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~~

jenna.is/at-lead-dev

@zeigenvector

~~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

Performance?!

What?

Make the app go fast! Quick load speeds! Reduce latency! Buttery smooth interactions! Low memory footprint! Not a lot of CPU usage!

How?

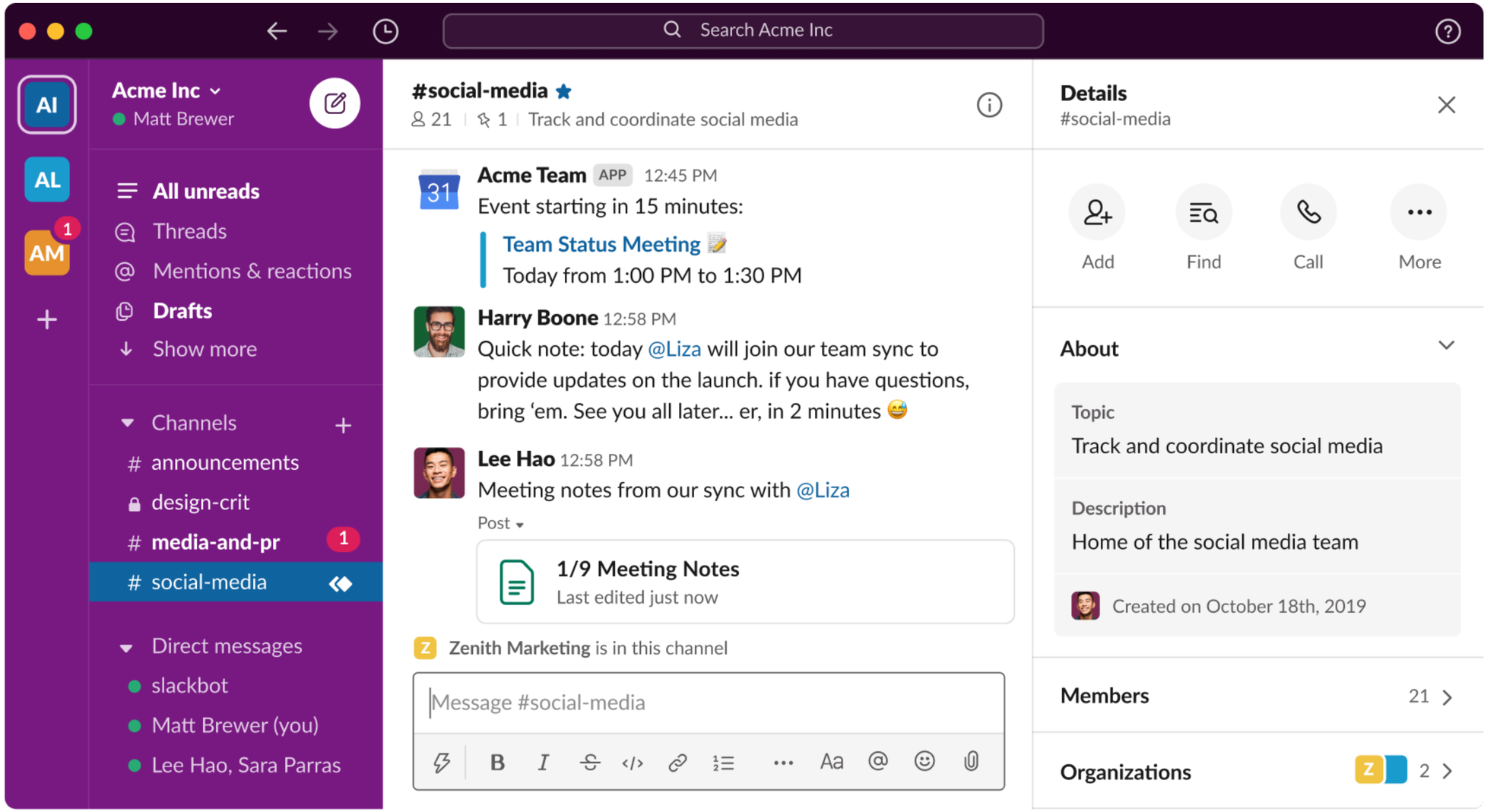
Doing less work!

Why?

✨ So our users have a great experience! ✨

First some stuff about the Slack Desktop app

Slack, a React app on your Desktop



Slack, a Long-Lived Single Page App

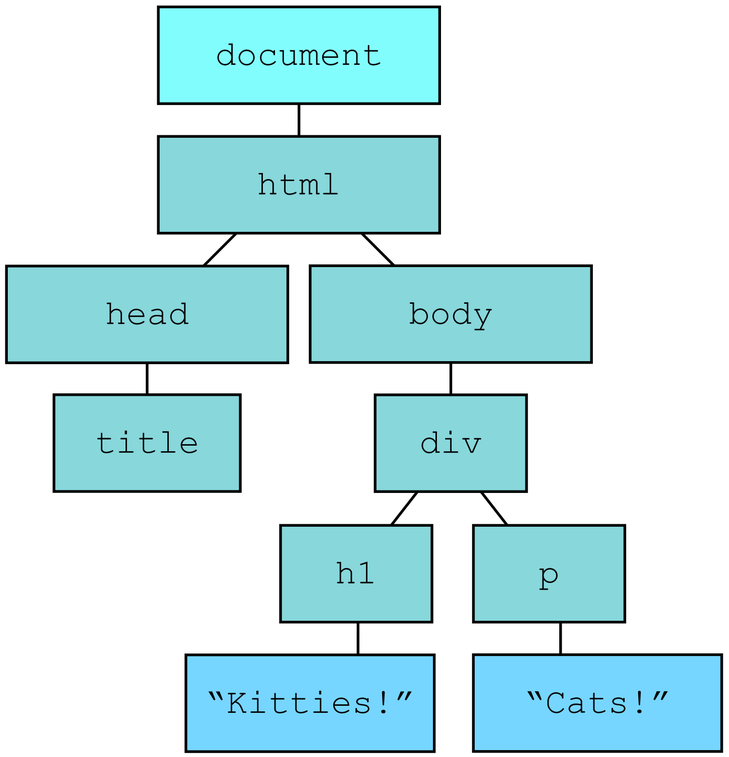
- Very long sessions
- No clear "page load" except at boot
- A lot of dynamic content
- Frequent updates via websocket

🎶
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

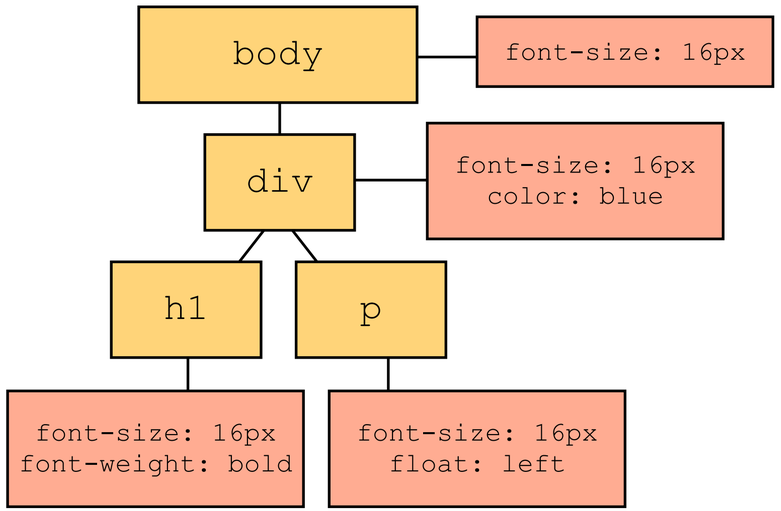
How Do Browsers Even?

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



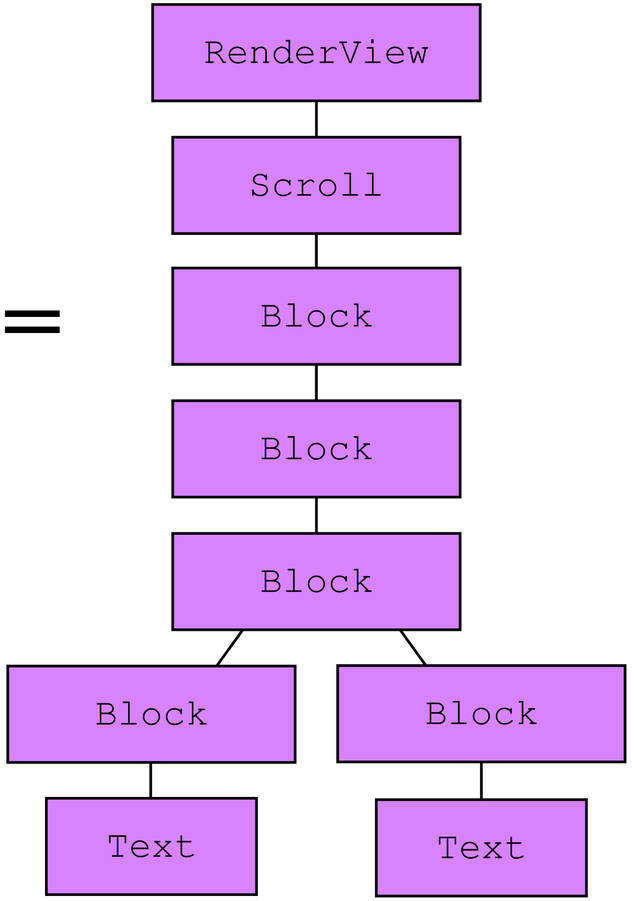
DOM

+



CSSOM

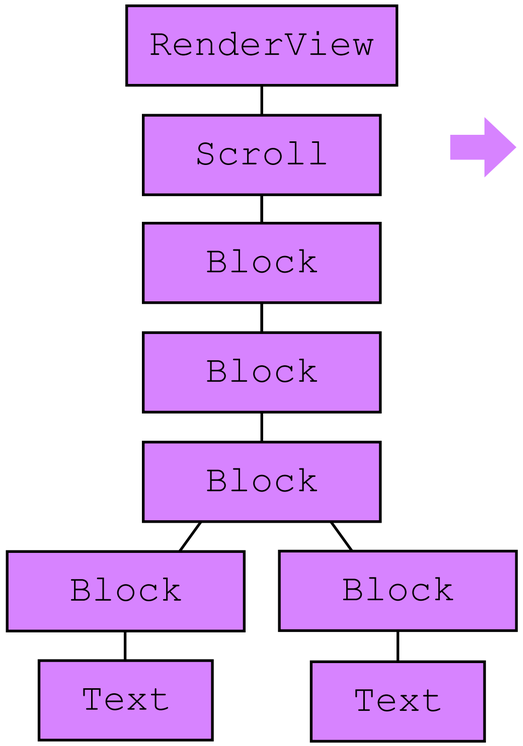
=

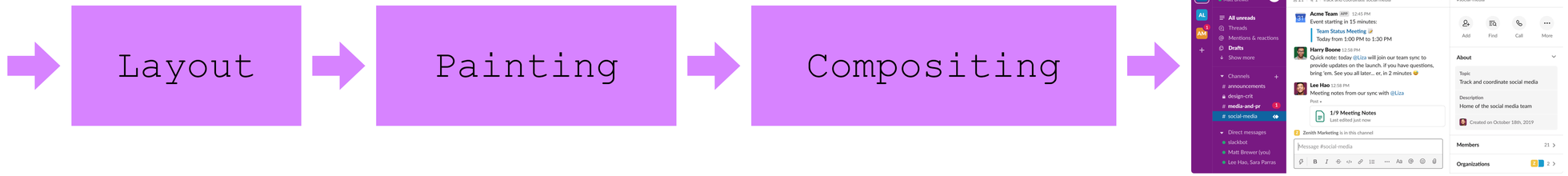


Render Tree

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 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** ✨

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

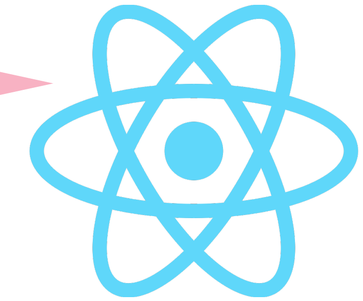
React and Redux 101

- **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

```
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)
♪♪



Source: https://react.dev/learn/passing-props-to-a-component

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

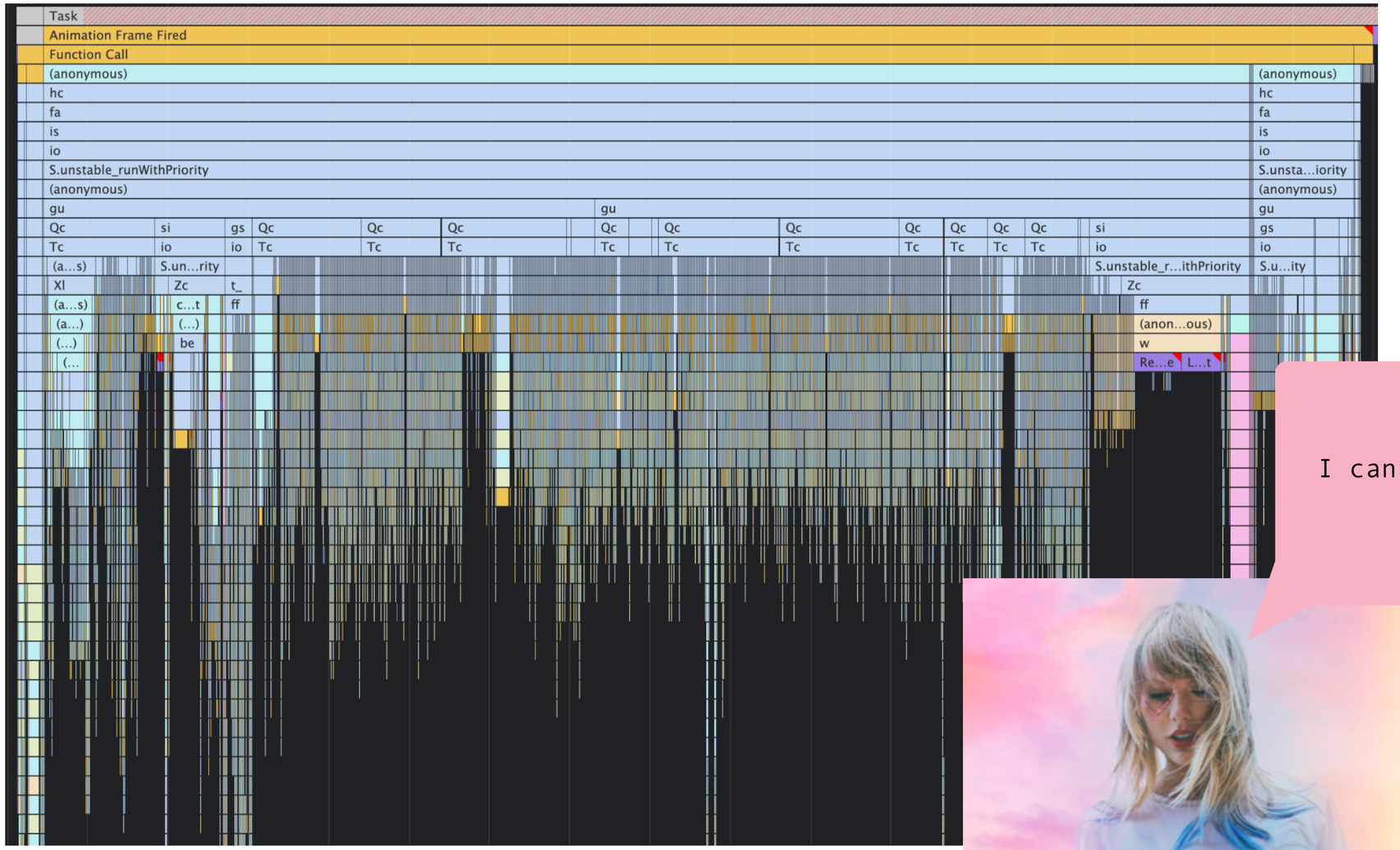
```
function Avatar({ id, size }) {
  const person = useSelector((state) =>
    getPersonById(state, id));

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

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

Source: https://react.dev/learn/passing-props-to-a-component

Ok, so those performance issues?

Papercuts?



♫♪

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

♫♪

Metrics, Metrics, Metrics

- Devised four top-line metrics that balanced performance state-of-the-art and understanding of the system with quantifying user experience
  - Keypress Lag ("Input delay")
    - Perennial KR-level metric, in some form)
  - Channel switch time
  - Number of "Long tasks" (> 50 ms)
  - Redux Loop time

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

Cool, how did we start making it better?

Doing Less Work!

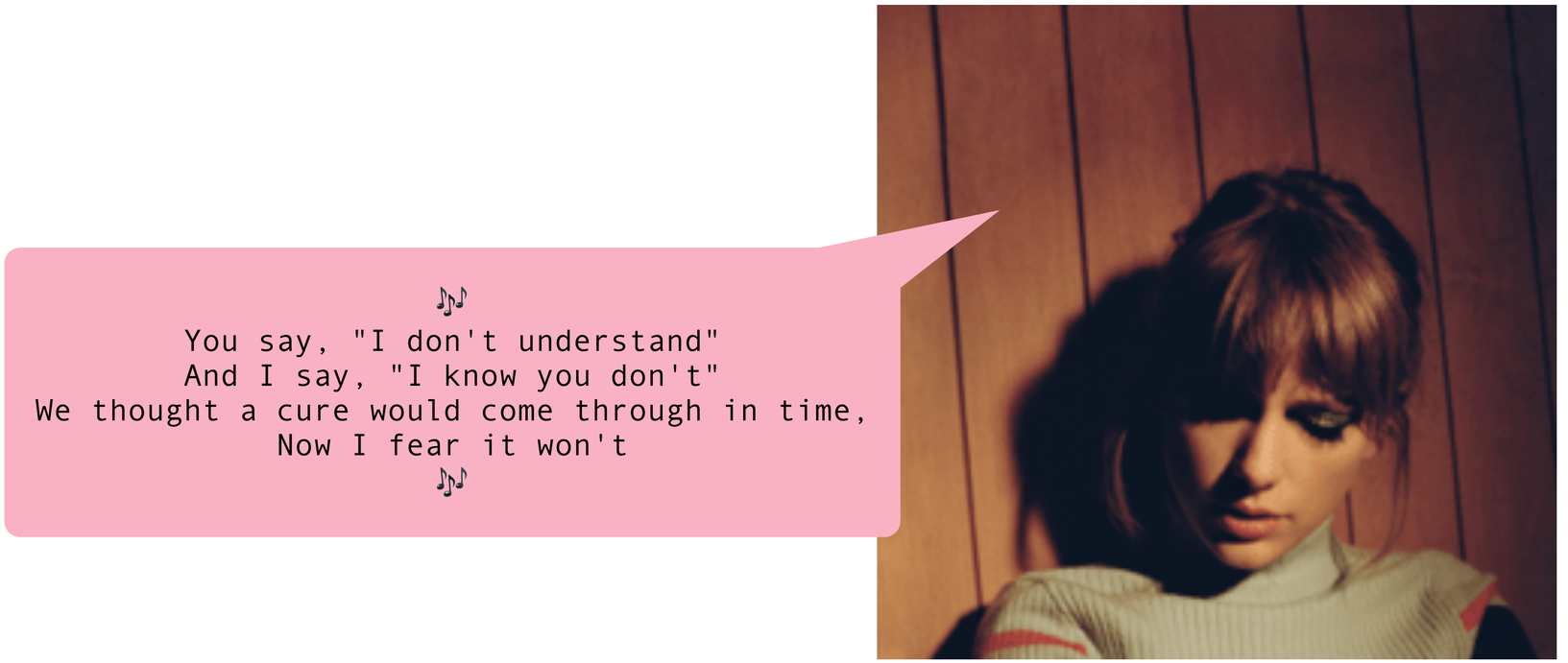
Thanks!

Thanks! (lol jk 😂)

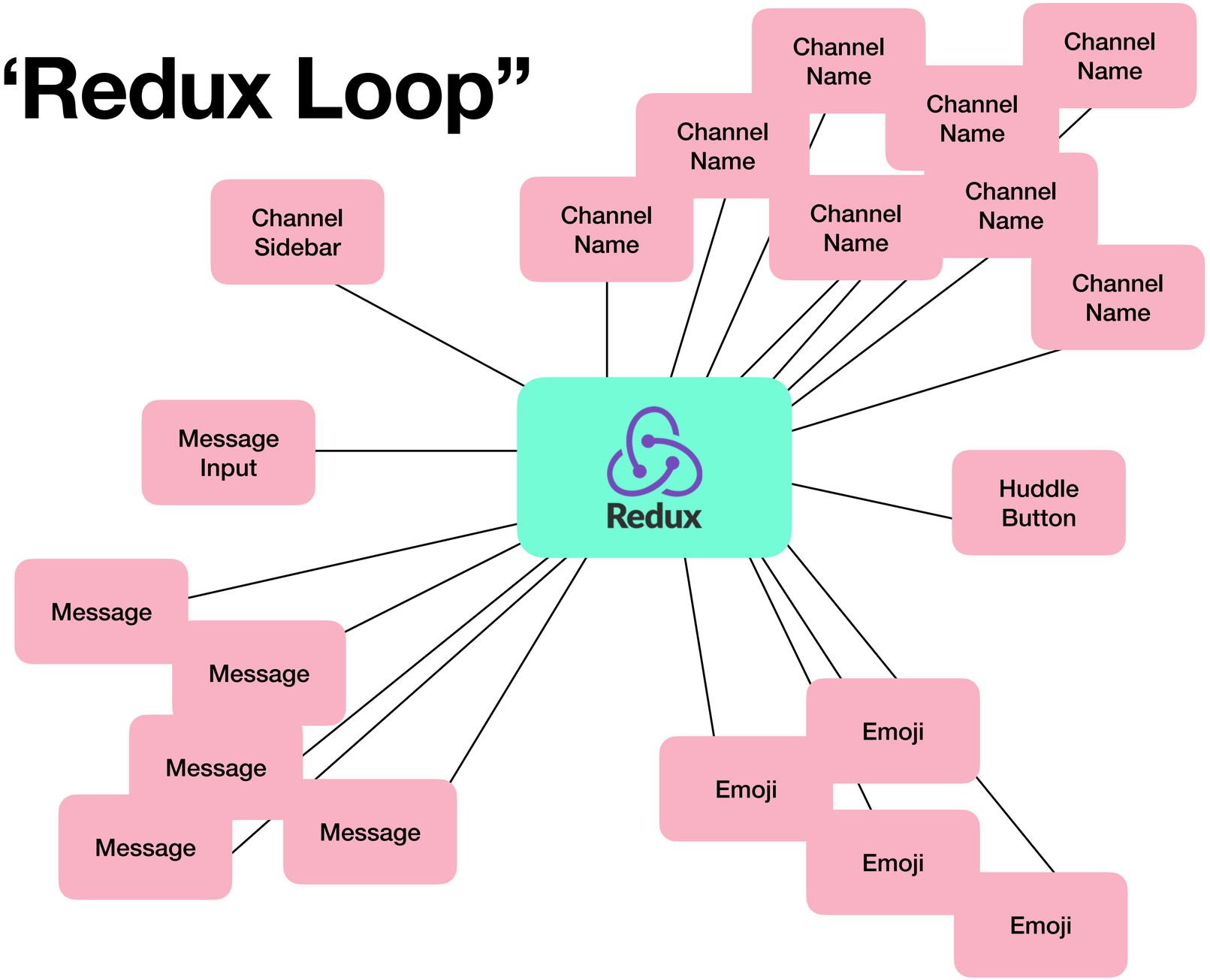
Doing Less Work!!!!

React and Redux Deep-Dive

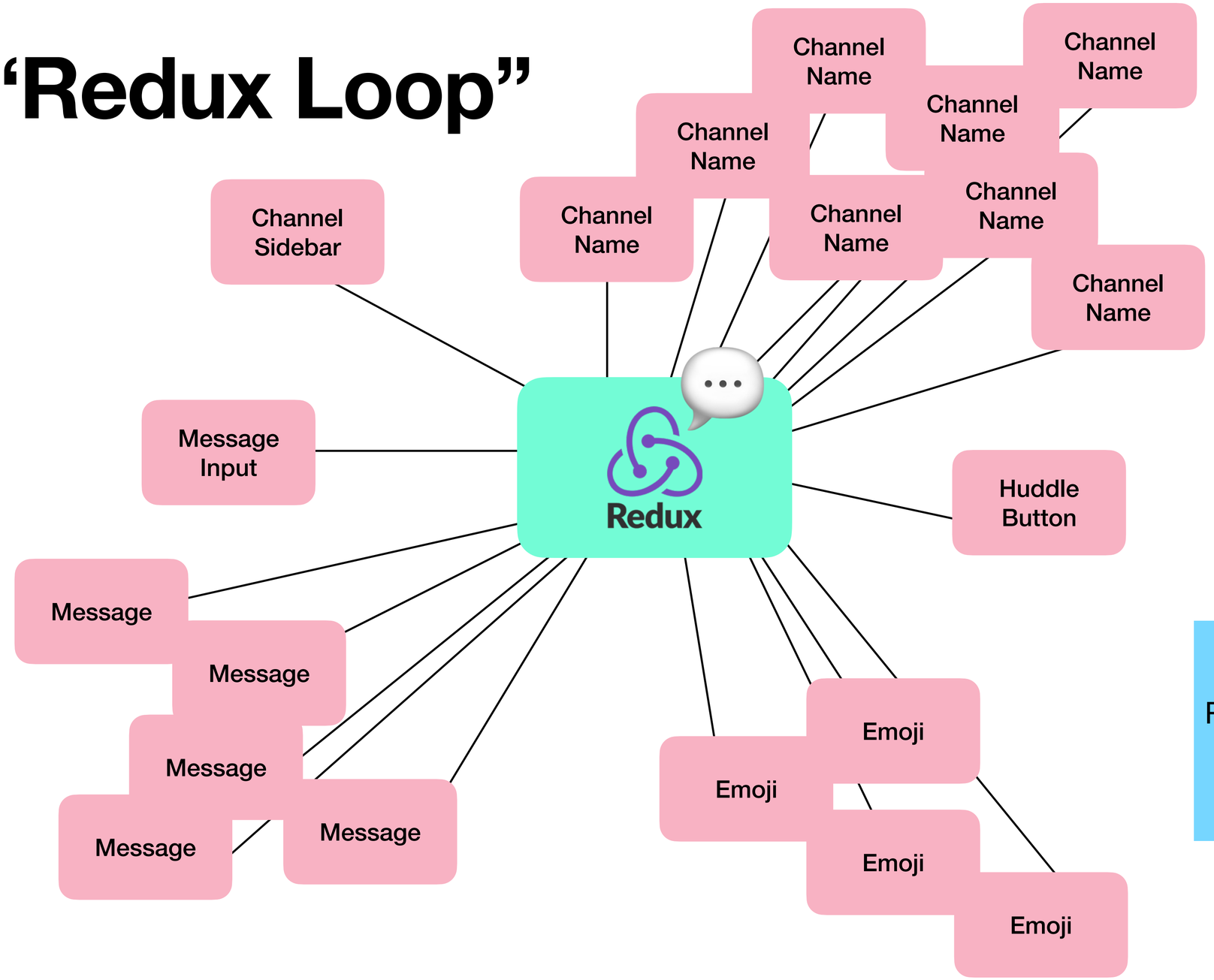
- For us to understand how the system was breaking, we needed a deeper understanding of the libraries and how they worked under-the-hood, via reading the docs and reading the code



The “Redux Loop”

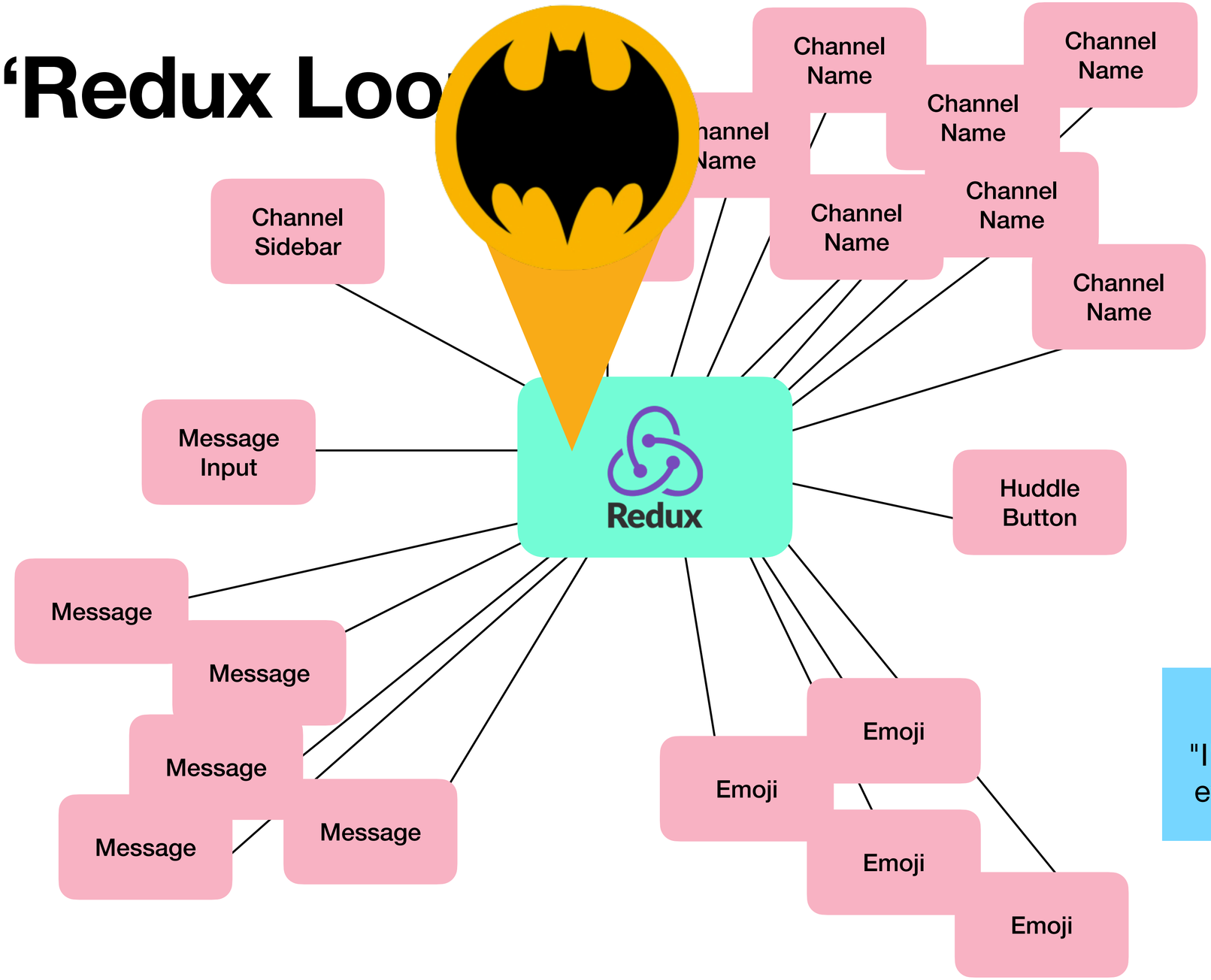


The “Redux Loop”



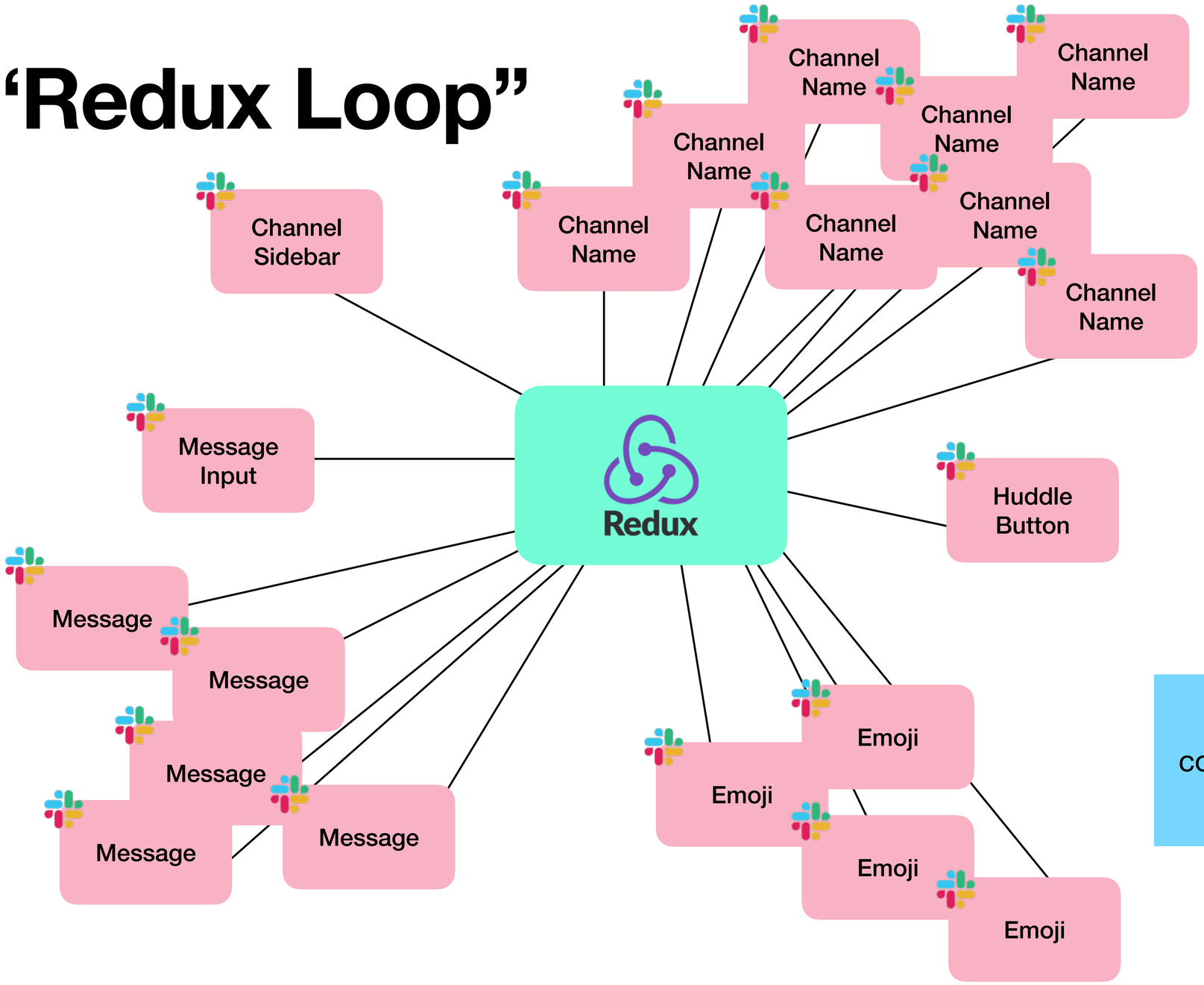
Actions are dispatched to Redux, causing “reducers” to run, which updates Redux state

The “Redux Loo



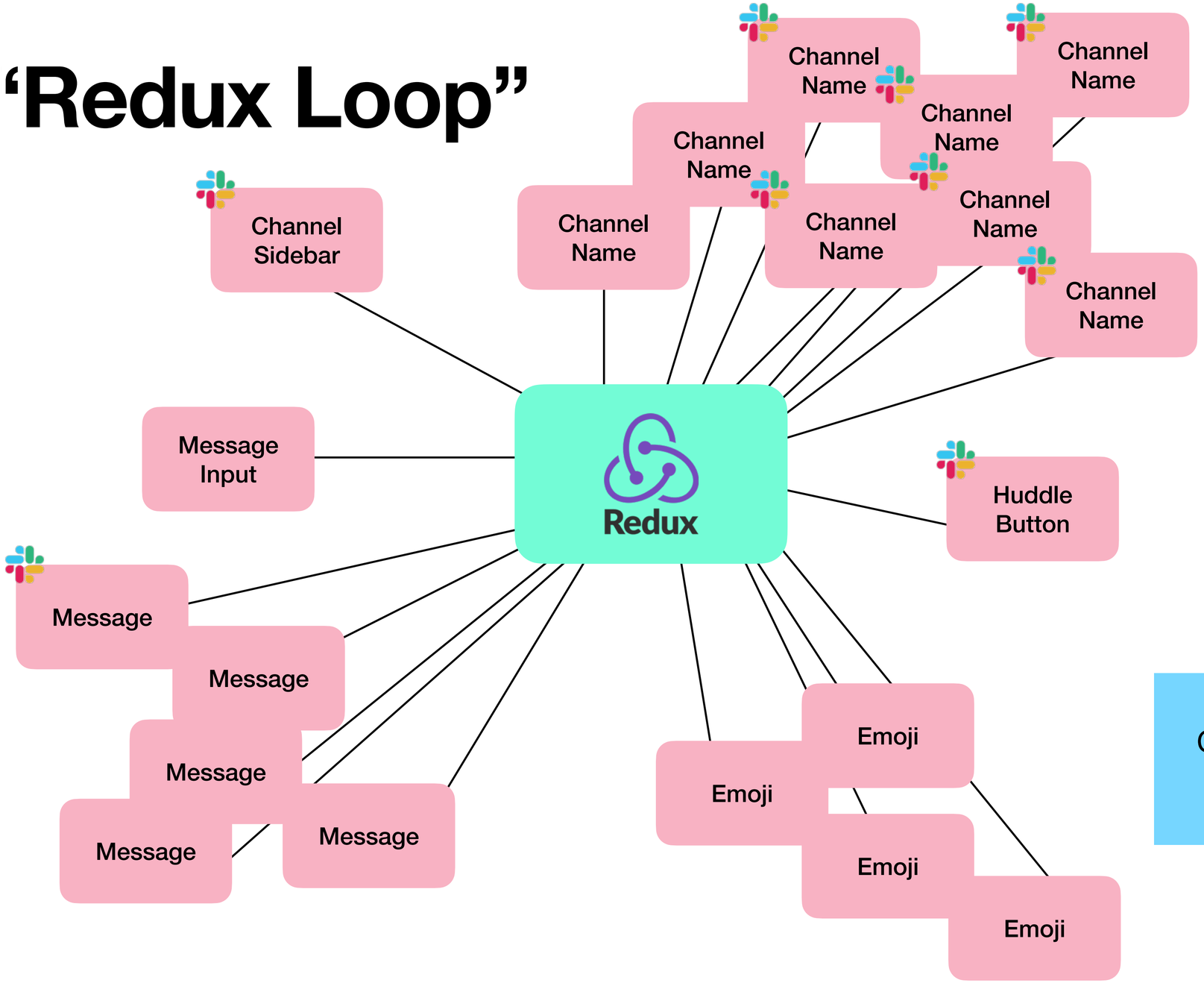
Redux sends out an "I've changed!" notification to every connected component

The “Redux Loop”



Components recalculate connected props to see if any values have changed

The “Redux Loop”

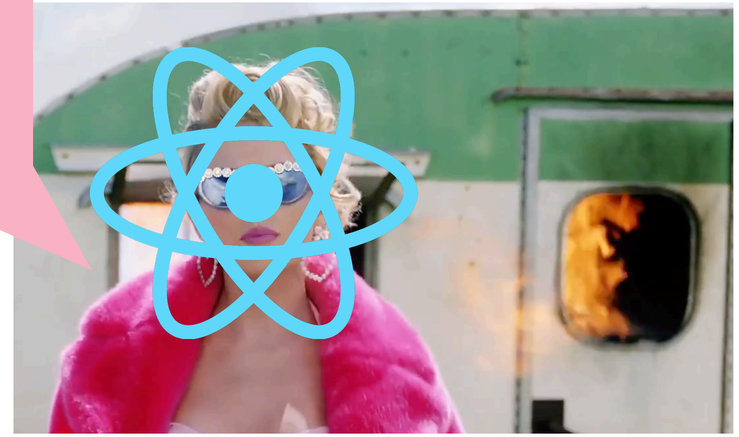


Components with changed props will re-render

Where Does Performance Break Down

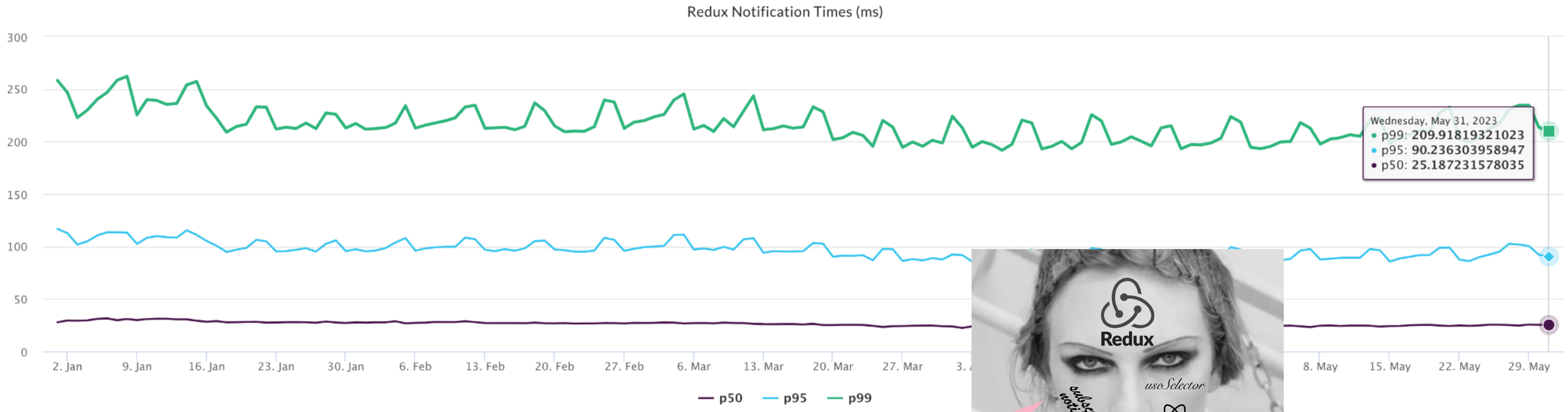
1. Every change to Redux results in a Redux notification firing
2. Redux notification means all selectors are running, which means spending too long running selectors
3. Spending too long re-rendering components (often, unnecessarily)

♫♪
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 an animation frame (16ms) so we’re not dropping frames and blocking inputs, but… we weren't there yet!



The Big Question:
Do we keep React + Redux?

Ideal State

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

Why React and Redux, Still?

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

- Me, about 15 minutes ago

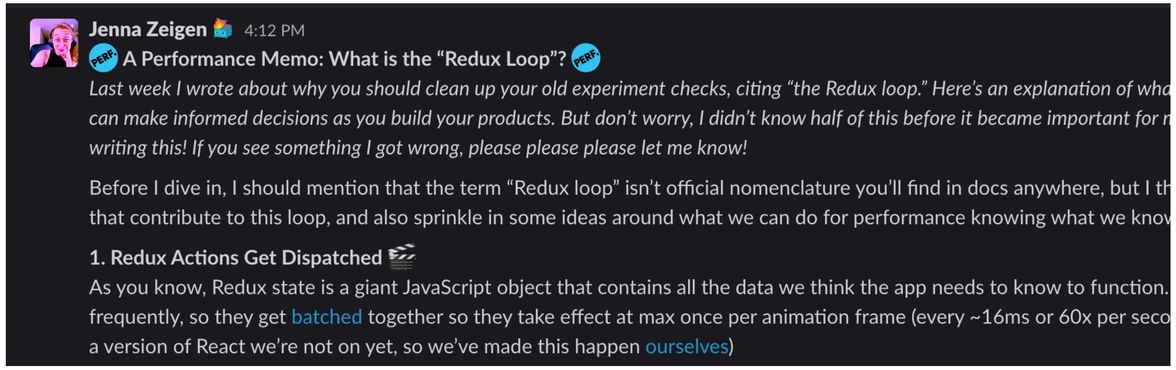
✨ **Is the cost of drastically changing our architecture worth it for the performance boosts?** ✨

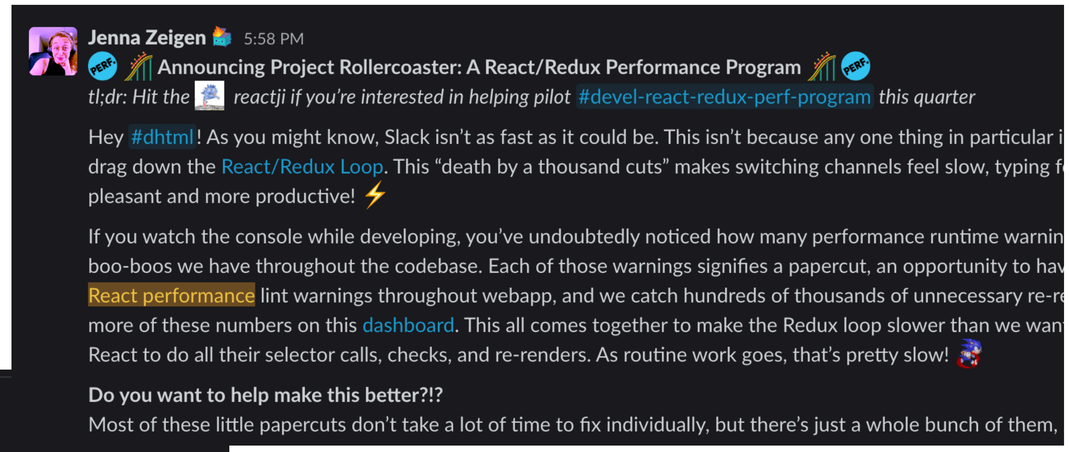
Everyone Loves Performance

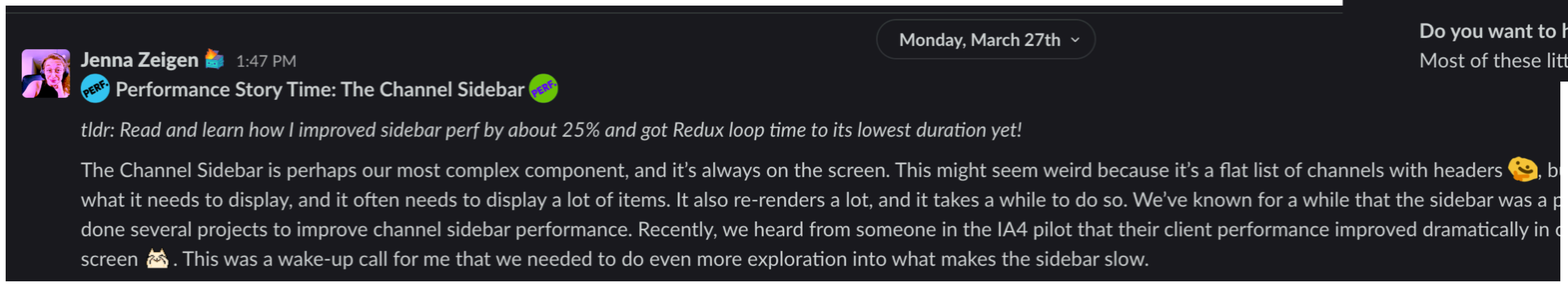
- ✨ **Engineers fundamentally want to create performant software, so let's give them the tools to set them up for success** ✨
- Creating a performance culture through education, tooling, and evangelism

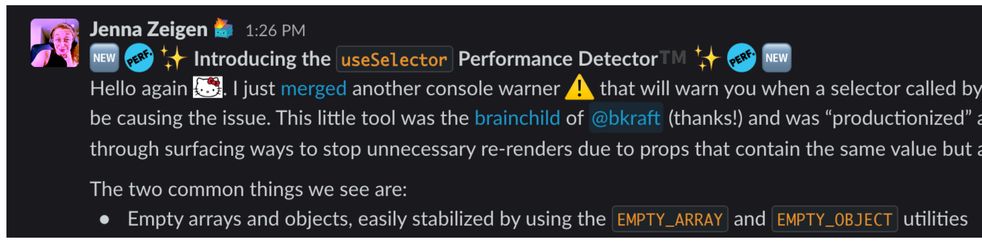
Education and Evangelism

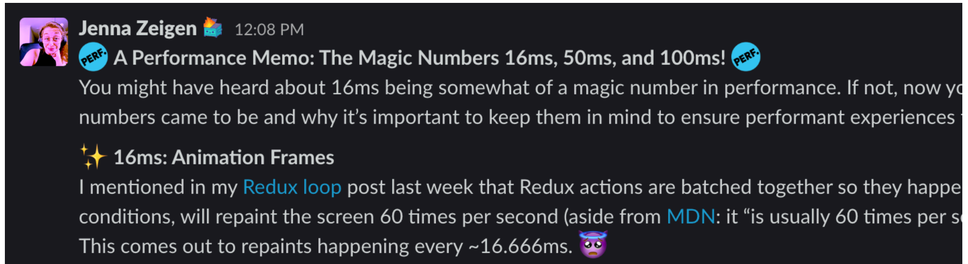
React and Redux abstract away internals but **understanding the system contextualizes and motivates performance work**

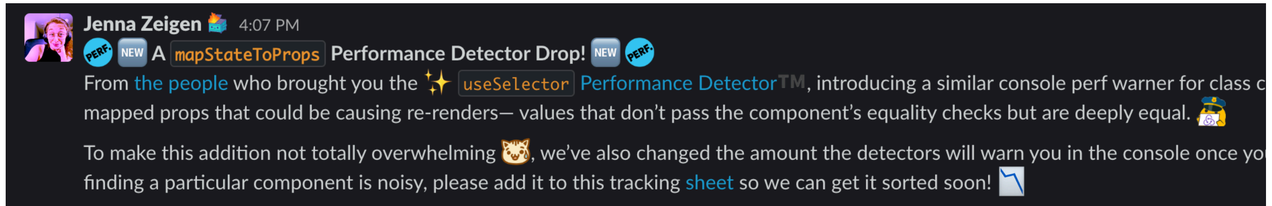












Lint Rules

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

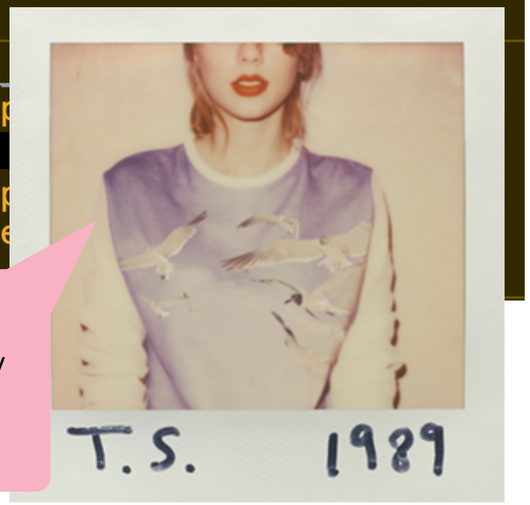
♪♪
But I got smarter, I got harder in the nick of time
Honey, I read all of your code, I do it all the time
I got a list of props, and yours is in red, underlined
I check it once, then I check it twice, oh!
♪♪

Runtime Console Warnings

Warnings in the Chrome console for performance opportunities best caught at runtime, such as unstable connected prop calculations

```
⚠ ▶Jun-7 17:24:58.144 [PERF DEBUG] (T5J4Q04QG) The toggle [redacted] (http   react_devtools_backend_compact.js:2367
[redacted] is finished as "on" and might be able to be cleaned up. We've
checked for this toggle 14400 times. :0:0
⚠ ▶Jun-7 17:24:58.176 [PERF DEBUG] A useSelector call in                    react_devtools_backend_compact.js:2367
[redacted] returned a deep-equal value
▶{[redacted]}
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    react_devtools
connects [redacted] returned a deep-equal value for the p
▶{[redacted]
equality checks. This means the component might be re-rendering unnecessarily and has happ
last refresh. This value might be a member, which are known to be unstable due to our upse
returning a more stable derived value from your selector. :0:0
```

♪♪
I found a blank list, baby
Consider EMPTY_ARRAY!
♪♪



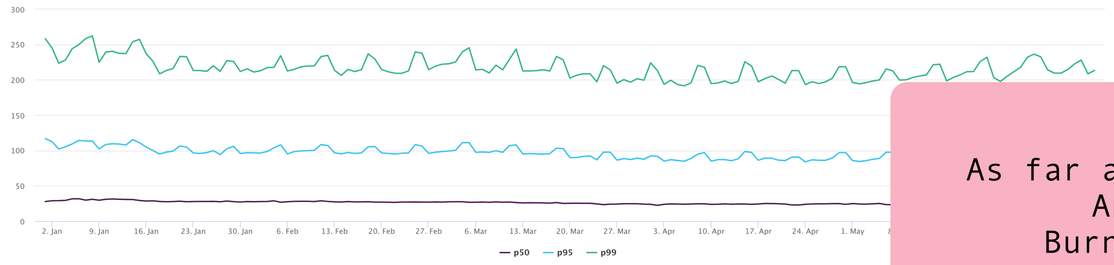
And, a Burndown Program

"Project Rollercoaster" (making the Loops go fast!)

Desktop Performance Burndowns by jenna

Edit

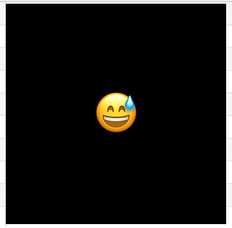
Redux Notification Times (ms)



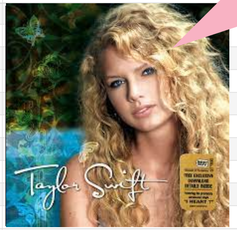
Total React Runtime Warnings

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 a
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 Ru

#	Component:Prop	
1		488485
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

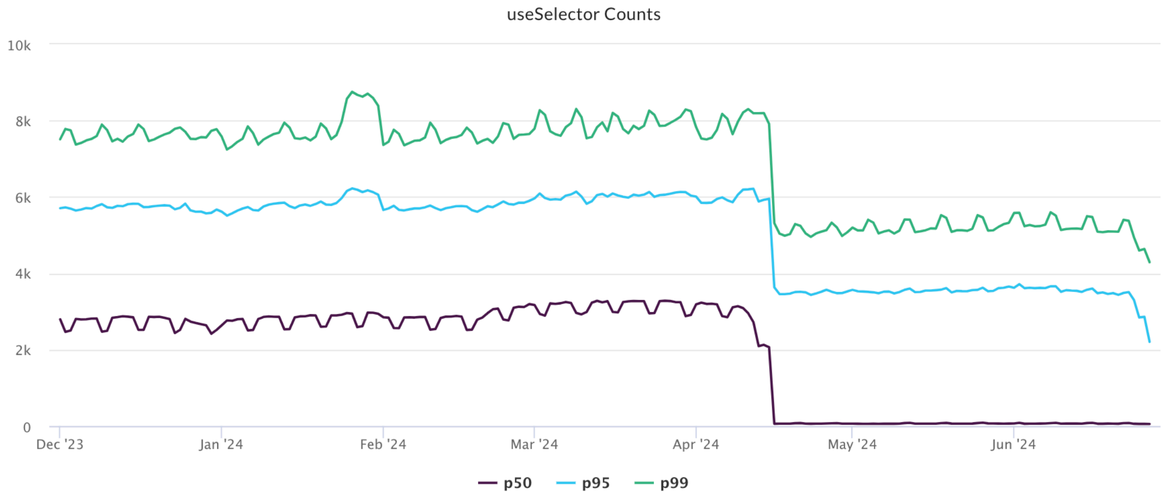
Mitigating A Problem of Scale at Scale

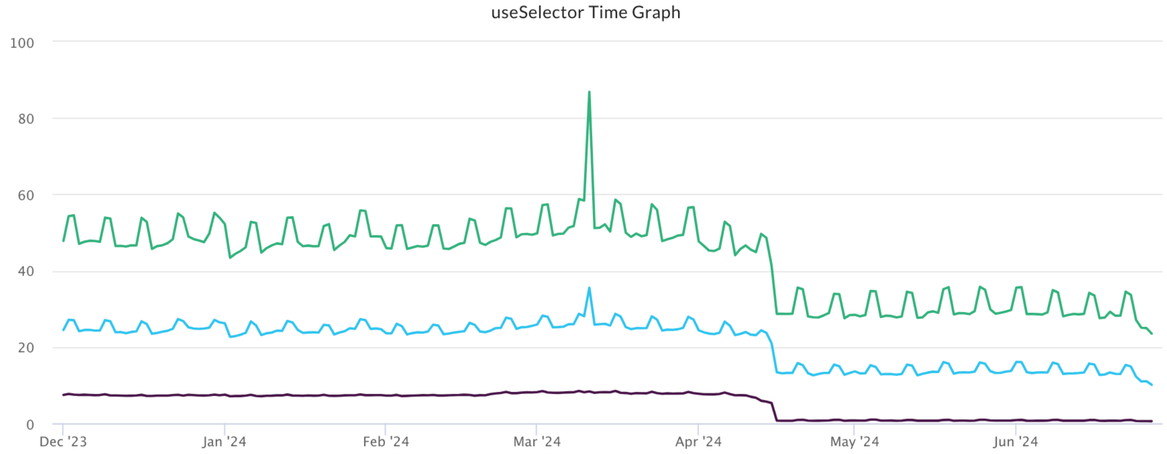
- Performance has been built up as a problem for the experts, often surrounded by an air of hero culture, but **we’re doing ourselves a disservice by keeping it an inaccessible discipline**
- Instead let’s get many people to fix many problems— architecting a distributed solution for a problem of scale!

♫♫
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





🎶
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 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 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%

♫♪
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
♫♪

What I'm bringing to Notion

- Ask "how does this affect our customers' experience?"
- Devise metrics that suit your app's needs
- Develop an understanding of your product's architecture and how it scales
- Make performance accessible to all and scale efforts by empowering teams to own performance of areas they own
- Slow and steady works to make things faster... but don't be afraid of doing the big scary thing too

♫♪
Let's fast forward to one collab software company later
Will I view perf profiles and see issues with extra re-renders
♫♪

It takes
a lot of work
to do less work.

Thanks!

jenna.is/at-lead-dev
@zeigenvector