Creating my personal website with NuxtJS

#javascript #vue #nuxt

Jimmy Kasprzak Sep 1 · 6 min read

Welcome to my first blog post using the content module offered by NuxtJS! In this post I'll describe my first real experience with NuxtJS creating my personal website.

The project

First, the project. My project was something I had in mind for a long time, but never managed to find the time to do: a simple personal website to gain visibility on the web and show a bit of what I do. Thanks to the lockdown in France I got started with this project. I had a rough idea of what I wanted in this website: a summary of what I do, my work experiences and skills; the different ways to contact me and a blog section where I could start writing.

My personal website

Why Nuxt JS?

real-world experiences with Angular and React, so I wanted to try Vue.js on a real project which goes a bit beyond classical tutorials (being a big fan of learning by doing). At the same time, I knew I wasn't good enough with Vue.js to create a whole project structure from scratch (without regretting it later at least). And here comes NuxtJS and the very first advantage I saw in this framework. Starting a project with NuxtJS is very simple and it takes care of all the set up pain for you. It allowed me to focus on developing and creating content from the get-go. I also knew a bit about what NuxtJS opinions and features were so I knew I could make my project with it.

Feedback on?

Now that we know why NuxtJS, it's time to dive into what I used in NuxtJS. Keep in mind it's an opinionated list of some of the main stuff I used. NuxtJS is much richer than just that.

Dynamic routing

Do you remember PHP? Yes I just said PHP, please do not be afraid. NuxtJS has a routing system very similar to what PHP proposed: based on file names and directory structures.

Everything starts with the **pages** directory. Inside, you create your files which will be used as pages. By pages, I mean Vue.js components linked to a route in vue-router. But how to define these routes? Well you don't! When NuxtJS builds your project, it looks inside your pages directory and generates it for you, based on the files name.

```
pages/
--| index.vue
--| contact.vue
--| blog/
-----| index.vue
-----| _slug.vue
```

What's happening here ?

First we have an index.vue file. This page will be displayed when navigating on the root of the app. You guessed it, the contact page will be displayed on the /contact path.

Next we have the subdirectory blog with an index file. This index page behaves the same as the previous one. Il will be displayed on the root but this time of the current directory : /blog.

Lastly, we have a _slug file. The _slug expression represents a dynamic parameter. It means, whenever I'll navigate to /blog/something, I'll display the _slug page in which I'll have access to a slug parameter which value will be (in this example) something.

Layout

Layout allows you to easily configure the aspect of your app. It places itself one step higher

```
<Header />

<Nuxt />

<Footer />
```

The `<Nuxt />` is replaced by the code of the page you're in, depending on your app and current route. It's only one use case of NuxtJS layouts, you can easily find more use cases on the NuxtJS documentation, such as creating a specific layout according to the resolution (mobile/tablet/desktop) for example.

Static mode

One of my favorite feature so far! I knew my personal website - at least in a first version - did not need to request any APIs at runtime to display my content. So all my content could be generated at build time. But how could I take advantage of that? Well the NuxtJS static mode does the exact job! It goes through all your pages and generates all your content, at build time. It also means I could use a static hosting services (I went with **Netlify** by the way).

How to use the static mode? Update the nuxt.config.js file with the following property and that's it.

```
target: 'static'
```

Nuxt content

I wanted one specific feature : the ability to easily write and display blog posts. To do so, I tried one of the latest NuxtJS module : @nuxt/content.

To install this module, it is as easy as:

```
npm install @nuxt/content
```

and add the new module in nuxt.config.js:

```
{
   modules: [
      '@nuxt/content'
   ],
}
```

At this point, the @nuxt/content module is installed. What's next? Creating your content. All you have to do is create files in a **content** directory. @nuxt/content supports all kinds of formats (markdown, json, csv, xml). I personally went with markdown as it is a format I'm comfortable with. Another thing to note is that, in addition to your content, you can add metadata to your content file. In my case, I added a title, a description and an image

used the asyncData method available on all pages of a NuxtJS app. This method is called when navigating to the matching route and before initializing the component. In this

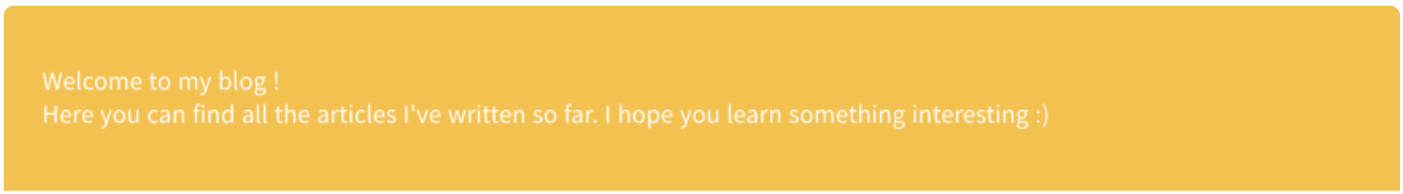
method, you have access to a context object. By adding the @nuxt/content module, this context object is enriched with a $content property which will help us interact with our content. To fetch a content, it looks like this:

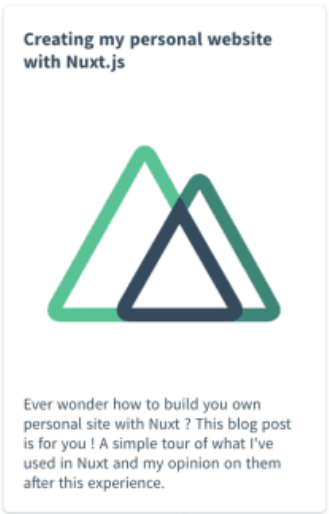
```
const article = await $content('articles', params.slug).fetch()

return { article }
```

The parameters define the path of the content you're looking for. If no parameters are defined, it defaults to the root of the content directory. In my case, I created an articles subdirectory to organize my content. So I'm asking for the content inside a subdirectory called articles and named as the slug parameter in my current url.

Here is what my blog introduction page looks like :





What I appreciated the most?

I appreciated a lot the simplicity of working with NuxtJS. No time lost on complex configuration or importing a number of libraries. You just get a concise, easy-to-read and working architecture and configuration from the get-go, allowing you to work on your ideas and your content. This also works very well thanks to the clarity of the documentation and the great articles you can find on the NuxtJS blog section. Big thumb up for that :)

One example is this article Creating a blog with @nuxt/content by **Debbie O'Brien**. It has

The biggest struggle I had?

The biggest struggle I faced on this project was about images loading. I decided to associate an image to every article I'll write. For the article you're currently reading, my first reflex was to add the image to the assets folder, like every other images of this site. But my image was never found. It turns out, the difference between this image and the other was that my image was loaded dynamically, according to the metadata of the article you're on.

Because of that, when Webpack bundles your app, it can not determine that this particular image is being used. Indeed, it never found an explicit use of it in your app at build time. So it removes it from your final bundle! One way to solve it was to put the image in the **static** folder instead of the **assets** folder. The content in this folder is always part of your final bundle, without build time analysis. I hope this explanation will save you some struggle time if you meet the same issue.

Conclusion

What to conclude? I learned a lot about NuxtJS in this project and I loved what I discovered. NuxtJS is really easy to get started with. It does not require a lot of experience, even with Vue.js and allows you to quickly focus on what matters: your ideas. This is something I'm looking for when working with a framework and NuxtJS greatly succeeded with it. Also, for the use case of my projects, the amount of features I needed (static mode, @nuxt/content, etc) and quickly available effortlessly was impressive, so I'm glad I chose NuxtJS.

I hope you learned some new stuff with this article and it gave you the desire to create your own personal website and to try out NuxtJS!

Please let me know your thoughts after this reading and in the meantime stay safe!

Posted on Aug 31 by:

Jimmy Kasprzak
@orodan

Consultant, trainer, web lover, raising pokemons when I'm not coding

Follow

PREVIEW SUBMIT

code of conduct - report abuse

Read Next

How to make your own (no template) personal website with React, Material UI, and Netlify

Sylvia Pap - Jun 27

React & Redux Project: Gomojii

Naya Willis - Jul 10

The new Keyword in JavaScript

Parwinder 👨🏻‍💻 - Jul 12

How I chose a programming language and beat bad habits

Kyle Martin - Jun 21

Trending on DEV 🔥


Load Balancer Tutorial 2020 - System Design Basics

#webdev #career #javascript #tutorial


You probably don't need Redux: Use React Context + useReducer hook

#react #redux #javascript #webdev


How do you manage clipboard history?

#webdev #discuss #productivity

Home Code of Conduct

4 2 6