Introduction

Nuxt is a free and open-source framework with an intuitive and extendable way to create type-safe, performant and production-grade full-stack web applications and websites with Vue.js.

We made everything so you can start writing `.vue` files from the beginning while enjoying hot module replacement in development and a performant application in production with server-side rendering by default.

Nuxt has no vendor lock-in, allowing you to deploy your application **anywhere, even to the edge**.

Automation and Conventions

Nuxt uses conventions and an opinionated directory structure to automate repetitive tasks and allow developers to focus on pushing features. The configuration file can still customize and override its default behaviors.

- ✓ **File-based routing:** define routes based on the structure of your `pages/` directory. This can make it easier to organize your application and avoid the need for manual route configuration.
- ✓ **Code splitting:** Nuxt automatically splits your code into smaller chunks, which can help reduce the initial load time of your application.
- ✓ **Server-side rendering out of the box:** Nuxt comes with built-in SSR capabilities, so you don't have to set up a separate server yourself.
- ✓ **Auto-imports:** write Vue composables and components in their respective directories and use them without having to import them with the benefits of tree-shaking and optimized JS bundles.
- ✓ **Data-fetching utilities:** Nuxt provides composables to handle SSR-compatible data fetching as well as different strategies.
- ✓ **Zero-config TypeScript support:** write type-safe code without having to learn TypeScript with our auto-generated types and `tsconfig.json`

- ✓ **Configured build tools:** we use Vite by default to support hot module replacement (HMR) in development and bundling your code for production with best-practices baked-in.

Nuxt takes care of these and provides both frontend and backend functionality so you can focus on what matters: **creating your web application**.

Server-Side Rendering

Nuxt comes with built-in server-side rendering (SSR) capabilities by default, without having to configure a server yourself, which has many benefits for web applications:

- ✓ **Faster initial page load time:** Nuxt sends a fully rendered HTML page to the browser, which can be displayed immediately. This can provide a faster perceived page load time and a better user experience (UX), especially on slower networks or devices.
- ✓ **Improved SEO:** search engines can better index SSR pages because the HTML content is available immediately, rather than requiring JavaScript to render the content on the client-side.
- ✓ **Better performance on low-powered devices:** it reduces the amount of JavaScript that needs to be downloaded and executed on the client-side, which can be beneficial for low-powered devices that may struggle with processing heavy JavaScript applications.
- ✓ **Better accessibility:** the content is immediately available on the initial page load, improving accessibility for users who rely on screen readers or other assistive technologies.
- ✓ **Easier caching:** pages can be cached on the server-side, which can further improve performance by reducing the amount of time it takes to generate and send the content to the client.

Overall, server-side rendering can provide a faster and more efficient user experience, as well as improve search engine optimization and accessibility.

As Nuxt is a versatile framework, it gives you the possibility to statically render your whole application to a static hosting with `nuxt generate`, disable SSR globally with the `ssr: false` option or leverage hybrid rendering by setting up the `routeRules` option.

Read more about the Nuxt rendering modes.

Server engine

The Nuxt server engine Nitro unlocks new full-stack capabilities.

In development, it uses Rollup and Node.js workers for your server code and context isolation. It also generates your server API by reading files in `server/api/` and server middleware from `server/middleware/`.

In production, Nitro builds your app and server into one universal `.output` directory. This output is light: minified and removed from any Node.js modules (except polyfills). You can deploy this output on any system supporting JavaScript, from Node.js, Serverless, Workers, Edge-side rendering or purely static.

Read more about Nuxt server engine.

Production-ready

A Nuxt application can be deployed on a Node or Deno server, pre-rendered to be hosted in static environments, or deployed to serverless and edge providers.

Discover more in the deployment section.

Modular

A module system allows to extend Nuxt with custom features and integrations with third-party services.

Discover more about modules.

Architecture

Nuxt is composed of different core packages:

- Core Engine: nuxt
- Bundlers: @nuxt/vite-builder and @nuxt/webpack-builder
- Command line interface: nuxi
- Server engine: nitro
- Development kit: @nuxt/kit

Nuxt 2 Bridge: @nuxt/bridge

We recommend reading each concept to have a full vision of Nuxt capabilities and the scope of each package.

Are you Nuxt?

Nuxt is the backbone of your Vue.js project, giving structure to build your project with confidence while keeping flexibility.

Extendable with a strong module ecosystem and hooks engine, it makes it easy to connect your REST or GraphQL endpoints, favorite CMS, CSS frameworks and more. PWA and AMP support is only a module away from your Nuxt project.

Ready to try? Head over to the Installation section.

Contribute

Do you want to get involved in the evolution of Nuxt?
Follow the contribution guide.

Edit on Github

Enterprise Design Kit NuxtLabs
Nuxt Studio