Create Vue Icon Package With Rollup

From the developer, by the developer, for the developer.

Press Space for next page →

Prerequisites

This talks is about a step by step guid for creating a Vue2 icon package (module) with Rollup. Finally is explained the basis for running, installing and then publishing locally or publicly.

- 👨‍💻 Basic knowledge of Javascript and Vue2 is required
- 💾 Basic knowledge of Vue SFC is required
- 👷 Knowledge of Rollup is good, but not needed
- 🛠️ You must be familiar and comfortable with the command line

You can read more about publishing simple package to npm here:
Simply Publish Your Package to npm

Read more about Vue SFC?

About Rollup

I found in rollup a great tool for creating npm modules. Is particularly easy to understand, no need so much configuration but it’s open for add more extras depending of your needs.

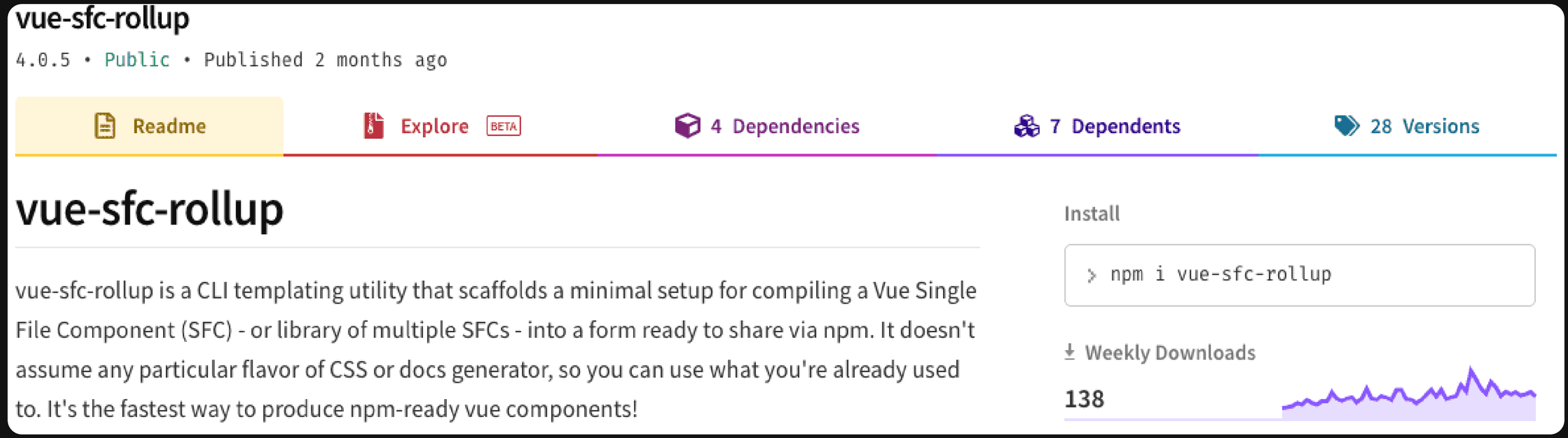
Rollup is a module bundler for JavaScript which compiles small pieces of code into something larger and more complex, such as a library or application. It can output CommonJs, EJS, and UMD.

We'll talk about it later.

Read more about rollup

About Vue SFC Rollup

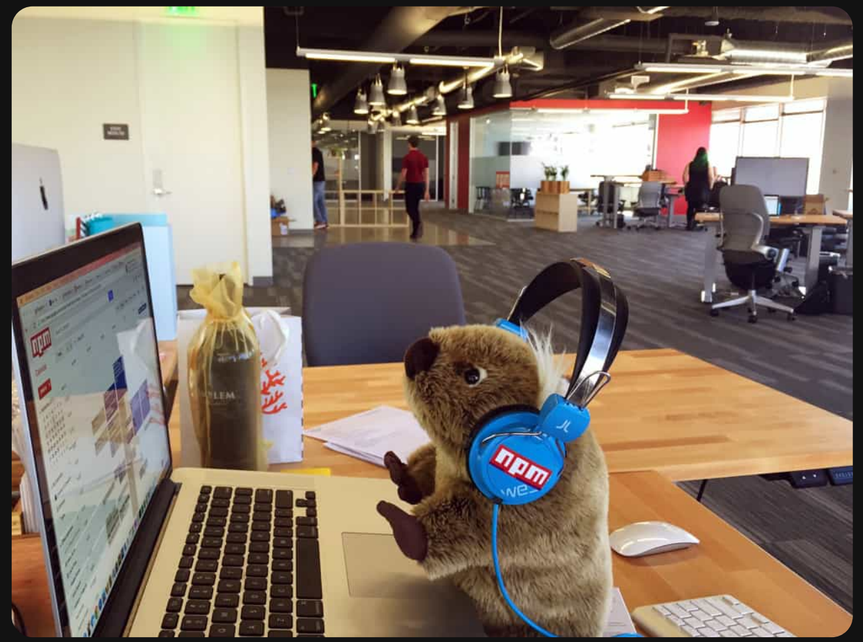
It’s the fastest way to produce npm-ready vue components! It shipped with a minimal rollup config and development environment. This package is a very useful utility when starting a component library.



Read more about vue-sfc-rollup

Let's begin with imagination.

LET'S ASSUME WE ARE WORKING ON 3 DIFFERENT PROJECTS USING ONE BEAUTIFUL DESIGN SYSTEM THAT OUR DESIGNER ALREADY CREATED FOR US. WE ARE GOING TO CREATE A SPA FOR 3 DIFFERENT PROJECTS. THE BUNDLE SIZE IS A CONSTRAINT. WE CANNOT HAVE TOO LARGE A BUNDLE SIZE SINCE IT DIRECTLY AFFECTS THE PERFORMANCE OF OUR APP. THEN WE NEED TO MAKE SURE EVERY APP IS CONSISTENT ESPECIALLY THE ICON. SO, LET'S LOOK AT HOW WE CAN CREATE A SINGLE CODEBASE THAT SHARED ACROSS 3 DIFFERENT PROJECT



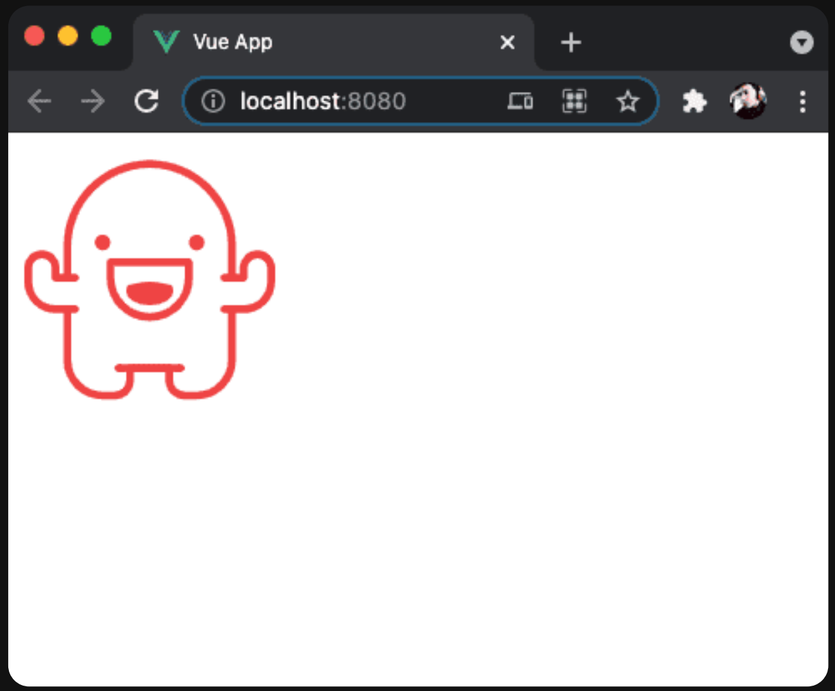
Project Overview

We want to create a simple component wrapper for our svg icon, then publish it locally with package file. They will make it easy to maintain a consistent look and feel across an application.

`npm i icon-pack-0.0.1.tgz`

```
<script lang="ts">
import Vue from 'vue';
import { OurIcon } from '@/entry.esm';
export default Vue.extend({
  name: 'ServeDev',
  components: {
    OurIcon,
  },
});
</script>

<template>
  <div id="app">
    <our-icon icon="monster" style="color:#ef4444; font-size:10em" />
  </div>
</template>
```



We will use free icons made by Darius Dan from www.flaticon.com

1.0 You'll want to install the vue-sfc-rollup globally

`npm install -g vue-sfc-rollup`

Once that is installed globally…

`cd develop`

run the following command to initialize the project.

`sfc-init`, see options on the right

`cd path/to/my-component-or-lib`

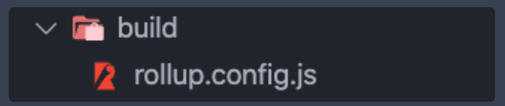
`npm install`

Once that is done, you can open the folder up in your editor of choice.

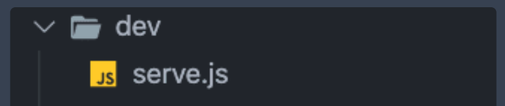
Choose the following options within the prompts:

- **Which version of Vue are you writing for?** Vue 2
- **Is this a single component or a library?** Library
- **What is the npm name of your library?** this will need to be unique on npm. I used my-icon-pack)
- **Will this library be written in JavaScript or TypeScript?** JavaScript (feel free to use TypeScript if you know what you're doing)
- **Enter a location to save the library files:** This is the folder name you want your library to have. It will default to the npm name you gave it above so you can just hit enter.

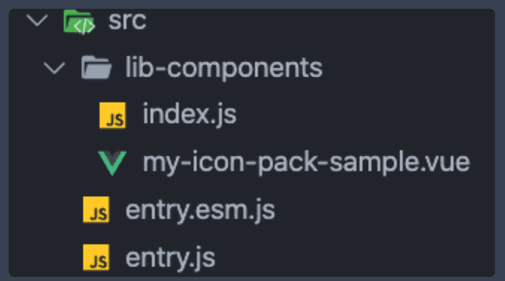
Folder Structure



A minimal rollup config from vue-sfc. It will output esm, cjs and iife format for our library



a sample usage file which can be used to load and test your component/library during development



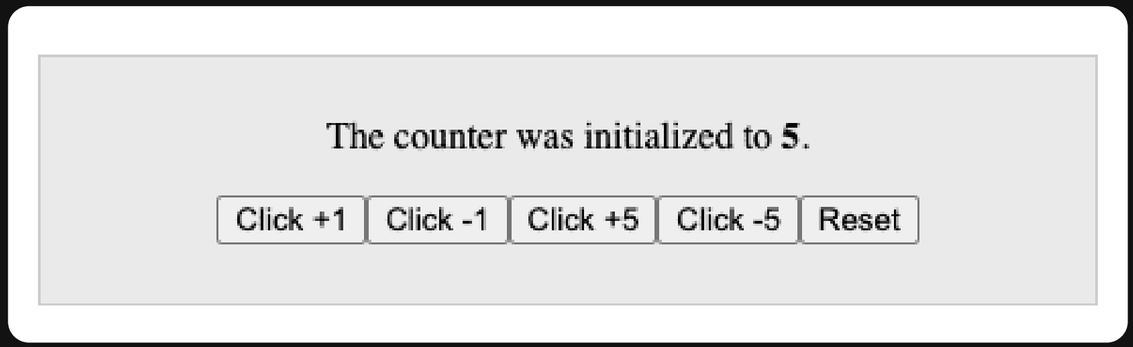
This is the place where our component library is present.

Babel config and generated package.json from vue-sfc

There is a sample Vue component built for us. You can find it inside of the `/src/lib-components` folder. To see what this component looks like, you can run `npm run serve` and navigate to http://localhost:8080/

Don't forget to `npm install` first!

Output:



ESM vs CJS vs IIFE

ESM

Keep the bundle as an ES module file. Suitable for other bundlers and inclusion as a tag in modern browsers (alias: esm, module).

- It can only be placed at the top of a file
- The export directive, on the other hand, can be used to explicitly make items public

CJS

Suitable for Node and other bundlers (alias: commonjs).

- It is widely used on the server side
- Were designed with server development in mind
- Cannot be used on the browser side unless it is packaged with a transpiler
- Can be recognized by the use of the require() function and module.exports

IIFE

A self-executing function suitable for inclusion in a script tag. If you want to create a bundle for your application, you probably want to use this.

- We can use this format to create a bundle for an application
- It helps us to put things into namespaces to avoid variable collisions and keep code private.

2.0 Let’s start with creating icon wrapper component

```
<template>
  <component :is="iconComponent" role="img" />
</template>
```

```
props: {
  icon: {
    type: String,
    required: true,
  },
  hasFill: {
    type: Boolean,
    default: true,
  },
  growByHeight: {
    type: Boolean,
    default: true,
  },
},
```

- Icon string prop to pass the .svg filename to import
- hasFill boolean prop which tells the component if fill property will be used to change the color of the svg element, default is false i.e. no fill
- growByHeight boolean prop to determine to use height or width to scale relative to the font-size, default is true i.e. use height

Add fill: currentColor so it will match text color

```
<style scoped>
.svg-class {
  display: inline-flex;
  fill: currentColor;
}
</style>
```

2.1 Add computed property to get the icon

```
computed: {
  iconComponent() {
    let component;
    if (process.env.NODE_ENV === 'production') {
      component = () => import(`../src/assets/icons/${this.icon}`);
    } else {
      component = () => import(`@/assets/icons/${this.icon}`);
    }
    return component;
  },
},
```

If in production mode we use '../' I still don't know why, but it works!

Using lazy import to import icon component

2.2 Add Function to properly convert svg on update lifecycle

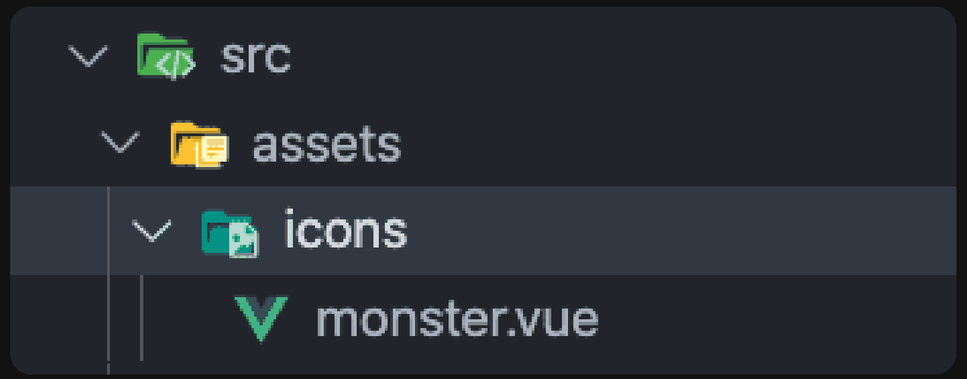
```
updated() {
    if (this.$el.firstElementChild && this.$el.nodeName === 'svg') {
      const svgElement = this.$el;
      // use `viewBox` attribute to get the svg's inherent width and height
      const viewBox = svgElement.getAttribute('viewBox').split(' ').map((n) => Number(n));
      const widthToHeight = (viewBox[2] / viewBox[3]).toFixed(2);
      if (this.hasFill) {
        // recursively remove all fill attribute of element and its nested children
        recursivelyRemoveFill(svgElement);
      }
      // set width and height relative to font size
      // if growByHeight is true, height set to 1em else width set to 1em
      // and remaining is calculated based on widthToHeight ratio
      if (this.growByHeight) {
        svgElement.setAttribute('height', '1em');
        svgElement.setAttribute('width', `${widthToHeight}em`);
      } else {
        svgElement.setAttribute('width', '1em');
        svgElement.setAttribute('height', `${1 / widthToHeight}em`);
      }
      svgElement.classList.add('svg-class');
    }
},
```

2.3 Add recursivelyRemoveFill function

```
function recursivelyRemoveFill(el) {
  if (!el) return;
  el.removeAttribute('fill');
  [].forEach.call(el.children, (child) => {
    recursivelyRemoveFill(child);
  });
}
```

We are trying to clear fill attribute from our svg

Convert our svg to vue components, then place our svg icons to folder bellow



I will give you script to automate that task in bonus 🥳

3.0 Test our component

Add our component in index.js so we can import

```
// index.js
export { default as OurIcon } from './our-icon.vue';
```

We can use it globally

```
// dev/server.js
import MyIconPack from '@/entry.esm';
Vue.use(MyIconPack);
```

OR

We can use it local in our component

```
// dev/server.js
import { OurIcon } from '@/entry.esm';
export default Vue.extend({
  name: 'ServeDev',
  components: {
    OurIcon,
  },
});
```

Usage in our component

```
<template>
  <div id="app">
    <our-icon icon="monster" style="color:#ef4444; font-size:10em" />
  </div>
</template>
```

- Run `npm run serve`
- Open in http://localhost:8080/
- See the result

Let's  it...

4.0 Publish it Locally

If we want to just use our package independently without publish it on npm so we can use `npm pack`.

It will convert our package to ~.tgz so we can share it later to our team

npm-pack

NPM CLI

For anything that's installable (that is, a package folder, tarball, tarball url, name@tag, name@version, name, or scoped name), this command will fetch it to the cache, and then copy the tarball to the current working directory as -.tgz, and then write the filenames out to stdout.

```
npm pack [[<@scope>/]<pkg>...]
```

Read more about npm-pack

1. First run command `npm run build`
2. Then run command `npm pack`

Result

Usage

`npm i my-icon-pack-1.0.0.tgz`

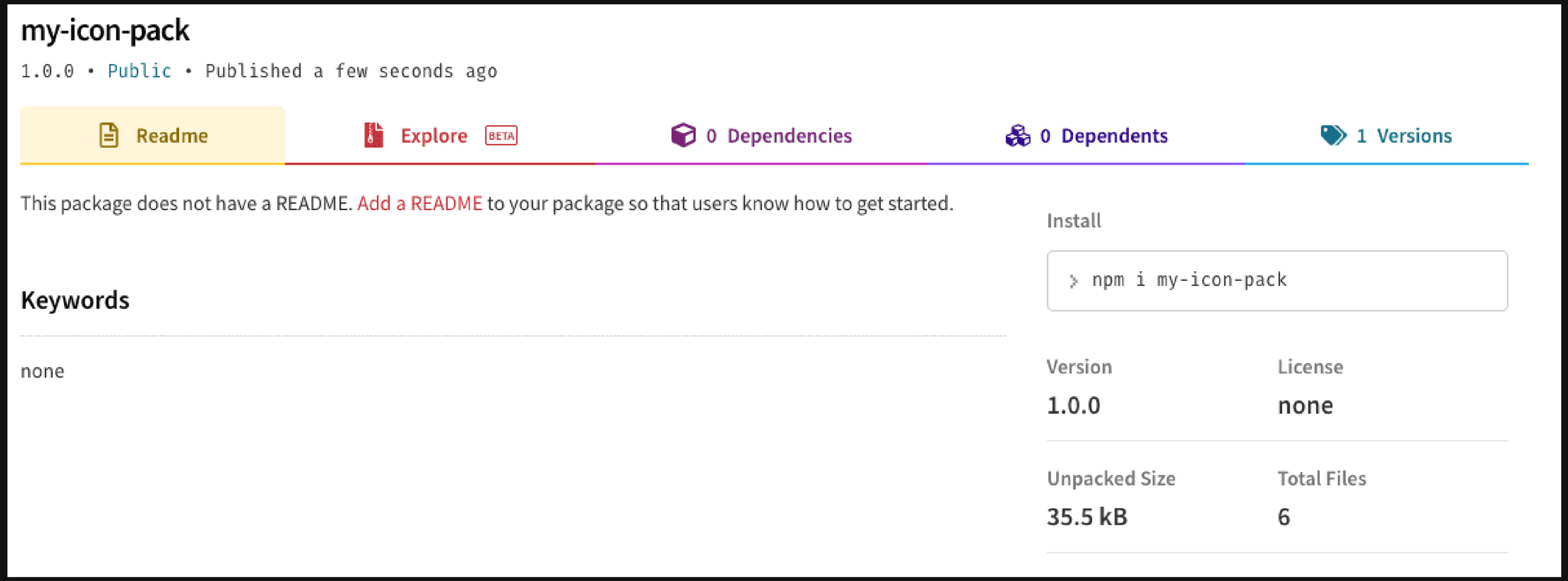
4.1 Publish publicly

`npm publish`

```
➜  my-icon-pack npm publish
npm notice
npm notice 📦  my-icon-pack@1.0.0
npm notice === Tarball Contents ===
npm notice 11.4kB dist/my-icon-pack.esm.js
npm notice 5.8kB  dist/my-icon-pack.min.js
npm notice 13.1kB dist/my-icon-pack.ssr.js
npm notice 1.4kB  package.json
npm notice 1.7kB  src/lib-components/my-icon-pack-sample.vue
npm notice 2.0kB  src/lib-components/our-icon.vue
npm notice === Tarball Details ===
npm notice name:          my-icon-pack
npm notice version:       1.0.0
npm notice filename:      my-icon-pack-1.0.0.tgz
npm notice package size:  8.1 kB
npm notice unpacked size: 35.5 kB
npm notice shasum:        ac278010370742a742cf1277567d6e63eca928f8
npm notice integrity:     sha512-9wv7R/mkSsYMB[...]8oiTFj4j1FJkg==
npm notice total files:   6
npm notice
This operation requires a one-time password.
Enter OTP: 031 462
+ my-icon-pack@1.0.0
➜  my-icon-pack
```

5.0 Check on npm registry

You can check on npm



https://www.npmjs.com/package/my-icon-pack

There is more 

Tips, you can automate convert svg to vue component automatically

```
const fs = require('fs');
const path = require('path');

fs.readdir('./svg/', function (err, files) {
  if (err) {
    return console.log('Unable to scan directory: ' + err);
  }
  files.forEach(async function (file) {
    const name1 = path.basename(file);
    const ext1 = path.extname(file);
    const nameWithoutExt1 = path.basename(name1, ext1);
    if (path.extname(file) === '.svg') {
      const svg = fs.readFileSync(`./svg/${file}`, 'utf8');
      fs.writeFile(
        `./icons/${nameWithoutExt1}.vue`,
        `<template>${svg}</template>`,
        function (err) {
          if (err) return console.log(err);
          console.log(`${file} > ${nameWithoutExt1}.vue`);
        }
      );
    }
  });
});
```

Result

```
➜  assets node compile.js
whatsapp.svg > whatsapp.vue
monster.svg > monster.vue
➜  assets
```

Thanks, Any feedback would be appreciated

Lucky DS
Fullstack Javascript Engineer
085156501865

TECH STACK FOR THIS PRESENTATION