Tutorial mastering-nuxt-3

# Chapter 1 - Building Our MVP

## 1-0 Getting Set Up

In this lesson, you’ll get set up to start working on the Mastering Nuxt 3 codebase.

We’ll download the code, install pnpm and nuxi, and see how we can use git to checkout the code from a specific lesson.

Update, March 2023: Only the most recent commits are shown when running git log, so you may need to scroll down (using the mouse wheel, enter, or space bar) in order to see the first commits.

pnpm: <https://pnpm.io>

Fig: <https://fig.io>

## 1-1 File based routing

In this lesson, we’ll learn how to use Nuxt to create routes in our app. We’ll see how Nuxt automatically turns our files and directories into routes using Vue Router.

TailwindCSS: <https://tailwindcss.com>

Tailwind Prose plugin: <https://tailwindcss.com/docs/typography-plugin>

NuxtTailwind: <https://tailwindcss.nuxtjs.org>

Nuxt 3 tự động tạo route path theo cấu trúc trong thư mục pages.

Mặc định **pages** là thư mục gốc.

|  |  |
| --- | --- |
|  | A screenshot of a computer  Description automatically generated with medium confidence |

## 1-2 Displaying Course Lessons

In this lesson, we figure out how we want to set up our routes in order to show course lessons. We’ll discuss what makes a good route, and outline how we’ll implement it.

A screen shot of a computer

Description automatically generated with low confidence

* first-chapter, second-lesson: is dinamic route
* soucre/chapter/lesson: is not dinamic route

## 1-3 Nested Routes

In this lesson, we’ll learn how we can use nested routes to create powerful app hierarchies. We’ll separate out the logic for displaying a lesson into it’s own child page.

**Example**: course/lesson; **course** is parent, **lesson** is children (nested route).

**app.vue**

A screen shot of a computer program

Description automatically generated with low confidence

**pages/course.vue**

A screen shot of a computer program

Description automatically generated with low confidence

**course/lesson.vue**

A screenshot of a computer program

Description automatically generated with medium confidence

**Summary to create nested route**

* course: course.vue, folder course.
* lesson: course/lesson.vue.

## 1-4 Dynamic Routes

In this lesson, we’ll explore dynamic routes in Nuxt. This will let us navigate to any chapter and lesson in our course. We’ll also see all the different ways we can use dynamic routes to create complex routing.

### Example 1

A screen shot of a computer

Description automatically generated with low confidence

**Do example 1**

Từ file lesson.vue, thay đổi tên file bằng:

A screen shot of a computer

Description automatically generated with medium confidence

Sau đó ấn enter, được kết quả:

A screen shot of a computer

Description automatically generated with low confidence

Truy cập từ URL

A screenshot of a computer

Description automatically generated with medium confidence

We can access params url:

A screen shot of a computer program

Description automatically generated with low confidence

### Example 2



**Do example 2**

Từ file course/lesson.vue, thay đổi tên file bằng:

A screenshot of a computer

Description automatically generated with medium confidence

Sau đó ấn enter, được kết quả:

A screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer program

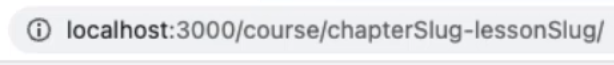
Description automatically generated with low confidence

Truy cập từ URL

A screenshot of a computer

Description automatically generated with medium confidence

### Example 3



A picture containing text, font, screenshot

Description automatically generated

### Example 4

Truy cập URL với tham số tùy chọn [[lessonSlug]]

A picture containing text, font, screenshot, line

Description automatically generated

A picture containing text, font, screenshot

Description automatically generated

Truy cập từ URL

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Vue devtool route**

A green and blue rectangle with black text

Description automatically generated with low confidence

## 1-5 Loading in Course Data

In this lesson, we’ll hook up some actual data so we have something more substantial to work with. We’ll also see how to use nuxi to generate files for us.

nuxi: <https://v3.nuxtjs.org/api/commands/add>

Configuring auto-imports: <https://v3.nuxtjs.org/api/configuration/nuxt.config#imports>

### Create compatable

nuxi add composable useCourse

A screen shot of a computer

Description automatically generated with low confidence

### Create data

A screen shot of a computer program

Description automatically generated with low confidence

A screenshot of a computer screen

Description automatically generated with low confidence

### Import data to component



## 1-7 Add VideoPlayer Component

### Create component

nuxi add component VideoPlayer

A screen shot of a computer

Description automatically generated with low confidence



## 1-8 What is Universal Rendering?

In this lesson we do a deep dive into how Nuxt renders our apps. We’ll take some time to understand how Universal rendering works, and why it gives us the best of SPA and SSR.

Nuxt 3 rendering modes: <https://v3.nuxtjs.org/guide/concepts/rendering>

Ref: <https://viblo.asia/p/so-sanh-server-side-rendering-vs-client-side-rendering-vs-pre-rendering-vs-dynamic-rendering-LzD5dWoOljY>

### Server side rendering (SSR)

A picture containing text, screenshot, jack, design

Description automatically generated

* Cơ chế: Server xử lý logic, render ra html rồi gửi về client hiển thị kết quả.
* Ưu: load lần đầu nhanh, SEO.
* Nhược:
  + Tốn băng thông, nhiều request server se quá tải.
  + Mỗi lần request, sang trang mới, server lại phải render lại toàn bộ trang 🡪 lâu.

### Single page app

A picture containing text, screenshot, design

Description automatically generated

Cơ chế: lần đầu client request đến server, server xử lý, trả về app bundle. Từ lần 2 trở đi, client tự request và render ra html.

Ưu: chia tải với server, việc xử lý logic ở server, client render và hiển thị kết quả. Server chỉ trả về dữ liệu json để client hiển thị ra.

Nhược: khởi động lần đầu chậm, không SEO.

### Universal rendering

A picture containing text, screenshot, design

Description automatically generated

Cơ chế: Request lần đầu, server xử lý logic và render ra html rồi gửi về client. Từ lần 2 trở đi, client client tự request, nhận kết quả json từ server trả về và render ra html.

## 1-9 Navigating with NuxtLink

In this lesson, we’ll see how and why NuxtLink is so useful. We’ll update the navigation of our app to be more efficient, and dig into what NuxtLink is actually doing for us.

NuxtLink: <https://v3.nuxtjs.org/api/components/nuxt-link>

### Example

|  |  |
| --- | --- |
|  |  |

### Active link

Cách 1

A picture containing text, font, screenshot, line

Description automatically generated

Cách 2

A screen shot of a computer program

Description automatically generated with low confidence

## 1-11 The useHead composable

In this lesson, we’ll investigate the useHead to reactively set the title of our page.

useHead: <https://v3.nuxtjs.org/api/composables/use-head>

## 1-12 Tracking progress with useState

In this lesson we’ll dip our toes into state management with useState. We’ll see how we can use it to track state across our app, and why we’d use it instead of ref.

useState: <https://v3.nuxtjs.org/getting-started/state-management>

### Create

A screen shot of a computer program

Description automatically generated with low confidence

### Using

A screen shot of a computer program

Description automatically generated with medium confidence

useState được đặt trong mục composables sử dụng để

* Truyền đi tất cả đến các compoment
* Khi giá trị trong useState thay đổi ở trang A (ví dụ là isLogin ở trên từ true sang fasle), sau đó chuyển qua trang B, rồi quay lại trang A thì giá trị false ở trang A đã chuyển trước đó không mất giá trị (trừ khi refresh lại trang). Note: sử dụng isLogin = ref(true) sẽ bị restart lại giá trị ban đầu là true, không là false.
* Nếu isLogin được gọi ở component A và B, thì nếu component A thay đổi isLogin thì ở component B cũng tự động được update và thay đổi theo.

# 1-13 Save Course Progress with VueUse and the ClientOnly

### Component

In this lesson we’ll see how we can work with third-party modules. We’ll use VueUse to help us store our state in localStorage. Then we’ll dive into understanding the ClientOnly component, why we’d want to use it, and a convenient alternative.

Using VueUse with Nuxt: <https://vueuse.org/guide/#nuxt>

useLocalStorage: <https://vueuse.org/core/uselocalstorage/#uselocalstorage>

ClientOnly: <https://v3.nuxtjs.org/api/components/client-only>

### useLocalStorage

Hình mục 1-12, sử dụng **useLocalStorage** để lưu giá trị **process** khi thực hiện **handleChangeLogin**.

A screenshot of a computer

Description automatically generated

### ClientOnly

Sử dụng để xác định đoạn mã js trong file hoặc file cần render ở Client (không render ở Server sau đó trả về khi refresh trang). Điều này giúp cho giữ được trạng thái của Client, không bị ảnh hưởng khi fresh trang.

**Cho js trong file**

A picture containing text, font, screenshot

Description automatically generated

**Cho cả file**

A picture containing text, screenshot, font, line

Description automatically generated

## 1-14 Deploying our MVP to Netlify

Finally, we get to deploy our app!

In this lesson, we’ll see how we can deploy our app using Netlify. From this point on, you’ll be able to deploy your app whenever you want throughout the course.

A great way to share your progress with friends and co-workers!

Update, March 2023: If you’re using pnpm you will need to add a .nprmc file to the root of your project with the single line shamefully-hoist=true. This will make sure that when Netlify builds your project it runs pnpm with the flag --shamefully-hoist.

Netlify: <https://www.netlify.com/>

Zero-configuration deploy with Nitro: <https://nitro.unjs.io/deploy/#zero-config-providers>

# Chapter 2 - Architecture and Organizing Our Code

## Chapter 1 - Building Our MVP

In this chapter, we’ll take a step back and look deeper into Nuxt 3 architecture.

We’ll see how we can use Pages, Layouts, and Components to organize our code better, and increase the maintainability of our code over time.

## 2-1 Smaller is better / but not too small

In this lesson, we explore how smaller pieces of functionality are generally easier to work with.

We also outline the different tools that Nuxt 3 gives us to help us organize our app code in an effective way.

## 2-2 Understanding App.vue

In this lesson, we take a quick look at the app.vue file, the component where everything begins.

app.vue: <https://v3.nuxtjs.org/guide/directory-structure/app>

* App.vue là file root
* Tuy nhiên, có thể xóa file app.vue. Từ url, nuxt sẽ căn cứ vào tên file để truy cập đến các file trong thư mục pages trùng tên theo đường dẫn url.
* File index.vue trong thư mục pages, ứng với router root ‘/’
* File pages/movies.vue tương ứng với pages/movies/index.vue

2-3 Understanding Layouts

**Tạo layout**

A screenshot of a computer

Description automatically generated with low confidence

**Sử dụng layout**

A screenshot of a computer program

Description automatically generated with medium confidence

Hoặc trong component

A picture containing text, font, screenshot, line

Description automatically generated

## 2-4 Pages vs. Layouts vs. Components

Ref: <https://nuxt.com/docs/getting-started/views>

### Layout

Sử dụng để share UI dùng chung cho nhiều pages.

Layouts are wrappers around pages that contain a common User Interface for several pages, such as a header and footer display. Layouts are Vue files using <slot /> components to display the **page** content. The layouts/default.vue file will be used by default. Custom layouts can be set as part of your page metadata.

A picture containing screenshot, rectangle, green, game

Description automatically generated

### Page

Pages represent views for each specific route pattern. Every file in the pages/ directory represents a different route displaying its content.

### Component

Thành phần unique một chức năng, sau đó import vào sử dụng trong pages và layout.

Component is **reusable pieces** of the user interface. you can create these components in the **components/ directory**, and they will be **automatically available across your application** **without import them.**

### Hướng dẫn tổ chức code in Nuxt 3

Bước 1: Tạo trang pages

Bước 2: Đưa các phần dùng chung cho nhiều pages sang layouts

Bước 3: Đưa các phần nhỏ hơn gắn với chức năng sang components

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

## 2-5 Understanding Composables in Nuxt

In this lesson we take a look at using Composables in Nuxt 3. We’ll see the three different types of composables, and why they’re useful to us.

Composables: <https://v3.nuxtjs.org/guide/directory-structure/composables>

Nuxt 3 uses the composables/ directory to automatically import your Vue composables into your application using auto-imports!

A picture containing font, text, line, typography

Description automatically generated

A white background with blue text

Description automatically generated with low confidence

# Chapter 3 - Making Our App Robust

## 3-0 Making Our App Robust

In this chapter, we’ll work on making our app more robust and reliable for our end users.

We start by adding in some TypeScript to avoid the most common types of problems with our code. Then we’ll look at adding a custom 404 page.

After that, we tackle handling client-side errors using the NuxtErrorBoundary component. We see how we can resolve errors in a couple different ways. Then we implement server-side error handling, as well as adding in a custom error page for our app.

Finally, we refactor some of our error handling into route validation instead, using the validate function on definePageMeta.

## 3-1 Adding in TypeScript

In this lesson, we’ll add types to our useCourse composable.

By making it fully typed, we make it easier to avoid basic errors like treating a string as an int (or vice versa), typos, and other mistakes that are easily avoidable when using types.

We also get much better autocomplete support, which makes our development experience better!

## 3-2 Handling Client-Side Errors with NuxtErrorBoundary

In this lesson, we look at how we can use the NuxtErrorBoundary component to elegantly handle client-side errors. By isolating errors to specific parts of our app, we’re able to have better error handling and error messages than a generic error page would allow.

We’ll look at how errors are created (and thrown) within Nuxt, how we can resolve these client-side errors, and how to show our own custom error messages.

NuxtErrorBoundary: https://v3.nuxtjs.org/api/components/nuxt-error-boundary createError: <https://v3.nuxtjs.org/api/utils/create-error>

## 3-3 Advanced Error Handling

In this lesson, we’ll see another example of dealing with client-side errors.

We can’t always resolve a client-side error by simply re-rendering the broken component. Often, we’ll need to do more to fix what happened. In many cases this will be navigating to a different part of the app.

Finally, we’ll also look at fatal client-side errors, and discuss where you might want to use the NuxtErrroBoundary component.

NuxtErrorBoundary: <https://v3.nuxtjs.org/api/components/nuxt-error-boundary>

createError: <https://v3.nuxtjs.org/api/utils/create-error>

navigateTo: <https://v3.nuxtjs.org/api/utils/navigate-to>

## 3-4 Handling Server Errors and 404s

In this lesson, we’ll shift our attention to server-side errors. Many of the same principles apply, but there are some differences between handling client and server-side errors.

We’ll see how we can add a custom error page with error.vue, and how we can use the clearError and useError composables to work with the server-side error object.

useError: <https://v3.nuxtjs.org/api/composables/use-error/>

clearError: <https://v3.nuxtjs.org/api/utils/clear-error/>

## 3-5 Route Validation

In this lesson, we’ll refactor our error handling code into a route validation method instead. The validate method on definePageMeta is a middleware shorthand that makes it easy to add route validation to any page.

We also discuss how to think about server-side error handling, and where we should be putting it.

validate method on definePageMeta: <https://v3.nuxtjs.org/api/utils/define-page-meta>

# Chapter 4 - Route Middleware and Authentication

## 4-Intro

In this chapter we tackle authentication, one of the most complex things we’ll tackle in this course.

But to get there we’ll first learn about route middleware, including inine, named and global middleware.

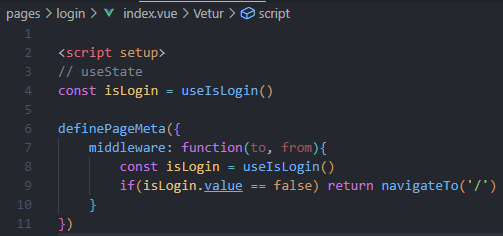
Then, we’ll set up our Supabase project and also set up Github, so we can use OAuth in our app. We’ll then look at environment variables and using Netlify’s CLI to easily import them into our production app.

Now comes the good part, where we finally implement logging and out, which is pretty straightforward thanks to the Supabase Nuxt module. Once we’ve done that, we can finally protect some routes in our application.

In the last lesson, we take some time to learn about what OAuth is, how it works, and why we’re using it in this application.

## 4-1 Route Middleware Basics

In this lesson, we refactor the route validation into route middleware. Middleware allows us to run code whenever the route changes.



## 4-2 Creating a Login Page

A screen shot of a computer program

Description automatically generated with low confidence

## 4-3 Inline, Named, and Global Route Middleware

In this lesson, we expand our middleware arsenal beyond inline middleware. Learn how to implement named and global middleware for re-usability.

## 4-4 Setting Up Supabase

## 4-5 Adding Environment Variables

We’re mostly done with setting up Supabase, but we still have to set up our environment variables. Environment variables are important in any app for two main reasons:

Not storing sensitive data in our git repo. Separating configuration from code. Adding environment variables to our local dev environment is pretty easy. And luckily for us, the awesome Netlify CLI tool makes it a breeze to instantly add them to our production environment as well!

Netlify CLI: <https://docs.netlify.com/cli/get-started>

Nuxt Supabase module: <https://supabase.nuxtjs.org/get-started>

## 4-6 Logging in with Github

## 4-7 Logging Out

## 4-8 Protecting Routes with Auth

Finally, we can now protect our routes using authentication!

It’s a fairly simple update to change our middleware to also check for a logged in user. Now we can freely access Chapter 1, but any other lessons require us to be logged in with Github. Once we build our checkout we’ll be able to restrict access only to those who have bought.

We also implement a UX improvement. If we redirect to the login page, we’ll redirect back to where the user was trying to go after they’ve logged in.

## 4-9 Understanding OAuth Basics

We’ve been discussing OAuth a lot in this chapter, but we haven’t really talked about what it is. In this lesson we cover why we need OAuth, what it does, and how we’ve been using it in our online course platform.

Understanding OAuth at a basic level helps us put all of these auth pieces together, even though a deep understanding of it just isn’t necessary. Auth is hard, and that’s why we outsource it instead of building it from scratch ourselves.

A diagram of a chatbot

Description automatically generated

# Chapter 5 - Server Routes

## 5-Intro

In this chapter, we’ll learn all about server routes with Nuxt 3.

We’ll start by refactoring our app to serve course data from an API. Then, we’ll set up Prisma and our database so we can fetch the data from there.

Along the way we’ll explore how we can type our endpoints, how data fetching works in Nuxt, and we’ll also dig deep into Nitro.

## 5-1 Server Route Basics

In this lesson, we look at the basics of how to create server routes in Nuxt 3, and start to build out our first endpoint for serving up lesson data.

We use a tool called Hoppscotch (written in Nuxt!) which lets us easily test and inspect our routes.

Hoppscotch: <https://hoppscotch.io>

## 5-2 Lesson Endpoint

## 5-3 Fully Typing Our Endpoint

## 5-4 Course Metadata Endpoint

## 5-5 Basic Data Fetching

## 5-6 Advanced Data Fetching

## 5-7 Creating the useFetchWithCache Composable

## 5-8 Fetch Course Metadata

## 5-9 Setting up Prisma

## 5-10 Creating the Prisma Schema

## 5-11 Initialize Prisma and Seed Database

## 5-12 Getting data from Prisma

## 5-13 Update Lesson Endpoint

## 5-14 Update Course Meta Endpoint

## 5-15 Auth and Server Routes

## 5-16 Understanding Nitro and h3

# Chapter 6 - State Management with Pinia

## 6-Intro

## 6-1 Setting up Pinia

## 6-2 Fixing the Meta Endpoint

## 6-3 Add LessonProgress Model to Our Schema

## 6-4 Add Endpoint to Update Progress

## 6-5 Add User Progress Endpoint

## 6-6 Show Course Completion Percentage

# Chapter 7 - Building a Sales Page

## 7-Intro

## 7-1 Adding the Landing Page

## 7-2 Static Generation and Pre-Rendering

## 7-3 Route Rules

## 7-4 Managing Static Assets

## 7-5 Lazy Loading Checkout Components

## 7-6 Runtime Config and App Config

## 7-7 Setting up Stripe

## 7-8 Handle Payments with Stripe

## 7-9 Add the PaymentIntent Endpoint

## 7-10 Overview of Granting Access

## 7-11 Create CoursePurchase in the Database

## 7-12 Verify Purchase with Stripe Webhooks

## 7-13 Testing our Webhook Handler

## 7-14 Link Purchase with Github

## 7-15 Grant Access to the Course