**Documentation link:**

[**https://nuxt.com/docs/getting-started/introduction**](https://nuxt.com/docs/getting-started/introduction)

**Prerequisites**

• node - We recommend you have either 16.x or 14.x installed.

• A text editor, we recommend VS Code with the Volar extension.

• A terminal, we recommend using VS Code's integrated terminal.

**Command to create new project for nuxt 3**

**With NPM**

**npx create-nuxt-app@latest (installs nuxt 2)**

**With NPX**

**npx nuxi@latest init <project-name> (installs nuxt 3)**

then do **npm install**

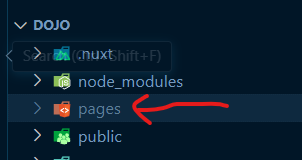
then to open development server type **npm run dev -- -o**

**UNDERSTANDING DIRECTORY STRUCTURE:**

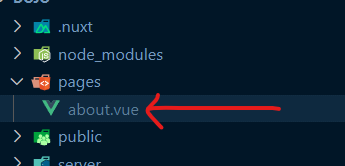
1. Nuxt.config.ts: - this file should always be .ts file (type script file). This file is used to extend or change the default NUXT configuration for your project. You can add custom meta data here, register modules here or add api keys here.
2. App.vue: - This is the default root component of the application. Means it is like index.php (entry point or starting point of an application). It’s the home page of application. You can see **<NuxtWelcome />** in this component which is displaying the default nuxt home page. To start making your app. Remove this component.
3. . nuxt folder: - the nuxt folder is used by nuxt joint development to generate the nuxt application. We do not need to change anything in this folder, leave it as it is.
4. . gitignore: - this is the file used to control what needs to be ignored while pushing to the remote repository.
5. Package. Json: - package. Json file is where you can see all the JavaScript libraries and dependencies installed and their version.

**Adding More pages to application:**

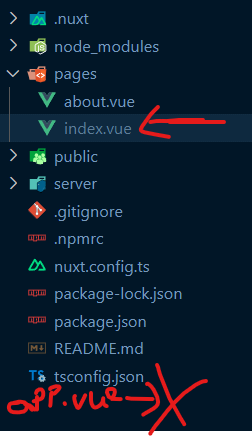
1. Create a folder named pages in root directory



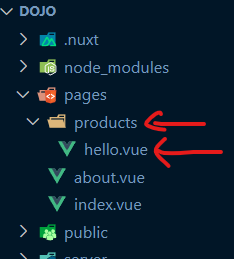
1. Inside this page’s folder we will create different Vue files. Each file will represent different page and routes. And the route path will be determined by file name. For example, if we create a component inside the pages folder called about. Vue, then nuxt will create a route for this page which will be /about in the URL. And if we visit this route localhost:3000/about we will see the about component displaying in the browser.



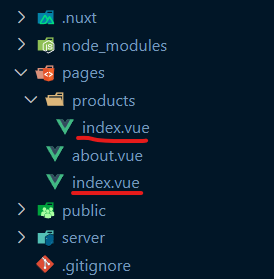
1. In order to let Nuxt use our page components in pages folder we must either delete the app.vue component or use a special component inside app.vue component called NuxtPage (like router-view in vue.js).
2. In our case we will delete the app.vue file (which behaves as a home page for now). And create a new file inside pages folder named index.vue file. this file (index.vue) will now behave as a home page for us now. Nuxt will no need to create a /index route for index.vue file, instead the route for index.vue file will be **“/”, home page**



1. Now write some code in about.vue and index.vue files (vue code (type vbase and press enter)). And start the development server by typing npm run dev in terminal. You will notice that In Browser, you will see home page on this URL localhost:3000, and about page on localhost:3000/about URL
2. If we create subfolders in pages directory, Nuxt will automatically add the name of the subfolder to the route path of the subfolder files. For example, if we create a folder inside pages folder named Products, then we add a component inside this folder, then the name of the folder will be added to the route path for those pages inside product folder. For example, if we create a file named hello.vue inside pages/products folder. Then the path for this hello.vue file will be /products/hello. And when we visit localhost:3000/products/hello, the hello.vu file will be rendered.



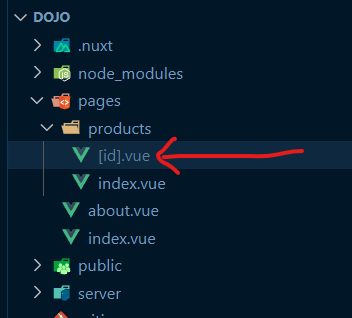
1. In our scenario we do not need hello.vue file inside pages/products folder. So, we will delete hello.vue. And instead create a file inside pages/products folder named index.vue. now this file will generate the following route for us localhost:3000/products. Same like the home page for the website is index.vue file inside the pages folder, the index.vue file inside the products folder, is the home page for the sub folder or subdomain.

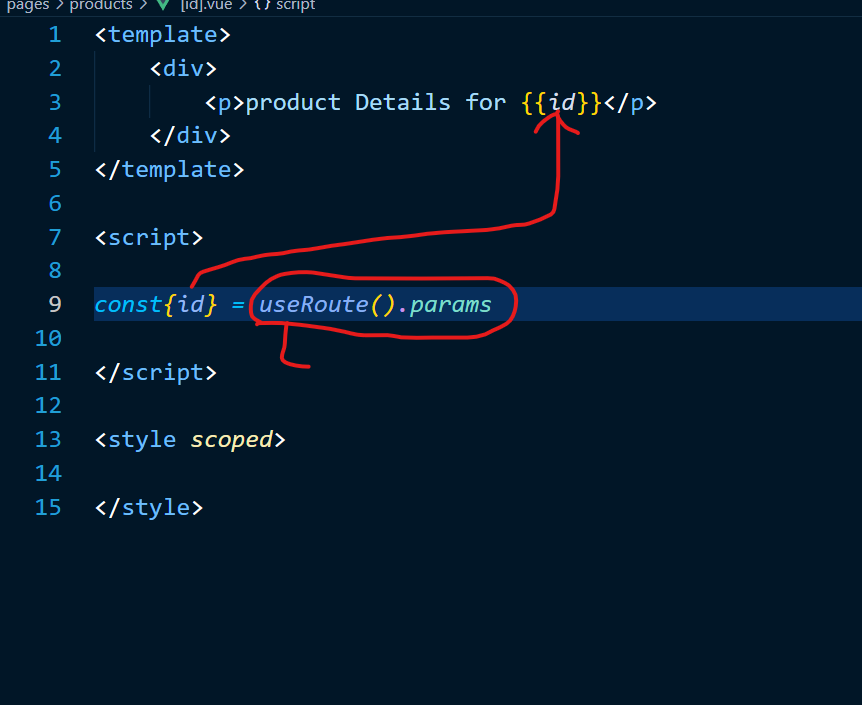


**Adding Dynamic Routes in Nuxt using ID Example: localhost:3000/products/1, localhost:3000/products/2: -**

1. The routes declared in the previous section are definite or static routes, they are not dynamic. What if we want dynamic routing for URL like localhost:3000/products/id, where “id” is the product id (1,2,3,4 etc.), and “id” is dynamic and changeable. We want to use the same structure of HTML and CSS to display every product, no matter what ever the product id is. So, we will be using the same component structure (html, CSS) inside the component, but the data will change based on the ID we pass to the component. So, products page is where you can see all the products localhost:3000/products, but when you click on one product on product page, you will see a page for that product details page specifically localhost:3000/products/id.

1. To achieve this dynamic routing, we will create a file inside a sub folder, for which we want to display data for different Id’s. The name of this file will be wrapped inside “[]”. We can name whatever we want but we will be using id here. We name it [id]. Vue here



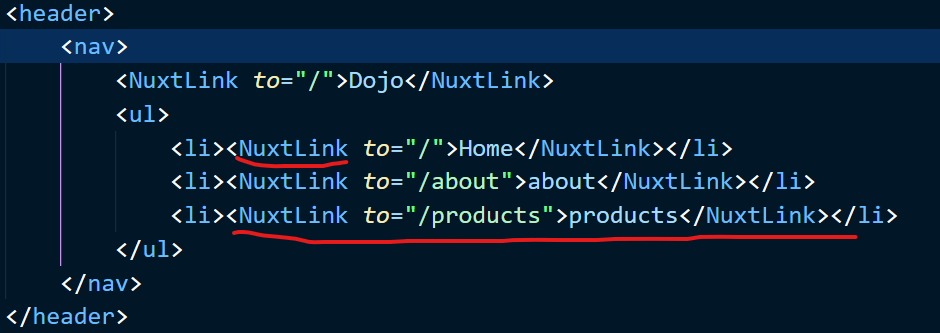
1. Now when we visit localhost:3000/products/id**,** the [id]. vue component will be rendered. Note here that “id” here can be anything, it can be 1,2,3 or a string like Mario or Hamza.
2. [id]. Vue file will pick up all the route variation of the id, because we used the square brackets “[ ]” as part of the file name. and this tells the Nuxt that this part of the route (route parameter) is changeable.
3. So, it does not matter whatever the value of “id” is in the URL, we will still display the [id]. vue component
4. We can capture the value of “id” inside [id]. Vue file by using use route composable function.

Here useRoute().params is the use route composable function.

Note here that the name inside const {id} must match the name inside [id] file name.

**Navbar in Nuxt using NuxtLink:**

We use <NuxtLink></NuxtLink> instead of anchor tag(<a></a>) for routing to another page



**Managing Layouts in Nuxt (header and footer):**

1. To manage layouts, create a folder in root directory called Layouts. You can have many types of layouts. But to create a default layout you need to create a file inside the layout folder named default.vue.

Now copy the header cut the header code in previous section and paste it in default.vue file.

Now you want to display the different pages dynamically in layouts/default.vue file. How can we do that?

We can make use of vue </slot> component. Like this the header will be displayed on all of the pages of nuxt, even on localhost:3000/products/123. <slot / > is a built-in component in Nuxt, so when Nuxt sees this <slot / > component inside a layout folder, it automatically knows that it needs to display HTML content from pages folder of an active page. Conceptually speaking layout folder is a child of pages folder.

## Slot

We have learned that components can accept props, which can be JavaScript values of any type. But how about HTML template content? In some cases, we may want to pass a template fragment to a child component, and let the child component render the fragment within its own template. Like props are for passing data to child component, slots are used to pass Html content to the child component. We write <slot /> in the child component as any data can be passed down to child component to be displayed. This means Layout/default.vue file is a child component of the files inside pages folder.

For detail listen to following lecture: <https://www.udemy.com/course/vuejs-2-the-complete-guide/learn/lecture/21526198#overview>

1. Currently we have a default layout for all the pages of the application. What if we want to create different layout for some pages?

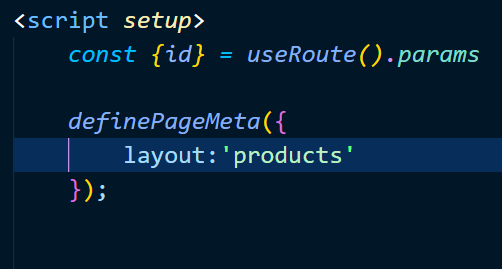
To create custom layouts create a new file in layouts folder for example products. Vue(we can name it anything we want names do not have to match with subfolder name), because we want a different layout for all the product pages.

Change the layout to your desire in layout/products.vue file and save it.

Now to apply this different layout to the different pages in pages folder.

Like for example if we want to apply layouts/products.vue layout to products/index.vue file, we can use a special composable function that Nuxt Gives us inside the <script setup></script> called

definePageMeta. this function accepts an object as a parameter, and there we can specify layout property and then mention the layout we want to use in that page.



To add tailwind CSS to Nuxt Project follow the link below

<https://tailwindcss.nuxtjs.org/getting-started/setup#>

command to install tailwind CSS:

npm install --save-dev @nuxtjs/tailwindcss

then add the tailwind CSS module to Nuxt Config file.

Inside nuxt.config.ts file add the module by typing  
  
modules: [‘@nuxtjs/tailwindcss’]

inside

*export* *default* *defineNuxtConfig* ({

Now if you want to add more modules. You can comma separate it.

modules: [

‘@nuxtjs/tailwindcss’,

‘asdasdasdasd’,

‘asdasdasd’

]

After this, create a folder in root directory named assets. Inside assets create a folder named CSS. Then create a file inside CSS folder named tailwind.css

Then open tailwind.css and paste the following inside the file

@tailwind base;

@tailwind components;

@tailwind utilities;

then create a tailwind config file by using tailwind init (optional).