

React 是一個 JavaScript 函數組件庫，用於構建用戶界面。

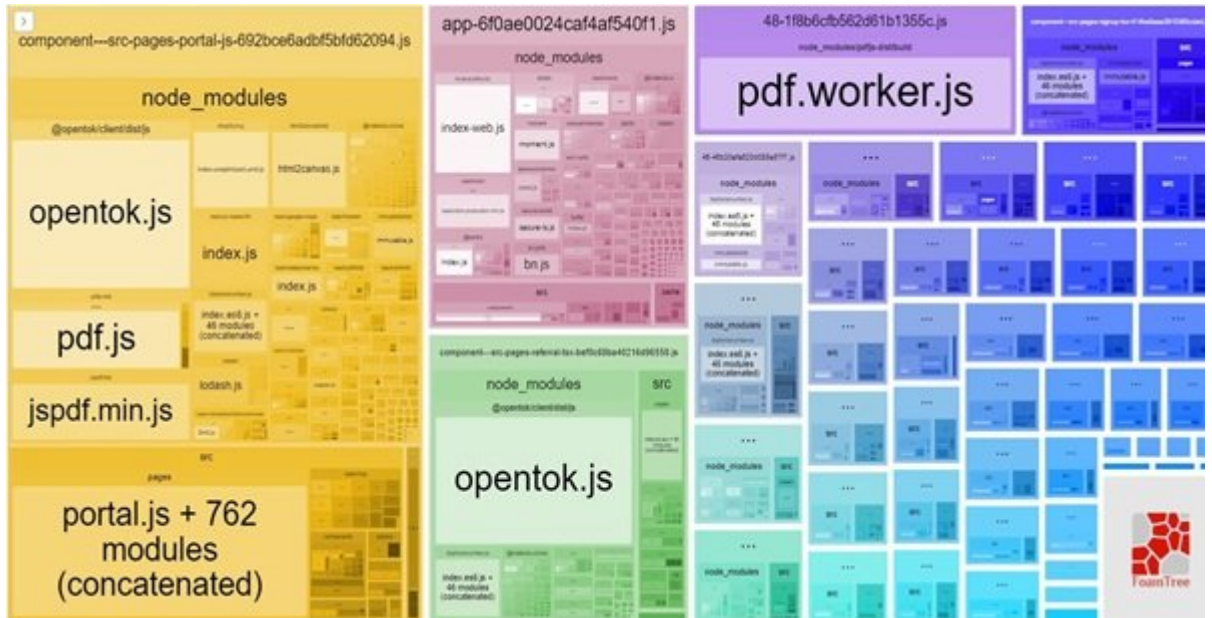
React 是一個 JavaScript 函數組件庫，用於構建用戶界面。



React 是一個 JavaScript 函數組件庫，用於構建用戶界面。React 是一個 JavaScript 函數組件庫，用於構建用戶界面。React 是一個 JavaScript 函數組件庫，用於構建用戶界面。

Webpack Bundle Analyzer .1

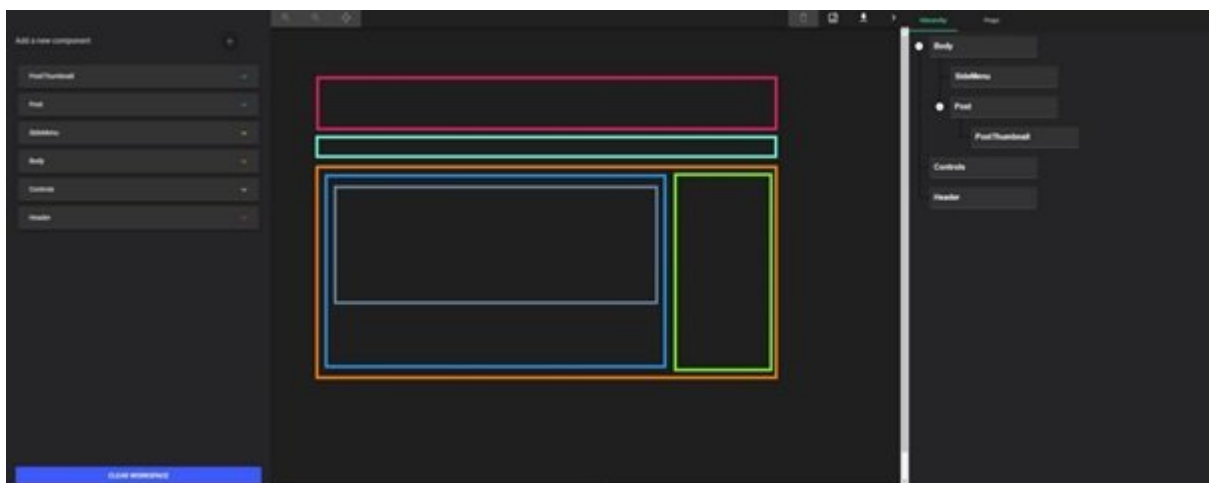
Webpack Bundle Analyzer 是一個用於分析 Webpack 打包結果的工具。它可以帮助你了解你的應用程序包的大小，並識別出哪些文件是過大的。你可以使用 Webpack Bundle Analyzer 來分析你的應用程序包，並生成一個可視化的報告。報告中會列出每個文件的尺寸，並提供一個可視化的餅圖來顯示文件的分布。你可以使用 Webpack Bundle Analyzer 來分析你的應用程序包，並生成一個可視化的報告。報告中會列出每個文件的尺寸，並提供一個可視化的餅圖來顯示文件的分布。你可以使用 Webpack Bundle Analyzer 來分析你的應用程序包，並生成一個可視化的報告。報告中會列出每個文件的尺寸，並提供一個可視化的餅圖來顯示文件的分布。



이러한 모듈 구조를 시각적으로 보여주는 도구를 사용하여, 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다.

React-Proto .2

이 도구는 React-Proto .2 버전에서 제공됩니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다.



이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다. 이 도구를 사용하여 애플리케이션의 모듈 구조를 분석하고, 불필요한 모듈을 제거하여 애플리케이션의 크기를 줄일 수 있습니다.



00000000 00 00000000 000000 00 000000 00 0000 React 00000 00000000 000000 00 [Why Did You Render](#) 000000
 00000000 000000 00 0000000000000000 00 00 00000000 000000 00 .000000 00000 00 0000 00000000 00000 00000
 0000 000000 00 0000000000 00 0000000000000000 .00000000 000000 00 react 00000000 00000 00 0 00000 0000 000000
 00 listener 00 000000 turn 000000 00 whyDidiYouRender 00 0000000 000000 00000 00 000000 00 0000000000
 .000000 00000 00 0000000 0000 0000 0000 .0000 0000000 00000000 000000 00

.
.

```

1  import React from 'react'
2  import Button from '@material-ui/core/Button'
3
4  const Child = (props) => <div {...props} />
5
6  const Child2 = ({ children, ...props }) => (
7    <div {...props}>
8      {children} <Child />
9    </div>
10 )
11
12 Child2.whyDidYouRender = true
13
14 const App = () => {
15   const [state, setState] = React.useState({})
16
17   return (
18     <div>
19       <Child>{JSON.stringify(state, null, 2)}</Child>
20       <div>
21         <Button type="button" onClick={() => setState({ hello: 'hi' })}>
22           Submit
23         </Button>
24       </div>
25       <Child2>Child #2</Child2>
26     </div>
27   )
28 }
29
30 export default App

```

wdy.jsx hosted with ❤ by GitHub

[view raw](#)

.jsx 파일을 jsx 파일로 변환하는 변환기인 babel을 사용하여 jsx 파일을 js 파일로 변환할 수 있습니다.

Create React App .4

간단하게 프로젝트를 생성하고 React 앱을 실행하는 방법을 알아보겠습니다. [Create React App](#)은 프로젝트를 생성하고 실행하는 데 필요한 모든 것을 제공합니다. Create React App을 사용하여 프로젝트를 생성하고 실행할 수 있습니다.

`npx create-react-app <name>`

.jsx 파일을 js 파일로 변환하는 변환기인 babel을 사용하여 jsx 파일을 js 파일로 변환할 수 있습니다. :jsx 파일을 js 파일로 변환하는 변환기인 babel을 사용하여 jsx 파일을 js 파일로 변환할 수 있습니다.

`npx create-react-app <name> - typescript`

간단하게 프로젝트를 생성하고 React 앱을 실행하는 방법을 알아보겠습니다. Create React App을 사용하여 프로젝트를 생성하고 실행할 수 있습니다.

React Lifecycle Visualizer .5

.[看看 React 如何渲染组件](#) 或者 [看看 npm 上的 **React Lifecycle Visualizer**](#)
[为什么你渲染了两次？](#) 这个页面可以帮助你理解为什么你的组件被渲染了两次。
 如果你已经了解了这些，那么你可以继续学习[React 的更多知识](#)。

: □□□□ □□□□ □□□ □□ □□□□□□ □□ □□□□ □□□□ □□□ □□□□ □□□□ □□□□ □□□□

SUBMIT

Child #2

Traced Component

Log

clear log



Delay: 0.5s

(hover to highlight, shift-up/down to navigate)

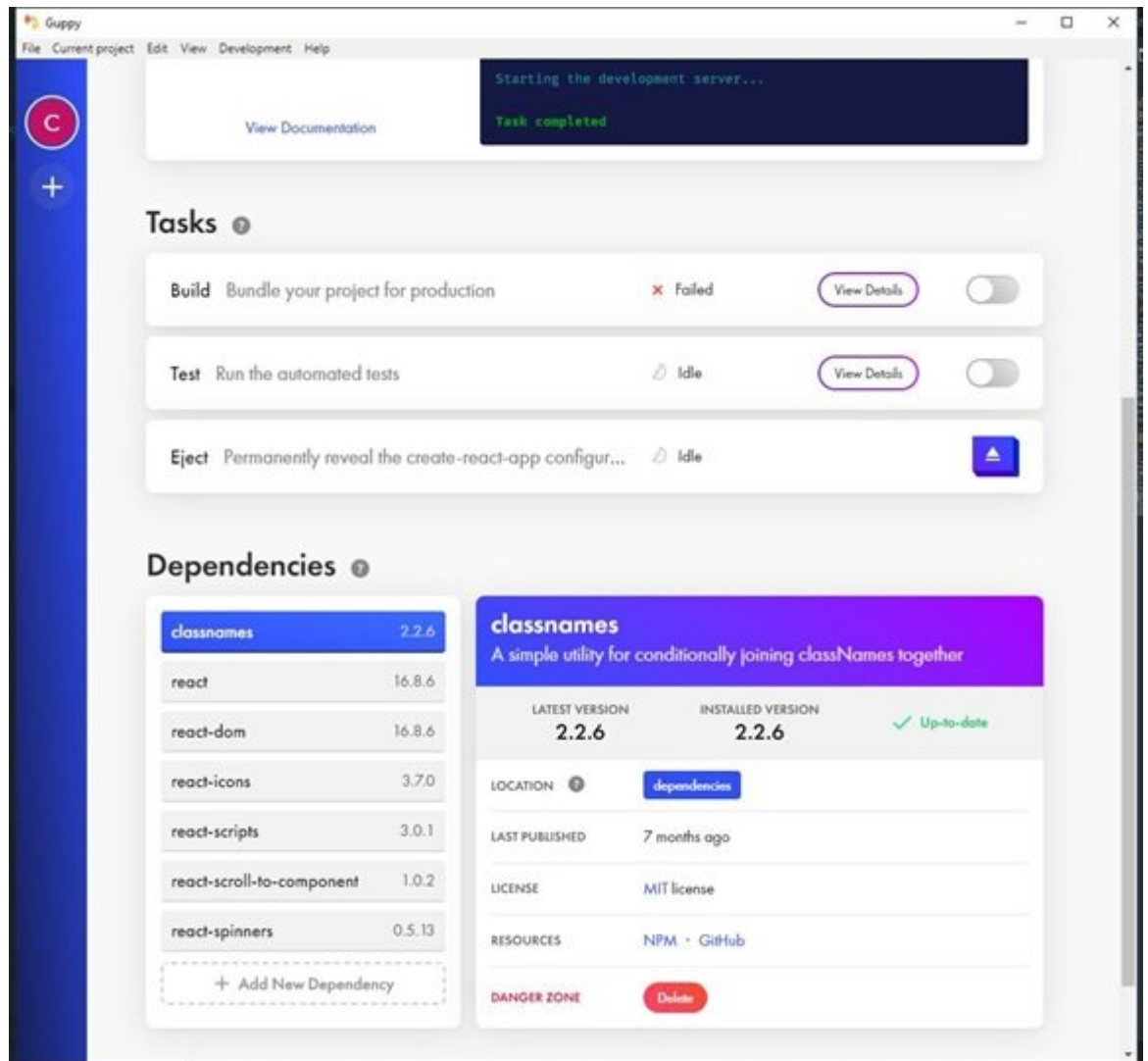
```
0 TracedComponent-1: constructor
1 TracedComponent-1: static getDerivedStateFromProps
2 TracedComponent-1: render
3 TracedComponent-1: componentDidMount
4 TracedComponent-1: static getDerivedStateFromProps
5 TracedComponent-1: shouldComponentUpdate
6 TracedComponent-1: render
7 TracedComponent-1: getSnapshotBeforeUpdate
8 TracedComponent-1: componentDidUpdate
9 TracedComponent-1: componentWillUnmount
10 TracedComponent-2: constructor
11 TracedComponent-2: static getDerivedStateFromProps
12 TracedComponent-2: render
13 TracedComponent-2: componentDidMount
14 TracedComponent-2: static getDerivedStateFromProps
15 TracedComponent-2: shouldComponentUpdate
16 TracedComponent-2: render
17 TracedComponent-2: getSnapshotBeforeUpdate
18 TracedComponent-2: componentDidUpdate
19 TracedComponent-2: componentWillUnmount
20 TracedComponent-3: constructor
```

በመጨረሻም በሀገር ውስጥ በሚኖሩ የሀገራችን ሕገ መንግሥት አንቀጽ 31 መሰረት የሚከተሉትን መብቶችና ንብረቶች ያላቸው ሕግ አገልግሎት ለማግኘት የሚችሉበትን ሁኔታ ለማረጋገጥና ለማረጋገጥ ይህ ስራ ይኖርባቸዋል፡፡

Guppy .6

[illegible]

.000000 000000 00 0000 000000 000000 0000 0000 0000 .000000 00 00 00000000 00 00000 000000 0000000000000000



react-testing-library .7

[illegible]


```
1 // Hoist helper functions (but not vars) to reuse between test cases
2 const renderComponent = ({ count }) =>
3   render(
4     <StateMock state={{ count }}>
5       <StatefulCounter />
6     </StateMock>,
7   )
8
9 it('renders initial count', async () => {
10   // Render new instance in every test to prevent leaking state
11   const { getByText } = renderComponent({ count: 5 })
12
13   await waitForElement(() => getByText(/clicked 5 times/i))
14 })
15
16 it('increments count', async () => {
17   // Render new instance in every test to prevent leaking state
18   const { getByText } = renderComponent({ count: 5 })
19
20   fireEvent.click(getByText('+1'))
21   await waitForElement(() => getByText(/clicked 6 times/i))
22 })
```

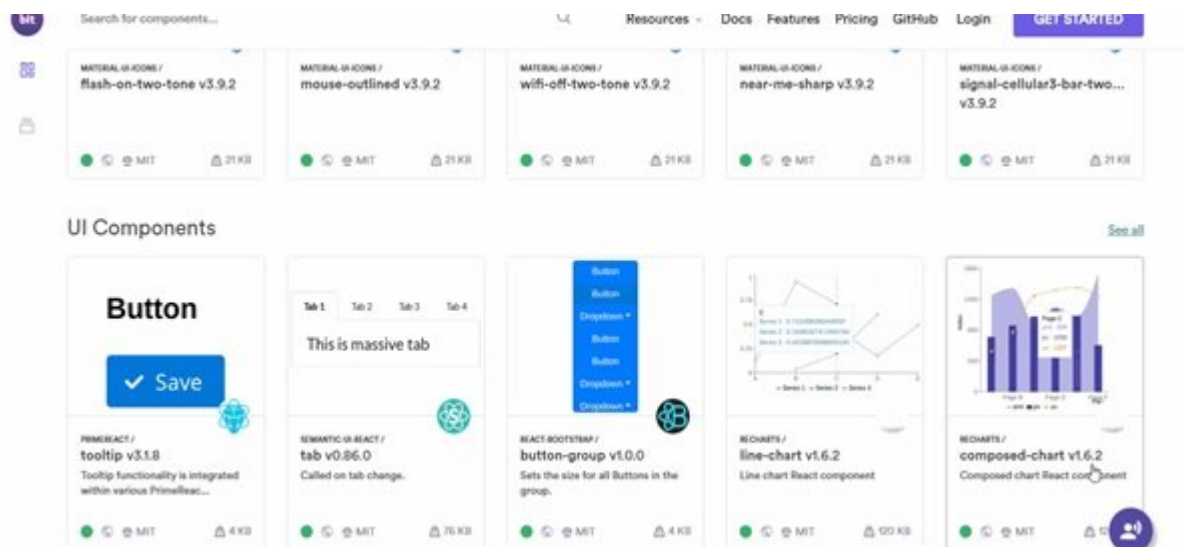
test.js hosted with ❤ by GitHub [view raw](#)

React Developer Tools .8

React Developer Tools는 React 애플리케이션을 개발하는 데 사용되는 도구입니다. 이 도구는 React 애플리케이션의 상태를 시각적으로 보여주고, 컴포넌트의 렌더링 과정을 추적할 수 있습니다. 또한, React의 디버깅 기능을 강화하여, 개발자가 애플리케이션의 동작을 더 깊이 이해할 수 있도록 도와줍니다. 이 도구는 React의 핵심 개념인 컴포넌트, 상태, props 등을 쉽게 이해하고 디버깅할 수 있는 환경을 제공합니다.

Bit .9

Bit은 Semantic UI React와 Material-UI와 같은 UI 라이브러리를 쉽게 배포하고 관리할 수 있는 도구입니다. 이 도구는 UI 컴포넌트를 모듈화하고, 이를 쉽게 배포하고 관리할 수 있도록 도와줍니다. 또한, Bit은 UI 컴포넌트의 배포와 관리를 단순화하여, 개발자가 더 빠르고 효율적으로 UI 컴포넌트를 개발하고 배포할 수 있도록 도와줍니다.



০০০০০০ ০(০০০০০০০০০০) ০০০০০০ ০০০০০০০০ ০০০০০০০০০ ০০০০০ ০০০০০০০০ ০০০০০ ০০০০০ ০০০০০০০০
 ০০০০ ০ ০০০০ ০০০ ০০ ০০০ ০০০ ০০ ০০০০০০ ০ ০০০০০০ ০০০০০০ ০০০০০০০০০০০০০ ০০০০০০ ০০০০০০০০ ০০০০০০০০০০০
 React ০০০০০০ ০০০ ০০০০০ ০০০ ০০০ ০০০০০০০০০ ০০০০ ০০০০ .০০০০০০ ০০০০ ০০০০০০০০০০০০০ ০০০০০০০ ০০ ০০০০০০০০০০০০
 .০০০ ০০০ ০০০০০

Storybook .10

00 0000000 00000000 000000 000000 000000 00000 00000 00000 00 00000000 0000 0000 00 0000 000000 000
 000 0000000 000000000 0000000 00 00 000000 000000 000000 0000 00 0000 000000.00000 000000000 [Storybook](#)
 0000 000000.0000 0000000 00 00 000000 0 0000000 0000 00 00 **React** 0000000000 000000 000000 0 0000 0000000000
 00 000000000 00 00000 000000.000000 000000 00 00000 00000000000 000 000000000000 000000 0 000 000000000000
 00000 **React** 0000000000 0000000 0000 00 **README** 000000000 00000 000000 **Storybook** **README** 00000
 .000000 00000 00 000000 000 000 0000.00000 00000 000000 00000 00000 00 00000000 000000000000

- LoadingSpinner
- **README**
- default
- dim (white bg)
- white (dark bg)
- renderContent
- fullscreen (white bg)
- fullscreen (dark bg)
- Typography
- default
- primary
- secondary
- thirdary
- link
- italic
- unselected
- centerOnSmall
- variants

```
const App = () => {
  <div style={{ width: 500, height: 300 }}>
    <LoadingSpinner margin={5} white>
      Loading...
    </LoadingSpinner>
  </div>
}
```

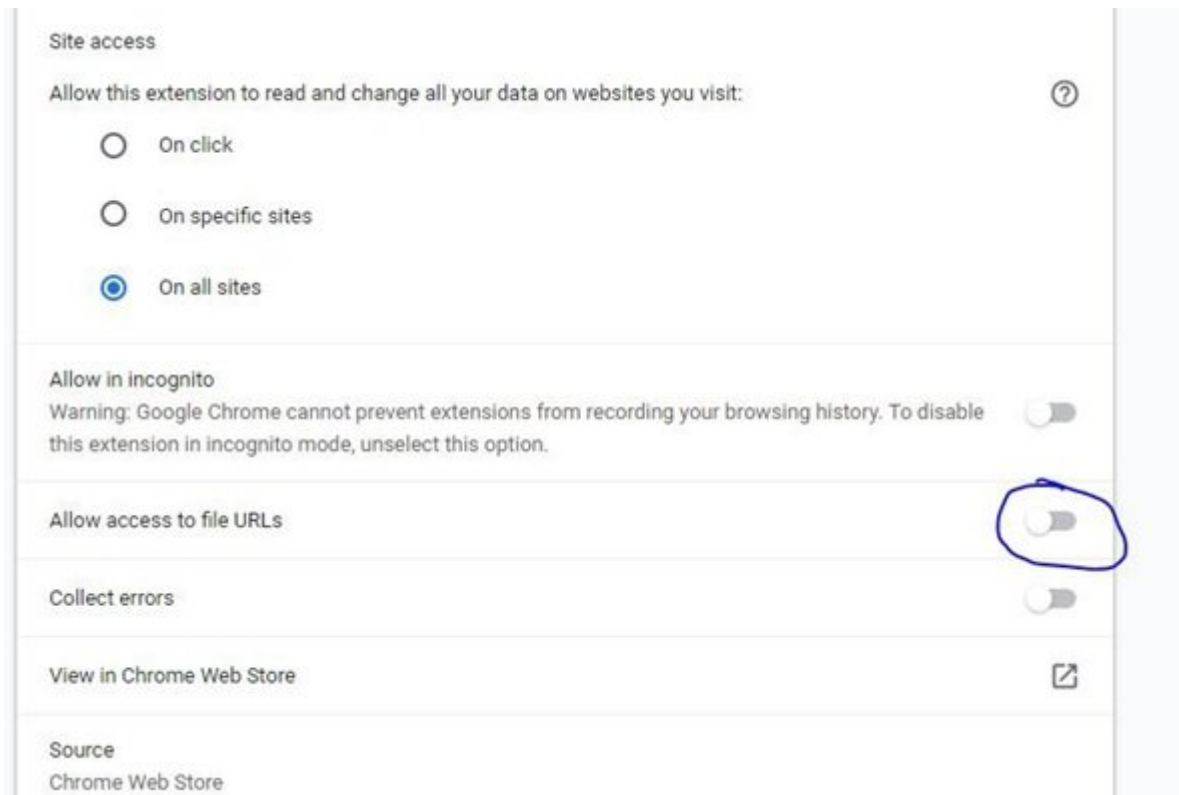
The loading spinner's logo will appear with dark text by default.

However, if you place this on a dark background the dark text will become camouflaged. You can pass in `white` as a prop, and the spinner the colors of the typography to white.

Prop	Type	Default	Description
children (optional)	React.ReactNode		
wrapperClassName (optional)	string		className passed to the wrapper
className (optional)	string		className passed to the loading spinner component
typographyClassName (optional)	string		className passed to the content component
wrapperStyle (optional)	React.CSSProperties		A CSS styles object passed to the wrapper

React Sight .11

0000 000000 000000 0000000000 00 00 0000 000000 0000000000 00 0000 000000 000000 00 000000 0000 0000 00 00
 0000 00 000000 00 React 0000000000 0000000000 00000000 [React Sight](#) 00000000 0000 00000000 0000 00
 00 0000 0000 0000000000 .000000 000000 00 00000000 00000000 00 00 0000 0000000000 00 0000 0000 0000000000
 0000 000000 000000 00000000000000 00 React Sight .000000 0000000000 React Fiber [] Redux[] react-router
 0000 00000 0000 000000 00000000 0000 .0000 0000 000000 000000 0000000000 00 00000000 0000 00 00 00000000
 React Sight 0000 000000 00 0 0000 0000 0000 00000000 0000 00 00[]chrome:extensions 000000 0000 0000
 .0000 00000 Allow access to file URLs 000000 0000 0000 0 00000000 box



React Cosmos .12

.این پکیج به شما امکان می‌دهد که React کامپوننت‌ها را در محیطی شبیه به مرورگر اجرا کنید. این پکیج به شما امکان می‌دهد که **React Cosmos** props را به کامپوننت‌ها بدهید و همچنین می‌توانید state و localStorage را شبیه‌سازی کنید. همچنین می‌توانید context را شبیه‌سازی کنید. این پکیج به شما امکان می‌دهد که

:نصب
نصب
:نصب
نصب
:نصب
06:35 - 27/10/1398
:نصب

[react](#) - [React](#) - [React](#) - [React](#) - [React](#) - [React](#)

<https://www.shabakeh-mag.com/workshop/16473/%DB%B1%DB%B2-%D8%A7%D8%A8%D8%B2%D8%A7%D8%B1-%DA%A9%D8%A7%D8%B1%D8%A8%D8%B1%D8%AF%DB%8C-%D8%A8%D8%B1%D8%A7%DB%8C-%D8%AA%D9%88%D8%B3%D8%B9%D9%87%E2%80%8C%D8%AF%D9%87%D9%86%D8%AF%DA%AF%D8%A7%D9%86%DB%8C-%DA%A9%D9%87-%D8%A7%D8%B2-react-%D8%A7%D8%B3%D8%AA%D9%81%D8%A7%D8%AF%D9%87-%D9%85%DB%8C%E2%80%8C%DA%A9%D9%86%D9%86%D8%AF>