

(Un)popular Opinions



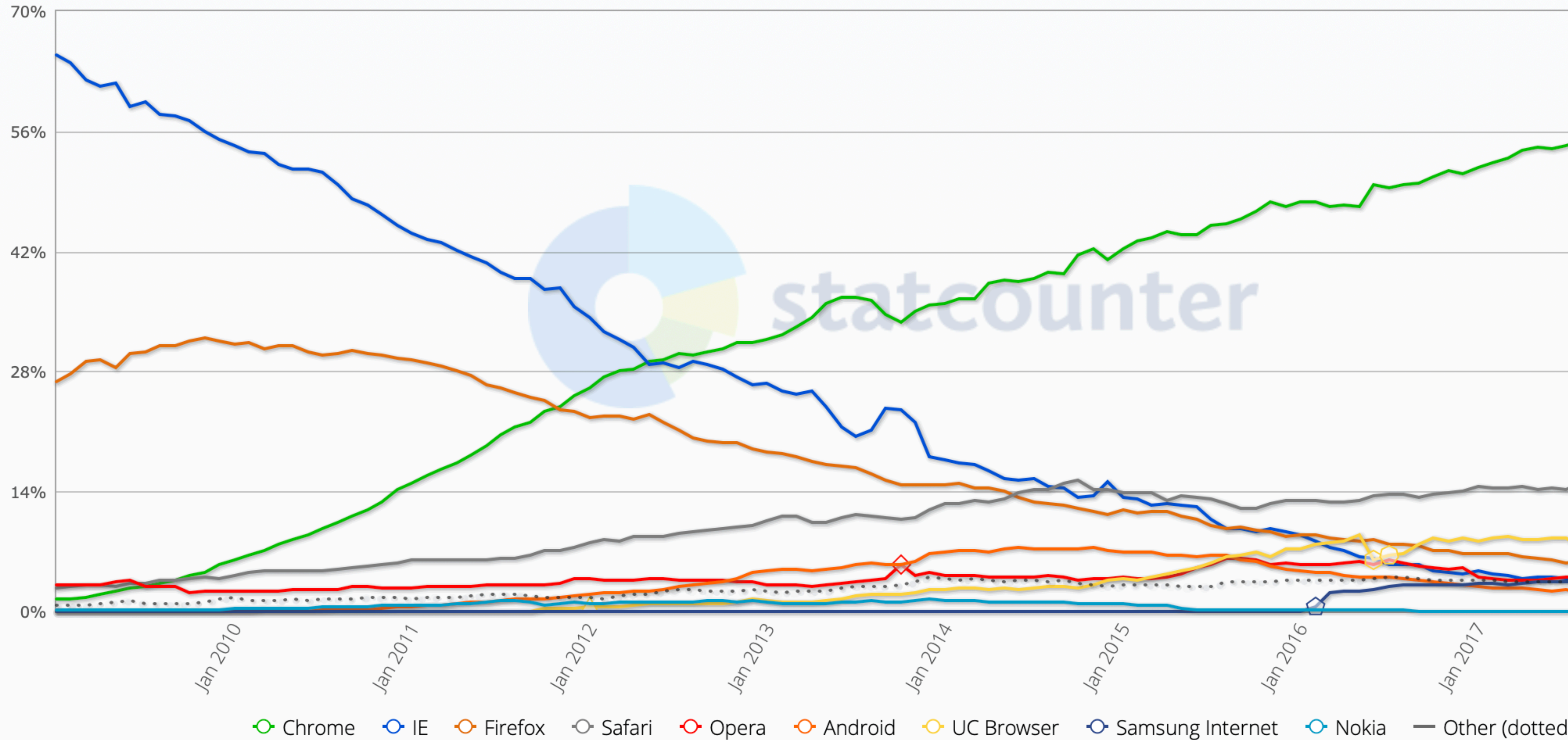
Who of you ever used has
Redux?

Would you use Redux for
your next project?

Are you a fan of Redux?



Jan 2009 - Sept 2018





BACKBONE.JS

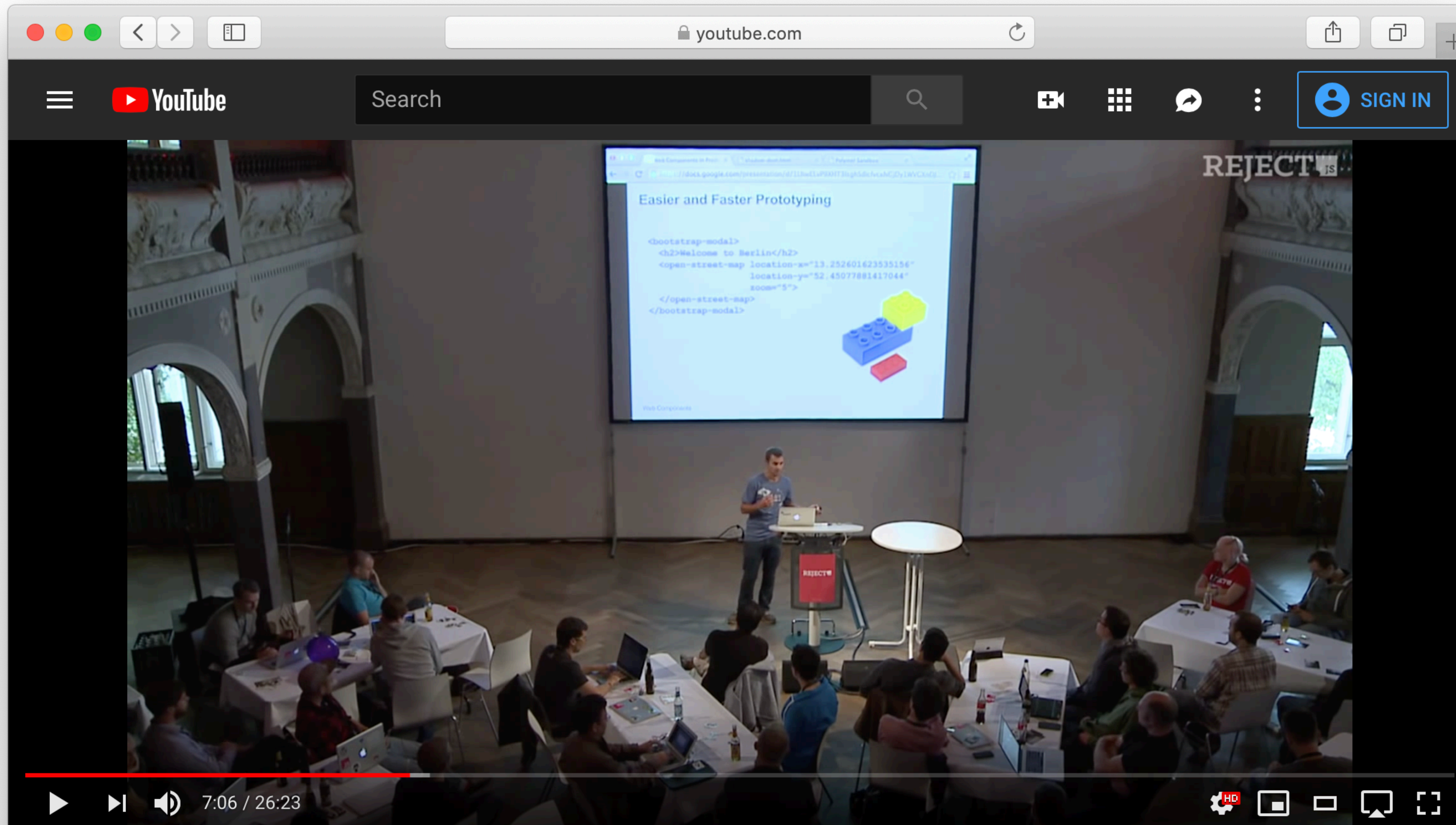
AWESOME



The image shows a web browser window with a macOS-style title bar (red, yellow, green buttons) and a navigation bar. The address bar displays 'github.com' with a lock icon. The page content is a code editor showing JavaScript code with line numbers 1123 through 1144. The code defines a function to throw an error when a URL is needed, a function to wrap an optional error callback, and a helper function to escape a string for HTML rendering. The code is syntax-highlighted with colors: red for keywords, blue for function names and strings, and black for comments and other code elements.

```
1123 // Throw an error when a URL is needed, and none is supplied.
1124 var urlError = function() {
1125     throw new Error("A 'url' property or function must be specified");
1126 };
1127
1128 // Wrap an optional error callback with a fallback error event.
1129 var wrapError = function(onError, model, options) {
1130     return function(resp) {
1131         if (onError) {
1132             onError(model, resp, options);
1133         } else {
1134             model.trigger('error', model, resp, options);
1135         }
1136     };
1137 };
1138
1139 // Helper function to escape a string for HTML rendering.
1140 var escapeHTML = function(string) {
1141     return string.replace(/&(?!\w+;|#\d+;|#x[\da-f]+;)/gi, '&').replace(/</g, '<').replace(/>/g, '>');
1142 };
1143
1144 }).call(this);
```


Web Components rock!

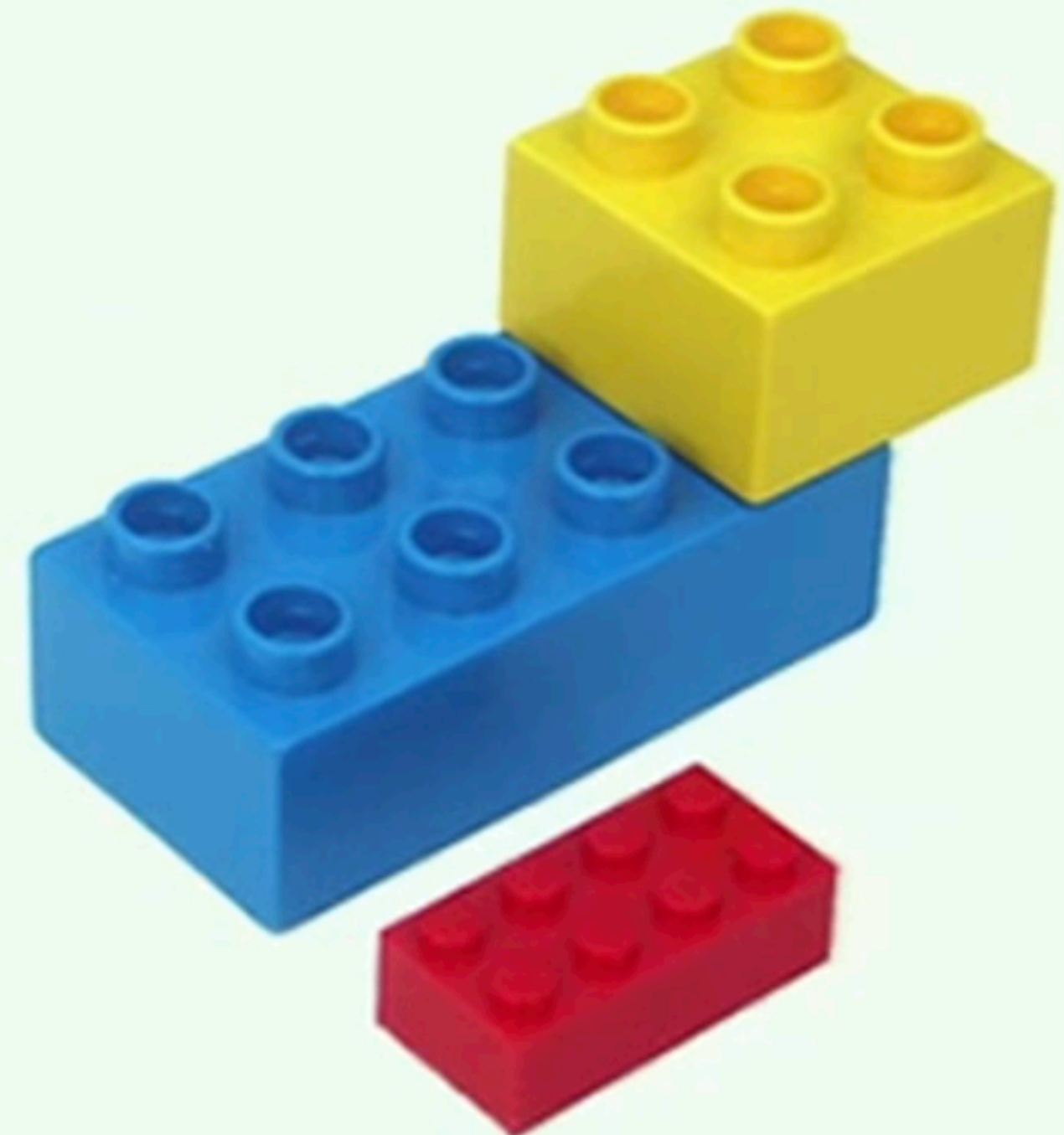


[Reject.JS 2013] Nik Graf - Using Web Components in production

Ad closed by Google

... looks familiar?

```
<bootstrap-modal>  
  <h2>Welcome to Berlin</h2>  
  <open-street-map location-x="13.252601623535156"  
                    location-y="52.45077881417044"  
                    zoom="5">  
    </open-street-map>  
</bootstrap-modal>
```





rocks

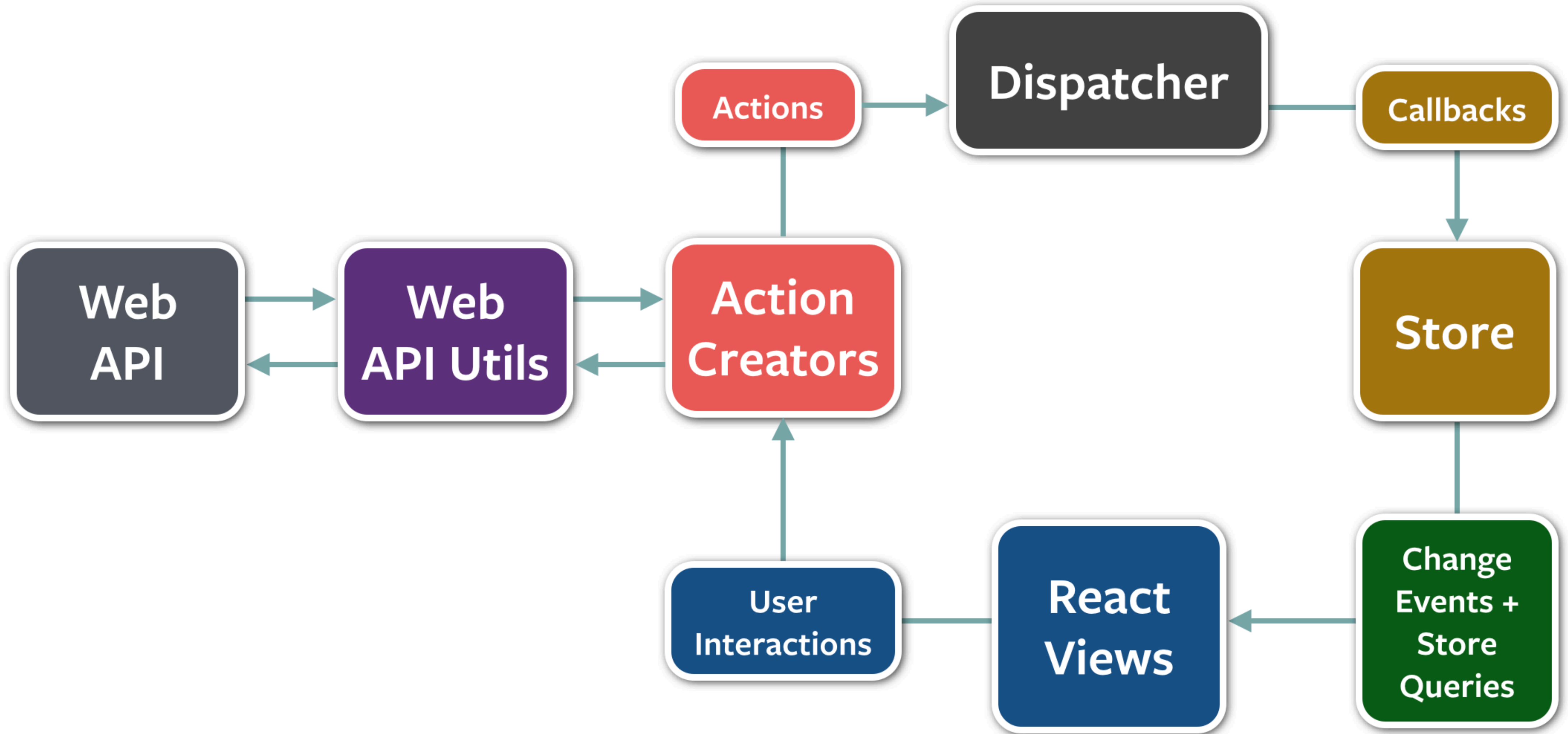


~~rocks~~

was kind of ok



$\text{fn}(\text{state}) \Rightarrow \text{ui}$





Lesson 1

Judge technology with
the context of time.






```
npm uninstall redux
```


npm uninstall redux-saga redux-loop
redux-thunk redux-effects react-intl-
redux react-redux redux-form

What now?



MobX

What are people really
excited about?

A man with dark, wavy hair and a serious expression is sitting on a yellow lifeguard stand. He is wearing a red lifeguard jacket with a dark blue collar and cuffs. His legs are spread wide, and he is looking directly at the camera. In the background, there is a clear blue sky, a seagull perched on the stand, and an American flag flying. The number '15' is visible on the side of the stand.

Hooks & Context

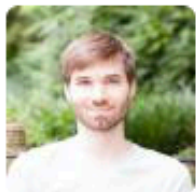
Is it really a good idea?

Dancing between state and effects - a real-world use case

#15240

New issue

Open jlongster opened this issue 20 days ago · 27 comments



jlongster commented 20 days ago • edited

Contributor

I started this as a gist but Dan mentioned that this would be a good discussion issue so here goes. I've been writing with and refactoring code into hooks for a while now. For 95% of code, they are great and very straight-forward once you get the basic idea. There are still a few more complex cases where I struggle with the right answer though. This is an attempt to explain them.

The use case

This is a real-world use case from an app I'm building: interacting with a list items. I've simplified the examples into codesandboxes though to illustrate the basic idea.

Here's the first one: <https://codesandbox.io/s/lx55q0v3qz>. It renders a list of items, and if you click on any of them, an editable input will appear to edit it (it doesn't save yet). The colored box on the right

Assignees

No one assigned

Labels

Type: Discussion

Projects

None yet

Milestone

No milestone

Sorry for the length of this. I figure I'd be over-detailed rather than under-detailed, and I've been brewing these thoughts since hooks came out. I'll try to conclude my thoughts here:

- Effects are **very nice**. It feels like we have easy access to the "commit" phase of React, whereas previously it was all in `componentDidUpdate` and not composable at all. Now it's super easy to throw on code to the commit phase which makes coordinating stuff with state easier.
- Reducers have interesting properties, and I can see how they are fully robust in a concurrent world, but for many cases they are too limited. The ergonomics of implementing many effect-ful workflows with them requires an awkward dance, kind of like when you try to track effect states in local state and split up workflows. Keeping a linear workflow in a callback is not only nice, but necessary in many cases for maintainability.
- Callbacks can be made memoizable without much work. In many cases I think it's easier to use the ref trick than reducers, but the question is: just *how* dangerous is it? Right now it's not that dangerous, but maybe concurrent mode really is going to break it.
- If that's the case, we should figure out a better way to weave together effects and state changes.

I hope all of this made sense. Let me know if something is unclear and I'll try to fix it.



46



2



6



2



4

Unpopular Opinions



Dan Abramov

@dan_abramov

Following



Unpopular opinion: The Durrells > Game of Thrones

2:55 PM - 17 Apr 2019

1 Retweet 16 Likes



 10

 1

 16



KEELEY HAWES

THE DURRELLS

SERIES ONE & TWO

As seen on
itv



★★★★★
DAILY MAIL

★★★★★
THE TIMES

★★★★★
THE TELEGRAPH

4 DISCS



KEELEY HAWES

THE DURRELLS

SERIES ONE & TWO

As seen on

itv



4 DISCS



★★★★★
DAILY MAIL

★★★★★
THE TIMES

★★★★★
THE TELEGRAPH



KEELEY HAWES

THE DURRELLS

SERIES ONE & TWO

As seen on

itv



4 DISCS



★★★★★
DAILY MAIL

★★★★★
THE TIMES

★★★★★
THE TELEGRAPH



Dragon

KEELEY HAWES

THE DURRELLS

SERIES ONE & TWO

As seen on

itv



4 DISCS



★★★★★
DAILY MAIL

★★★★★
THE TIMES

★★★★★
THE TELEGRAPH



White Walker



Dragon



KEELEY HAWES

THE DURRELLS

SERIES ONE & TWO

As seen on

itv



Dog

White Walker

Dragon

4 DISCS



★★★★★
DAILY MAIL

★★★★★
THE TIMES

★★★★★
THE TELEGRAPH





Dan Abramov

@dan_abramov

Following



Unpopular opinion: @reactjs edition

Bring it on 🔥

3:24 PM - 23 Mar 2019

183 Retweets 1,117 Likes



 589

 183

 1.1K





Sebastian Markbage @sebmarkbage · Mar 23

Replying to @dan_abramov @reactjs

React doesn't deprecate and delete things quickly enough.



13

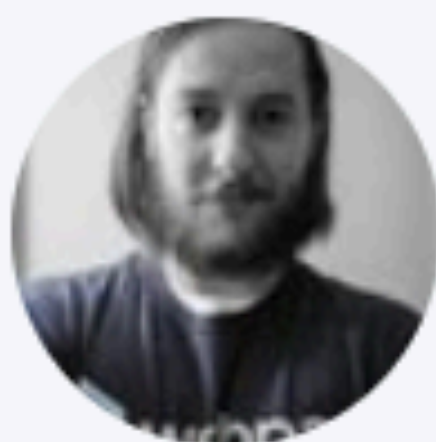


4



298





Sean Thomas Larkin (廖肖恩) @TheLarkInn · Mar 23



Replying to @dan_abramov @reactjs

Synthetic eventing is a wasted over-abstraction that hurts userland load times. Only about 15-30% of events are actually used.



3



3



107



Dan Abramov @dan_abramov · Mar 23



Agree in its current form. One solution could be to yank it out. Another interesting approach is to see what it could enable if it utilized to its full potential.



1



19





Ryan Florence @ryanflorence · Mar 23



Replying to @dan_abramov @reactjs

The majority of logic people write in React is for data loading and hooks makes it more confusing for people than classes.

Suspense maybe should have come before hooks.

Excited for when we get it though!



5



7



239



Andrew Clark @acd lite · Mar 23



One is harder than the other Ryan!



1



41



Ryan Florence @ryanflorence · Mar 23



Haha, I know. It's gonna be great though.



14





Paul Henschel @0xca0a · Mar 23



Replying to @dan_abramov @reactjs

I think the new context api was a little premature. Without selectors, a sure way to bail out of updates and consuming multiple providers it shouldn't have been greenlit. Now we have libs like Redux going back to subscribers in order to serve hooks, no hooks for React-Router, etc



6



5



38



Dan Abramov @dan_abramov · Mar 24



I think the problem here is that context API isn't meant as a catch-all solution. It's meant primarily for relatively infrequent updates. Would you prefer it didn't exist? For fast updates you can still use subscriptions which works the same way as in legacy context.



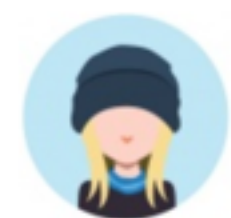
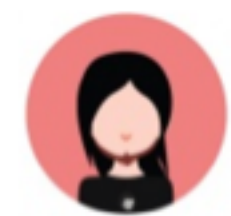
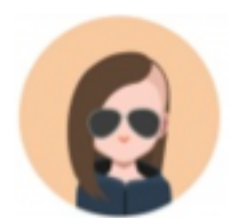
5



17



Context



[Redacted text bar]

[Redacted text bar]

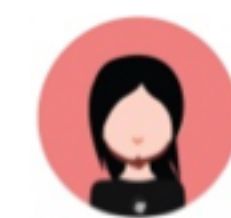
[Redacted text bar]

[Redacted text bar]

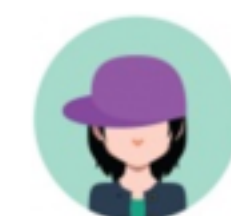
[Redacted text bar]

[Redacted text bar]

[Redacted text bar]

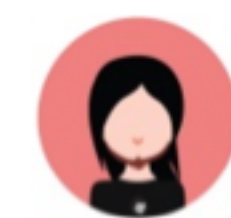


[Redacted text bar]



[Redacted text bar]

[Redacted text bar]



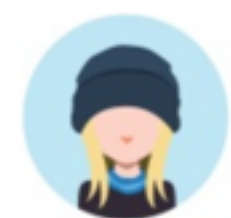
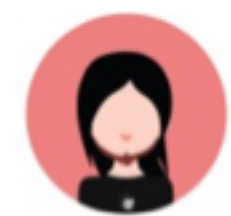
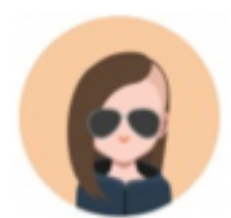
[Redacted text bar]

```
function Nav() {  
  const { me } = useContext(StoreContext);  
  ...  
}
```

```
function Sidebar() {  
  const { friends } = useContext(StoreContext);  
  ...  
}
```

```
function History(props) {  
  const { histories } = useContext(StoreContext);  
  const history = histories[props.historyId];  
  const { friends } = useContext(StoreContext);  
  ...  
}
```





[Redacted text bar]

[Redacted text bar]

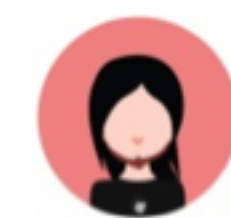
[Redacted text bar]

[Redacted text bar]

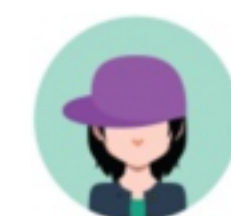
[Redacted text bar]

[Redacted text bar]

[Redacted text bar]

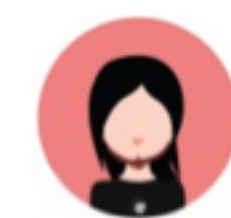


[Redacted text bar]

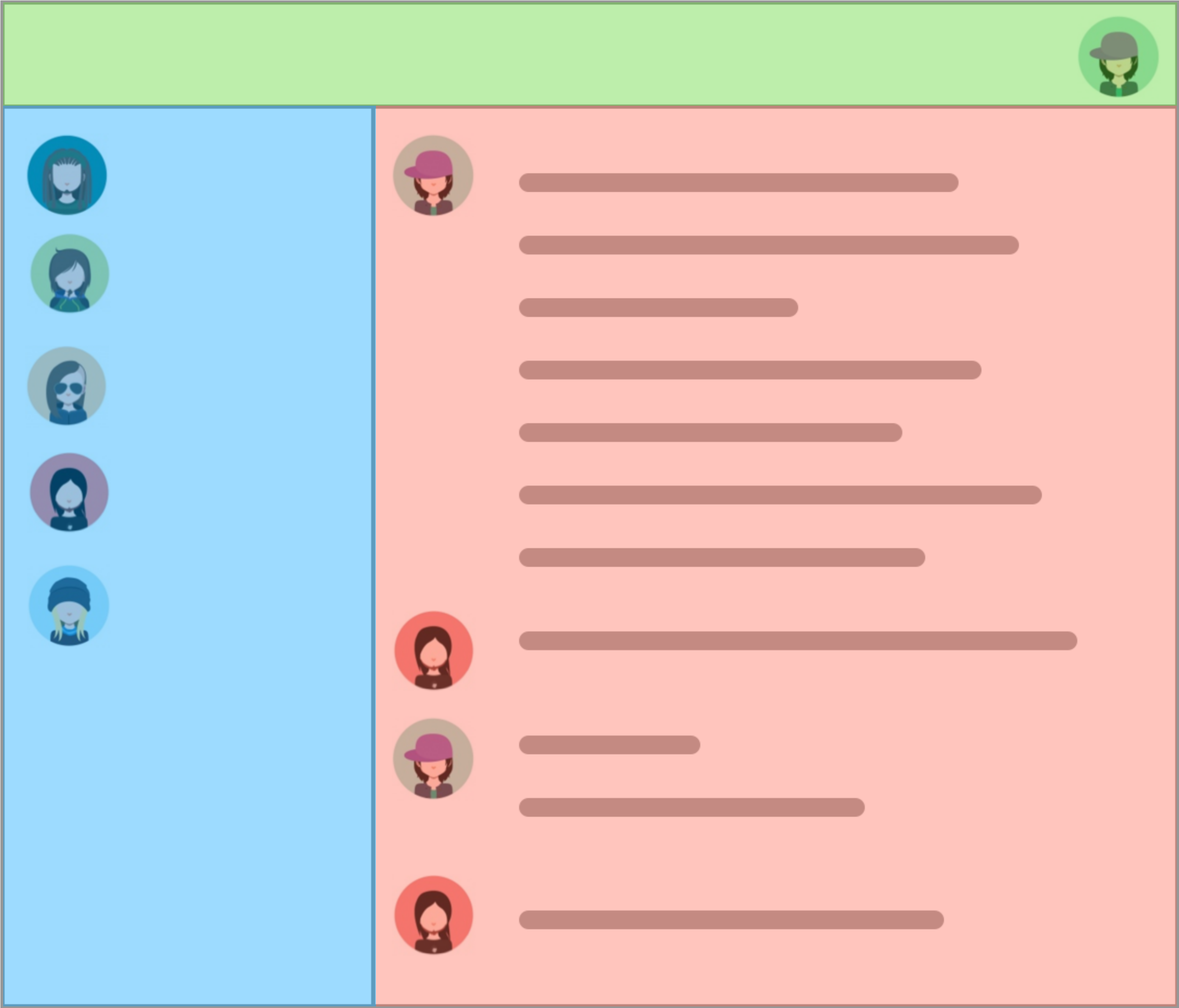


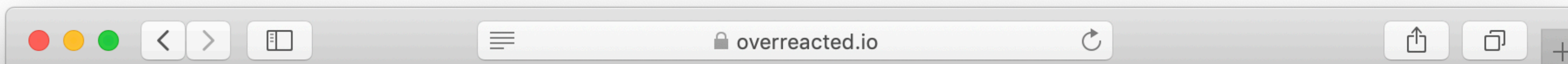
[Redacted text bar]

[Redacted text bar]



[Redacted text bar]





Not a Hook: `useBailout()`

As an optimization, components using Hooks can bail out of re-rendering.

One way to do it is to put a `React.memo()` wrapper around the whole component. It bails out of re-rendering if props are shallowly equal to what we had during the last render. This makes it similar to `PureComponent` in classes.

`React.memo()` takes a component and returns a component:

```
function Button(props) {  
  // ...  
}  
  
export default React.memo(Button);
```

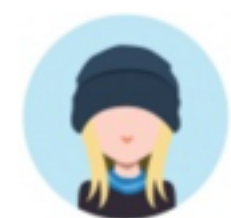
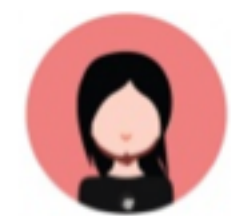
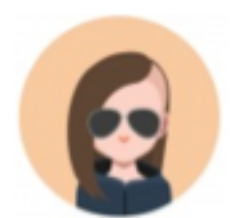
But why isn't it just a Hook?

Whether you call it `useShouldComponentUpdate()`, `usePure()`,

MeContext

FriendsContext

HistoriesContext



[Redacted message text]

[Redacted message text]

[Redacted message text]

[Redacted message text]

[Redacted message text]

[Redacted message text]

[Redacted message text]



[Redacted message text]



[Redacted message text]

[Redacted message text]



[Redacted message text]



Message text line 1

Message text line 2

Message text line 3

Message text line 4

Message text line 5

Message text line 6

Message text line 7



Message text line 8



Message text line 9

Message text line 10

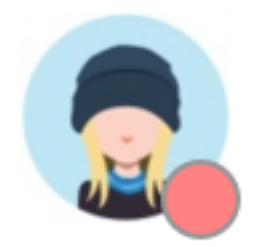


Message text line 11



Feature

Online indicators



[Redacted text bar]

[Redacted text bar]

[Redacted text bar]

[Redacted text bar]

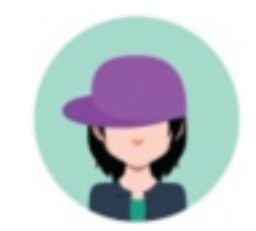
[Redacted text bar]

[Redacted text bar]

[Redacted text bar]



[Redacted text bar]

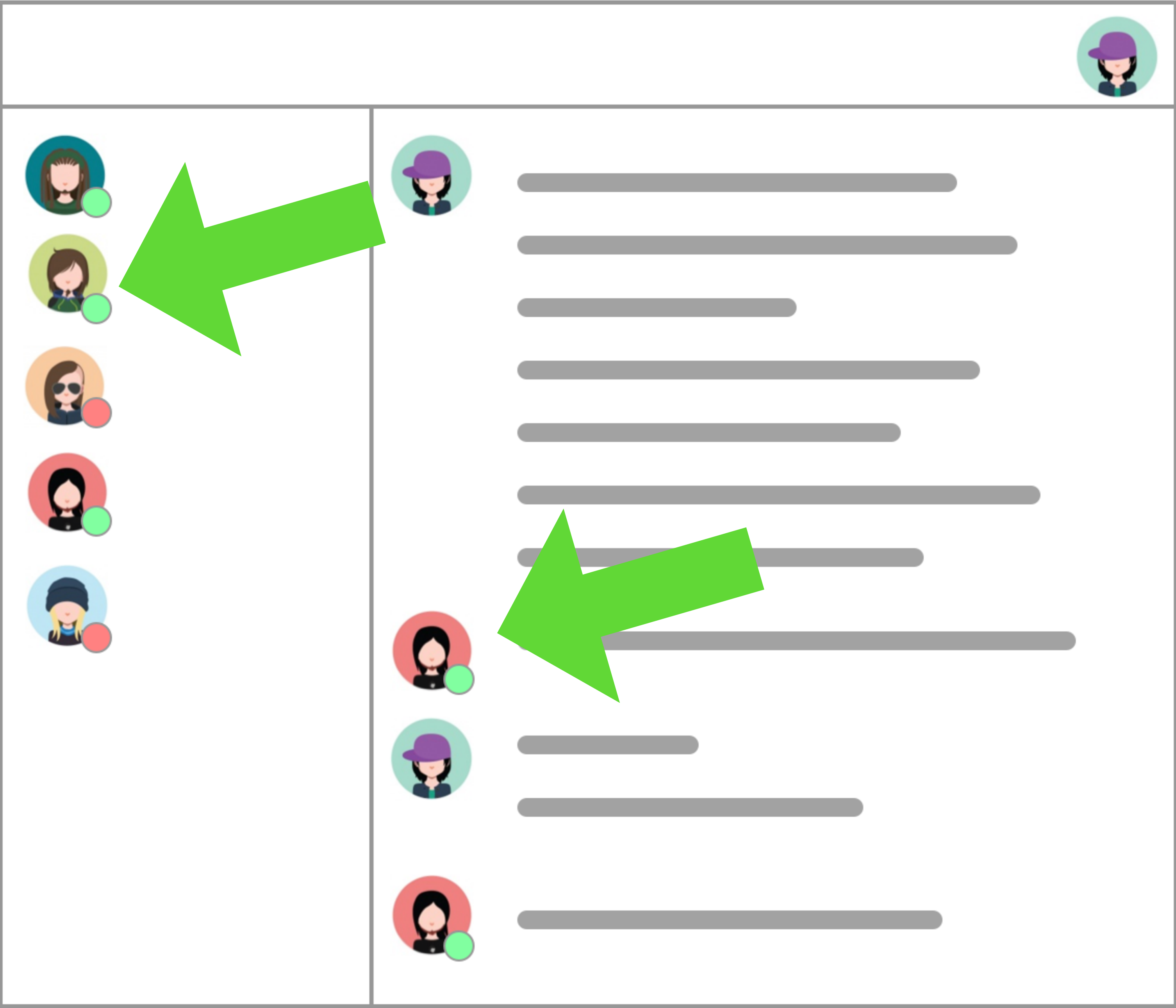


[Redacted text bar]

[Redacted text bar]



[Redacted text bar]



```
function History(props) {  
  const histories = useContext(HistoriesContext);  
  const history = histories[props.historyId];  
  const friends = useContext(FriendsContext);  
  ...  
}
```




[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]



[Redacted text line]



[Redacted text line]

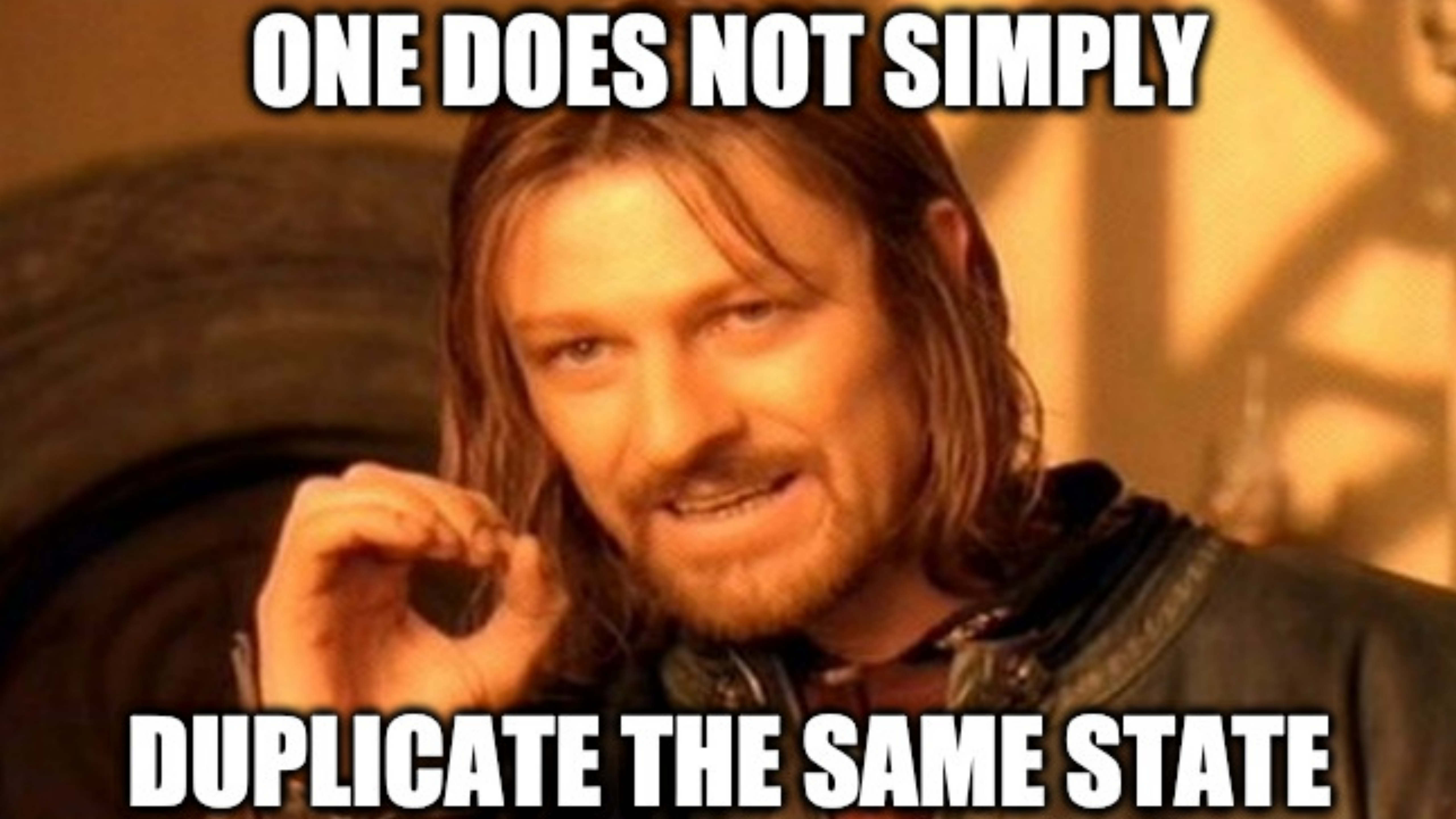
[Redacted text line]



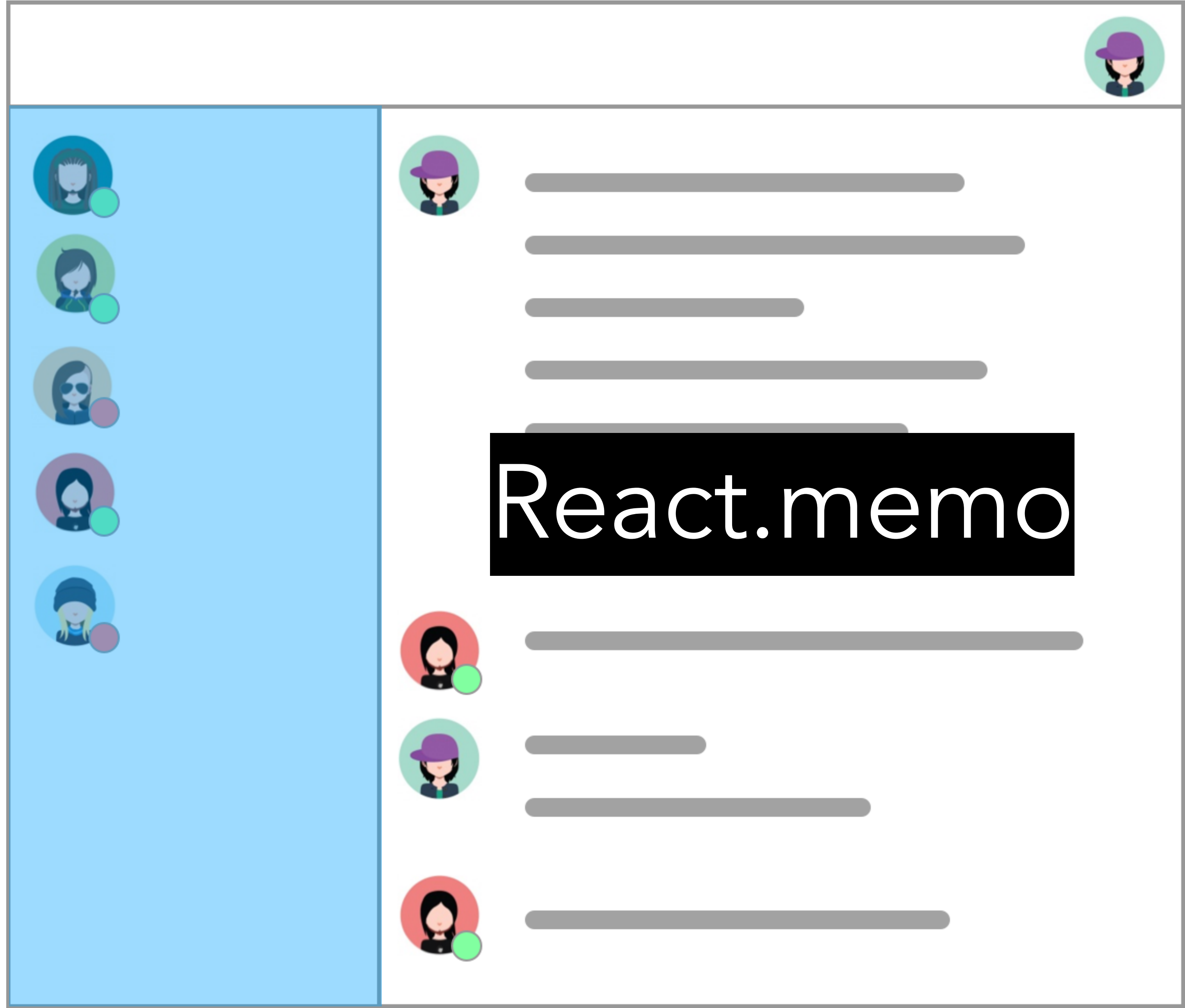
[Redacted text line]

We can duplicate the state!

ONE DOES NOT SIMPLY



DUPLICATE THE SAME STATE





Storing all data in Context



Dan Abramov

@dan_abramov

Following



Replying to @nikgraf @acemarke @kentcdodds

I think the idiomatic solution for “entities” like this is separate caches. It’s not truly state — because it can always be restored from server (although with latency). Relay/Apollo are examples of such caches but there could be simpler ones.

12:02 PM - 23 Apr 2019

1 Retweet 8 Likes



2



1



8





What can I store in Context?

Lesson 2

Think twice before you
move data into Context.

So what now?

Is it the right time to jump ship?



Dan Abramov

@dan_abramov

Following



Replying to @nikgraf @acemarke @kentcdodds

I think the idiomatic solution for “entities” like this is separate caches. It’s not truly state — because it can always be restored from server (although with latency). Relay/Apollo are examples of such caches but there could be simpler ones.

12:02 PM - 23 Apr 2019

1 Retweet 8 Likes



2



1

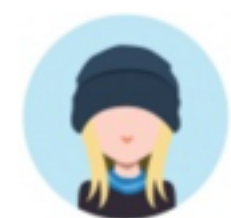
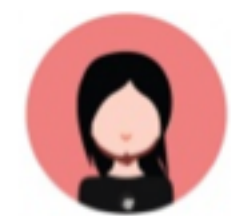
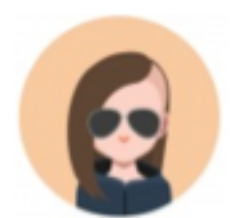


8



Lesson 3

Think ~~one~~ two steps ahead
before switching technology.



[Redacted text bar]

[Redacted text bar]

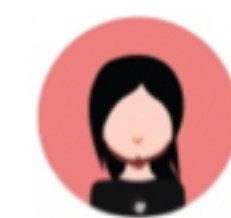
[Redacted text bar]

[Redacted text bar]

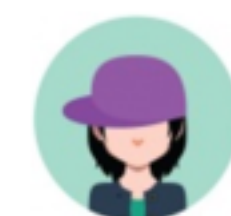
[Redacted text bar]

[Redacted text bar]

[Redacted text bar]

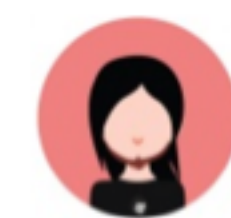


[Redacted text bar]



[Redacted text bar]

[Redacted text bar]



[Redacted text bar]

Lesson 1

Judge technology with the context of time.

Lesson 2

Think twice before you move data into Context.

Lesson 3

Think ~~one~~ two steps ahead before switching technology.

**Let's get rid of ego and
make good decisions.**

Thanks!
@nikgraf