

# Migration to React *Suspense*

闫畅

twitter@cyan\_binary

Instead of learning a new feature...

Instead of learning a new feature...

**let's fix real problems 😊**

# Suspend

*/sə'spend/*

*verb*

- a. to cause to stop temporarily
- b. to set aside or make temporarily inoperative

A riddle:

Why is it called “**Suspense**” ?

**Demo**

# Demo

- Manage your code budget with **code splitting**
- A new way to do **async data fetching**
- Make the application more “**performant**”, with a better user experience

***code splitting***



bundle.js

...

1.chunk.js


0.chunk.js

bundle.js

# Recap:

- `React.lazy()`
- `<Suspense fallback={<Spinner />}>`

***Async data fetching***  
*(not ready, yet)*

Let's go back to the demo  and see if there's any anti-patterns

What's the problems behind the spinners?



1. **Fetching data** and **showing the spinners** are tightly coupled in a single component

1. **Fetching data** and **showing the spinners** are tightly coupled in a single component
2. Overuse of **local state** and **life-cycle methods**



1. **Fetching data** and **showing the spinners** are tightly coupled in a single component
2. Overuse of **local state** and **life-cycle methods**
3. It's **an anti-pattern to use state as cache**

***Suspense*** coming to rescue, again!

**Demo**

1. **Decoupled** fetching data and showing spinners

1. **Decoupled** fetching data and showing spinners
2. No more **local state** or **life-cycle**.

1. **Decoupled** fetching data and showing spinners
2. No more **local state** or **life-cycle**.
3. **Global cache** instead of local state as cache

What's the magic behind <Suspense> ?

```
<ErrorBoundary>  
  <Foo />  
</ErrorBoundary>
```



```
<Suspense fallback>  
  <Foo />  
</Suspense>
```

```
<Suspense fallback>  
  <Bar />  
  <Foo />  
  <SyncContent />  
</Suspense>
```

Is the loading spinner *always* necessary?

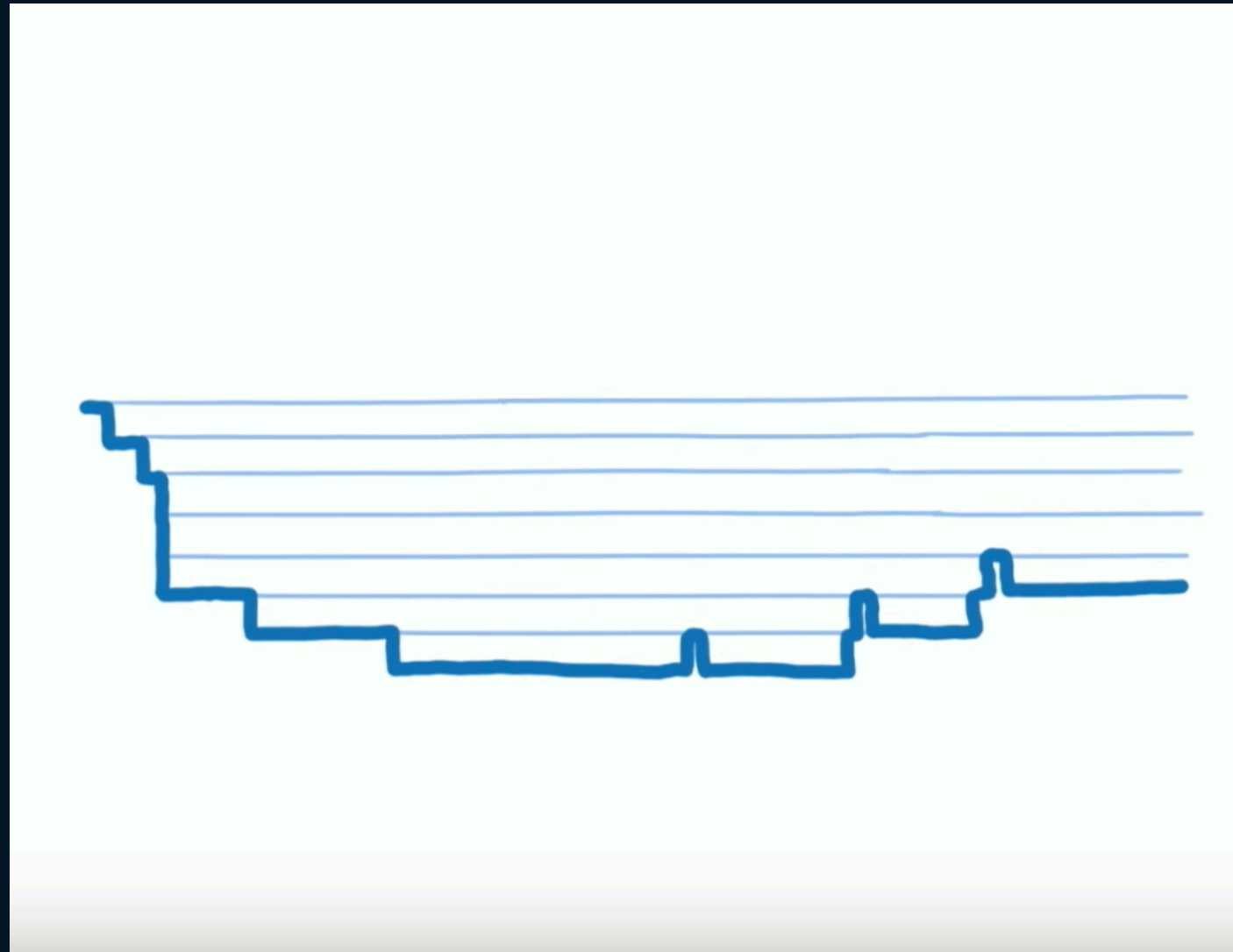
Show the spinner only after a certain threshold

# *Concurrent Rendering*

**Demo**

***Sync mode***  
vs.  
***Concurrent mode***

When updating a React component,  
the JavaScript call stack might look like this:



@linclark





Rendering



Committing

*User input* 😭



Rendering



Commiting



Rendering

Commiting





Rendering

Committing



*User input* 😊

In “Sync Mode”, the “**fallback**” is shown immediately.

In “Concurrent Mode”, React “*pauses*” rendering if a child component of `<Suspense>` is suspended.

In “Concurrent Mode”, React “***pauses***” rendering if a child component of `<Suspense>` is suspended

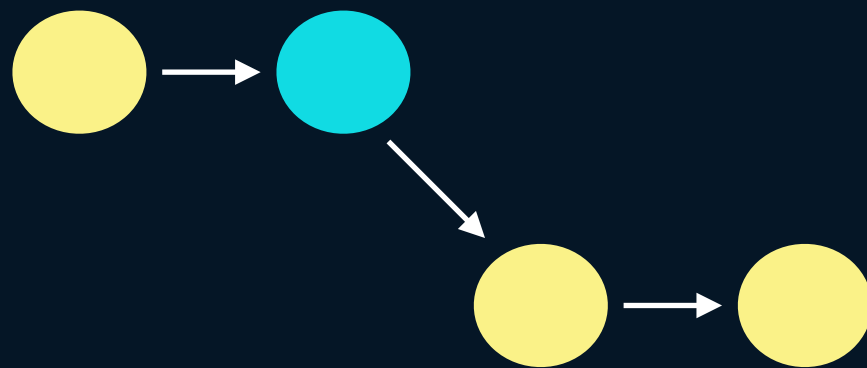
React will wait as long as ***maxDuration***, while the data is being loaded

# *Git as a Metaphor*



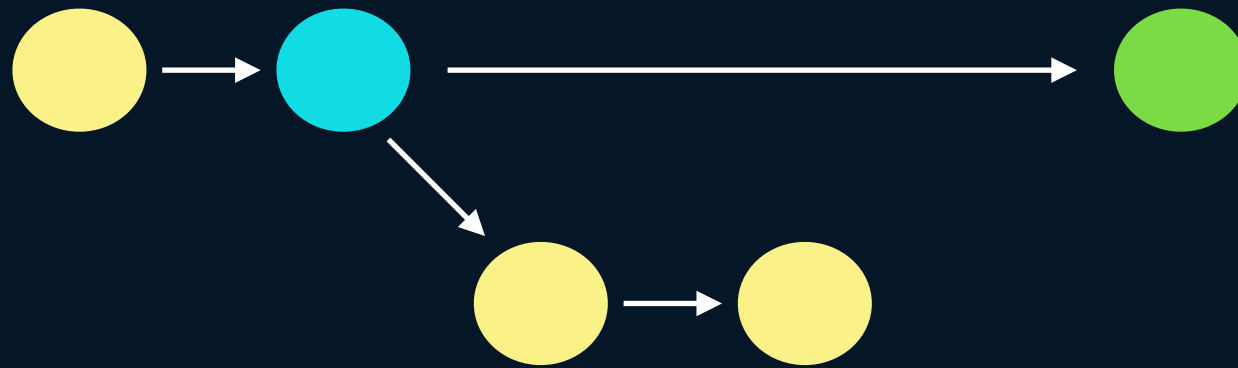


# *Git as a Metaphor*



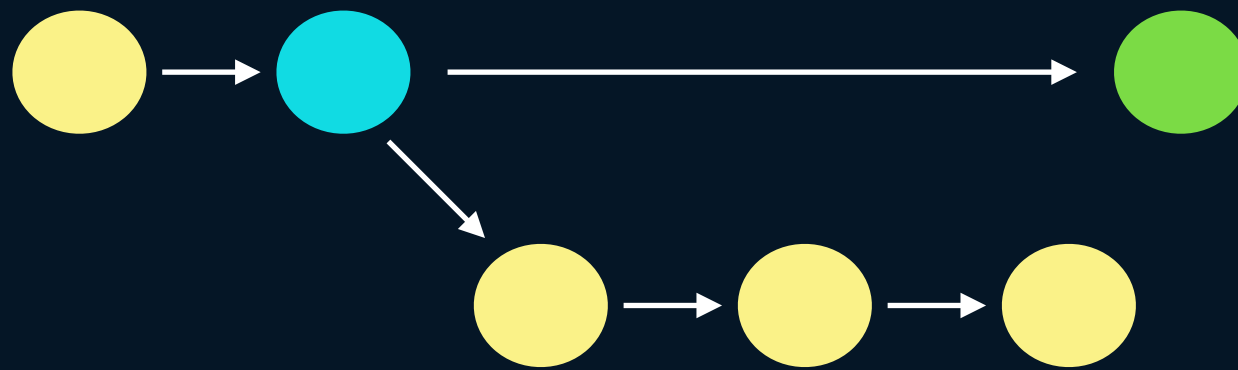
We work on our own branch

# *Git as a Metaphor*



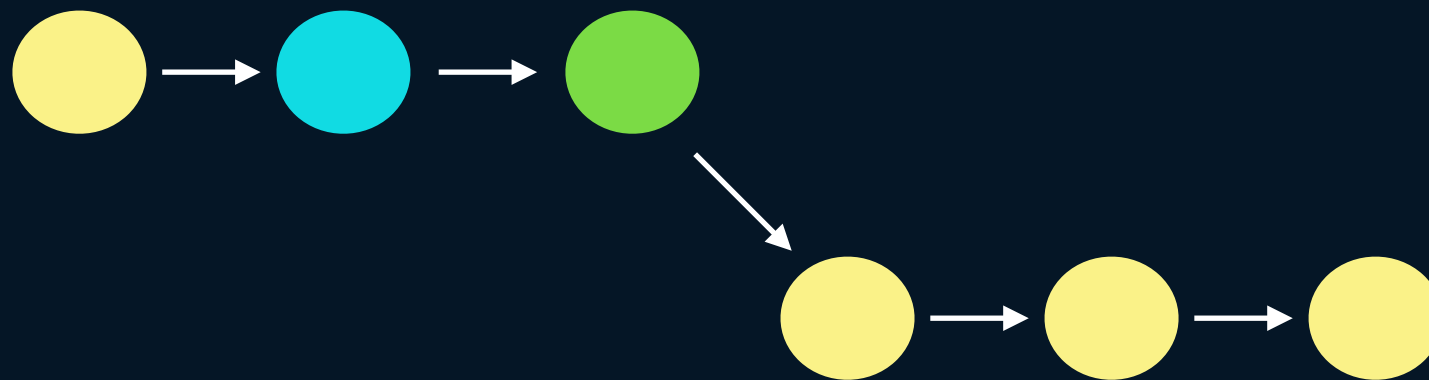
Other people can continue working,  
even if we got suspended in our own branch

# *Git as a Metaphor*



Later, the blocker gets resolved

# *Git as a Metaphor*



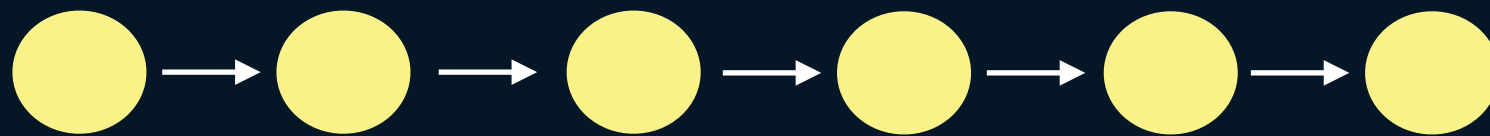
Rebase our branch to master

# *Git as a Metaphor*



Our pull request gets merged

# *Git as a Metaphor*



Our pull request gets merged

# *Suspense*

Hands-on **code splitting** API

Much easier to **compose** loading states

More idiomatic React, less state or life-cycles

Feels faster, with a better **user experience**

# Roadmap

v16.6: Suspense for code splitting 

v16.8: Concurrent Mode (~Q1 2019)

v16.9: Suspense for data fetching (~mid 2019)

<https://reactjs.org/blog/2018/11/27/react-16-roadmap.html>



# Thank you

See slides and code at: [github.com/cyan33](https://github.com/cyan33)