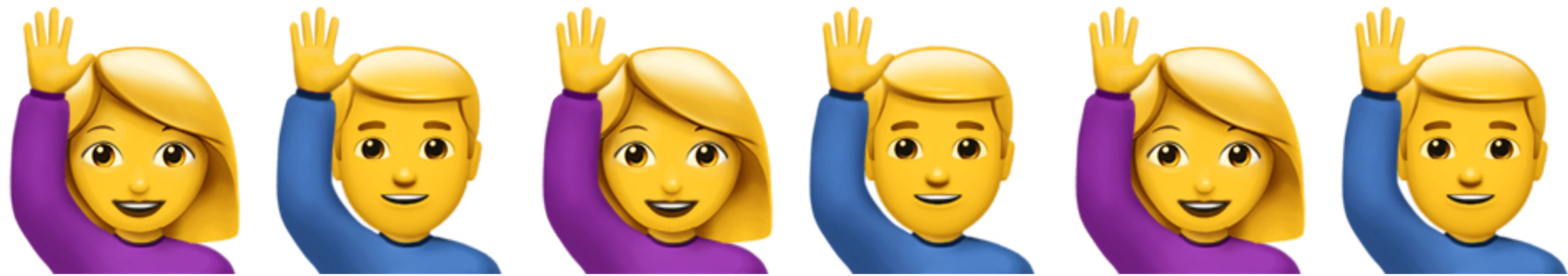


REACT 16.X

WAY BEYOND HOOKS

REVISITED

@YTHECOMBINATOR



(9) A Cartoon Intro to React Fil x +

https://www.youtube.com/watch?v=VLAqywvHpD0

YouTube BR Pesquisar

Front-Trends
24-26 May, 2017
in Warsaw, Poland

reconciler renderer

ReactNative Reacthardware
ReactBlessed
React ART react-pdf
ReactAframe ReactDOM
ReactThreeRenderer

A Cartoon Intro to React Fiber
-Lin Clark

3:13 / 30:36

A Cartoon Intro to React Fiber – Lin Clark / Front-Trends 2017

1.786 visualizações 31 4 COMPARTILHAR SALVAR

Crie o Seu Próprio Site

Wix

Comece seu site Anúncio Wix.com CRIE JÁ

Próximo REPRODUÇÃO AUTOMÁTICA

Programming in Visual Basic .Net...
iBasskung
Recomendado
19:11

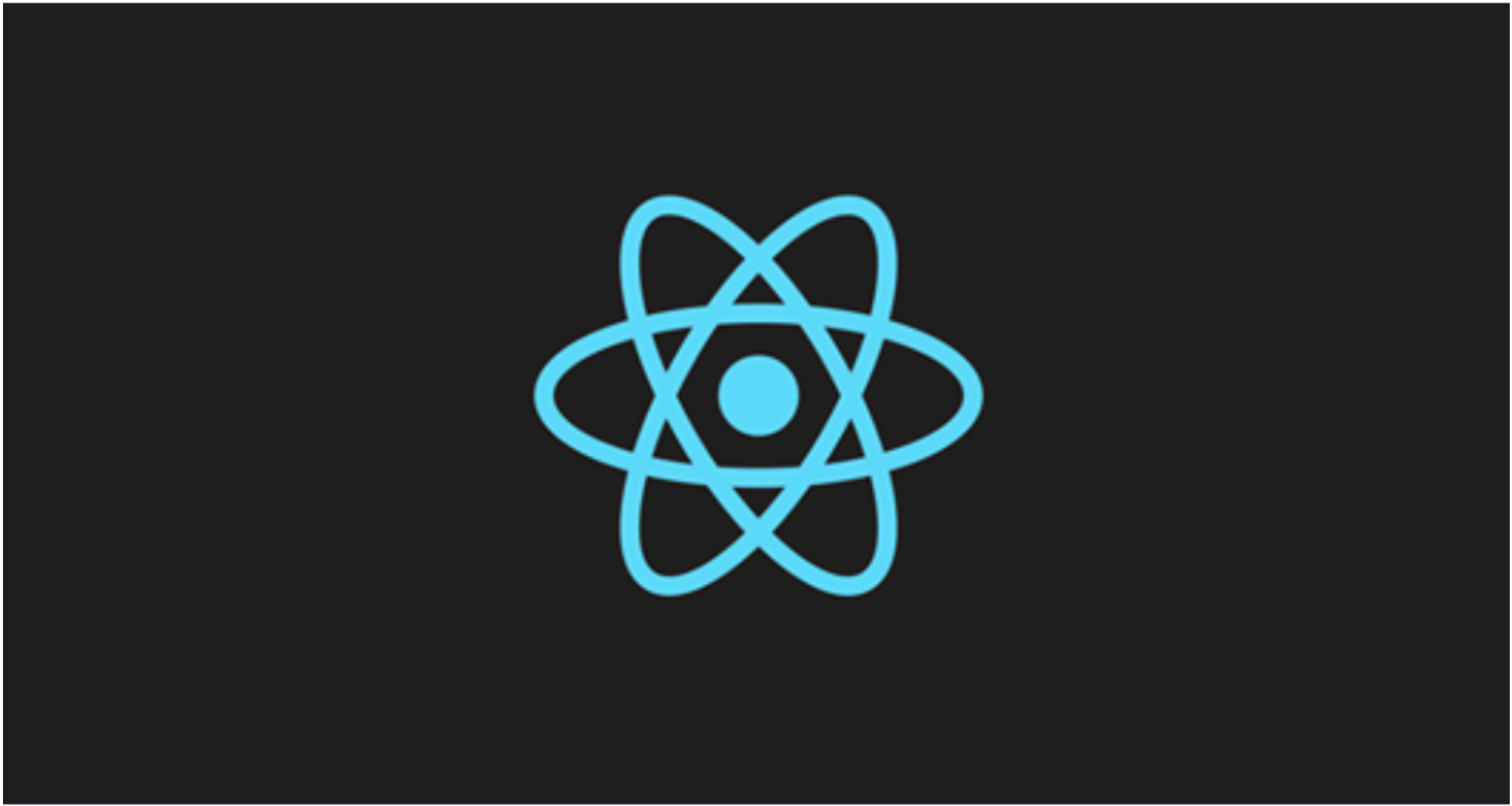
How To Insert Image Into Anoth...
Recipes
Recomendado
14:13

Autodesk Inventor - BMW M5 Rim...



6 de fevereiro às 10:43 · 

Passando só pra avisar que a melhor versão do React acabou de sair.



REACTJS.ORG

React v16.8: The One With Hooks – React Blog

With React 16.8, React Hooks are available in a stable release! What Are...



4 comentários

 Curtir

 Comentar

 Compartilhar

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

contextType createRef() **forwardRef()**

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

```
import React, { useRef } from 'react';

const InputComponent = ({ inputRef }) => {
  return <input ref={inputRef} type="text" />;
};

const Parent = props => {
  let textInput = useRef();

  const inputFocus = () => {
    textInput.current.focus();
  };

  return (
    <div>
      <InputComponent
        inputRef={textInput}
      />
      <input type="button" value="Click to focus" onClick={inputFocus} />
    </div>
  );
};

export default Parent;
```

```
import React, { useRef, forwardRef } from 'react';

const InputComponent = forwardRef((props, ref) => {
  return <input ref={ref} type="text" />;
});

const Parent = props => {
  let textInput = useRef();

  const inputFocus = () => {
    textInput.current.focus();
  };

  return (
    <div>
      <InputComponent ref={textInput} />
      <input type="button" value="Click to focus" onClick={inputFocus} />
    </div>
  );
};

export default Parent;
```


The Differences:

1. Instead of passing the `inputRef` attribute, we are able to pass `ref`
2. We use `forwardRef` to encapsulate the entire component, and we pass it props and then the `ref` attribute
3. We set the `ref` attribute on the input element equal to the passed in `ref`.

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

Deprecated now (will be removed in React 17):

1. `componentWillMount`
2. `componentWillReceiveProps`
3. `componentWillUpdate`

These will work:

1. `UNSAFE_componentWillMount`
2. `UNSAFE_componentWillReceiveProps`
3. `UNSAFE_componentWillUpdate`

Why?

The original lifecycle model was not intended for some of the upcoming features like **async rendering**. With the introduction of async rendering, some of these lifecycle methods will be **unsafe** if used.

e.g.

Why?

The original lifecycle model was not intended for some of the upcoming features like **async rendering**. With the introduction of async rendering, some of these lifecycle methods will be **unsafe** if used.

e.g.

Calls from **componentWillReceiveProps** that change the store, leading to call component's **componentWillReceiveProps** again - leading to an infinite loop and many useless render calls.

Why?

The original lifecycle model was not intended for some of the upcoming features like **async rendering**. With the introduction of async rendering, some of these lifecycle methods will be **unsafe** if used.

e.g.

Similarly calling **setState** in **componentWillUpdate** will trigger **shouldComponentUpdate** and then again **componentWillUpdate**, which also leads to infinite methods calls.

Why?

The original lifecycle model was not intended for some of the upcoming features like **async rendering**. With the introduction of async rendering, some of these lifecycle methods will be **unsafe** if used.

e.g.

Async rendering will cause **componentWillMount** to **trigger multiple rendering of your component tree**. This makes it unsafe.

What now?

1. `getDerivedStateFromProps`
2. `getSnapshotBeforeUpdate`


```
static getDerivedStateFromProps(props, state) {  
  if (state.value !== props.value) {  
    return {  
      derivedValue: deriveValueFromProps(props),  
      mirroredProp: props.value  
    }  
  }  
  return null;  
}
```

1. Used to keep the state synced with incoming props
2. Is expected to return an object to update the state of the component
3. If null is returned then, nothing is changed in the state.
4. A safer replacement of **componentWillReceiveProps**
5. Is a pure function

```
getSnapshotBeforeUpdate(prevProps, prevState) {  
  if (prevProps.list.length < this.props.list.length) {  
    return this.listRef.value.scrollHeight;  
  }  
  return null;  
}
```

1. Gets called after the render created the React element and before it is actually updated from virtual DOM to actual DOM
2. Is useful if you want to keep sync in-between state of current DOM with the updated DOM
3. A safer replacement of **componentWillUpdate**
4. The snapshot value is passed on to `componentDidUpdate`





contextType createRef() forwardRef()

Lifecycle Changes **<Strict Mode/>** act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>  
document.getElementById('root')
```

1. Does not render any visible UI
2. Identifying components with unsafe lifecycles, legacy string ref API, unexpected side effects, findDOMNode occurrences etc.
3. You'll get these errors from dependencies, too

Warning: Unsafe lifecycle methods were found within a strict-mode tree:
 in div (created by ExampleApplication)
 in ExampleApplication

componentWillMount: Please update the following components to use componentDidMount instead: ThirdPartyComponent

Learn more about this warning here:

<https://fb.me/react-strict-mode-warnings>

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache **Profiler** <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

Profiler

React has two phases:

Render: React determines what DOM changes need to be made by comparing render results with a previous render

Commit: React applies any changes that need to happen. Add/remove from the DOM and call lifecycle methods like **componentDidMount** and **componentDidUpdate**

Profiler

The profiler collects timing information about components, the time rendered and committed in order to identify when each component actually rendered and at what speed.

When profiling, we'll be looking closely at the **commit phase** to see where performance dips are.

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

```
import { ThemeConsumer } from "../ThemeContext";

class App extends React.Component {
  render() {
    return (
      <ThemeConsumer>
        {theme => {
          let mainColor = theme.color.blue["400"];
          return <h1 style={{ color: mainColor }}>Hello!</h1>;
        }}
      </ThemeConsumer>
    );
  }
}
```

```
import { ThemeContext } from "../ThemeContext";

class App extends React.Component {
  static contextType = ThemeContext;
  render() {
    const mainColor = this.context.color.blue["400"];
    return <h1 style={{ color: mainColor }}>Hello!</h1>;
  }
}
```

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler **memo()** Create React App v3

ReactDOM.createRoot() Migrating Stuff



~ Demo time ~

PureComponent

```
export default class Card extends React.PureComponent
{
  render() {
    return (
      <div>
        <p>{this.props.name}</p>
        <p>{this.props.description}</p>
      </div>
    )
  }
}
```


memo()

```
function Card(props) {  
  return (  
    <div>  
      <p>{props.name}</p>  
      <p>{props.description}</p>  
    </div>  
  )  
}  
export default React.memo(Card);
```

So...

1. `React.memo` provides a higher order component which prevents re-rendering from happening unless the props change
2. Much like `PureComponent` for class components
3. It's a tool you can use for improving performance

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() **react-cache** Profiler **<Suspense/>**

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

Suspense

1. Allows you to defer rendering part of your application tree until some condition is met
2. Takes a fallback prop that accepts the React elements you want rendered as placeholder

lazy()

1. Makes it easy to create components that are loaded using `dynamic import()`
2. Takes a function as its argument that must return a promise by calling `import()` to load the component



~ Demo time ~

github.com/palmerhq/the-platform

facebook/jscoreshift: A JavaScript... | reactjs/react-codemod: React... | palmerhq/the-platform: Web. C... x

GitHub, Inc. [US] | https://github.com/palmerhq/the-platform

Search or jump to... / Pull requests Issues Marketplace Explore

palmerhq / the-platform Watch 62 Unstar 3,428

Code Issues 7 Pull requests 2 Insights

Web. Components. 😂

react react-suspense react-hooks hooks suspense

162 commits 7 branches 9 releases 16 contributors

Branch: master New pull request Create new file Upload files Find file Clone

jaredpalmer Merge pull request #72 from palmerhq/bugfix/update-gitignore-add-react... Latest commit f89d...

.circleci	Correctly name file
.github	[ImgBot] Optimize images
example	Update the gitignore and add react-cache build to example vendor
src	Fix hashing functions
tests	Fix test infrastructure
typings	Use official React 16.7 type declaration
.babelrc	Rewritten in TypeScript

```
import React, { Suspense } from 'react';
import { Router } from '@reach/router';
import Loading from './Loading';

const Home = React.lazy(() => import('./Home'));
const Dashboard = React.lazy(() => import('./Dashboard'));
const Overview = React.lazy(() => import('./Overview'));
const History = React.lazy(() => import('./History'));
const NotFound = React.lazy(() => import('./NotFound'));

function App() {
  return (
    <div>
      <Suspense fallback={<Loading />}>
        <Router>
          <Home path="/" />
          <Dashboard path="dashboard">
            <Overview path="/" />
            <History path="/history" />
          </Dashboard>
          <NotFound default />
        </Router>
      </Suspense>
    </div>
  )
}
```


But...

1. At the moment, `React.lazy()` does not support using named exports for React components.
2. `React.lazy()` and `Suspense` are not yet available for server-side rendering
3. `Suspense` for Data Fetching is not released yet
(estimated time: ~mid 2019)

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> **act()**

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

```
function App() {  
  let [ctr, setCtr] = useState(0);  
  useEffect(() => {  
    setCtr(1);  
  }, []);  
  return ctr;  
}
```

```
ReactDOM.render(<App />, document.getElementById("app"));
```

```
it("should render 1", () => {  
  const el = document.createElement("div");  
  ReactDOM.render(<App />, el);  
  expect(el.innerHTML).toBe("1"); // this fails!  
});
```

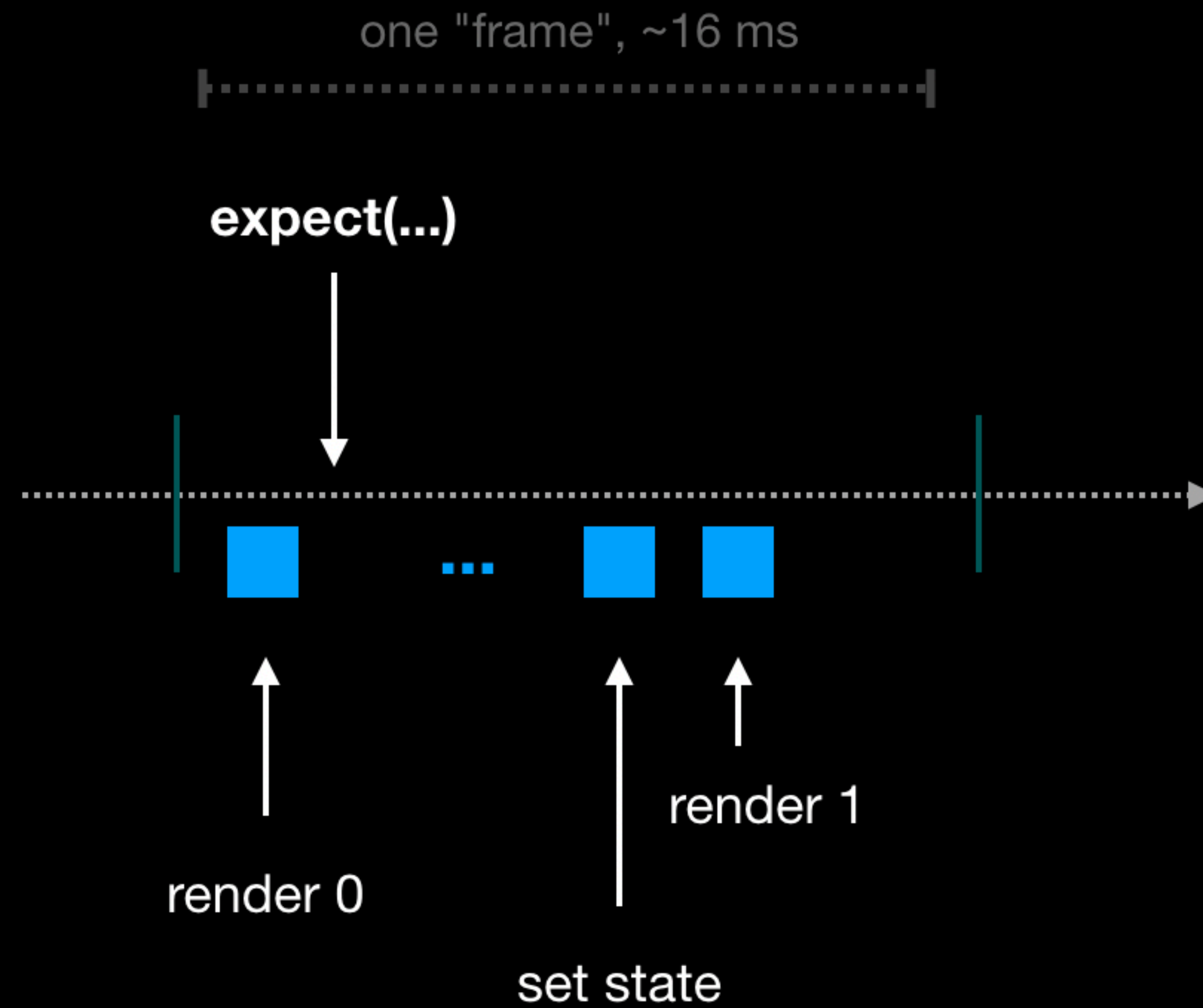

useEffect()

"By using this Hook, you tell React that your component needs to do something **after render**".

useEffect()

- the 'first' render where react **outputs 0**,
- the bit where it runs the effect and **sets state to 1**
- the bit where it rerenders and **outputs 1**

useEffect()



Hacks

- using **useLayoutEffect** instead of **useEffect**
- **waiting for some time**, like 100ms or so

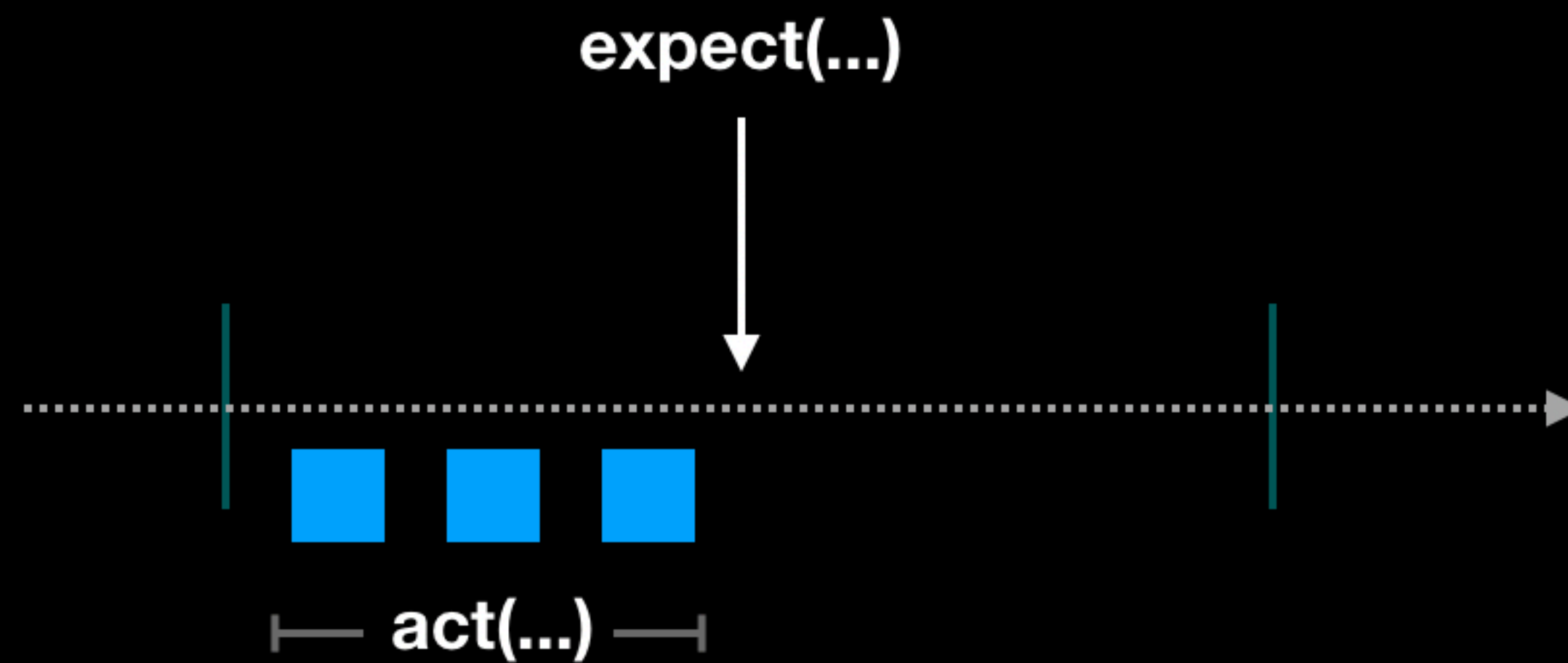
act()

- any state updates will be executed
- any enqueued effects will be executed

```
it("should render 1", () => {  
  const el = document.createElement("div");  
  act(() => {  
    ReactDOM.render(<App />, el);  
  });  
  expect(el.innerHTML).toBe("1"); // this passes!  
});
```

act()

- any state updates will be executed
- any enqueued effects will be executed



contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

Time Slicing

1. DOM Updates have high and low priority

2. Concurrent React sets priorities for you by default

```
scheduleCallback(() => {  
  this.setState({ update: 'lowPriority' });  
});
```

```
flushSync(() => {  
  this.setState({ update: 'highPriority' });  
});
```



~ Demo time ~

INTERMISSION

CONCURRENT

REACT

What?

It's an umbrella name for a new set of APIs resulting from the React Fiber rewrite.

What?

Time Slicing

A generic way to ensure that high-priority updates don't get blocked by a low-priority update.

Problems it solves: When rendering is **CPU-bound**.

What?

Suspense

A generic way for components to suspend rendering while they load data from a cache.

Problems it solves: When rendering is **I/O-bound**.

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() **Create React App v3**

ReactDOM.createRoot() Migrating Stuff

Cool Things

1. **Detect** and **lint** .ts files
2. **Absolute** imports
3. Latest version of **Jest** (v24)

Other stuff...

1. **browserslist** tools to target specific browsers

2. **PostCSS** Normalize

3. Linting for **Hooks**

contextType createRef() forwardRef()

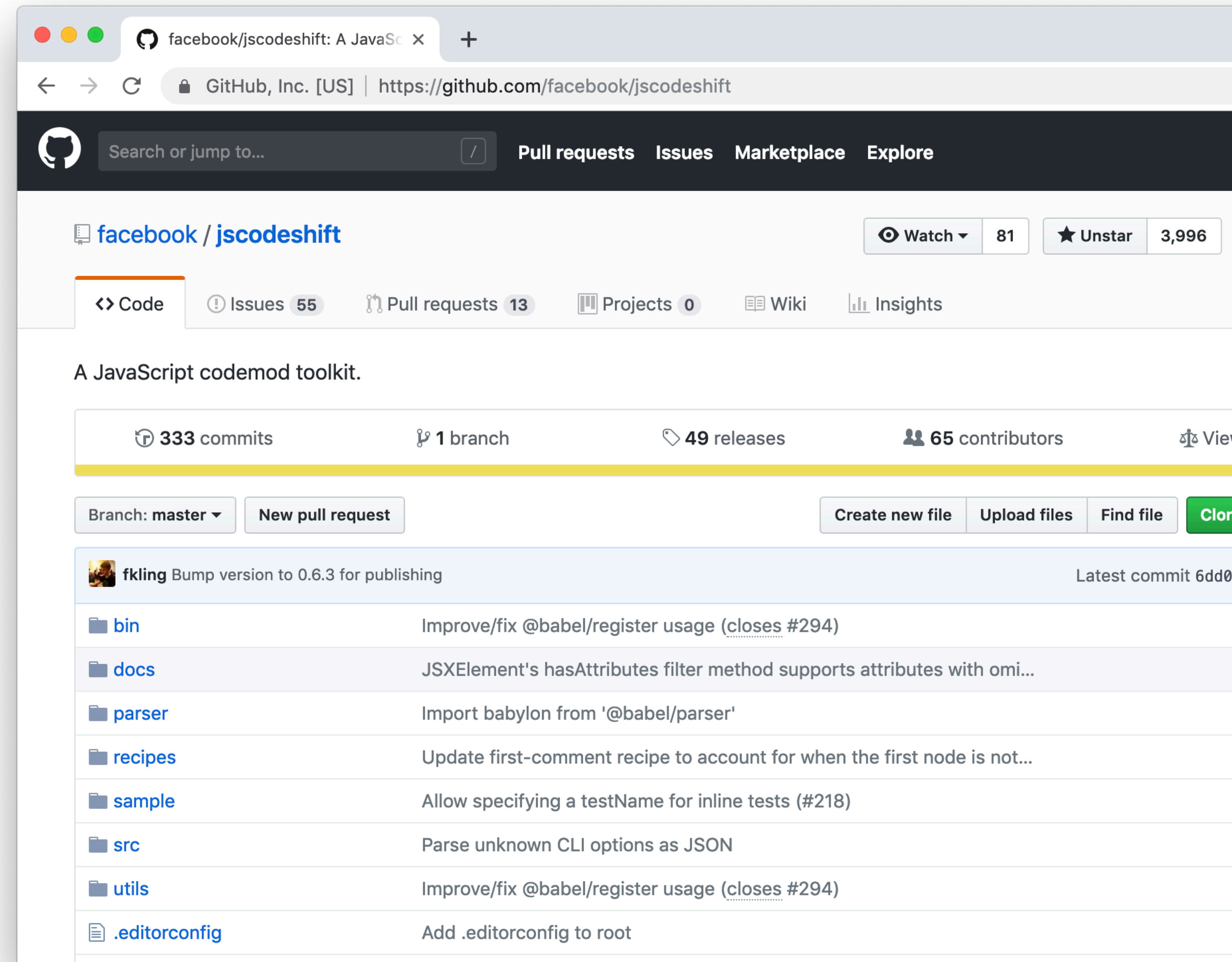
Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

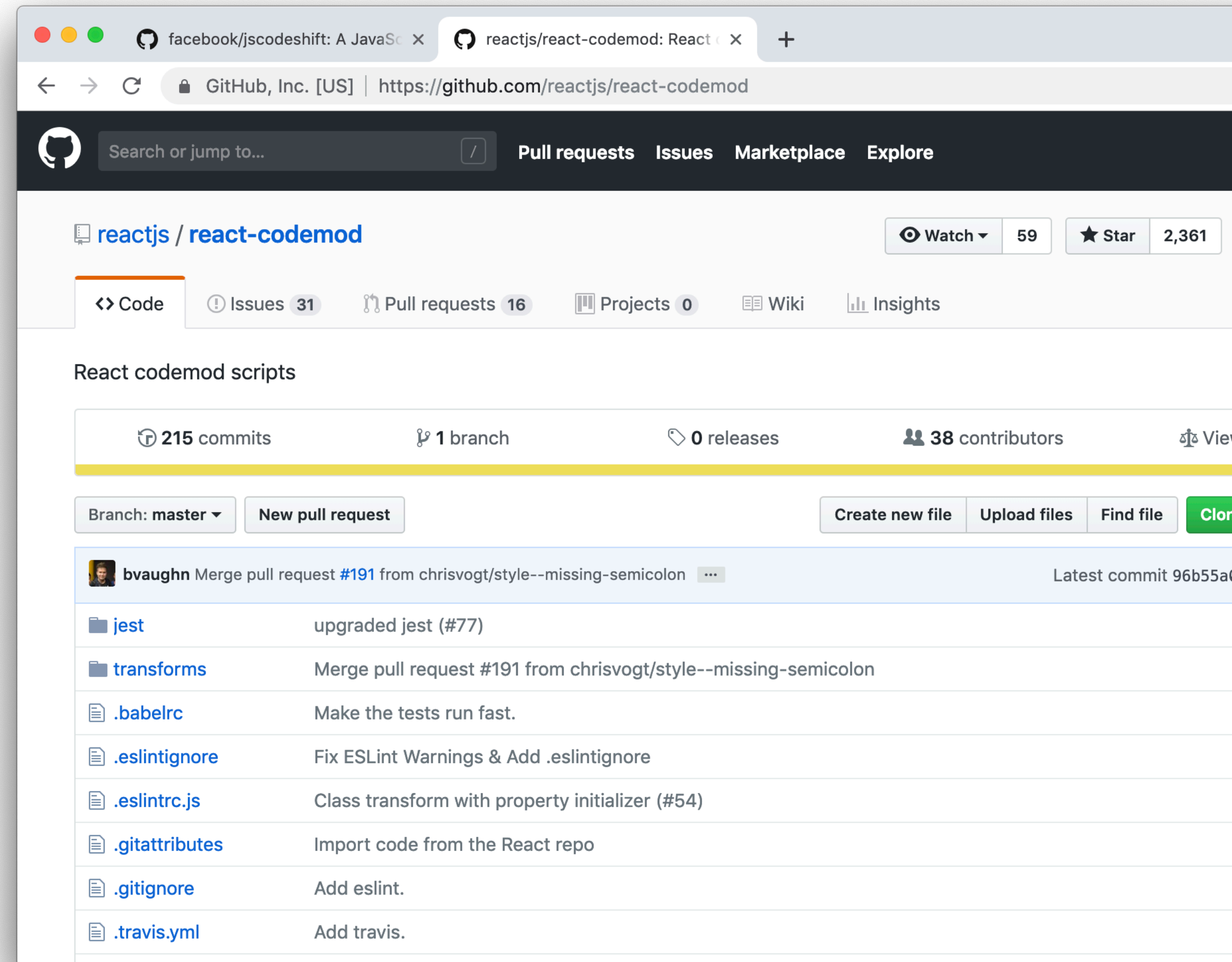
scheduler memo() Create React App v3

ReactDOM.createRoot() **Migrating Stuff**

github.com/facebook/jscodeshift



github.com/reactjs/react-codemod



jscodeshift

```
jscodeshift -t react-codemod/transforms/rename-unsafe-lifecycles.js
```

Path to transform file

Files/directories to
be transformed

```
jscodeshift -t react-codemod/transforms/rename-unsafe-lifecycles.js ./src
```

Path to transform file

```
jscodeshift -t react-codemod/transforms/rename-unsafe-lifecycles.js ./src
```

...

-t react-codemod/transforms/manual-bind-to-arrow.js

-t react-codemod/transforms/pure-component.js

jscodeshift -t react-codemod/transforms/rename-unsafe-lifecycles.js ./src

-t react-codemod/transforms/error-boundaries.js

-t react-codemod/transforms/findDOMNode.js

...

IN THE END...

React is such a good idea that we will spend the rest of the decade continuing to explore its implications and applications.



Guillermo Rauch @rauchg


7:14am - 22 Nov 2016



Xcode - SwiftUI - Apple Develo x ComponentKit | A React-inspir x +

https://componentkit.org

ComponentKit DOCS API SUPPORT GITHUB




ComponentKit

A React-Inspired View Framework for iOS

[GET STARTED](#)

Litho | A declarative framework x +

Docs API Tutorial GitHub




Litho: A declarative UI framework for Android

[GET STARTED](#) [LEARN MORE](#) [TUTORIAL](#)

Xcode - SwiftUI - Apple Develo x +

← → ↻ <https://developer.apple.com/xcode/swiftui/> ☆ ⚙️ 📄 👤 ⋮

Xcode Xcode 11 SwiftUI IDE Features What's New Resources



SwiftUI

Better apps. Less code.

SwiftUI is an innovative, exceptionally simple way to build user interfaces across all Apple platforms with the power of Swift. Build user interfaces for any Apple device using just one set of tools and APIs. With a declarative Swift syntax that's easy to read and natural to write, SwiftUI works seamlessly with new Xcode design tools to keep your



 SIGN UP TO OUR WEEKLY NEWSLETTER [SIGN UP HERE](#)

Software

Apple's tailored SwiftUI makes coding Mac and iOS apps RAD again

Developing for iGiant's platform just got way easier

By [Tim Anderson](#) 4 Jun 2019 at 10:40 18  SHARE ▼

 SIGN UP TO OUR WEEKLY NEWSLETTER

contextType createRef() forwardRef()

Lifecycle Changes <Strict Mode/> act()

lazy() react-cache Profiler <Suspense/>

scheduler memo() Create React App v3

ReactDOM.createRoot() Migrating Stuff

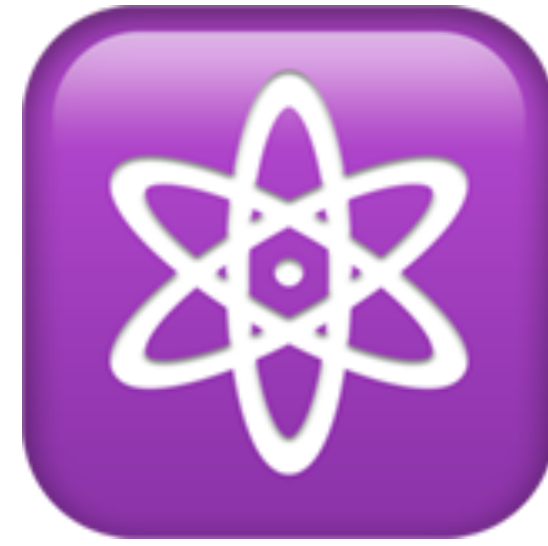
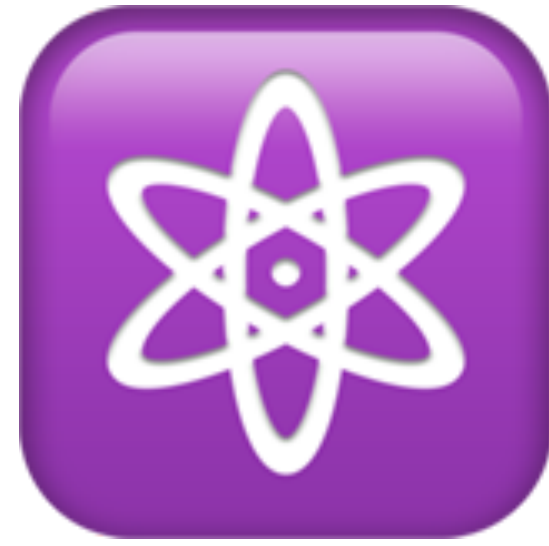
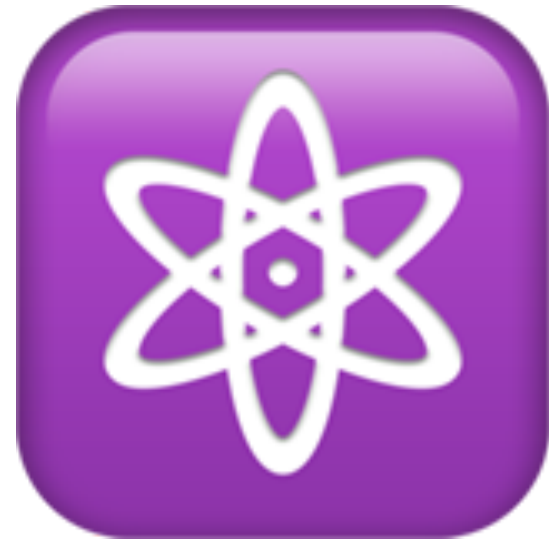
contextType createRef() **React Flare**

Lifecycle Changes **React Fusion** act()

lazy() react-cache Profiler **React Fire**

scheduler memo() Create React App v3

React Native Fabric Migrating Stuff

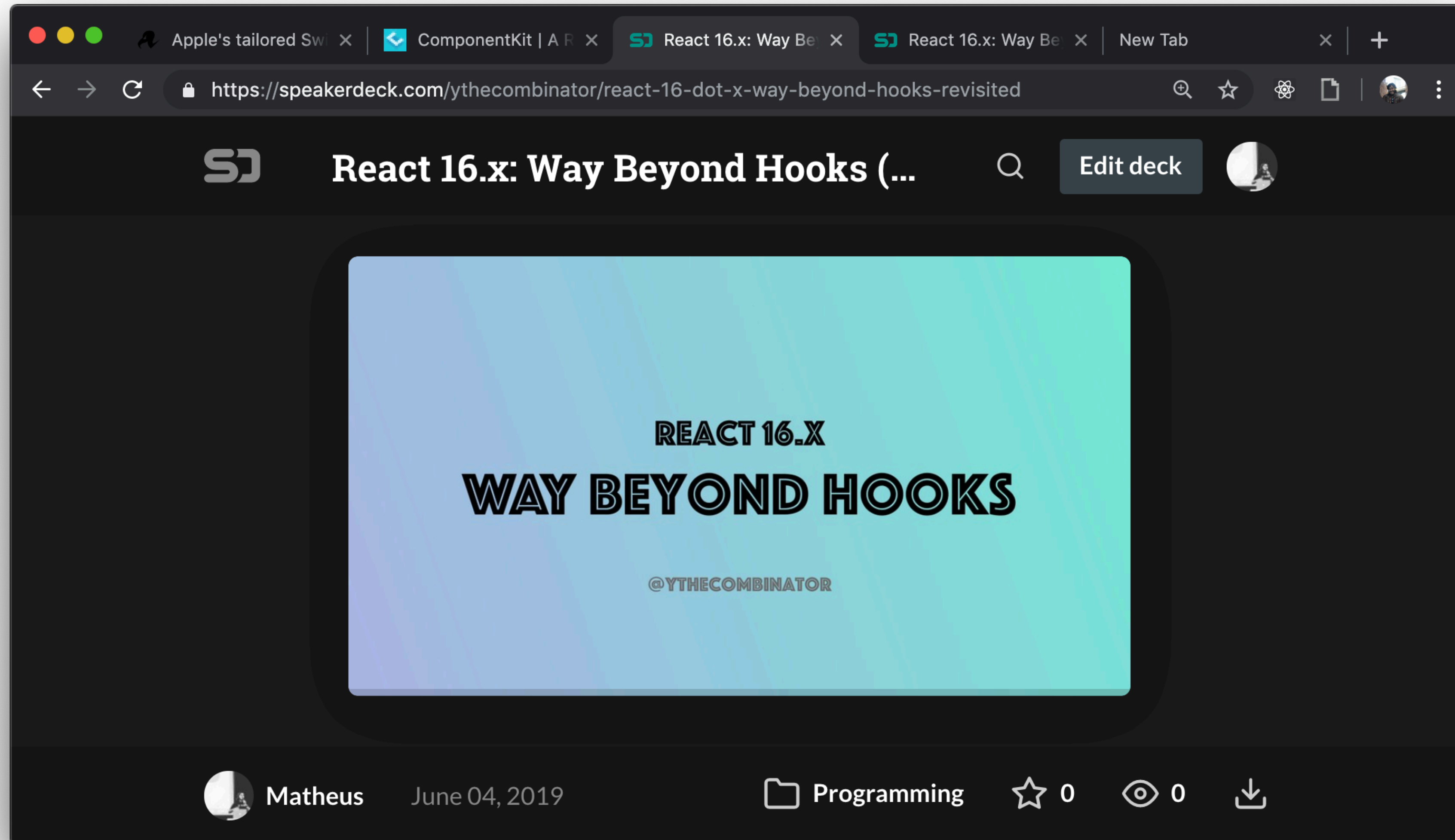




~ senior software engineer,
Front-End @STRV

~ **@ythecombinator** on the
webs

~ addicted to emojis, memes
and beer



<http://bit.ly/way-beyond-hooks-2019>

THANK YOU!

@YTHECOMBINATOR