
Team Status Meeting 📝
Today from 1:00 PM to 1:30 PM

Harry Boone 12:58 PM
Quick note: today @Liza will join our team sync to provide updates on the launch. If you have questions, bring 'em. See you all later... er, in 2 minutes 😅

Lee Hao 12:58 PM
Meeting notes from our sync with @Liza
Post
1/9 Meeting Notes
Last edited just now

Zenith Marketing is in this channel

Message #social-media

Details

#social-media

Add Find Call More

About

Topic
Track and coordinate social media

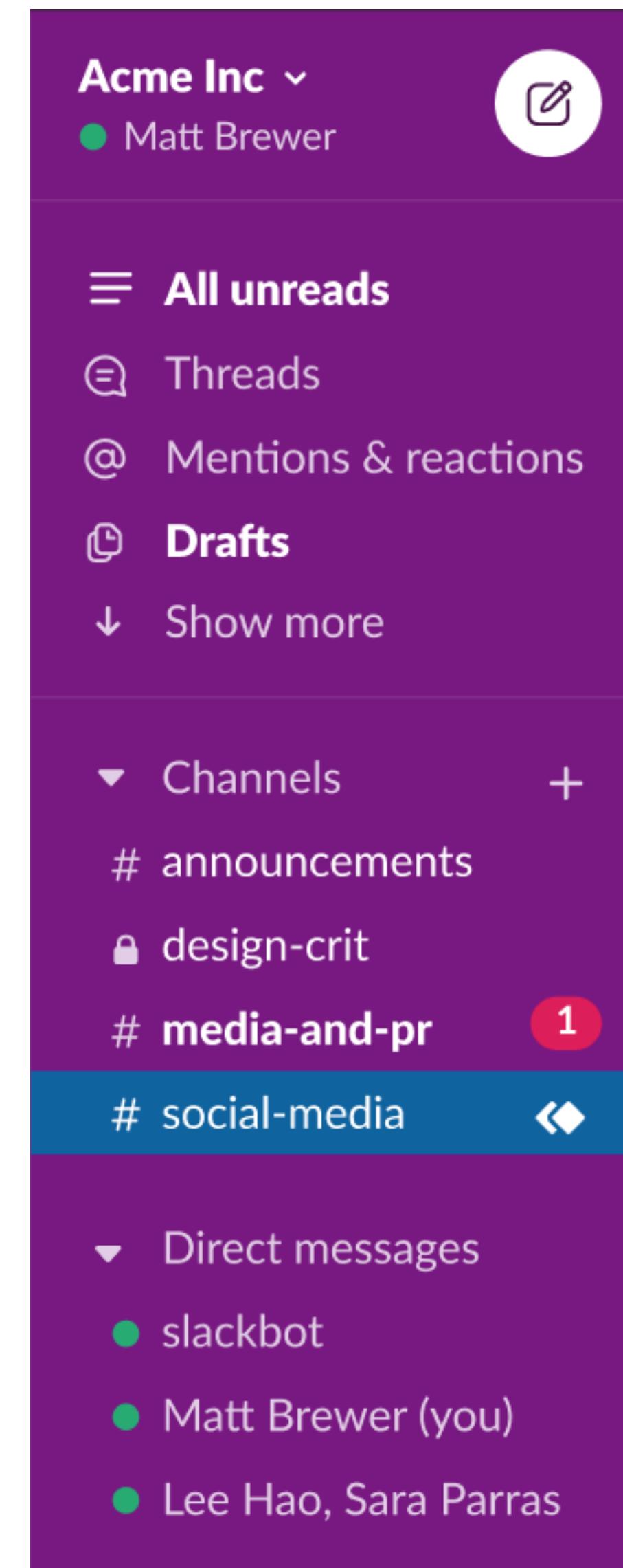
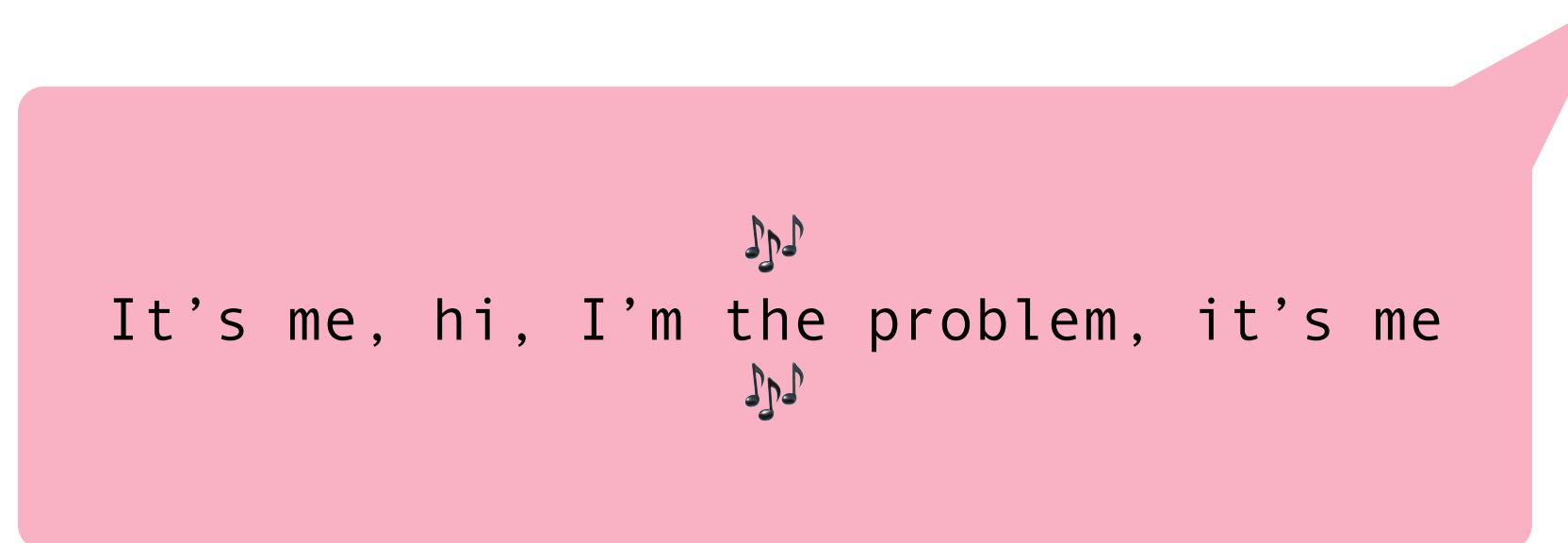
Description
Home of the social media team

Created on October 18th, 2019

21 >

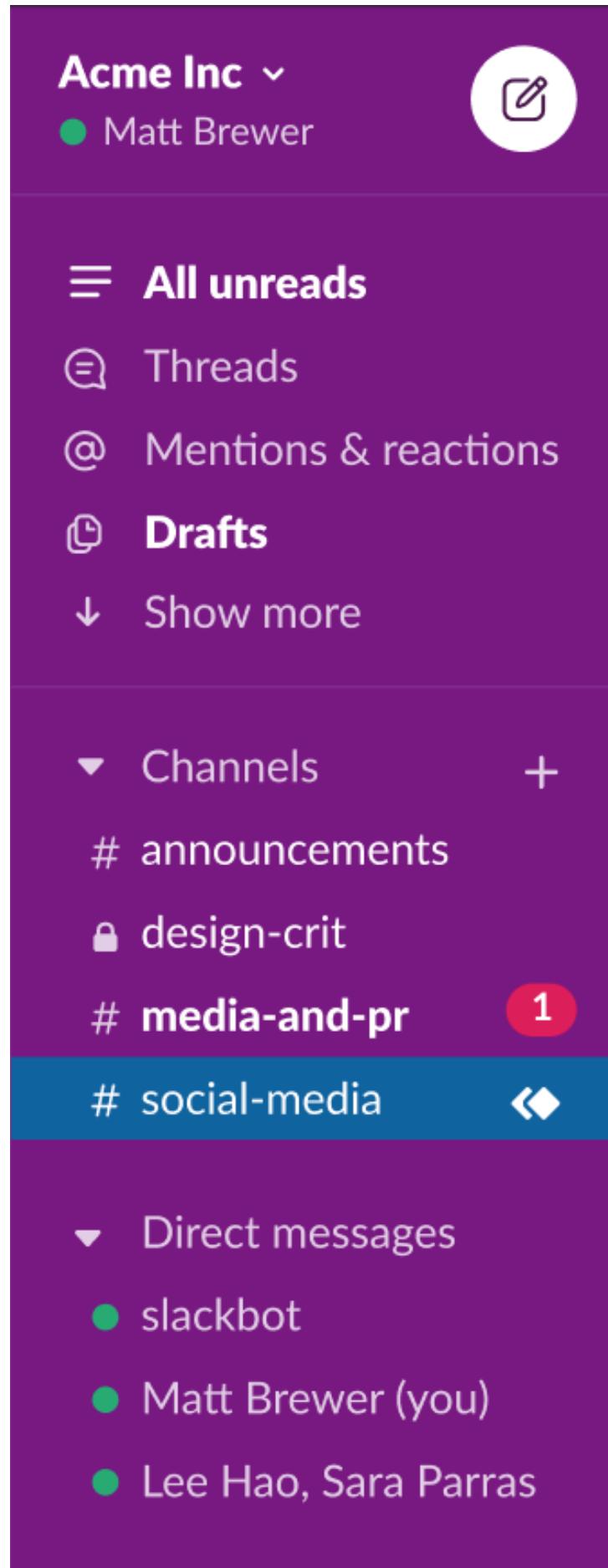
Organizations 2 >

The Channel Sidebar



A Sidebar About Performance

The sidebar looks like a simple list of channels with some icons and headers...



```
// simplified code :)

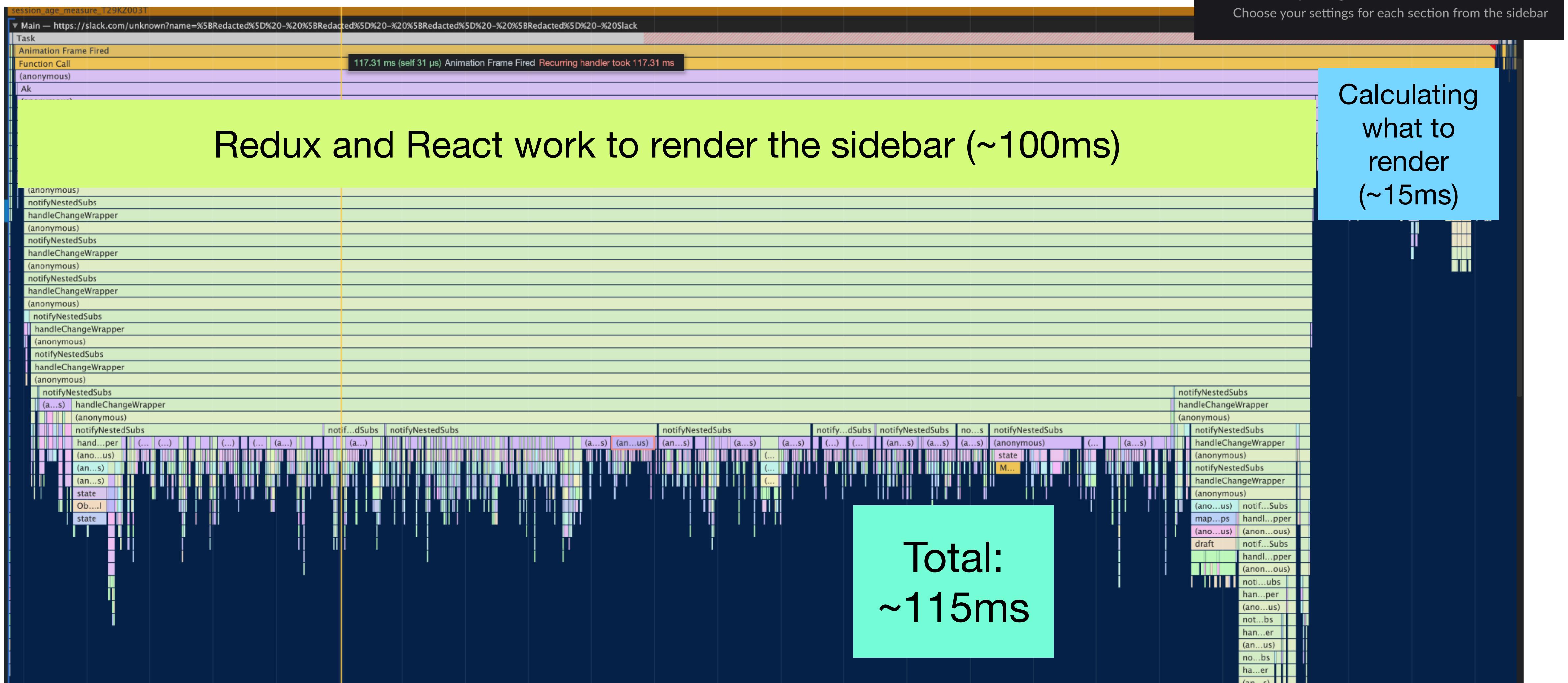
<ChannelSidebar>
  <Icon /><ViewName />
  <Icon /><ViewName />
  <Icon /><ViewName />
  <Icon /><ViewName />
  <Icon /><ShowMore />

  <Icon /><SectionHeading /><Icon />
  <ChannelIcon /><ChannelName />
  <ChannelIcon /><ChannelName />
  <ChannelIcon /><ChannelName /><NotificationBadge />
  <ChannelIcon /><ChannelName /><Icon />

  <Icon /><SectionHeading />
  <PresenceIcon /><ChannelName />
  <PresenceIcon /><ChannelName />
  <PresenceIcon /><ChannelName />
</ChannelSidebar>
```

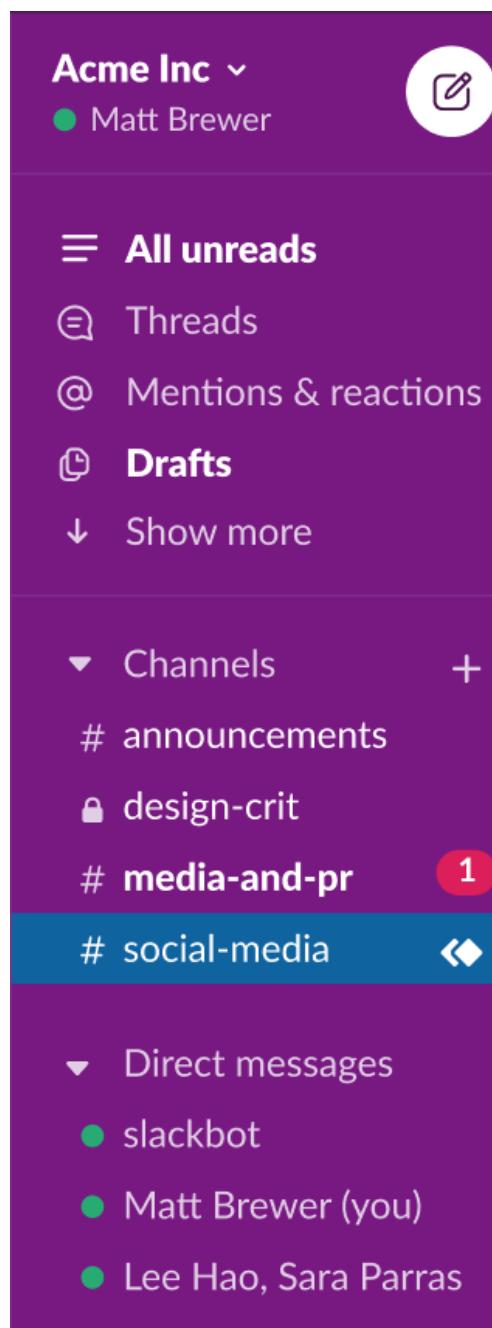
A Sidebar About Performance

...but it's so much more



Sooo Many Selectors...

- ⚛ 20,000 selector calls drops to 2,000 when the sidebar is hidden
- ⚛ Inefficiencies in lists compound quickly
 - ⚛ 40 connected prop calculations in the channel sidebar item component, times however many channels in your sidebar...



♪♪
It's born from just one single update
But it runs, and it runs, and it runs
A million little times
♪♪



... But Many of Them Are Unnecessary!

Biggest issue was repeated work

- ⚛ i.e., checking if users were in experiments

And also unnecessary work:

- ⚛ i.e. DMs needed data public and private channels didn't, and vice versa

♪
And maybe we got lost in translation
Maybe I asked for too much
But maybe this thing was a masterpiece
'til you tore it all up
♪



... But Many of Them Are Unnecessary!

Some solutions

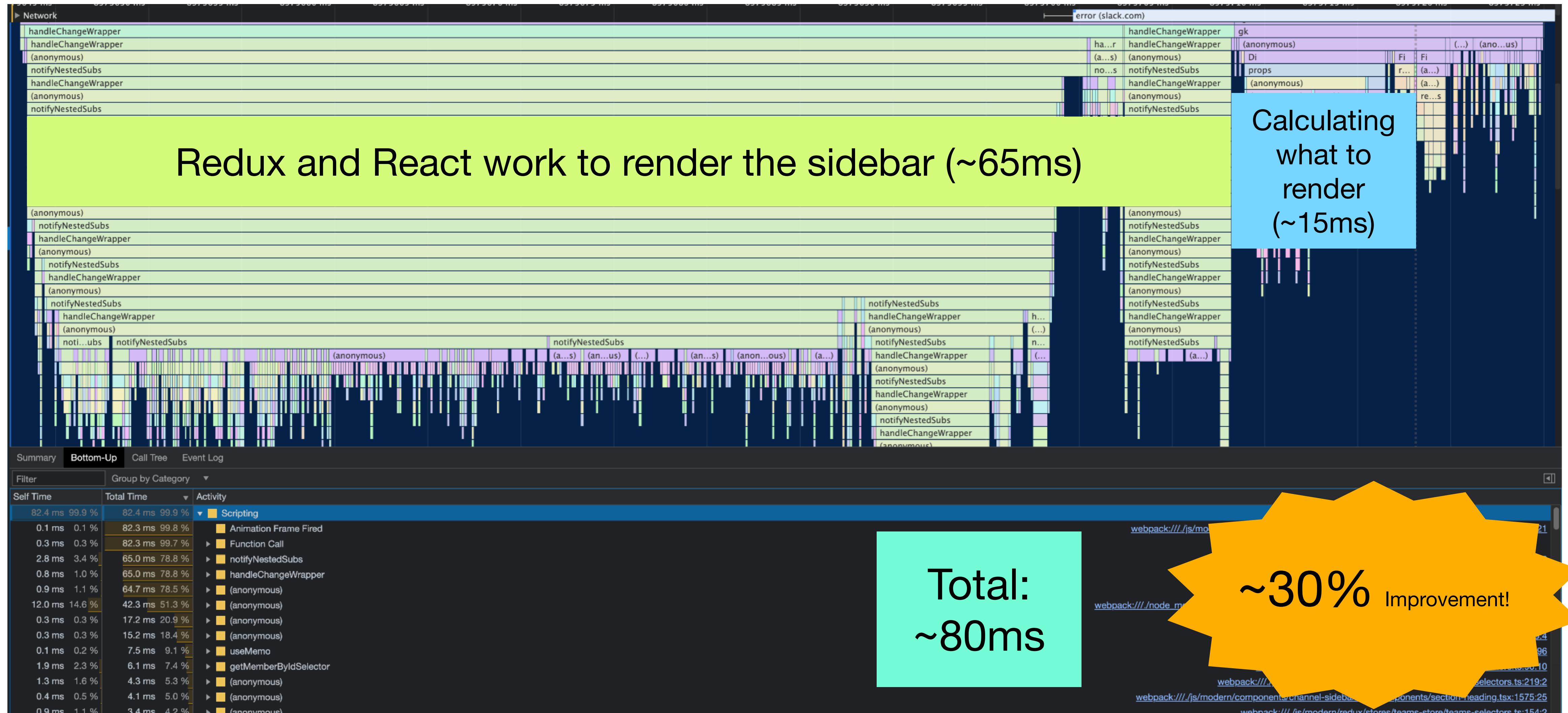
- ⚛️ Moving repeated work to the list level (call ‘em once instead of n times!)
- ⚛️ Creating more specialized components
- ⚛️ Deleting unused props (lol)

♪♪
I can feel the flame graph on my skin
She says, "Don't throw away a good thing"
♪♪



A Sidebar About Performance

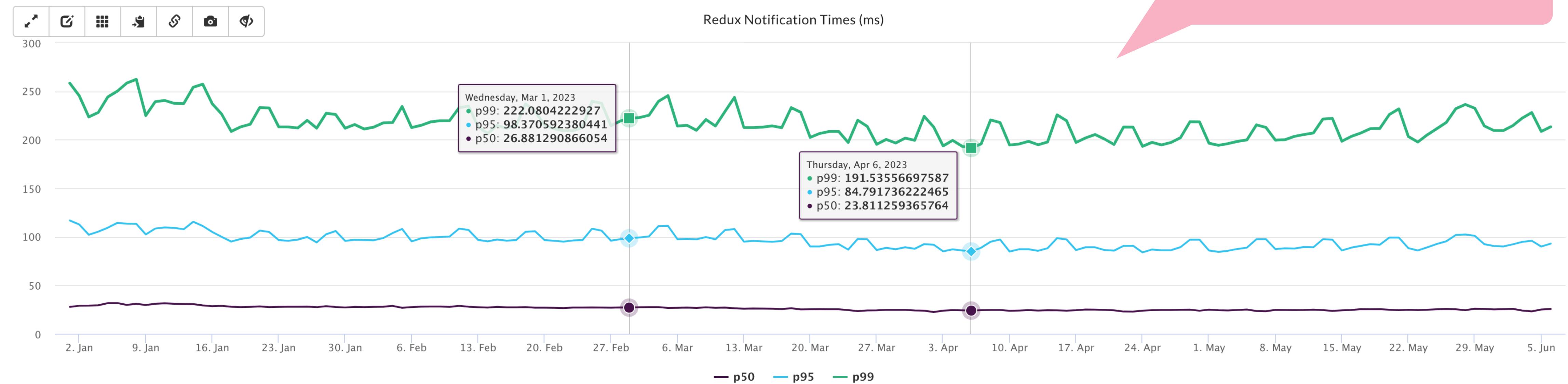
And now...



Impact on Overall Redux Loop Time

Redux Loop time overall saw improvements just from focusing on this one component

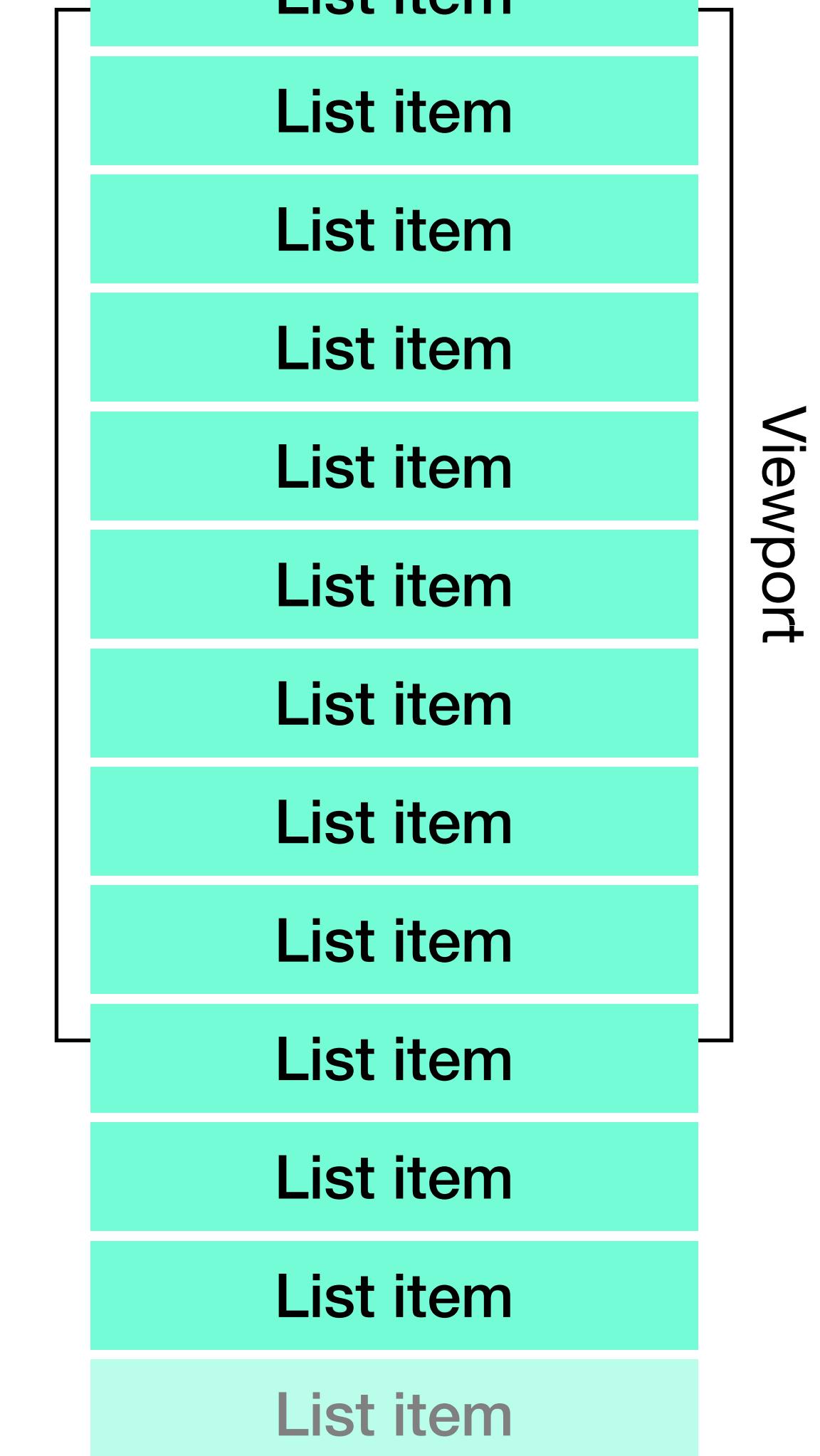
- ⚛️ 14% improvement at p99
- ⚛️ 13% improvement at p95
- ⚛️ 11% improvement at p50



List Virtualization

- ⌚ Only render what's going to be on screen, with a buffer to allow for smooth scrolling
- ⌚ Fewer connected components means fewer selectors firing on every loop
- ⌚ Tradeoff on scroll speed

I was there to watch you leave
And all the times I let you in
Just for you to go again
Disappear when you come back
Everything is better



2. Broad-Spectrum Solutions

Batched Updates

Invoking the Redux subscriber notification on the animation frame by using
ReactDOM.unstable_batchedUpdates

```
// simplified code :)
import { createStore } from 'redux';
import { batchedSubscribe } from 'redux-batched-subscribe';

const invokeOnNextAnimationFrame = (func: Function) => {
  callbacks.push(func);
  if (!requestId) {
    requestId = requestAnimationFrame(() => {
      ReactDOM.unstable_batchedUpdates(invokeAllCallbacks);
    });
  }
  return requestId;
};

const store = createStore(reducer, initialState, batchedSubscribe(invokeOnNextAnimationFrame));
```



Codemods

Performance problems can be detected and fixed at the AST level?!

- ⚛️ “Hoisting” static unstable props being passed to children
- ⚛️ Re-writing prop calculations in a way that facilitates memoization
- ⚛️ Replacing unstable empty values with constants like EMPTY_OBJECT

Perf: Codemod to stabilize `{ ok: boolean, ... }` and `EMPTY_OBJECT` returns
#153311

Merged slackbot merged 1 commit into master from jenna-perfy-remodeling on Sep 15, 2022

Conversation 2 Commits 1 Checks 0 Files changed 30 +616 -218

jenna commented on Aug 31, 2022 · edited

Description

This PR creates and runs a codemod that aims to clean up cases where we're return fresh static objects in the `boolean, ... }`, as well as empty objects (`{ }`)

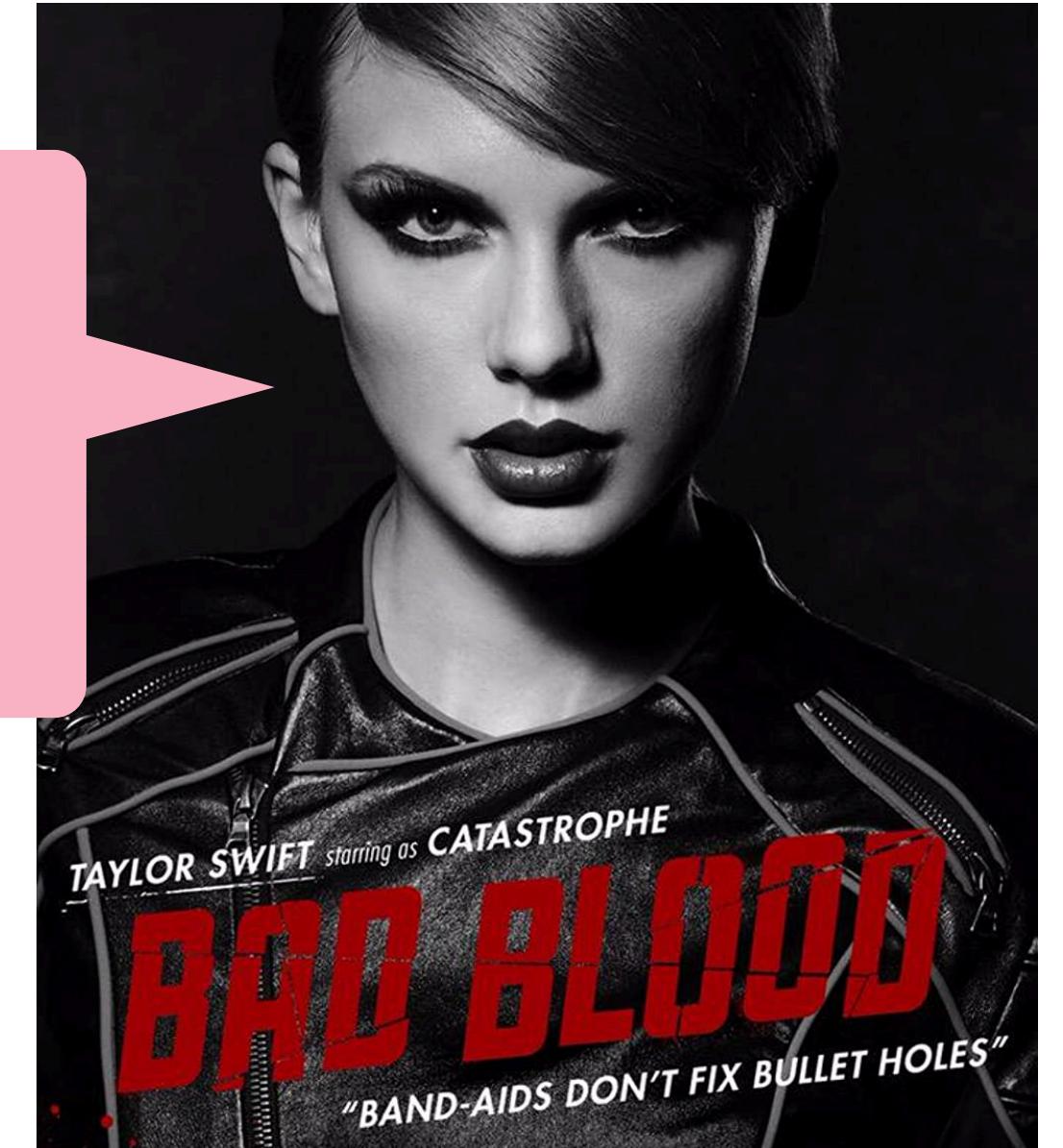
♪♪
Remember when props failed equality?
I wrote code, I said "I'm making you free"
The monsters turned out to be just trees
Amazing what you can do with an AST
♪♪



Using Redux Less?

- ⚛️ Investigating using IndexedDB to store more evicted items
 - ⚛️ Less data in Redux means fewer loops as a result of keeping items fresh
- ⚛️ Cache eviction is fun, but have you tried not storing it in the first place?
- ⚛️ Finer-grained subscription would be cool, but it's a departure from the Redux model

♪
Band-aids don't fix bullet holes
You say sorry just for show
If you live like that, you live with ghosts
If you code like that, your app runs slow!
♪



**Okay big question...
Why still use Redux?**

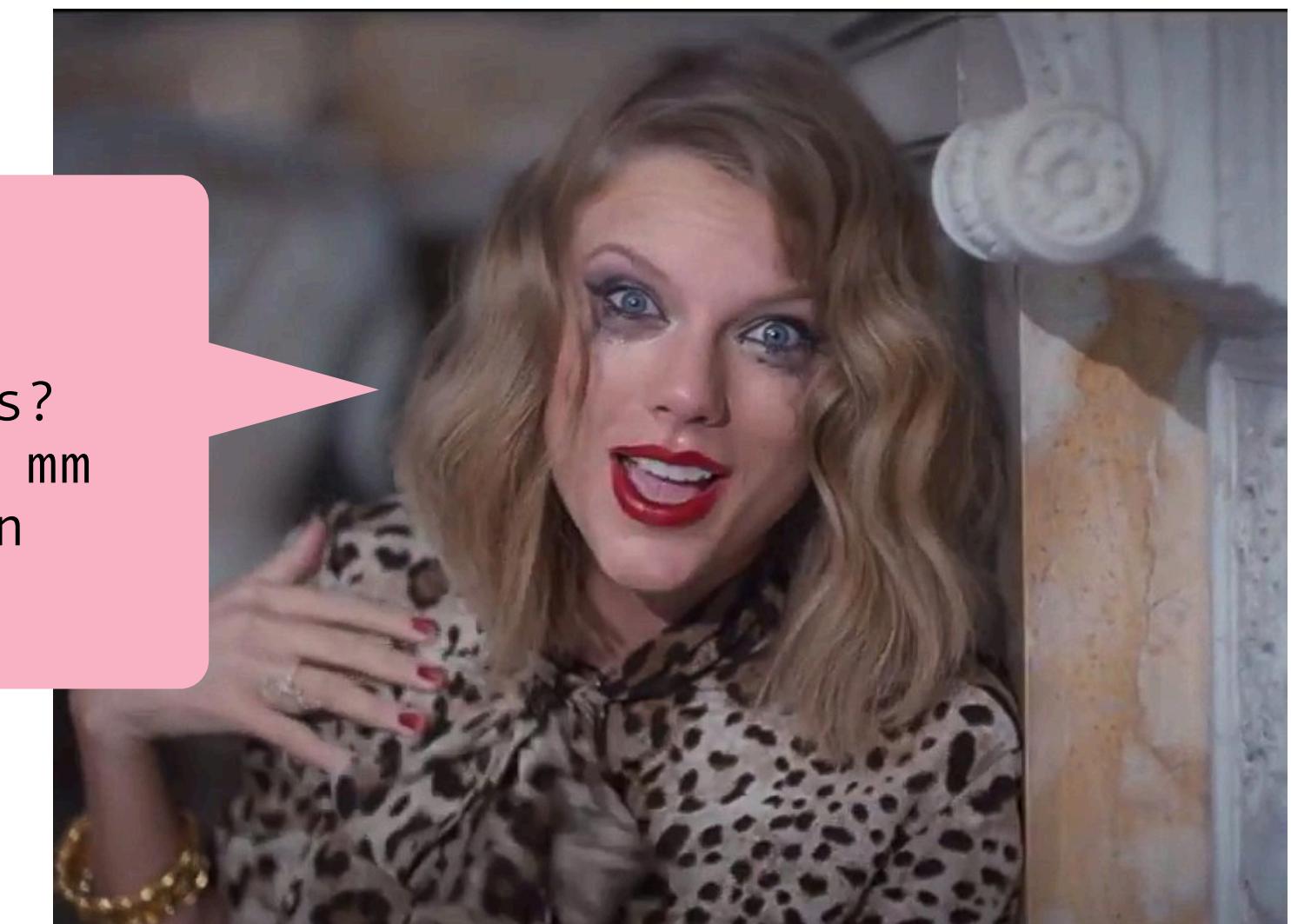
Why React and Redux, Still?

“React is a popular, well-maintained, easy-to-use component-based UI framework that promotes modularity”

- Me, about 30 minutes ago

✨ Is the cost of drastically changing our architecture worth it for the performance boosts? ✨

♪♪
So, it's gonna be forever
Or it's gonna go down in flames?
You can tell me when it's over, mm
If the high was worth the pain
♪♪



Everyone Loves Performance

- ⚛️ ✨ Engineers fundamentally want to create performant software, so let's give them the tools to set them up for success ✨
- ⚛️ Creating a performance culture through education, tooling, and evangelism

♪♪
Everybody here wanted somethin' more
Searchin' for a speed we hadn't had before
♪♪



Education and Evangelism

React and Redux abstract away internals but **understanding the system contextualizes and motivates performance work**

Jenna Zeigen 4:12 PM

PERF. A Performance Memo: What is the “Redux Loop”? **PERF.**

Last week I wrote about why you should clean up your old experiment checks, citing “the Redux loop.” Here’s an explanation of what can make informed decisions as you build your products. But don’t worry, I didn’t know half of this before it became important for me writing this! If you see something I got wrong, please please let me know!

Before I dive in, I should mention that the term “Redux loop” isn’t official nomenclature you’ll find in docs anywhere, but I think that contribute to this loop, and also sprinkle in some ideas around what we can do for performance knowing what we know.

1. Redux Actions Get Dispatched

As you know, Redux state is a giant JavaScript object that contains all the data we think the app needs to know to function. frequently, so they get **batched** together so they take effect at max once per animation frame (every ~16ms or 60x per second). a version of React we’re not on yet, so we’ve made this happen [ourselves](#)

Jenna Zeigen 5:58 PM

PERF. Announcing Project Rollercoaster: A React/Redux Performance Program **PERF.**

tl;dr: Hit the [reactji](#) if you’re interested in helping pilot [#devel-react-redux-perf-program](#) this quarter

Hey [#dhtml](#)! As you might know, Slack isn’t as fast as it could be. This isn’t because any one thing in particular is dragging down the [React/Redux Loop](#). This “death by a thousand cuts” makes switching channels feel slow, typing faster, and more pleasant and more productive!

If you watch the console while developing, you’ve undoubtedly noticed how many performance runtime warnings we have throughout the codebase. Each of those warnings signifies a papercut, an opportunity to have [React performance](#) lint warnings throughout webapp, and we catch hundreds of thousands of unnecessary re-renders. more of these numbers on this [dashboard](#). This all comes together to make the Redux loop slower than we want React to do all their selector calls, checks, and re-renders. As routine work goes, that’s pretty slow!

Monday, March 27th

Jenna Zeigen 1:47 PM

PERF. Performance Story Time: The Channel Sidebar **PERF.**

tldr: Read and learn how I improved sidebar perf by about 25% and got Redux loop time to its lowest duration yet!

The Channel Sidebar is perhaps our most complex component, and it’s always on the screen. This might seem weird because it’s a flat list of channels with headers , but what it needs to display, and it often needs to display a lot of items. It also re-renders a lot, and it takes a while to do so. We’ve known for a while that the sidebar was a pain in the neck, and done several projects to improve channel sidebar performance. Recently, we heard from someone in the IA4 pilot that their client performance improved dramatically in one of our screens . This was a wake-up call for me that we needed to do even more exploration into what makes the sidebar slow.

Jenna Zeigen 1:26 PM

PERF. NEW **PERF.** Introducing the **useSelector** Performance Detector™ **PERF.** NEW

Hello again . I just merged another console warner that will warn you when a selector called by **useSelector** is being caused the issue. This little tool was the [brainchild](#) of [@bkraft](#) (thanks!) and was “productionized” through surfacing ways to stop unnecessary re-renders due to props that contain the same value but are different objects.

The two common things we see are:

- Empty arrays and objects, easily stabilized by using the **EMPTY_ARRAY** and **EMPTY_OBJECT** utilities

Jenna Zeigen 12:08 PM

PERF. A Performance Memo: The Magic Numbers 16ms, 50ms, and 100ms! **PERF.**

You might have heard about 16ms being somewhat of a magic number in performance. If not, now you do! These numbers came to be and why it’s important to keep them in mind to ensure performant experiences for everyone.

16ms: Animation Frames

I mentioned in my [Redux loop](#) post last week that Redux actions are batched together so they happen in discrete conditions, will repaint the screen 60 times per second (aside from [MDN](#): it “is usually 60 times per second”). This comes out to repaints happening every ~16.666ms.

Jenna Zeigen 4:07 PM

PERF. NEW **PERF.** A **mapStateToProps** Performance Detector Drop! **PERF.** NEW

From the people who brought you the **useSelector** Performance Detector™, introducing a similar console perf warner for class components that map state to props that could be causing re-renders— values that don’t pass the component’s equality checks but are deeply equal.

To make this addition not totally overwhelming , we’ve also changed the amount the detectors will warn you in the console once you’re finding a particular component is noisy, please add it to this tracking [sheet](#) so we can get it sorted soon!

Lint Rules

Show engineers right in their editor when they're writing code that's a performance liability

- ⚛️ Unstable props being passed to children
 - ⚛️ i.e. react-perf/jsx-no-new-object-as-prop
- ⚛️ Unstable props being computed for connected components
- ⚛️ Functions and values that break memoization but don't have to

♪♪
But I got smarter, I got harder in the nick of time
Honey, I read all of your code, I do it all the time
I got a list of props, and yours is in red, underlined
I check it once, then I check it twice, oh!
♪♪

ESLint



Runtime Console Warnings

Warnings in the Chrome console for performance opportunities best caught at runtime, such as:

- ⚛️ Unstable connected prop calculations
- ⚛️ Checks for finished experiments

```
⚠️ ► Jun-7 17:24:58.144 [PERF DEBUG] (T5J4Q04QG) The toggle [REDACTED] is finished a
checked for this toggle 14400 times. :0:0
⚠️ ► Jun-7 17:24:58.176 [PERF DEBUG] A useSelector call in
[REDACTED] returned a deep-equal value
► {
that fails equality checks. This means the component might be re-rendering unnecessarily and has happened 1 times
since the last refresh. :0:0
⚠️ ► Jun-7 17:28:41.069 [PERF DEBUG] mapStateToProps in the component that [REDACTED] react_devtools_backend_compact.js:2367
connects [REDACTED] returned a deep-equal value for the prop "member"
► [
} that fails
equality checks. This means the component might be re-rendering unnecessarily and has happened 4 times since the
last refresh. This value might be a member, which are known to be unstable due to our upsert strategy. Consider
returning a more stable derived value from your selector. :0:0
```



Regression Testing

- ⚛️ Usually, perf testing involves measuring the amount of time something takes
 - ⚛️ Variations in each run
 - ⚛️ Hard to detect papercuts this way
- ⚛️ Instead, our tests ensured that there are no selectors that are detected by the modules that power the runtime warnings

♪♪
My team only breaks my favorite tests, tests, oh
I'm queen of sand castles everyone detests, oh, oh
♪♪



Education Equals Empowerment

Empowering engineers to fix performance issues as they build their features

- ❖ Wrap components in `React.memo` to skip re-rendering if props are equal
- ❖ Use `EMPTY_ARRAY` and `EMPTY_OBJECT` in place of unstable `[]` and `{ }`
- ❖ The homegrown `useShallowEqualSelector` hook gives another layer of equality check if needed
- ❖ `useCallback` and `useMemo` stabilize props passed from function components to children

Memoizing it is as easy as knowing all the words
To your old favorite song!



And, a Burndown Program

It's called Project Rollercoaster (making the Loops go fast!)



Mitigating A Problem of Scale at Scale

- ⚛️ Performance has been built up as a problem for the experts, often surrounded by an air of hero culture, but **we're doing ourselves a disservice by keeping it an inaccessible discipline**
- ⚛️ Instead let's get many people to fix many problems— architecting a distributed solution for a problem of scale!

♪♪
Can't you see that I'm the one
Who understands you?
Been here all along
So, why can't you see?
You belong with me
♪♪



But, in the end it turned out... fine

- ❖ We did end up figuring out how to simulate finer-grained subscription in Redux, which was a far bigger win than chipping away at the papercuts

DESKTOP_PERF Redux Subscriber Notification Timings v2							
		control	treatment				
2024-07-16	Metric	Mean	Mean	Absolute Change	Relative Change	P-Value ⓘ	MDE ⓘ
	% of redux subscriber notif > 3ms	0.76 1.56m/2.07m	0.53 1.08m/2.06m	-0.229641 +/- 0.001364	-30.41% +/- 0.16%	< 0.001 Significant	0.21%
	% of redux subscriber notif > 15ms	0.26 541.98k/2.07m	0.19 383.37k/2.06m	-0.07614 +/- 0.001183	-29.07% +/- 0.38%	< 0.001 Significant	0.51%
	% of redux subscriber notif > 30ms	0.14 279.67k/2.07m	0.11 220.21k/2.06m	-0.028442 +/- 0.000903	-21.05% +/- 0.59%	< 0.001 Significant	0.79%
	% of redux subscriber notif > 100ms	0.04 80.6k/2.07m	0.03 71.49k/2.06m	-0.004306 +/- 0.000507	-11.06% +/- 1.23%	< 0.001 Significant	1.63%
	% of redux subscriber notif > 300ms	0.01 17.2k/2.07m	0.01 15.66k/2.06m	-0.000723 +/- 0.000234	-8.7% +/- 2.69%	< 0.001 Significant	3.57%
	% of redux subscriber notif > 600ms	1.93e-3 3.99k/2.07m	1.65e-3 3.41k/2.06m	-0.000279 +/- 0.000111	-14.44% +/- 5.33%	< 0.001 Significant	7.07%
	% users w/redux subscriber notif > 300 ms	0.01 16.59k/1.44m	0.01 15.09k/1.43m	-0.001021 +/- 0.00032	-8.85% +/- 2.65%	< 0.001 Significant	3.51%

I forgot that you existed
And I thought that it would kill me,
But it didn't
And it was so nice
So peaceful and quiet



**It takes
a lot of work
to do less work.**

Thanks!

jenna.is/at-ny-perf
@zeigenvector