TypeScript

Nuxt 3 is fully typed and provides helpful shortcuts to ensure you have access to accurate type information when you are coding.

Type-checking

By default, Nuxt doesn't check types when you run `nuxi dev` or `nuxi build`, for performance reasons. However, you can enable type-checking at build or development time by installing `vue-tsc` and `typescript` as devDependencies and either enabling the `typescript.typeCheck` option in your `nuxt.config` file or manually checking your types with nuxi.

```
> yarn nuxi typecheck
```

Auto-generated Types

When you run `nuxi dev` or `nuxi build`, Nuxt generates the following files for IDE type support (and type checking):

`.nuxt/nuxt.d.ts`

This file contains the types of any modules you are using, as well as the key types that Nuxt 3 requires. Your IDE should recognize these types automatically.

Some of the references in the file are to files that are only generated within your `buildDir` (`.nuxt`) and therefore for full typings, you will need to run `nuxi dev` or `nuxi build`.

.nuxt/tsconfig.json

This file contains the recommended basic TypeScript configuration for your project, including resolved aliases injected by Nuxt or modules you are using, so you can get full type support and path auto-complete for aliases like `~/file` or `#build/file`.

Read more about how to extend this configuration.

> 👉 Nitro also auto-generates types for API routes. Plus, Nuxt also generates types for globally available components and auto-imports from your composables, plus other core functionality.

> Keep in mind that all options extended from `./.nuxt/tsconfig.json` will be overwritten by the options defined in your `tsconfig.json`. Overwriting options such as `"compilerOptions.paths"` with your own configuration will lead TypeScript to not factor in the module resolutions from `./.nuxt/tsconfig.json`. This can lead to module resolutions such as `#imports` not being recognized.
>
> In case you need to extend options provided by `./.nuxt/tsconfig.json` further, you can use the `alias` property within your `nuxt.config`. `nuxi` will pick them up and extend `./.nuxt/tsconfig.json` accordingly.

Stricter Checks

TypeScript comes with certain checks to give you more safety and analysis of your program.

Once you've converted your codebase to TypeScript and felt familiar with it, you can start enabling these checks for greater safety (read more).

In order to enable strict type checking, you have to update `nuxt.config`:

```ts
export default defineNuxtConfig({
  typescript: {
    strict: true
  }
})
```

Edit on Github