|  |  |
| --- | --- |
| **ch44.ReactJS React-icons** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# React-icons



Icons in the computing world are defined as **pictorial representations** or **ideograms** displayed on a computer screen that conveys a meaning. Icons convey characteristics that separate them from each other and make a thing exist uniquely. Under all of these aspects, they are also called symbols popularly that quickly comprehend thoughts about a software tool, a function, a file, a data. Icons represent real-world entities that contain detailed illustrations and are designed so that one symbol speaks it all.

In this article, we will be talking about React Icons specifically. React is a powerful library that is dynamic and smart. It has all such assets that make it unique. We would also be learning about how what, and why of React Icons in this article and carry out the discussion by practically implementing them. Let's learn more about React Icons.

## **What are React Icons?**

React is a JavaScript library. Similarly, React Icons is also an open-source library that can bring all the icons present in the world of applications all at once place in one directory.

## **Why React Icons?**

The following reasons can correctly justify why React Icons are chosen.

24.4M

477

History of Java

1. In some instances, one library might not contain all the essential icons for usage. Thus, there arises a problem of installing some other libraries to import them into your project directory. React Icons saves developers from having to install different types of packages for such icons.
2. ES6 imports are greatly utilized by React Icons that allow us to include only those necessary icons for the current project. Hence, it removes space-related confusion.
3. React Icons are designed solely for React. This means that it is tailored only to React, and there exist no issues using it in our react project.
4. It is straightforward to implement.

## **Getting Started**

To get started with the practical visualization of React Icons, all we need to do is add a node package and import the component we want. To do that, we need to install the package using npm through the below-given command.

### **Installation (for standard projects)**

1. $ npm i react-icons--save    // NPM
3. $ yarn add react-icons  // Yarn

### **Usage**

1. **import** { FaBeer } from 'react-icons/fa';
2. **class** Question **extends** React.Component {
3. render() {
4. **return** <h3> Lets go **for** a <FaBeer />? </h3>
5. }
6. }

### **Installation(for meteorjs, gatsbyjs, etc)**

1. $ npm install @react-icons/all-files --save

### **Usage**

1. **import** { FaBeer } from "@react-icons/all-files/fa/FaBeer";
2. **class** Question **extends** React.Component {
3. render() {
4. **return** <h3> Lets go **for** a <FaBeer />? </h3>
5. }
6. }

### **Importing React Icons**

Importing React icons is very simple. It follows this rule or syntax.

1. **import** { IconName } from "react-icons/<library-initials>";

For instance, consider that we are importing more than one icon from the library. We can do that in just one line of code using the ES6 destructing feature. See the code below for reference.

1. **import** { IconName1, IconName2, ..., IconNameN } from "react-
2. icons/<library-initials>";

Now, to take this forward, add the below code in the App.js file and import all the necessary icons. The method of importing multiple icons is shown below.

1. **import** { AiFillTwitterCircle } from "react-icons/ai";
2. **import** { DiGithubBadge } from "react-icons/di";
3. **import** { FaCodepen } from "react-icons/fa";
4. **import** { IoLogoLinkedin } from "react-icons/io";

### **Entering icon tags**

After importing the icons from React Icons in the App.js file, we can use it anywhere in the file, like any HTML tag we know. Consider the below code, for instance, where we would be working only in the nav section within the div tag of the navbar-header. Enter two icons before the h1 tag and two after it, as shown below.

1. <div className="navbar-header">
2. <AiFillTwitterCircle />
3. <FaCodepen />
5. <h1>Navigation</h1>
7. <IoLogoLinkedin />
8. <DiGithubBadge />
9. </div>

The icons would appear something like this.



Just like this, we can add multiple icons across different sections of our application. It is simple as that.

### **Icon Styling**

There are usually various ways to style the icons. In this case, we would be using only two ways to style React icons to keep them neat and easy to understand.

1. React Context API.
2. Styled component.

**Using React Context API**

We can either choose to style an icon or more than one icon in React. We need to wrap up all the icons in the tag and style the desired icon residing in the tag.

To understand how it is done, import the Context API and add it to the top of the file, as shown below.

1. **import** { IconContext } from "react-icons";
2. Styling more than one Let's wrap the first 2 icons and assign a
3. className to it like so:
4. <div className="navbar-header">
5. <IconContext.Provider value={{ className: "top-react-icons" }}>
6. <AiFillTwitterCircle />
7. <FaCodepen />
8. </IconContext.Provider>
9. <h1>Navigation</h1>
10. <IoLogoLinkedin />
11. <DiGithubBadge />
12. </div>

Let's style the added react icons in the App.js file as shown below.

1. .top-react-icons{
2. font-size: 5rem;
3. }

The above CSS code would increase the font size. To visualize this, open the browser and the results should look something like this.



Let's move ahead by styling individual elements. To do that, first, change the color of each icon present on the other side. The overall code snippet is given below to help us out.

1. <div className="navbar-header">
2. {/\* styling multiple icons \*/}
3. <IconContext.Provider value={{ className: "top-react-icons" }}>
5. <AiFillTwitterCircle />
6. <FaCodepen />
7. </IconContext.Provider>
8. <h1>Navigation</h1>
10. {/\* styling individual icons \*/}
11. <IconContext.Provider value={{ color: "blue" }}>
12. <IoLogoLinkedin />
13. </IconContext.Provider>
14. <IconContext.Provider value={{ color: "green" }}>
15. <DiGithubBadge />
16. </IconContext.Provider>
17. </div>

### **Styled Components in React Icons**

Assuming that we are familiar with Styled components, let's install this component through the command line as shown below.

1. $ npm install --save styled-components --save

**Importing Styled Components**

1. **import** styled from from 'styled-components

**Usage**

1. <div className="jumbotron">
2. <BsFillAlarmFill />
3. <h1>Header</h1>
4. <BsFillArchiveFill />
5. </div>

Now, move down to the last line in App.js and add the styles as shown below.

1. **const** Archive  = styled(BsFillArchiveFill)`
2. color: purple;
3. transform: scale(2);
4. margin: 5%;
5. `;

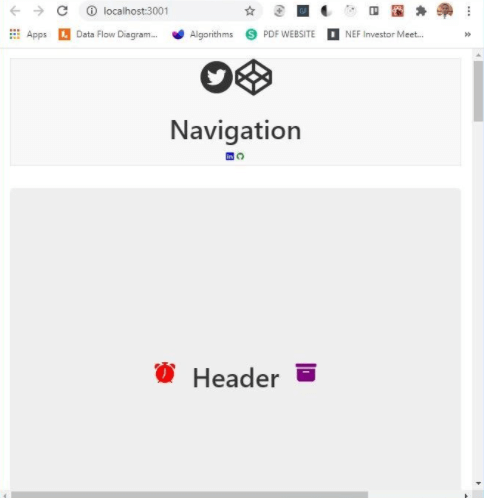
Adding another styled component like an **alarm,** as given below.

1. **const** Alarm = styled(BsFillAlarmFill)`
2. color: red;
3. transform: scale(2);
5. margin: 5%;
6. `;

To apply these styled components to the icons used in the App.js file, use the below method as shown.

1. <div className="jumbotron">
2. <Alarm />
3. <h1>Header</h1>
4. <Archive />
5. </div>

It is important to note that all we did was rename the components to the styled-components and save the changes. It is simple as that. The final output might look something like this.



This was one of the examples to make us familiar with React Icons and how different components of React Icons can be used for adding different types of icons on a website or web application.

Let's look at some of the advantages and disadvantages of using React Icons.

### **Advantages**

The advantages are as follows:

1. React Icons have the feature of importing only required icons one at a time or more than one depending on the requirement.
2. Icons in React are easy to use. With React Icons, one library serves all the icons in one place. Other icon generators or directories are required to add or install dependencies on the existing projects, but it is not the case with React Icons.
3. React is easy to learn for JavaScript developers, and with the feature of React Icons, they need not write any extra CSS for generating icons on the application they are working on.
4. CSS needs precision, but with React Icons, we don't need to care much about CSS. Although size and color can be easily modified, it is relatively easy than React Icons.

### **Disadvantages**

1. React Icons need multiple imports. Even for one or more than one icon that needs to be imported in the App.js file, the number of imports is directly proportional to the number of icons.
2. Apart from imports, React Icons are confined only to React. Although they can be used without libraries and frameworks, they suit React best, which can be another disadvantage for other frameworks.

### **Conclusion**

Icons are an integral part that needs no explanations at all. They play a crucial role in separating software's UI and make them unique in their class. It also appeals to the users about some of the important features that users see and how they use them. Many icons remain disposed in React Icons, and users can use it anytime with just simple import.

Beyond this, it is interesting to note that we can get all the required icons for our react project in just a single package. Another reason that makes React Icons unique is that they are very easy to use. In this tutorial, we saw how different types of other components apart from React Icons could be used to design and develop our project. We also came across how to get started with React icons and install them for standard projects and other frameworks. We later learned about the advantages and disadvantages of React Icons. React is easy to learn and implement, provided one needs to have good command over JavaScript and add a bit more. The react Icons also help the ideas get clear.

Next Topic[React Date Picker](https://www.javatpoint.com/react-date-picker)