better-docs
in JavaScript

Preview your Vue or React components using JSDoc (playground included)

Wojciech Krysiak | Follow
Sep 17 · 6 min read ★

Have you ever thought about documenting your Vue or React application? Most probably you haven't. If that is the case — I would like to convince you that documenting your Vue/React application can be useful and fun.

How I am going to achieve that?

First, I will try to define what elements a perfect documentation should have and point out the benefits.

Next, there will be a **fun part**. We will dive into an example, where, in literally 5 minutes, we will create and publish documentation for a Vue app.

!!!Spoiler alert. This is how it will look:

https://softwarebrothers.github.io/example-design-system/ProgressBar.html

We will use JSDoc along with better-docs (open-source extension created by my company).

Intrigued? So let's start!

Perfect documentation

Let's think — what would you like to have in a perfect frontend documentation? I would say that:

- **components interface**: props, slots, events, methods,
- **components description** — what is the purpose of each component, where it can be used,
- **preview** of components in **different states**,
- **playground** for modifying components,
- **interfaces** of all the supporting classes and objects (like client libraries).

All of the above can be described as a **Pattern Library**.

Anything else?

We can also add a list of colours, sizes, fonts etc. used in the application. These things form the **Style Guide**.

Do we need anything more?

Designers would also add design purpose, brand values and promise etc.

Having all of that we would create something called a **Design System** — sounds serious.

What would we gain by having a documented frontend app?

This is the list of things coming to my mind:

- We will have one place where all the people, no matter their background, can discuss the frontend: programmers, designers, clients.
- New people coming to the team will have an easier start with one source of truth (your documentation).
- Having components in different states will simplify testing them in different use cases.
- Playground, where you can try out a component with your data, is extremely useful for developers who will use the components you've built.

Now we know what a perfect documentation is and why you should invest 5 minutes of your time to add this to your project. So let's do this now!

An example

In the example, we will set up the documentation for Vue app created with vue-cli.

Set up the starter Vue app.

(Assuming that you have vue-cli installed) create a new Vue app:

```
vue create my-documented-app
cd my-documented-app
```

Next, we have to install dependencies: jsdoc with better-docs and also parcel-bundler — because it is a better-docs dependency:

```
yarn add --dev jsdoc better-docs
yarn global add parcel-bundler
```

Configure JSDoc

JSDoc can be configured with a config file. Copy this file and paste it to your project:

Here we define that we use 2 plugins from better-docs (component and category) and its theme. Furthermore, we tell JSDoc to parse files in `./src` directory and put the created documentation to `./docs` folder.

We can now generate the documentation like this: `node_modules/.bin/jsdoc -c 'jsdoc.json'`, but to simplify things let's create a new command in our `package.json` file, below the `lint` script:

```
"docs": "jsdoc -c 'jsdoc.json'"
```

now you can see the documentation by opening `./docs/index.html` file.

It has only a readme file, so let's add the first component.

Document component

I've already prepared a component called ProgressBar. Copy it to the app to `./src/components/ProgressBar.vue`

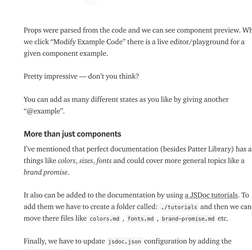
Please pay attention to the JSDoc tags I added above the component object: '@component' and '@example'. The first one marks the object as a component and the second one defines an example (use case) of the component usage.

Now regenerate the documentation:

```
yarn docs
```

And revisit the documentation page.

Now you can see a new *components* section in the sidebar and a newly created component:



Props were parsed from the code and we can see component preview. When we click "Modify Example Code" there is a live editor/playground for a given component example.

Pretty impressive — don't you think?

You can add as many different states as you like by giving another "@example".

More than just components

I've mentioned that perfect documentation (besides Patter Library) has also things like *colors*, *sizes*, *fonts* and could cover more general topics like a *brand promise*.

It also can be added to the documentation by using a JSDoc tutorials. To add them we have to create a folder called: `./tutorials` and then we can move there files like `colors.md` , `fonts.md` , `brand-promise.md` etc.

Finally, we have to update `jsdoc.json` configuration by adding the `opts.tutorials` property:

```
...
opts: {
  ...
  tutorials: "./tutorials"
}
...
```

and rerun the documentation:

```
yarn docs
```

You should now see all the tutorials on the sidebar menu.

Summary

Those were just the basics. JSDoc is a very powerful tool and with better-docs you can use it with your frontend apps — no matter if you use React or Vue, it works the same.

I genuinely hope that some of you are now thinking about setting up documentation in your current project.

If you do — don't forget to star the repo on GitFlub and raise an issue when you think that something could be improved.

JavaScript | Documentation | Vuejs | Vue | React

WRITTEN BY
Wojciech Krysiak
Follow