How to use SVGs in React

January 29, 2020 · 6 min read

What is an SVG?

SVG is a vector graphics image format based on XML. SVG stands for Scalable Vector Graphics. It was developed in the late 1990s and was poorly supported until around 2016. Today a huge percentage of icon libraries such as Flaticon, Font Awesome, Material Icon, etc., have full support for SVG. Brands such as Twitter, Youtube, Udacity, Netflix, etc., use SVG for some of their images and icons.

Why SVG over other image formats?

There are other image formats that we are more familiar with such as JPEGs, GIFs, PNGs, etc. We will take a look at some advantages of using SVG over other image formats.

- **Scalability and resolution**: This is the utmost advantage of SVG over others, SVG uses shapes, numbers, and coordinates instead of pixel grids like other image formats do. This makes it possible to zoom in and out of SVG images without losing any quality and gives SVG the ability to scale infinitely
- **Minimal file size**: The file sizes of SVG images are usually small compared to other file formats and they are easily compressible, giving it a possibility to be even smaller
- **High performance and speed**: Because of the small size nature of SVG images, it becomes very easy and fast for browsers to render them. It is

not have to make a request to get the image and renders it just like every other code in your file. In this case, no request was made. But in a situation where you have a complex image SVG file such as the Monalisa photo, I would suggest using, PNGs or JPEGs as the load time and performance for SVGs fall drastically

- **DOM-like and style-able**: SVG images are like code, so this means it can be navigated like a DOM element and also styled, although some properties have different names eg(you might want to use fill instead of color). It can also be styled with CSS
- **Editable**: Since SVGs are DOM-like, SVG's can be created and edited and animated with any text editor
- **Integration**: SVGs can be used in various ways, it can be used to display logo images and icons. It can be used as graphs, animations, and effects
- **Animatable**: SVGs can be animated. This can be done with tools like Web Animation API's, WebGL, CSS animations, etc. For animating SVG with CSS, check out this detailed article by Hope Armstrong
- **Accessibility and SEO**: SVGs contain text which improves the accessibility of a website and it also means they can be searched, indexed, scripted, etc.

How to use SVGs in React

Below we will go through various ways we can use or render this React SVG logo on a webpage.

Using image tag for static SVGs

In order to be able to use SVGs or any other image format in the img `<img src={} />` we have to set up a file loader system in whichever module bundler we are using(Webpack, Parcel, etc). Here I will show you how to set it up in a few steps if you are already using Webpack.

First, we install the file-loader library `$ npm install file-loader --save-dev` , this will install it as a dev-dependency.

```
const webpack = require('webpack');

module.exports = {
  entry: './src/index.js',
  module: {
    rules: [
      //...
      {
        test: /\.(png|jp(e*)g|svg|gif)$/,
        use: [
          {
            loader: 'file-loader',
            options: {
              name: 'images/[hash]-[name].[ext]',
            },
          },
        ],
      },
    ],
  },
```

Now you can import your SVG and use it as a variable, like this:

```
import React from 'react';
{/*images*/}
import ReactLogo from './logo.svg';

const App = () => {
  return (
    <div className="App">
      <img src={ReactLogo} alt="React Logo" />
    </div>
  );
}
export default App;
```

With the same Webpack settings above we can use the SVG tag, basically copying and pasting the content of the `.svg` file into your code. Here is a sample use case:

```
import React from 'react';

const App = () => {
  return (
    <div className="App">
      <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 841.9 595.3">
        <g fill="#61DAFB">
          <path d="M666.3 296.5c0-32.5-40.7-63.3-103.1-82.4 14.4-63.6 8-
114.2-20.2-130.4-6.5-3.8-14.1-5.6-22.4-5.6v22.3c4.6 0 8.3.9 11.4 2.6 13.6
7.8 19.5 37.5 14.9 75.7-1.1 9.4-2.9 19.3-5.1 29.4-19.6-4.8-41-8.5-63.5-
10.9-13.5-18.5-27.5-35.3-41.6-50 32.6-30.3 63.2-46.9 84-46.9V78c-27.5 0-
63.5 19.6-99.9 53.6-36.4-33.8-72.4-53.2-99.9-53.2v22.3c20.7 0 51.4 16.5 84
46.6-14 14.7-28 31.4-41.3 49.9-22.6 2.4-44 6.1-63.6 11-2.3-10-4-19.7-5.2-
29-4.7-38.2 1.1-67.9 14.6-75.8 3-1.8 6.9-2.6 11.5-2.6V78.5c-8.4 0-16 1.8-
22.6 5.6-28.1 16.2-34.4 66.7-19.9 130.1-62.2 19.2-102.7 49.9-102.7 82.3 0
32.5 40.7 63.3 103.1 82.4-14.4 63.6-8 114.2 20.2 130.4 6.5 3.8 14.1 5.6
22.5 5.6 27.5 0 63.5-19.6 99.9-53.6 36.4 33.8 72.4 53.2 99.9 53.2 8.4 0
16-1.8 22.6-5.6 28.1-16.2 34.4-66.7 19.9-130.1 62-19.1 102.5-49.9 102.5-
82.3zm-130.2-66.7c-3.7 12.9-8.3 26.2-13.5 39.5-4.1-8-8.4-16-13.1-24-4.6-8-
9.5-15.8-14.4-23.4 14.2 2.1 27.9 4.7 41 7.9zm-45.8 106.5c-7.8 13.5-15.8
```

With this, we can already see the disadvantage of using this method, when the image is more complex the SVG file becomes larger and since SVG is stored in text, that means we have a whole bunch of text in our code.

Using SVG as a component

SVGs can be imported and used directly as a React component in your React code. The image is not loaded as a separate file, instead, it's rendered along the HTML. A sample use-case would look like this:

```
import React from 'react';
import {ReactComponent as ReactLogo} from './logo.svg';

const App = () => {
  return (
    <div className="App">
      <ReactLogo />
    </div>
  );
}
export default App;
```

Note this approach only works with create-react-app, If you are not using create-react-app, I would recommend using other approaches. Create-react-app uses SVGR under the hood to make it possible to transform and import SVG as a React component.

Using SVGR

SVGR is an awesome tool that converts your SVG into React components that you can use. So how do we set it up?

First, we install the package `$ npm install @svgr/webpack --save-dev` .

Then we update our Webpack configuration rule to use SVGR for SVGs:

```
const webpack = require('webpack');

module.exports = {
  entry: './src/index.js',
  module: {
    rules: [
      //...
      {
        test: /\.svg$/,
        use: ['@svgr/webpack'],
      },
    ],
  },
  //...
};
```

Now we can import our SVG images as a React component and use it in our code like this:

```
import React from 'react';
import ReactLogo from './logo.svg';

const App = () => {
  return (
    <div className="App">
      <ReactLogo />
    </div>
  );
}
export default App;
```

Using SVG as a data-url

Data-urls are URLs prefixed with the `data:` scheme that allow content creators to embed small files inline in documents. You can read more about it

So how do we achieve this? First, we need an appropriate Webpack loader in our bundler, in the use case, I will be using svg-url-loader. We can grab it into our project by running this command `$ npm install svg-url-loader --save-dev` .Then we will update the Webpack configuration file rules section with these rules:

```
const webpack = require('webpack');

module.exports = {
  entry: './src/index.js',
  module: {
    rules: [
      //...
      {
        test: /\.svg$/,
        use: [
          {
            loader: 'svg-url-loader',
            options: {
              limit: 10000,
            },
          },
        ],
      },
    ],
  },
```

Now we can import our SVG file and use it in our React component like this:

```
import ReactLogo from './logo.svg';

const App = () => {
  return (
    <div className="App">
      <img src={ReactLogo} alt="React Logo" />
    </div>
  );
}

<img src="ebbc8779488b4073081931bd519478e8.svg" alt="" />
```

This usually complies into something like this in the DOM

```
<img src="a34r2dad4234dad7824sa4284r0s2345.svg" alt="React Logo" />
```

See how SVGs impact your users with LogRocket

SVGs help with performance and add a great level of animation, interactivty, and more, but sometimes have unintended concequences for end-users. If you're interested in monitoring how your SVGs render in your React apps, tracking Redux state, automatically surfacing JavaScript errors, and tracking slow network requests and component load time, try LogRocket.

LogRocket is like a DVR for web apps, recording literally everything that happens on your React app. Instead of guessing why problems happen, you can aggregate and report on what state your application was in when an issue occurred. LogRocket also monitors your app's performance, reporting with metrics like client CPU load, client memory usage, and more.

The LogRocket Redux middleware package adds an extra layer of visibility into your user sessions. LogRocket logs all actions and state from your Redux

Things to note

- **Complicated images**: The more complex the image the larger the SVG file gets like we saw while trying to use the SVG tag. Here I will recommend you go with PNG or JPEG
- **Backwards support on the web**: SVG doesn't have backwards browser support, which means that not all older versions of browsers support SVG, so SVG might not display correctly in those browsers. This is due to the fact that initially when SVG was first introduced most browsers did not support SVG. If you are targeting older versions of browsers SVG might not be your go-to image format

Conclusion

SVGs are really useful for building fast, performant, and accessible webpages. Knowing when and when not to use them will serve you even better. I recommend you use SVGs for low-intensity images like logos, icons, and less-complex images.

Nedy Udombat Follow

Software engineer.

#react #svg

Looks like a lot of editing of Webpack. What's the best way to use SVG's without ejecting?

Iris Says:
April 29, 2020 at 3:53 am

Reply

Great article! help me a lot, thanks!

Rafael Says:
May 10, 2020 at 6:11 pm

Reply

I tried this, but the SVG doesn't show when using an image tag. I don't know what I am missing. I installed file-loader, too.

Valentin Says:
June 3, 2020 at 8:37 am

Reply

Great article! One usage not mentioned above, using as inline background image:
.elem {
background-image: url('data:image/svg+xml, ...');
}

Amir Says:
June 10, 2020 at 3:27 pm

Reply

I second, this is a great article!
Would also like to add the the object tag can be used (...)

But, more important – this article does not relate the fact that some of the methods listed here inline the SVG into the DOM – which allows developers to use CSS and JS to manipulate the SVG which is amazing !

You can use something like react-app-rewired to provide access to a webpack config file without ejecting.

Alfonsus Says:
July 9, 2020 at 7:40 am

Reply

"export 'ReactComponent' (imported as 'ReactLogo') was not found in '../static/images/icon/svg/feather-white.svg'
i go this error messages after using ReactComponent step, what should i do?

Lewisvrobinson Says:
November 16, 2020 at 8:10 pm

Reply

If you're using CRA just import it as a component like in the example. All the SVGR business is handled for you already

Mcarbonera Says:
November 17, 2020 at 4:29 pm

Reply

"Then we update our Webpack configuration rule to use SVGR for SVGs". What file is that? With witch name I need to create it?

Mel Says:
April 9, 2021 at 1:04 pm

Reply

" With this, we can already see the disadvantage of using this method, when the image is more complex the SVG file becomes larger and since SVG is stored in text, that means we have a whole bunch of text in our code."

What are the disadvantages? Is it solely the amount of text in the code? Or is there another disadvantage?

Thanks for sharing mate! Using SVG as a component worked like dream for me. One cool thing is that it creates an inline SVG which to which we can pass props such as className and control its presentation that way.

Leave a Reply

Enter your comment here...