

# ReactJS Optimization Techniques and Development Resources

*By Nilanth*



# **ReactJS Optimization Techniques and Development Resources**

*eBook by Nilanth*

# Packages to Optimize and Speed Up During Development

## 1. Why Did You Render (6.6K ★)

Why-did-you-render is a React package that allows you to find potentially avoidable re-renders. Most of the performance issues will arise from unwanted re-renders. If a big list component re-renders multiple times, It will make the app unresponsive.

To avoid these issues, we will use `pureComponents` or `useMemo` but in some cases, those also re-render due to misuse of state updates. We can avoid these using the **why-did-you-render package**. It will notify when and why the component re-rendered!

**Note: For development use only!**

```
► {App: Object} "Re-rendered because of hook changes:"
```

```
[hook useState result]
```

```
different objects that are equal by value. (more info at http://bit.ly/wdyr3)
```

```
▼ {prev : Object}      "!==      ▼ {next : Object}
  ▼ prev : Object      ▼ next : Object
    firstName: "Bob"    firstName: "Bob"
    lastName: "Sawyer"  lastName: "Sawyer"
```

## 2. Source Map Explorer (3.3K )

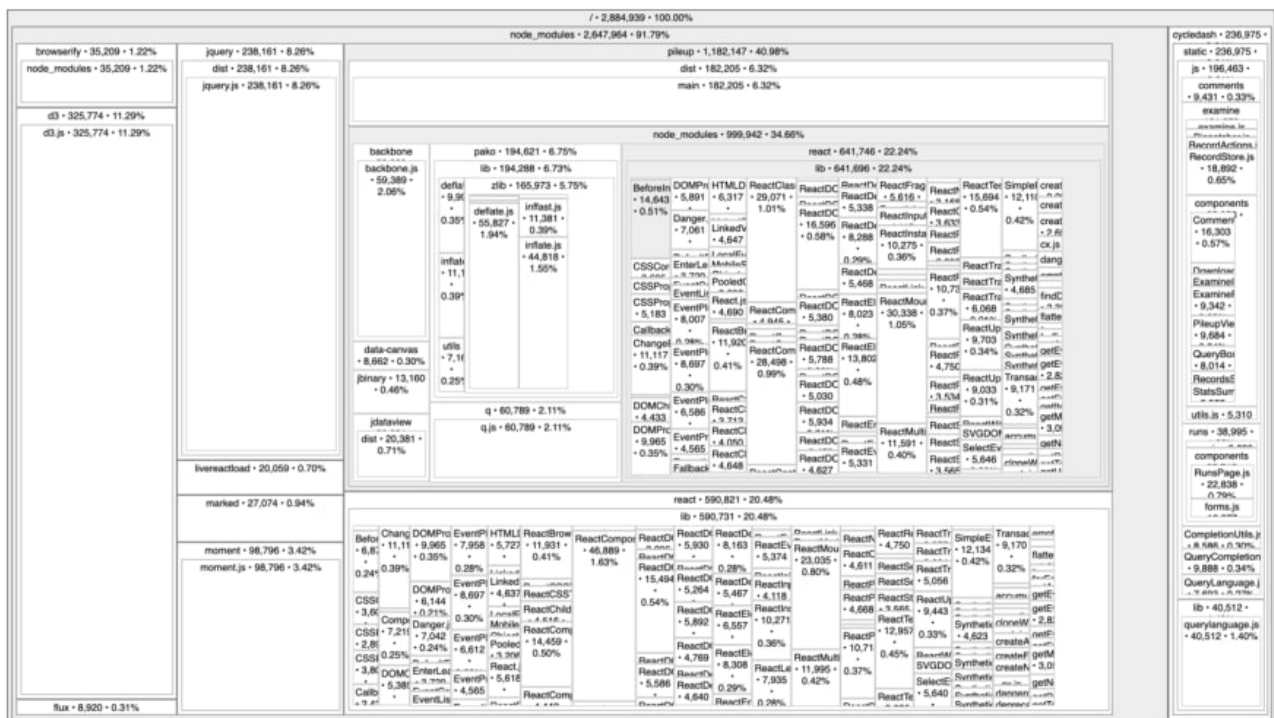
Source Map Explorer gives a view of the build with each file size that occupies the build. It allows knowing which dependency occupies a larger file size in the bundle. We could optimize the file based on the view.

### **Why this important?**

As your app features grow, the build size also increases. A large build size will take more time to build.

We need to keep the build size small as much as we can. Using Source Map Explorer, we can analyze the build and optimize it. It also supports Sass and LESS files.

**Note: For development use only!**



### 3. Redux Immutable State Invariant (800+★)

Redux Immutable State Invariant is a Redux middleware. It detects mutations between and outside Redux dispatches.

If you are using Redux for state management, you **should not mutate the state** in the reducer or outside. As **reducer** always returns a **new state object**.

Mutating the state will lead to several issues in your app. To avoid this, we can use Redux Immutable State Invariant middleware.

This package will throw an error if the state is mutated. So we can fix these issues in the development stage itself.

**Note: For development use only!**

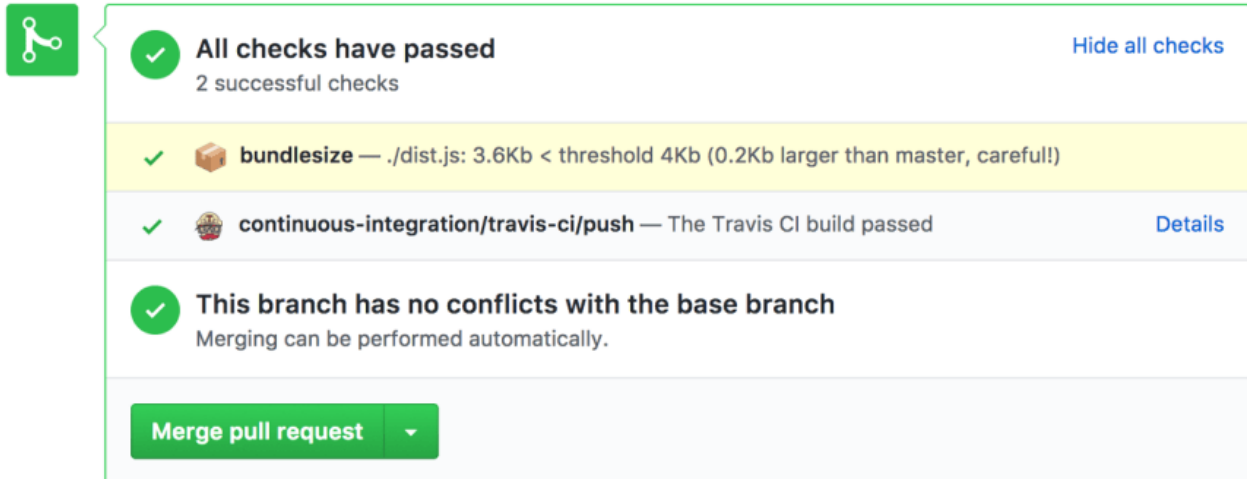
## 4. Bundle Size (4.2K ★)

Bundle Size allows keeping your bundle size in check. We can configure each file size in the config file. So we will get a warning if the size exceeds. It ensures the bundle size is in control.


Check the below bundle size config used by bootstrap.


```
1  {
2    "files": [
3      {
4        "path": "./dist/css/bootstrap-grid.css",
5        "maxSize": "7.25 kB"
6      },
7      {
8        "path": "./dist/css/bootstrap-grid.min.css",
9        "maxSize": "6.5 kB"
10     },
11     {
12       "path": "./dist/css/bootstrap-reboot.css",
13       "maxSize": "2 kB"
14     },
15     {
16       "path": "./dist/css/bootstrap-reboot.min.css",
17       "maxSize": "2 kB"
18     },
19     {
20       "path": "./dist/css/bootstrap-utilities.css",
21       "maxSize": "7.25 kB"
22     },
23     {
24       "path": "./dist/css/bootstrap-utilities.min.css",
25       "maxSize": "6.6 kB"
26     },
27     {
28       "path": "./dist/css/bootstrap.css",
29       "maxSize": "25 kB"
```


You can also add it to your GitHub to check on every pull request. Check the below image.



The image shows a GitHub pull request status bar. On the left is a green square icon with a white branching diagram. To its right is a white box with a green border. Inside the box, at the top, is a green circle with a white checkmark, followed by the text "All checks have passed" and "2 successful checks". In the top right corner of this box is a blue link "Hide all checks". Below this is a list of checks. The first check has a green checkmark, a brown box icon, and the text "bundlesize — ./dist.js: 3.6Kb < threshold 4Kb (0.2Kb larger than master, careful!)". The second check has a green checkmark, a Travis CI logo, and the text "continuous-integration/travis-ci/push — The Travis CI build passed". To the right of this second check is a blue link "Details". Below the checks is a green circle with a white checkmark, followed by the text "This branch has no conflicts with the base branch" and "Merging can be performed automatically.". At the bottom of the box is a green button with the text "Merge pull request" and a small downward arrow.

 **All checks have passed** [Hide all checks](#)  
2 successful checks

✓  **bundlesize** — ./dist.js: 3.6Kb < threshold 4Kb (0.2Kb larger than master, careful!)

✓  **continuous-integration/travis-ci/push** — The Travis CI build passed [Details](#)

✓ **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** ▾



## 5. Immer ( 20.7K ★ )

Mutating a state will lead to more issues. To avoid this, we need to manually clone every state object and mutate it.

Instead of a manual clone, We can use Immer, which will handle it in a more efficient and optimized way. Immer allows mutating states more conveniently. You can also use Immer to mutate states in Redux.

Check the below code for Immer usage with Redux.

```
import produce from "immer"

// Reducer with initial state
const INITIAL_STATE = [
  /* bunch of todos */
]

const todosReducer = produce((draft, action) => {
  switch (action.type) {
    case "toggle":
      const todo = draft.find(todo => todo.id === action.id)
      todo.done = !todo.done
      break
    case "add":
      draft.push({
        id: "todo_" + Math.random(),
        title: "A new todo",
        done: false
      })
      break
    default:
      break
  }
})
```

# Custom Hooks to Make Your Development Easier

Here we will see 15 react-use package custom hooks that make our development easier.

## 1.useIdle

The useIdle hook tracks if the user on the page is idle. You can pass two params — one is time to consider idle and initialState, which allows the setting user is idle initially.

```
import {useIdle} from 'react-use';
const Demo = () => {
  const isIdle = useIdle(3e3);
  return (
    <div>
      <div>User is idle: {isIdle ? 'Yes 🤖' : 'Nope'}</div>
    </div>
  );
};
```


## 2. useInterval

This hook to use for interval-related functionalities. Which handles `clearInterval` on component unmount automatically. It also allows pausing the interval by setting the delay to null.

```
import * as React from 'react';
import {useInterval} from 'react-use';
const Demo = () => {
  const [count, setCount] = React.useState(0);
  const [delay, setDelay] = React.useState(1000);
  const [isRunning, toggleIsRunning] = useBoolean(true);
  useInterval(
    () => {
      setCount(count + 1);
    },
    isRunning ? delay : null
  );
  return (
    <div>
      <div>
        delay: <input value={delay} onChange={event => setDelay(Number(event.target.value))} />
      </div>
      <h1>count: {count}</h1>
      <div>
        <button onClick={toggleIsRunning}>{isRunning ? 'stop' : 'start'}</button>
      </div>
    </div>
  );
};
```

### 3. useScroll

This hook is used to listen to the scroll event of the element and rerenders on scrolling. No required to add the JavaScript event listeners manually.



```
import {useScroll} from 'react-use';
const Demo = () => {
  const scrollRef = React.useRef(null);
  const {x, y} = useScroll(scrollRef);
  return (
    <div ref={scrollRef}>
      <div>x: {x}</div>
      <div>y: {y}</div>
    </div>
  );
};
```

## 4. useToggle

This hook is used to toggle between two states like TRUE, FALSE. This approach reduces the manual logic.

```
import {useToggle} from 'react-use';
const Demo = () => {
  const [on, toggle] = useToggle(true);
  return (
    <div>
      <div>{on ? 'ON' : 'OFF'}</div>
      <button onClick={toggle}>Toggle</button>
      <button onClick={() => toggle(true)}>set ON</button>
      <button onClick={() => toggle(false)}>set OFF</button>
    </div>
  );
};
```

## 5. useTitle

This hook is used to set the page title.



```
import {useTitle} from 'react-use';  
const Demo = () => {  
  useTitle('Hello world!');  
  return null;  
};
```

## 6. usePrevious

This hook is used to get the previous state. We might not require to write custom logic to get the previous state.

```
import {usePrevious} from 'react-use';  
const Demo = () => {  
  const [count, setCount] = React.useState(0);  
  const prevCount = usePrevious(count);  
  return (  
    <p>  
      <button onClick={() => setCount(count + 1)}>+</button>  
      <button onClick={() => setCount(count - 1)}>-</button>  
    <p>  
      Now: {count}, before: {prevCount}  
    </p>  
  </p>  
);  
};
```

## 7. useState

This hook is used to merge objects into their current state, similar to the `this.setState` in the class component. If you are using multiple states, it can be brought down to a single object state using `useState`

```
import {useState} from 'react';
const Demo = () => {
  const [state, setState] = useState({});
  return (
    <div>
      <pre>{JSON.stringify(state, null, 2)}</pre>
      <button onClick={() => setState({hello: 'world'})}>hello</button>
      <button onClick={() => setState({foo: 'bar'})}>foo</button>
      <button
        onClick={() => {
          setState((prevState) => ({
            count: (prevState.count || 0) + 1,
          }))
        }}
      >
        count
      </button>
    </div>
  );
};
```



## 8. useCookie

This hook is used to return the current value of a cookie, a callback to update the cookie and a callback to delete the cookie.

```
import { useCookie } from "react-use";
const Demo = () => {
  const [value, updateCookie, deleteCookie] = useCookie("my-cookie");
  const [counter, setCounter] = useState(1);
  useEffect(() => {
    deleteCookie();
  }, []);
  const updateCookieHandler = () => {
    updateCookie(`my-awesome-cookie-${counter}`);
    setCounter(c => c + 1);
  };
  return (
    <div>
      <p>Value: {value}</p>
      <button onClick={updateCookieHandler}>Update Cookie</button>
      <br />
      <button onClick={deleteCookie}>Delete Cookie</button>
    </div>
  );
};
```

## 9. usePermission

This hook is used to get the permission status of the browser API. Pass the API name to get the permission status.



```
import {usePermission} from 'react-use';
const Demo = () => {
  const state = usePermission({ name: 'microphone' });
  return (
    <pre>
      {JSON.stringify(state, null, 2)}
    </pre>
  );
};
```

## 10. useDebounce

This hook is used to delay the event until the wait time is completed. In the below code the setState is performed after the wait time is completed.

```
const Demo = () => {
  const [state, setState] = React.useState('Typing stopped');
  const [val, setVal] = React.useState('');
  const [debouncedValue, setDebouncedValue] = React.useState('');
  const [, cancel] = useDebounce(
    () => {
      setState('Typing stopped');
      setDebouncedValue(val);
    },
    2000,
    [val]
  );
  return (
    <div>
      <input
        type="text"
        value={val}
        placeholder="Debounced input"
        onChange={({ currentTarget }) => {
          setState('Waiting for typing to stop ... ');
          setVal(currentTarget.value);
        }}
      />
      <div>{state}</div>
      <div>
        Debounced value: {debouncedValue}
        <button onClick={cancel}>Cancel debounce</button>
      </div>
    </div>
  );
};
```

## 11. useGeolocation


This hook is used to get the user geolocation. useGeolocation returns latitude, longitude, altitude, and more info.



```
import {useGeolocation} from 'react-use';
const Demo = () => {
  const state = useGeolocation();
  return (
    <pre>
      {JSON.stringify(state, null, 2)}
    </pre>
  );
};
```

## 12. useNetworkState

This hook is used to get the network status of the browser. useNetworkState can be used to show the connection status to the user.



```
import {useNetworkState} from 'react-use';
const Demo = () => {
  const state = useNetworkState();
  return (
    <pre>
      {JSON.stringify(state, null, 2)}
    </pre>
  );
};
```

## 13. useCopyToClipboard

useCopyToClipboard hook is used to copy the text to the clipboard.

```
const Demo = () => {
  const [text, setText] = React.useState('');
  const [state, copyToClipboard] = useCopyToClipboard();

  return (
    <div>
      <input value={text} onChange={e => setText(e.target.value)} />
      <button type="button" onClick={() => copyToClipboard(text)}>copy text</button>
      {state.error
        ? <p>Unable to copy value: {state.error.message}</p>
        : state.value && <p>Copied {state.value}</p>}
    </div>
  )
}
```

## 14. useFavicon

The useFavicon hook is used to set the favicon of the page.

```
import {useFavicon} from 'react-use';
const Demo = () => {
  useFavicon('https://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico');
  return null;
};
```

## 15. useError

useError hook is used to dispatch errors.



```
import { useError } from 'react-use';
const Demo = () => {
  const dispatchError = useError();
  const clickHandler = () => {
    dispatchError(new Error('Some error!'));
  };
  return <button onClick={clickHandler}>Click me to throw</button>;
};
// In parent app
const App = () => (
  <ErrorBoundary>
    <Demo />
  </ErrorBoundary>
);
```

# Free Hosting Services

## 1. Netlify



# netlify

Netlify is an all-in-one platform for automating modern web projects. It provides continuous deployment using GitHub, Bitbucket and Gitlab. React App can be deployed in 3 steps.

It also provides a free automatic HTTPS. You can also add a custom domain. With Netlify Edge, Your React Apps are accessed to the client blazing fast.

Other than deployment, Netlify also provides serverless functions, Forms, Analytics, CLI, API and more. Most of these features are provided for free with some limitations.



## 2. Vercel



Vercel creator of Next.js, A modern react Framework. You can deploy React App with Zero configuration in Vercel. It will boost the app performance with its global edge network.

Vercel provides a preview link for Pull Request in Bitbucket, Github and GitLab to test the feature before deploying.

Vercel also provides some starter templates to create a new App and deploy it. It Provides continuous deployment, Serverless functions, HTTPS and more.

### 3. AWS S3



Amazon Web Services is the world's leading cloud service provider. It provides almost every cloud services, and some services are given only by AWS.

S3 is also one of the AWS services. S3 Bucket is known for storing static assets. The most common use of S3 is to save images. It also provides static site hosting.

You just need to upload the React Build files to the bucket. Once Upload is completed. You can access the app using the bucket URL. You can also configure CloudFront to add a custom domain and HTTPS.

AWS provides 12 months of free credits on the new account. You can use that free credits for using S3 and also other AWS services.

## 4. AWS Amplify



Amazon Web Services also provides another service called Amplify to host your React App. Amplify will build and host your React App with global CDN.

Amplify can be integrated with Git services to make the continuous deployment. It also provides HTTPS, Custom Domain, Monitoring, Password Protection, Pull-Request previews and More.

Amplify comes with 12 months free of cost. As it used Amazon CloudFront CDN, the Deployed Apps are faster and cached in nearby edge locations to serve very quickly.

## 5. Microsoft Azure



Microsoft provides Azure Static Web Apps to host your React App. The hosted app is served from distributed points globally to provide better performance.

Azure provides Free HTTPS, Custom Domains, Versioning, Git Integration and more.

Azure has a free plan with these services. It provides continuous deployment using Git integration.

## 6. GitHub Pages



GitHub Pages is a service by GitHub, The largest and most advanced development platform in the world.

You can directly host your React app from the GitHub repository. You just need to make your changes and push to make your React App live.

GitHub Pages provide Free HTTPS and Custom Domain. You can configure the GitHub pages with some simple steps.

## 7. Google Cloud Storage



Google Cloud provides Cloud Storage Bucket to host static sites. All you need to create a bucket, upload the code and make it public. Now your React App is deployed.

Cloud Storage Bucket does not provide HTTPS and custom domain. You can configure it with the HTTPS Load Balancing service.

Cloud Storage Bucket has a Free Tier plan with monthly limits. Google also provides **300\$** free credits for new Account.

## 8. Render



Render is a Cloud Service Provider, Provides services for both Static and Dynamic Site. You can host your React App with three simple steps.

Render provides Free SSL, Global CDN, Custom Domain, Auto Deploy with Git Integration.

Render provides a Free Plan to Host Static Site and Competitive Pricing for Other services too.

## 9. Surge



Surge is a Static Website Hosting Platform. You can deploy using the surge CLI.

Surge doesn't provide any Web Console to host the Web Pages. You can host your React App from your CLI.

You can host it with a few steps using the surge CLI. It provides Free SSL, Custom Domain Configuration. Hosting in surge might require some CLI skills.



## 10. Heroku



Heroku is a container-based cloud Platform as a Service. Heroku provides mostly all cloud services like AWS. It has a Free Plan for most of the services.

You can deploy your React App using Heroku Buildpack for create-react-app. Using Heroku CLI, You can deploy the React App with few commands.

The Buildpack is used for Automatic Deployment and a built-in Bundler to make the deployment less complicated.

Heroku provide free SSL, Custom Domain and Git Integration.

## **Additional Hosting Services**

- 1. Firebase**
- 2. CloudFlare Pages**
- 3. Gitlab Pages**

# UI Components Packages

## 1. Ant Design


Ant Design is an enterprise-class UI design language and React UI library. Which is the most popular React UI library based on GitHub Stars.

It has 100 plus components from typography to tables. Ant Design document is very clean and has clear examples.

Ant Design does not save only developer time, it saves designers time also, As it includes Sketch and Figma files for all components.

Ant Design component supports both JSX and TypeScript. Customizing the ant theme is also very simple.

Ant Components save a lot of time for developers in handling forms and validation as it has prebuilt form components. Ant Design also supports hooks.


***GitHub - 75.4K*** 

## 2. Material-UI

Material-UI is also the most popular React UI library, It is a simple and customizable component library to build faster, beautiful, and more accessible React applications.

Material-UI contains 100 plus components. It also includes 1K plus icons.

Material UI also provides paid Sketch, Figma, Adobe Xd files for designers. Material UI is also used by top organizations like Spotify, NASA, Netflix, Amazon and more. Material UI has well-prepared documentation with code samples.


***GitHub - 72.7K*** 

### 3. Chakra UI

Chakra UI provides a set of accessible, reusable, and composable React components that make it super easy to create websites and apps.

Chakra UI components follow the WAI-ARIA guidelines specifications and have the right aria-\* attributes. Chakra UI community is growing faster due to its performance and experience.

Chakra UI has well-prepared documentation with code samples.

***GitHub - 21.7K*** 

## 4. React Bootstrap

React Bootstrap enables to use of Bootstrap JS for React Component. React Bootstrap components are build from scratch with react and not contains jquery.

React Bootstrap contains all the bootstrap components which we used with JavaScript. Now it includes Bootstrap 5 in the beta stage. React Bootstrap has well-prepared documentation with code samples.

***GitHub - 20.1K*** ★

## 5. Semantic UI React

Semantic is a UI component framework based around useful principles from natural language.

Semantic UI React is the official Semantic-UI-React integration. It contains 50 plus components, JQuery Free, Auto Controlled State, Sub Components and more. If your react App needs Semantic UI you can prefer this package.

***GitHub - 12.5K*** ★

## 6. Fluent UI

Fluent is an open-source, cross-platform design system that gives designers and developers the frameworks they need to create engaging product experiences-accessibility, internationalization, and performance included. Fluent design is used for Windows 10 devices, tools and also for Windows 11.

Fluent UI is developed by Microsoft, It has a collection of utilities, React components and web components for building web applications. It has good documentation.

***GitHub - 12.4K*** ★


## 7. Evergreen

Evergreen is the UI framework upon is build product experiences at Segment. It serves as a flexible framework, and a lot of its visual design is informed through plenty of iteration with the segment design team and external contributors. Evergreen has 30 Plus components and the documentation also seems good.

***GitHub - 11.2K*** ★

## 8. Reactstrap


Reactstrap helps to use Bootstrap 4 Components with react. This is simple to configure and use. It has good documentation for using components.

***GitHub - 10.1K*** 

## 9. Grommet

Grommet is a react-based framework that provides accessibility, modularity, responsiveness, and theming in a tidy package. It has 60 Plus components.

It also provides Sketch, Figma, AdobeXd files and 600 plus SVG icons. Grommet is used by Netflix, Samsung, Uber, Boeing, IBM and more organizations.

***GitHub - 7.6K*** 



## 10. Reakit

Reakit is a lower-level component library for building accessible high-level UI libraries, design systems and applications with React. Reakit is tiny and fast.

***GitHub - 5K*** ★

## 11. Mantine

Mantine is a React components and hooks library with native dark theme support and focus on usability, accessibility and developer experience. Mantine includes more than 100 customizable components and hooks.

***GitHub - 2.9K*** ★

## 12. Blueprint

Blueprint is a React-based UI toolkit for the web.

It is optimized for building complex, data-dense web interfaces for *desktop applications* which run in modern browsers and IE11. This is not a mobile-first UI toolkit.

***GitHub - 18.3K*** ★