# GIỚI THIỆU

## NUXT

Là một frameword của Vue, Nuxt giúp bạn xây dựng ứng dụng web hiện đại, mạnh mẽ, đặc biệt là **SSR (Server Side Rendering)**, **Static Site Generation (SSG)**, và cả **Hybrid Rendering**.

Có 2 bản là Nuxt 2 và Nuxt 3 hỗ trợ Vue 2 và Vue 3 tương ứng.

### Tự động hóa và quy ước

* Định tuyến: định tuyến tự động dựa trên cấu trúc nằm trong thư mục pages/
* Tự động tách mã nhỏ hơn
* Đi kèm khả năng tích hợp SSR (tự thiếp lập máy chủ)
* Tự động import component

### Công cụ máy chủ

Cung cấp khả năng lập trình fullstack, sử dụng **Rollup** và **Nodejs**. Tạo server API bởi đọc file **server/api/** và server middleware từ **server/middleware**

Khi build project Nitro sẽ build app và server thành 1 file phổ quát là **.output**

## Installation

### Yêu cầu

* Node.js phiên bản 18. trở lên
* IDE lập trình như VSCode hoặc Webstorm

### Cài đặt

npx nuxi@latest init <project-name>

## Configuration

Các tệp cấu hình dự án thường nằm tại thư mục gốc của dự án

VD: myproject/các tệp cấu hình

### nuxt.config.ts

Ghi đè hoặc mở rộng các hành vi của ứng dụng

export default defineNuxtConfig({

// My Nuxt config

})

Hoặc có thể khai báo các biến môi trường tại đây thay vì các file .env như trước kia và được đặt trong biến runtimeConfig (private), runtimeConfig/public.

Các dữ liệu công khai cho phép cả server và client đọc được thì lưu ở **runtimeConfig/public** còn chỉ server đọc được thì lưu ở **runtimeConfig**

export default defineNuxtConfig({

runtimeConfig: {

// The private keys which are only available server-side

apiSecret: '123',

// Keys within public are also exposed client-side

public: {

apiBase: '/api'

}

}

})

Sử dụng bằng cách gọi useRuntimeConfig()

<script setup lang="ts">

const runtimeConfig = useRuntimeConfig()

</script>

Tuy nhiên nếu bạn có file .env nó sẽ ghi đè file config bởi .env (ưu tiên .env hơn)

### app.config.ts

Khá giống với nuxt.config.ts là đều khai báo các biến có thể dùng nhiều nơi trong ứng dụng, tuy nhiên với app.config.ts sẽ có vài điểm khác biệt.

Thường dùng cho cấu hình giao diện của ứng dụng: tên app, màu sắc, cấu hình mặc định cho UI components, và các cấu hình public khác. Khác với runtimeConfig, những thứ này sẽ không thể bị ghi đè bởi các cấu hình môi trường

export default defineAppConfig({

title: 'My Awesome Nuxt App',

theme: {

primaryColor: '#3490dc'

}

})

Cách sử dụng

<script setup lang="ts">

const appConfig = useAppConfig()

console.log(appConfig.title) // "My Awesome Nuxt App"

console.log(appConfig.theme.primaryColor) // "#3490dc"

</script>

### Các bản config khác

Thông thường có thể bạn sẽ có những file config khác như: postcss.config.js, vite.config.ts, tsconfig.json, tailwind.config.js, … Tuy nhiên trong nuxt sẽ gộp tất cả các file cấu hình đó vào 1 nut.config.ts, tuy nhiên chỉ những file liên quan trực tiếp với build system

VD: postcss.config.js thì được viết trong nuxt.config.ts như sau

export default defineNuxtConfig({

postcss: {

plugins: {

tailwindcss: {},

autoprefixer: {}

}

}

})

Tuy nhiên với những file không liên quan trực tiếp đến build system thì không cần cho vào mà vẫn giữ nguyên file: tsconfig.json, eslint.config.js, prettier.config.js, stylelint.config.js, tailwind.config.js, vitest.config.ts, …

## Views

## Styling

### Css toàn dự án

Để import 1 file css global cho toàn bộ dự án, truy cập file nuxt.config.ts

export default defineNuxtConfig({

...

css: ['~/assets/css/main.css']

})

Trong đó ~/assets/css/main.css là đường dẫn đến file css global

### Css một file cụ thể

Có 3 cách import 1 file style thông thường

* Cách 1 (phổ biến nhất)

<script lang="ts" setup>

import "~/assets/css/first.css";

</script>

* Cách 2

<script lang="ts" setup>

import("~/assets/css/first.css");

</script>

* Cách 3

<style>

@import url("~/assets/css/first.css");

</style>

### Font

Nếu bạn có mẫu font riêng, bạn có thể lưu chúng trong mục /public/fonts

VD: /fonts/FarAwayGalaxy.woff

Lúc này trong file main.css (css toàn cục và phải import vào nuxt.config.css):

@font-face {

font-family: 'FarAwayGalaxy';

src: url('/fonts/FarAwayGalaxy.woff') format('woff');

font-weight: normal;

font-style: normal;

font-display: swap;

}

Bây giờ bạn có thể sử dụng font ở bất kì file css nào trong dự án

<style>

h1 {

font-family: 'FarAwayGalaxy', sans-serif;

}

</style>

### Stylesheet

Có thể dùng thư viện css khác để sử dụng

VD: thư viện animate

npm install animate.css

<script lang="ts" setup>

import "animate.css";

</script>

<template>

<section>

<p class="animate\_\_animated animate\_\_shakeX">This page will be displayed at the /about route.</p>

</section>

</template>

Hoặc có thể import trong nuxt.config.ts để có thể sử dụng toàn bộ dự án:

export default defineNuxtConfig({

...

css: ['~/assets/css/main.css','animate.css']

})

### External stylesheet

Bạn cũng có thể sử dụng liên kết ngoài thay vì tải về, sử dụng trong app / head vì đây sẽ là phần khai báo trong thẻ head

export default defineNuxtConfig({

app: {

head: {

link: [{ rel: 'stylesheet', href: 'https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css' }]

}

}

})

Ngoài ra nuxt cung cấp cho bạn composable useHead() để có thể import cho file cụ thể

useHead({

link: [

{

rel: 'stylesheet',

href: 'https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css'

}

]

})

### Nitro plugin

Nếu muốn can thiệp sâu hơn vào quá trình render html có thể dùng Nitro plugin để chèn trực tiếp vào head. Tuy nhiên không khuyến khích vì phải tải xong CDN bên ngoài mới render có thể gây chậm lag, nên nếu cài npm sẽ tối ưu hơn.

Tạo 1 file plugin: server/plugins/my-plugin.ts

export default defineNitroPlugin((nitro) => {

nitro.hooks.hook('render:html', (html) => {

html.head.push('<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">')

})

})

### Preprocessor

Cài preprocessor bạn muốn dùng, ví dụ sass:

npm install -D sass

Sau khi cài có thể sử dụng các file .scss .sass

VD:

<style lang="scss">

@use "~/assets/scss/main.scss";

</style>

@use cú pháp scss tốt hơn @import

Hoặc có thể import trong config

export default defineNuxtConfig({

css: ['~/assets/scss/main.scss']

})

**Inject biến/mixins vào mọi file SCSS**

Giả sử bạn có file biến như sau:

assets/\_colors.scss

$primary: #49240F;

$secondary: #E4A79D;

Bạn muốn dùng ở mọi file scss mà không muốn import nó lại mỗi lần

export default defineNuxtConfig({

vite: {

css: {

preprocessorOptions: {

scss: {

additionalData: '@use "~/assets/\_colors.scss" as \*;'

}

}

}

}

})

### Styling Single File Component

Mọi cú pháp tương tự như trong vue

VD:

<script setup lang="ts">

const color = ref("red")

</script>

<template>

<div class="text">hello</div>

</template>

<style>

.text {

color: v-bind(color);

}

</style>

## Routing

Khác với vue thông thường là phải khai báo, import thì nuxt sẽ tự động tải các mục trong folder /pages

VD:

-| pages/

---| about.vue

---| index.vue

---| posts/

-----| [id].vue

### NuxtLink

Giống như <router-link> trong vue. Khi render sẽ thành thẻ a, nhưng khi chuyển trang sẽ mượt mà và không tải lại trang (client-side)

VD:

<NuxtLink to="/about">About Us</NuxtLink>

Vì Component này sẽ được tải trước nên việc nhấn vào sẽ xuất hiện ngay lập tức, có thể tắt tính năng này bằng thuộc tính prefetch="false" trong NuxtLink

### Router có tham số

Trong Nuxt, bạn có thể tạo một file trong thư mục pages/ với tên có **dạng dấu ngoặc vuông** để định nghĩa một **route động**.

VD:

pages/posts/[id].vue

Kết quả:

/posts/:id