Several Components are Rendering

Client Performance at Slack-Scale

Jenna Zeigen
Lead Dev New York
9/4/2024

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

~~Senior Staff Software Engineer~~
~~on Slack’s Client Performance Infrastructure Team~~

Software Engineer
on Notion's Web Infrastructure Team

jenna.is/at-lead-dev

@zeigenvector

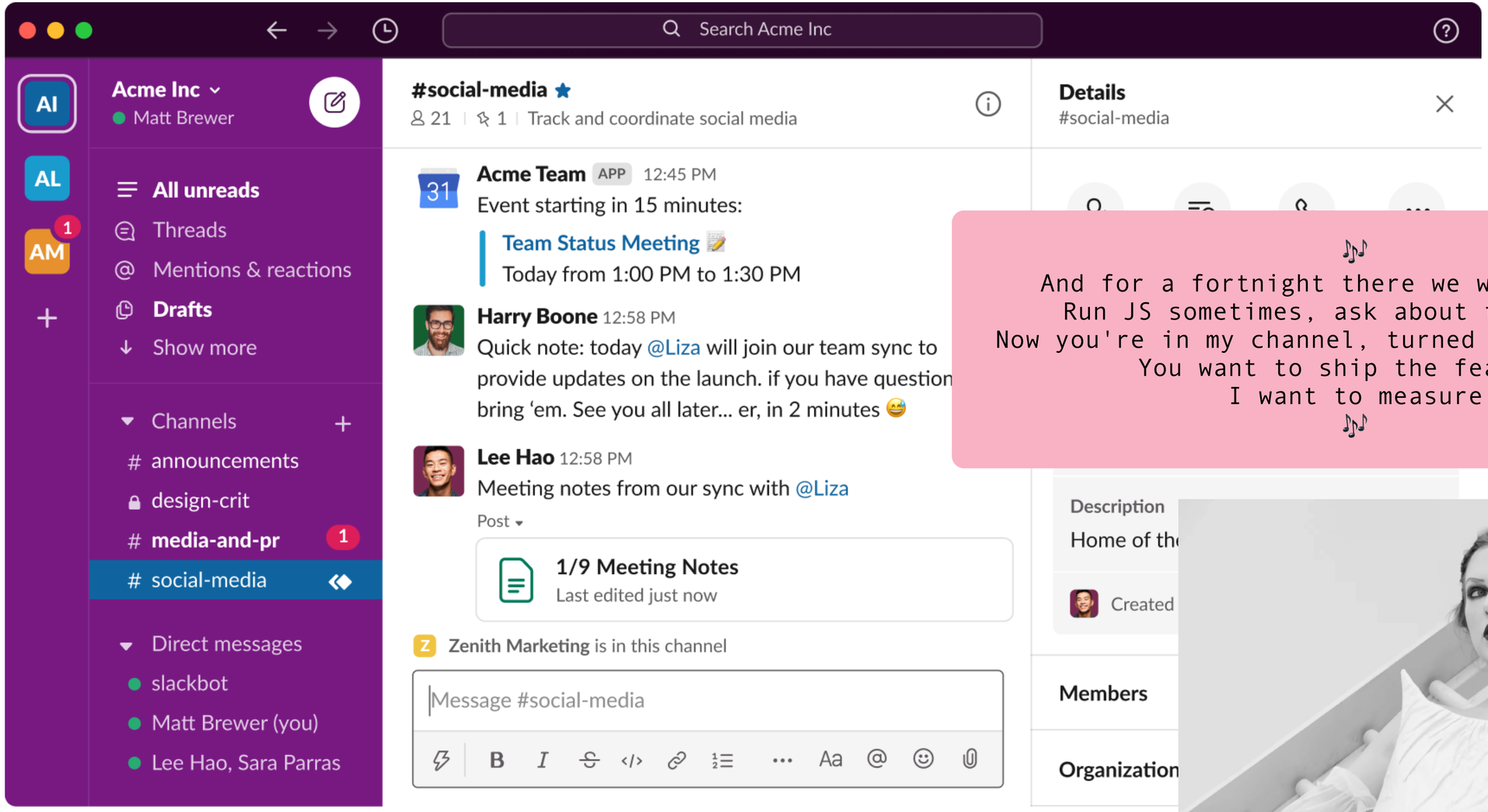
Happy Ten Year Conference Talk-iversary to Me!

JSConf EU 2014

and thus begins a performance on performance

First some stuff about the Slack Desktop app

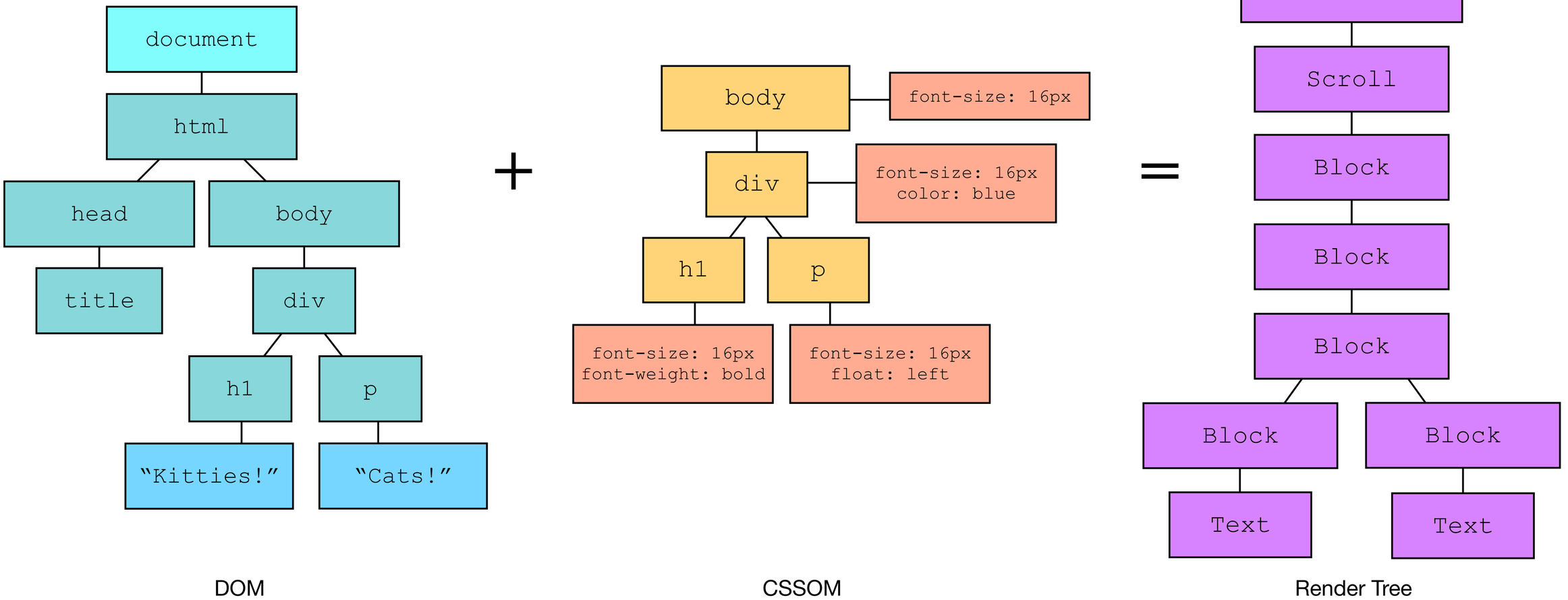
Slack, a React app on your Desktop



Now, some stuff about browsers

How Do Browsers Even?

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



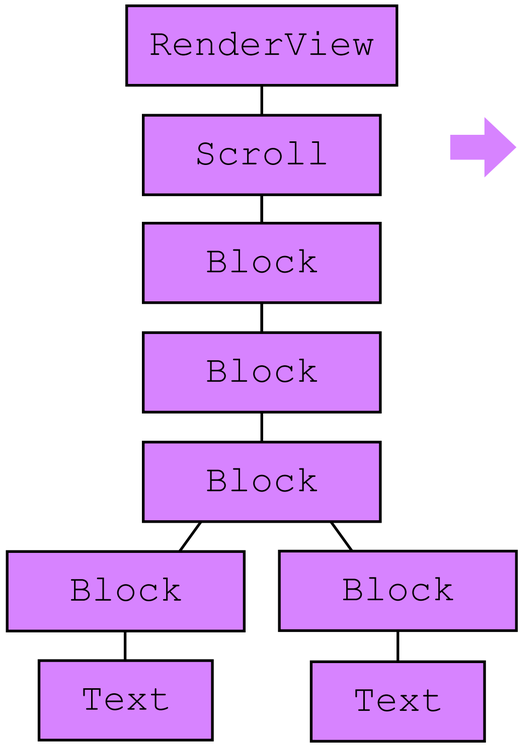
DOM

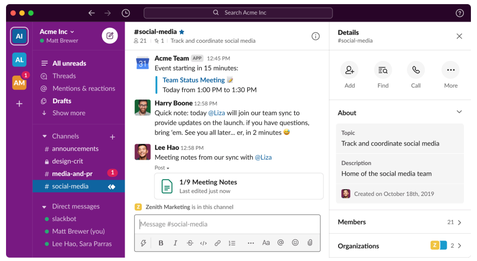
CSSOM

Render Tree

How Do Browsers Even?

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





♪♪
Cause the render's gotta rend, rend, rend?
And the painter's gotta paint, paint, paint
And the compositor's gotta composite,
composite, composite
♪♪

How Do Browsers Even?

tl;dr JavaScript is single threaded

- ⚛ All your JavaScript also has to happen on that same thread
- ⚛ The browser won't complete a render if there's JavaScript that needs to run
- ⚛ ✨ **If your JavaScript takes longer than 16ms to run, you can end up with dropped frames and laggy inputs** ✨

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)
🎶

Why “Do” Performance

- So the graphs go in the right direction?
- So we make more money??
- So people don’t write things about us on Hacker News???

✨ So our users have a ~~great~~ ~~pleasant~~ not-bad experience! ✨

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

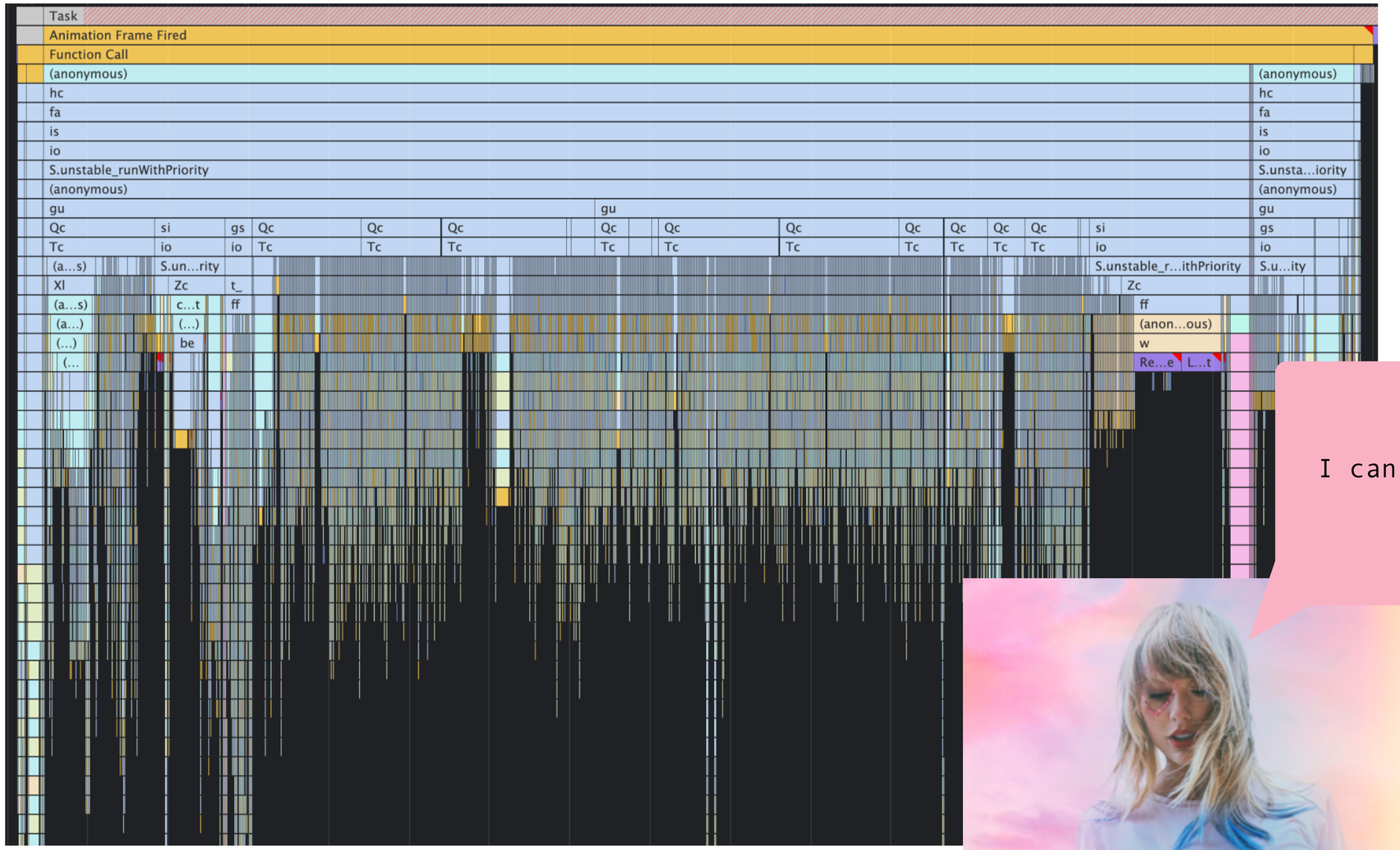
Ok, so those Slack performance issues?

bb perf team (Circa 2021)

- First pitched as a Frontend Performance Regression Testing initiative
- Quickly realized our problems were “papercuts” not “catastrophes”

♪♪
I wanna be your endgame (endgame)
I wanna be your first string (first string)
I wanna be your perf team (perf team)
I wanna be your endgame, endgame
♪♪

Papercuts?



♫♪

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

♫♪

bb perf team (Circa 2021)

- First pitched as a Frontend Performance Regression Testing initiative
- Quickly realized our problems were “papercuts” not “catastrophes”
- Pivoted to Frontend Performance Observability
- Eventually became Client Performance Infrastructure

♪♪
I wanna be your endgame (endgame)
I wanna be your first string (first string)
I wanna be your perf team (perf team)
I wanna be your endgame, endgame
♪♪

Metrics, Metrics, Metrics

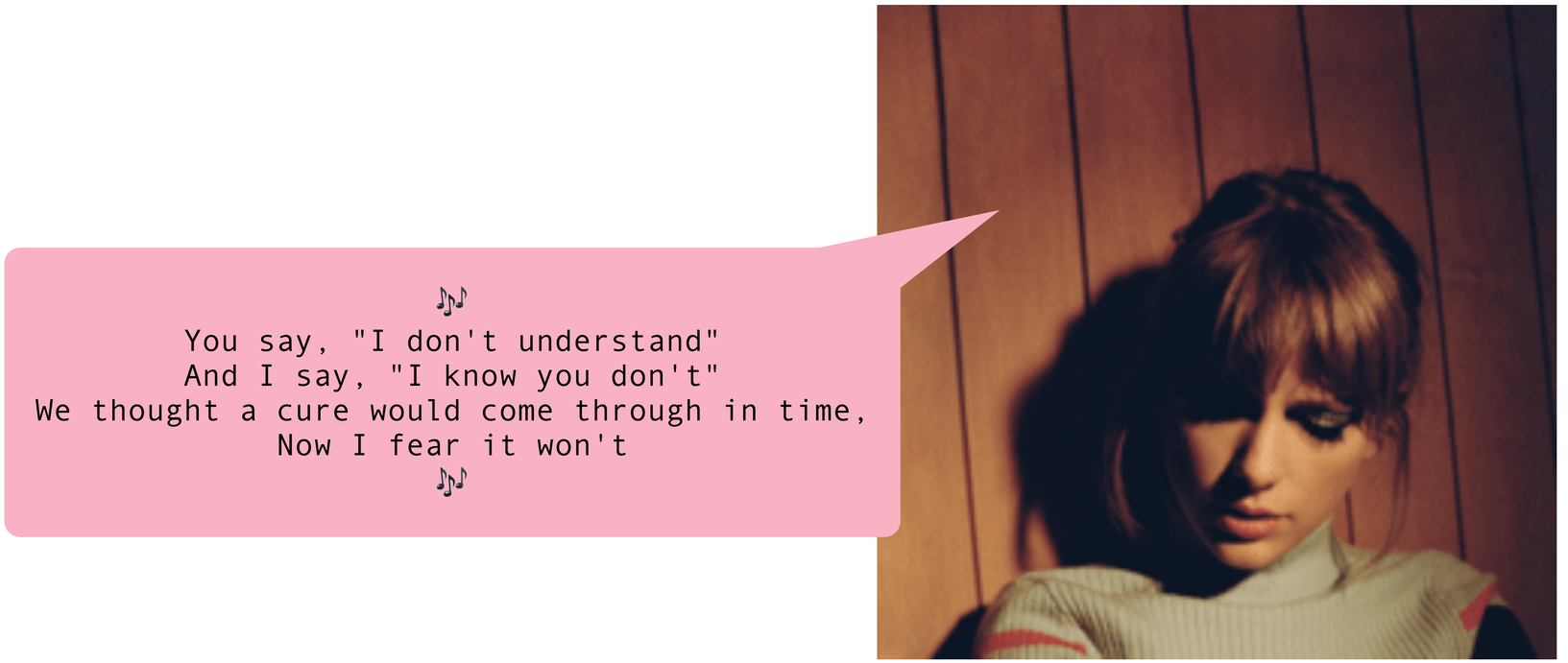
- Devised four top-line metrics that balanced performance state-of-the-art and understanding of the system with quantifying user experience in a way that allowed us to gain buy-in
  - Keypress Lag ("Input delay")
    - Perennial KR-level metric, in some form
  - Channel switch time
  - Number of JavaScript "long tasks" (> 50 ms)
  - React/Redux Loop time

♪♪
Time, mystical time
Cuttin' me open, then healin' me fine
♪♪

Cool, how did we start making it better?

React and Redux Deep-Dive

To understand how the system was scaling and breaking, we needed a deeper understanding of the libraries and how they worked under-the-hood



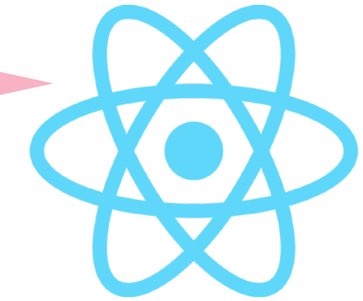
React and Redux 101

- **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

```
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'
  }}
/>
```

♪♪
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)
♪♪



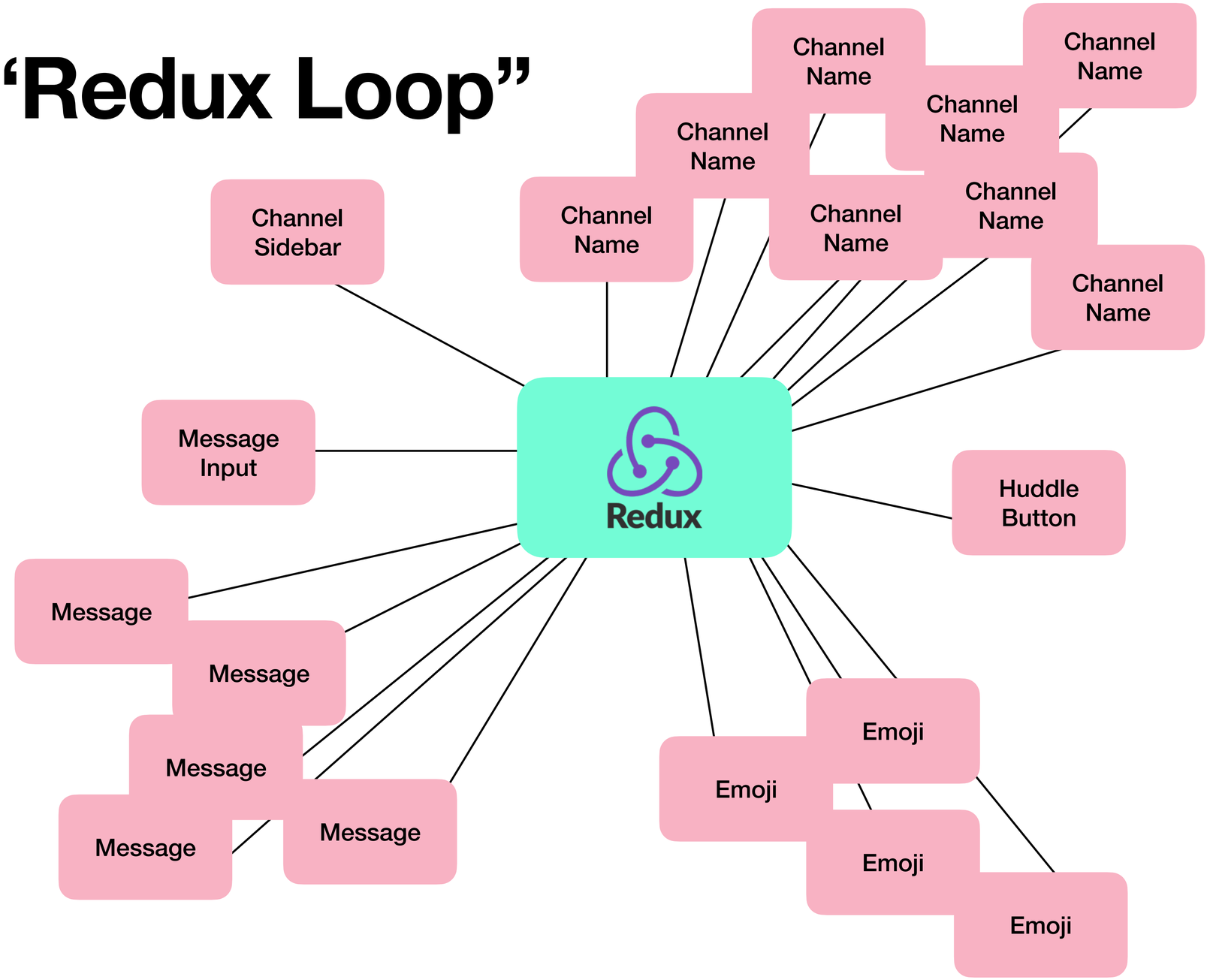
React and Redux 101

- **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

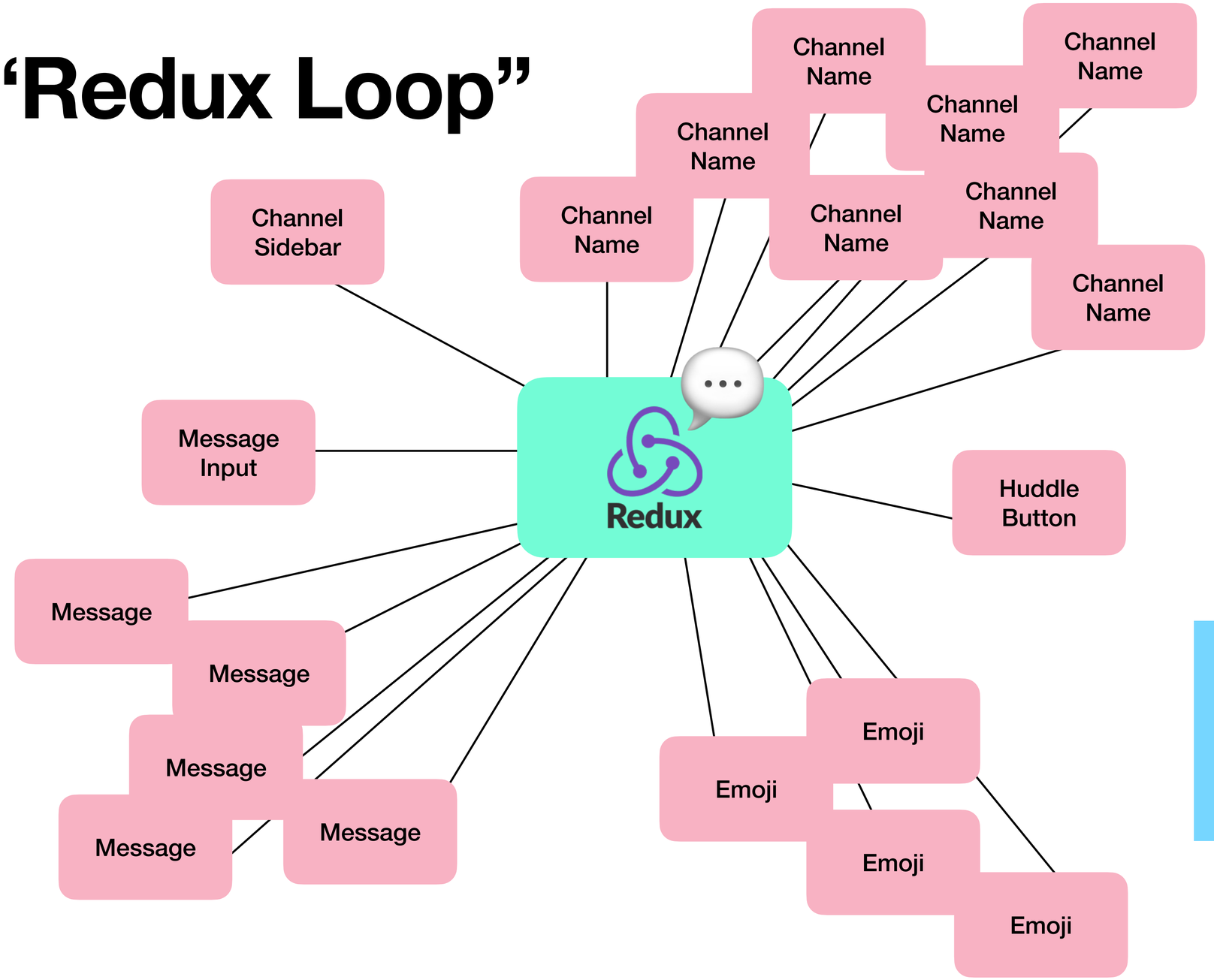
```
function Avatar({ id, size }) {
  const person = useSelector((state) =>
    getPersonById(state, id));

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

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

The “Redux Loop”

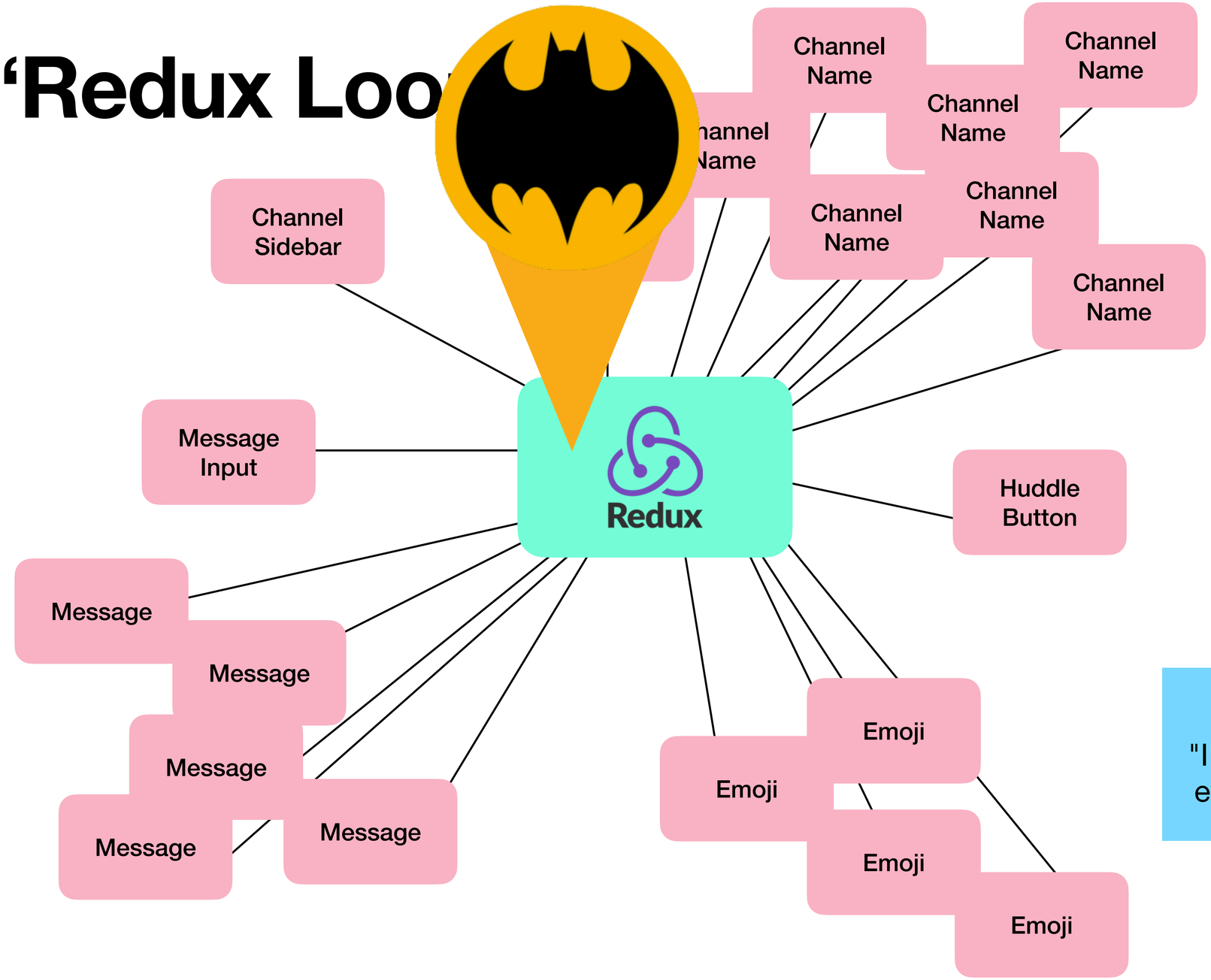


The “Redux Loop”



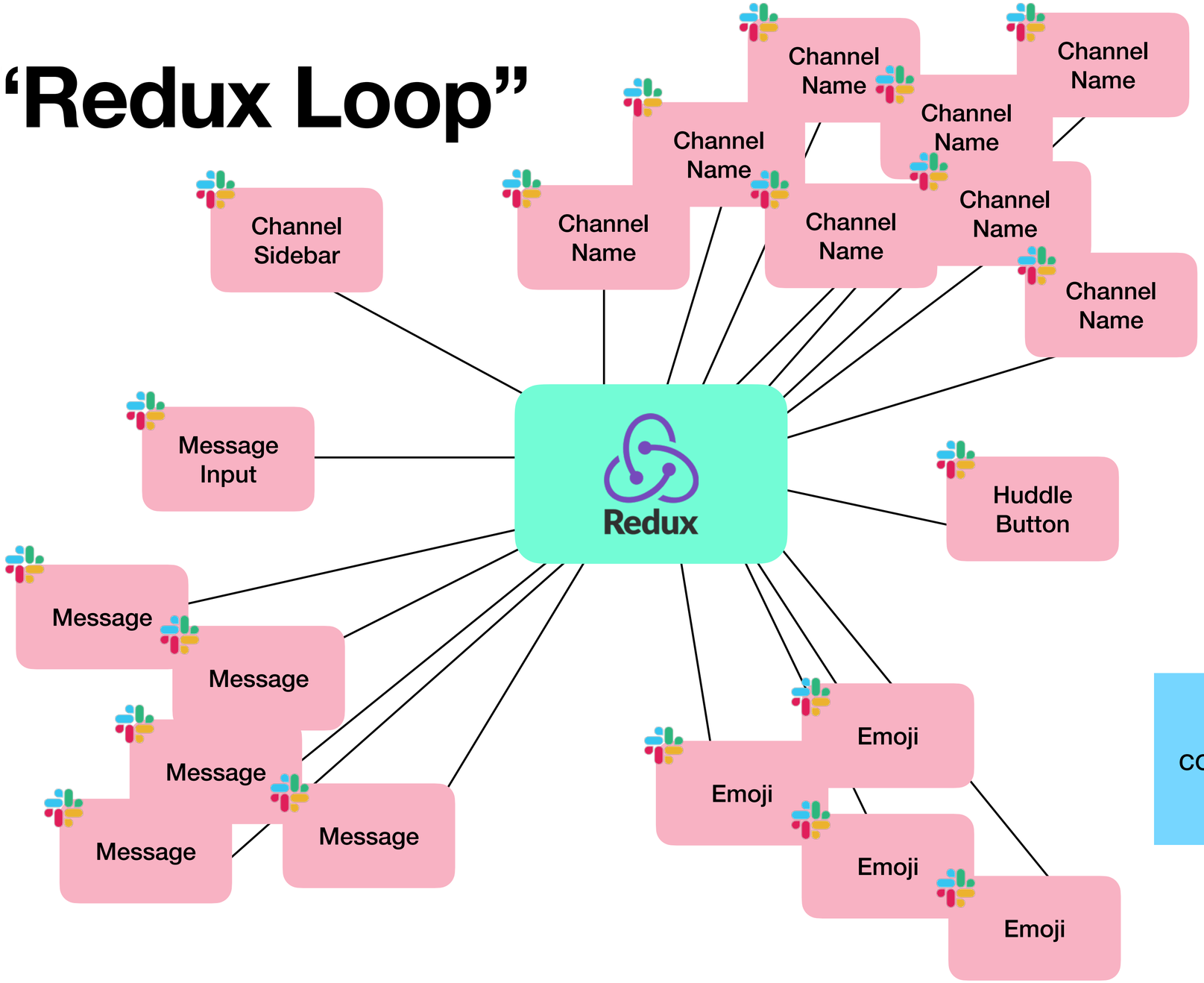
Redux state gets updated due to an API call, websocket event, user interaction, etc.

The “Redux Loo



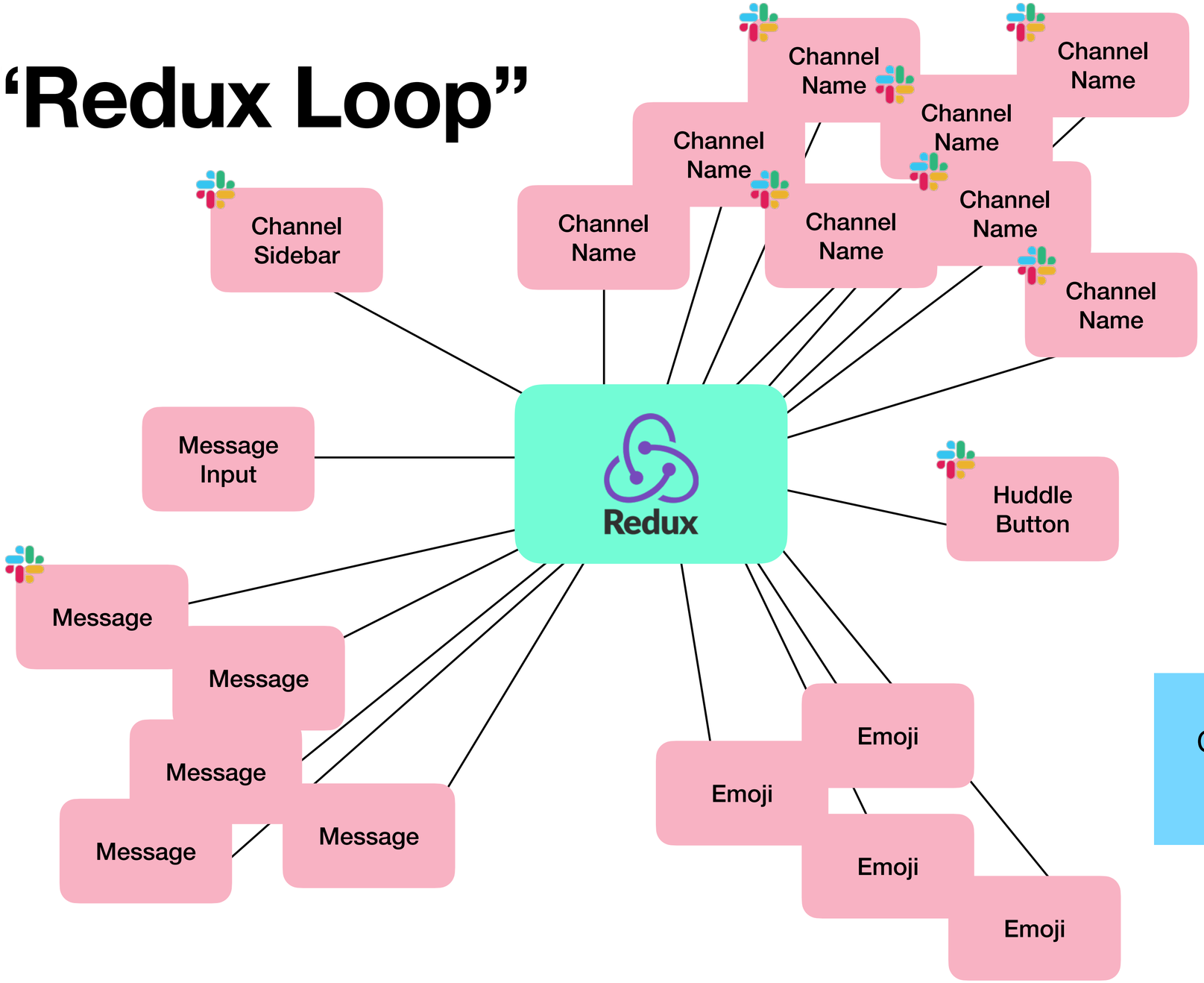
Redux sends out an "I've changed!" notification to every connected component

The “Redux Loop”



Components recalculate connected props to see if any values have changed

The “Redux Loop”

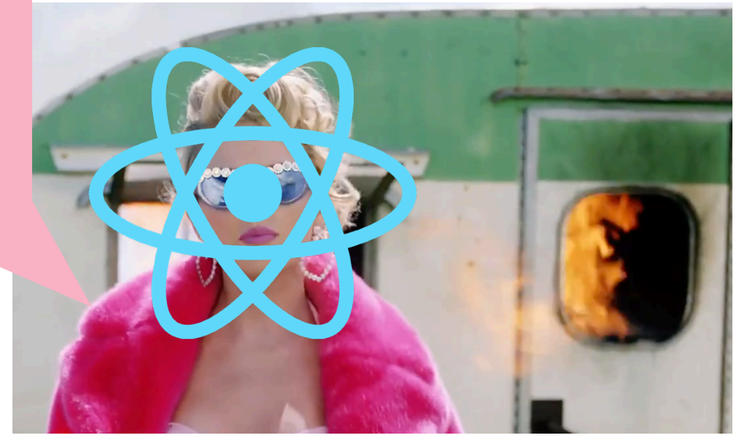


Components with changed props will re-render

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)

🎶
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
🎶

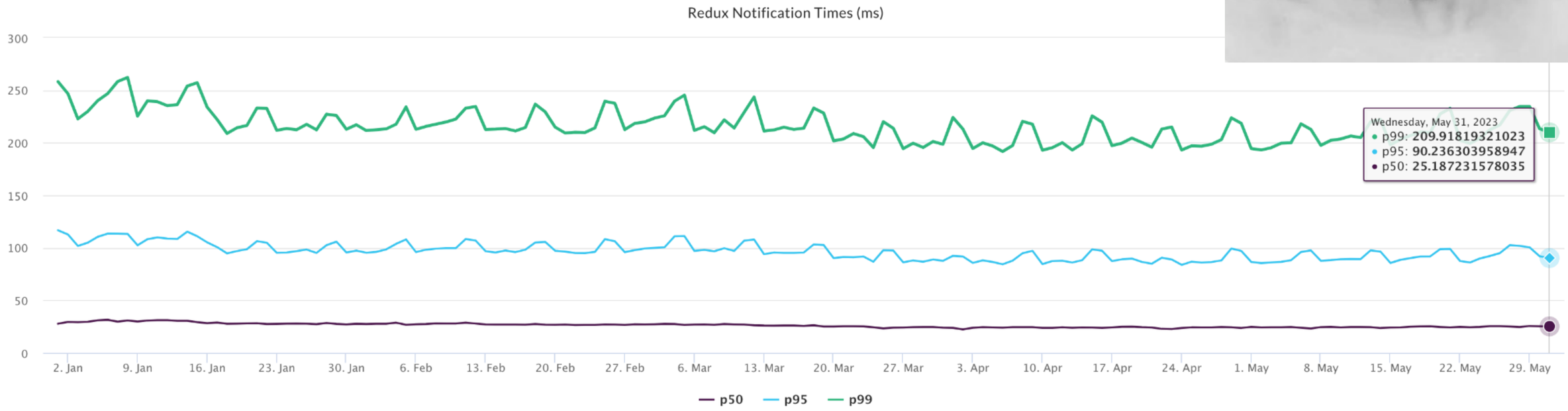


Redux Loop Scoop

Ideally, all Redux work happens within 16ms so we're not dropping frames and blocking inputs, but

- p50 at 25ms
- p99 at 209ms

♪♪
I love you
It's ruining my life
♪♪



The Big Question:
Do we keep React + Redux?

Dream State

- Finer-grained subscription
- Supports multiple stores (client-level and workspace-level)
- Not a total re-write?
- No seriously, finer-grained subscription

Why Keep React and Redux?

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

- Me, about 5 minutes ago

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

maybe not.

but wait the client perf team is only two people!

Scaling Ourselves

How do we fix all of these performance papercuts while also attending to escals, on-call, and other performance issues outside of the React/Redux ecosystem?

Step 1: Assemble a performance cabal!

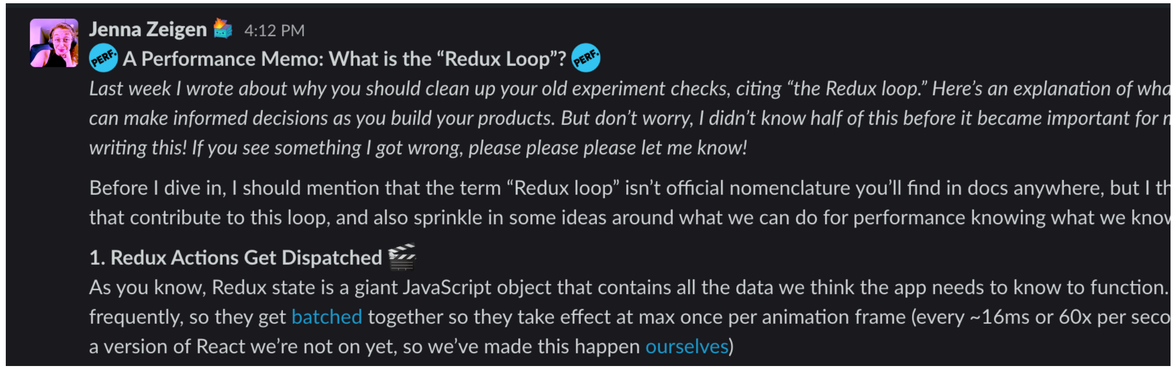
Step 2: …

Introducing:
A “Performance Program”

engineers fundamentally want to create performant software, so let’s give them the tools to set them up for success

Education and Evangelism

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



Jenna Zeigen 4:12 PM

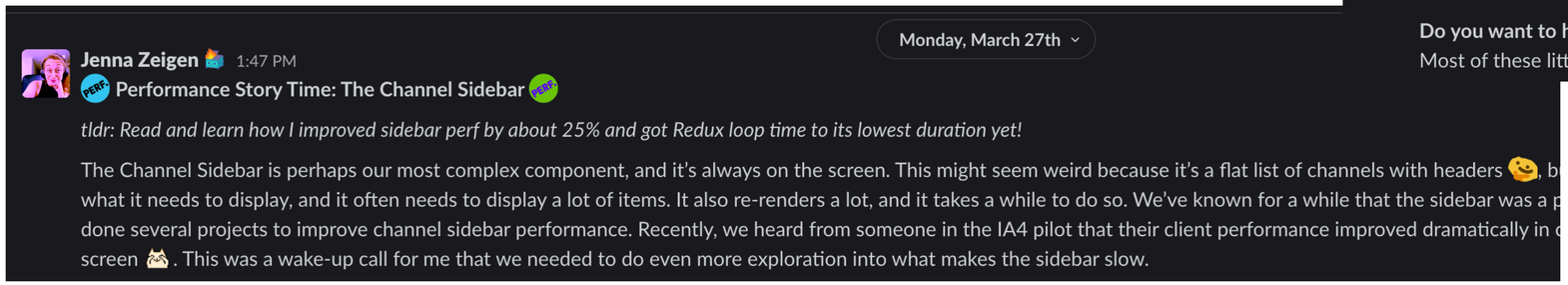
A Performance Memo: What is the "Redux Loop"?

Last week I wrote about why you should clean up your old experiment checks, citing "the Redux loop." Here's an explanation of wha
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 r
writing this! If you see something I got wrong, please 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 th
that contribute to this loop, and also sprinkle in some ideas around what we can do for performance knowing what we knov

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 seco
a version of React we're not on yet, so we've made this happen ourselves)



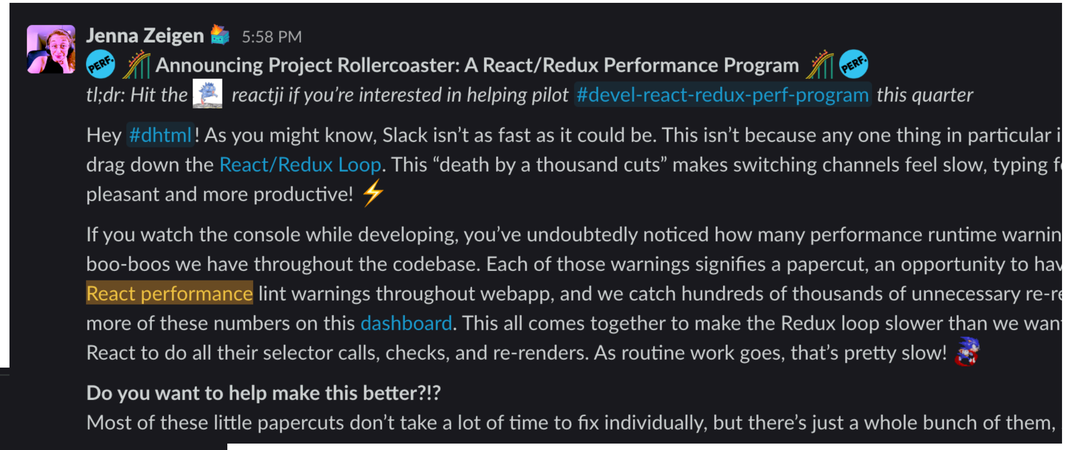
Monday, March 27th

Jenna Zeigen 1:47 PM

Performance Story Time: The Channel Sidebar

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, b
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 p
done several projects to improve channel sidebar performance. Recently, we heard from someone in the IA4 pilot that their client performance improved dramatically in c
screen. This was a wake-up call for me that we needed to do even more exploration into what makes the sidebar slow.



Jenna Zeigen 5:58 PM

Announcing Project Rollercoaster: A React/Redux Performance Program

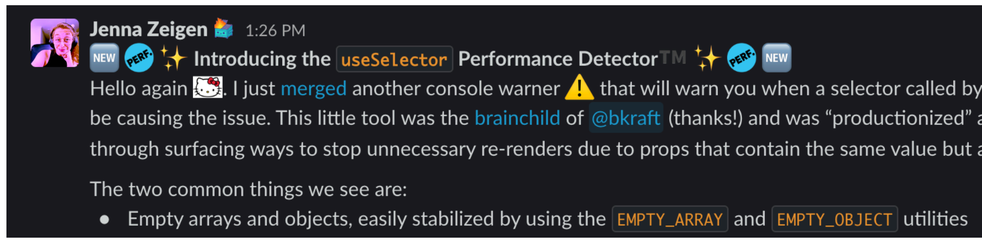
tl;dr: Hit the reactji if you're interested in helping pilot #devel-react-redux-perf-program this quarter

Hey #dhtml! As you might know, Slack isn't as fast as it could be. This isn't because any one thing in particular i
drag down the React/Redux Loop. This "death by a thousand cuts" makes switching channels feel slow, typing f
pleasant and more productive!

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

Do you want to help make this better?!?

Most of these little papercuts don't take a lot of time to fix individually, but there's just a whole bunch of them,



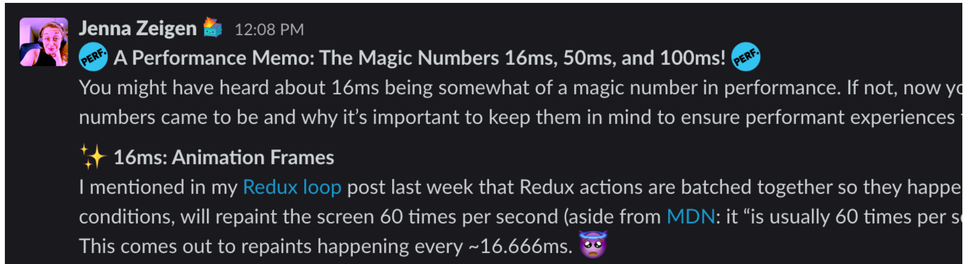
Jenna Zeigen 1:26 PM

Introducing the `useSelector` Performance Detector™

Hello again. I just merged another console warner that will warn you when a selector called by
be causing the issue. This little tool was the brainchild of @bkraft (thanks!) and was "productionized" a
through surfacing ways to stop unnecessary re-renders due to props that contain the same value but a

The two common things we see are:

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



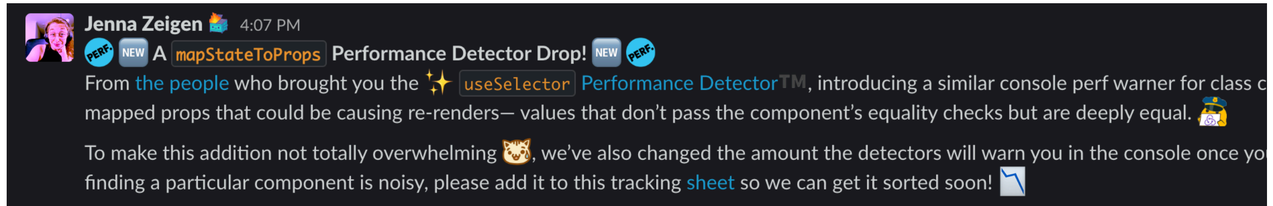
Jenna Zeigen 12:08 PM

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

You might have heard about 16ms being somewhat of a magic number in performance. If not, now yo
numbers came to be and why it's important to keep them in mind to ensure performant experiences

16ms: Animation Frames

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



Jenna Zeigen 4:07 PM

A `mapStateToProps` Performance Detector Drop!

From the people who brought you the `useSelector` Performance Detector™, introducing a similar console perf warner for class c
mapped 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 yo
finding a particular component is noisy, please add it to this tracking sheet so we can get it sorted soon!

Performance Guardrails

- Lint rules that highlight code that has the potential to cause unnecessary re-renders, e.g. `react-perf/jsx-no-new-object-as-prop`
- Runtime console warnings for performance opportunities best caught at runtime, e.g. unstable connected prop calculations
- And, yes, finally, regression testing

♪♪
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!
♪♪

For more nitty-gritty details about what we did, check out my QCon talk from last year

Performance Toolbox

- Provided “stability selectors” that provide engineers with performant options
- Promoted use of existing performance helpers from React like `useMemo`, `useCallback`, and `React.memo` (no we weren’t on React 18)
- Encouraged use of `EMPTY_ARRAY` and `EMPTY_OBJECT` constants in place of unstable `[]` and `{}`

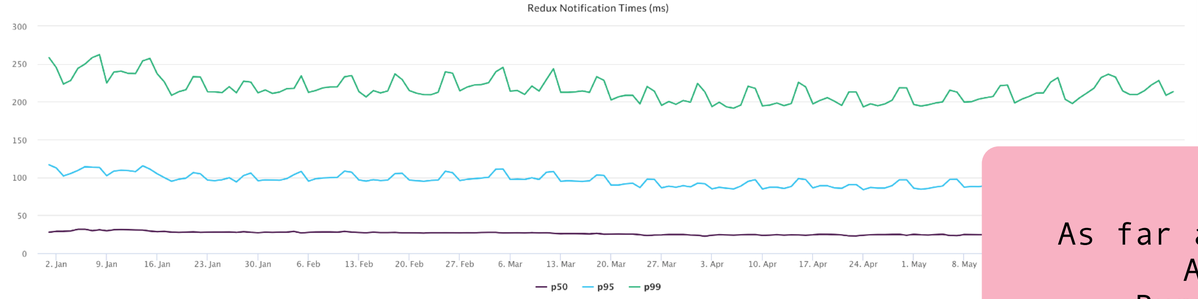
For more nitty-gritty details about what we did, check out my QCon talk from last year

Flamegraph, meet Burndown Chart

Desktop Performance Burndowns by jenna

Edit

Redux Notification Times (ms)



Total React Runtime Warnings

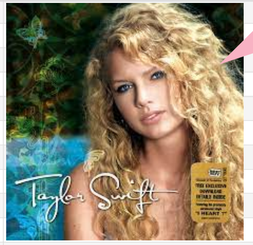
This data comes from Slack engineers' development environments and gives us a rough sense of components that are re-rendering unnecessarily in the wild. Since dev use does not match the averag
numbers should not be taken too seriously but can be considered alongside common sense intuitions about what components might be on the page and re-rendering a lot in production.

React Perf useSelector Runtime Warnings

#	Component	Re-render Count	Owner
1		728514	CliCap
2		196699	Files
3		26250	Platform
4		19791	CSC
5		17962	(null)
6		17213	CSC
7		10959	Platform
8		10431	Platform
9		10179	(null)
10		8997	(null)

React Perf mapStateToProps Runtime Wa

#	Component:Prop	
1		[illegible]
2		37213
3		37213
4		32896
5		23325
6		22746
7		18857
8		11307
9		10591
10		8722



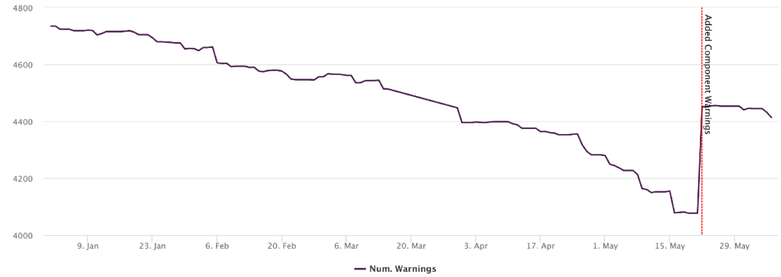
🎶

As far as I'm concerned, you're just
Another picture to burn
Burn, burn, burn, baby, burn
Just another picture to burn
Baby, burn

🎶

Total React Perf Lint Warnings

Total React Perf Lint Rule Counts Over Time Graph



Latest React Perf Warning Total

4,413

Performance Program Performance Review

- 44% net decrease in performance lint rule violations
- 40% fewer slow (>250ms) channel switches
- 50% gain in Redux performance
- #escal-desktop-performance became… quiet

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!

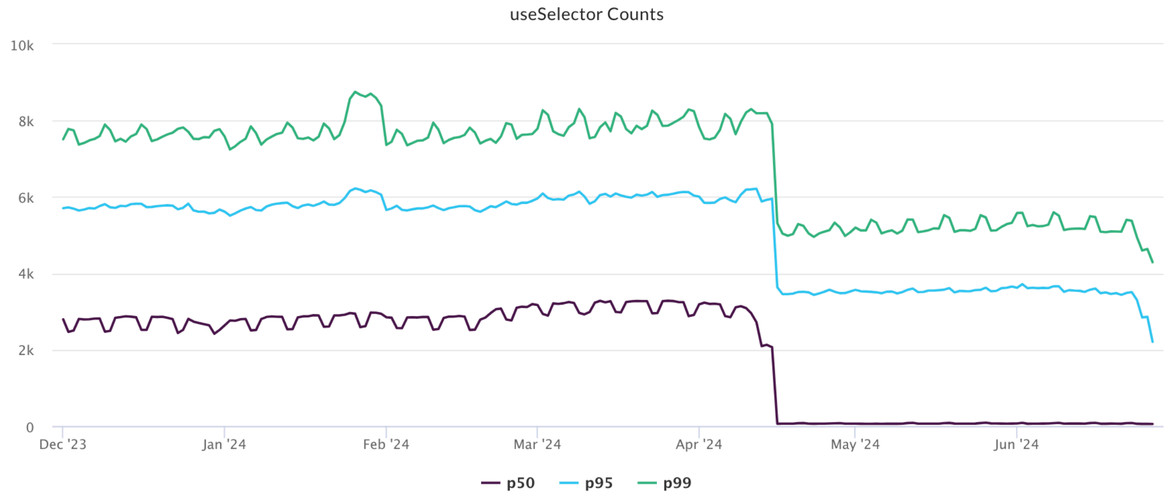
♪♪
Can't you see that I'm the one
Who understands you?
Been here all along
So, why can't you see?
You belong with me
♪♪

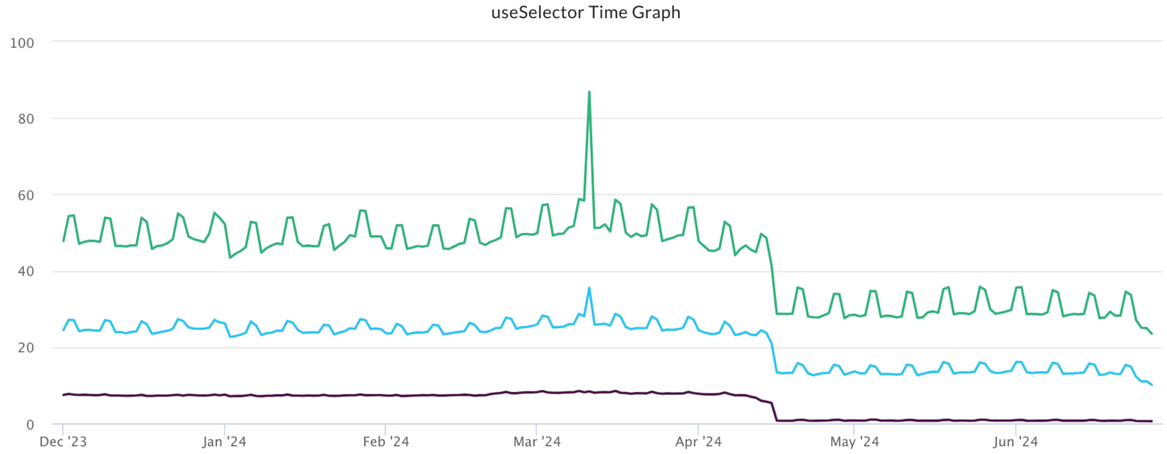
your performance culture cannot be a hero culture if you want it to scale

an update 🥲

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

We ended up figuring out finer-grained subscription in Redux, which was a far bigger, and quicker, win than chipping away at the papercuts





🎶
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!
🎶

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

We ended up figuring out finer-grained subscription in Redux, which was a far bigger, and quicker, win than chipping away at the papercuts

DESKTOP_PERF Redux Subscriber Notification Timings v2 guardrail performance fe

Reprocess Metric Group ···

Measurements of the redux subscriber notification timings. Derived from performance tracing data. Addresses filtering of autospan traces. https://corp.quip.com/5WamARZdbqZ5/Performance-Guardrail-Metric-Documentation.

2024-07-16	control	treatment				
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%

(statistical significance is brat)

♫♪

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

♪♫

oh no did we
waste three years
of everyone's life? 

What I'm bringing to Notion

- Focus on understanding the architecture deeply
- Learn from teammates about how performance issues are impacting users
- Make performance accessible
- Scale efforts by empowering teams to own performance of areas they own

♫♫
Let's fast forward to one collab software company later
Will I view chrome profiles and once again build a perf culture
♫♫

fin(e)


♪♪
Come one, come all
It's happening again
The heroic hunger descends
We'll tell no one
Except all of our friends
We must know
How did it end?
♪♪

Thank you.

jenna.is/at-lead-dev
@zeigenvector