Nuxt Configuration Reference

Discover all the options you can use in your nuxt.config.ts file.

This file is auto-generated from the source code.

alias

- **Type**: `object`
- **Default**

```
{
  "~": "/<rootDir>",
  "@": "/<rootDir>",
  "~~": "/<rootDir>",
  "@@": "/<rootDir>",
  "assets": "/<rootDir>/assets",
  "public": "/<rootDir>/public"
}
```

> You can improve your DX by defining additional aliases to access custom directories within your JavaScript and CSS.

Note: Within a webpack context (image sources, CSS - but not JavaScript) you *must* access your alias by prefixing it with `~` .

Note: These aliases will be automatically added to the generated `.nuxt/tsconfig.json` so you can get full type support and path auto-complete. In case you need to extend options provided by `./.nuxt/tsconfig.json` further, make sure to add them here or within the `typescript.tsConfig` property in `nuxt.config`.

Example:

```
export default {
  alias: {
    'images': fileURLToPath(new URL('./assets/images', import.meta.url)),
    'style': fileURLToPath(new URL('./assets/style', import.meta.url)),
    'data': fileURLToPath(new URL('./assets/other/data', import.meta.url))
  }
}
```

analyzeDir

- **Type**: `string`
- **Default:** `"/<rootDir>/.nuxt/analyze"`

> The directory where Nuxt will store the generated files when running `nuxt analyze`.

If a relative path is specified, it will be relative to your `rootDir`.

app

> Nuxt App configuration.

baseURL

- **Type**: `string`
- **Default:** `"/"`

> The base path of your Nuxt application.

This can be set at runtime by setting the NUXT_APP_BASE_URL environment variable.

Example:

```
> NUXT_APP_BASE_URL=/prefix/ node .output/server/index.mjs
```

buildAssetsDir

- **Type**: `string`
- **Default:** `"/_nuxt/"`

> The folder name for the built site assets, relative to `baseURL` (or `cdnURL` if set). This is set at build time and should not be customized at runtime.

cdnURL

- **Type**: `string`
- **Default:** `""`

> An absolute URL to serve the public folder from (production-only).

This can be set to a different value at runtime by setting the `NUXT_APP_CDN_URL` environment variable.

Example:

```
> NUXT_APP_CDN_URL=https://mycdn.org/ node .output/server/index.mjs
```

head

- **Type**: `object`
- **Default**

```
{
  "meta": [
```

```
    {
      "name": "viewport",
      "content": "width=device-width, initial-scale=1"
    },
    {
      "charset": "utf-8"
    }
  ],
  "link": [],
  "style": [],
  "script": [],
  "noscript": []
}
```

> Set default configuration for `<head>` on every page.

Example:

```
app: {
  head: {
    meta: [
      // <meta name="viewport" content="width=device-width, initial-scale=1">
      { name: 'viewport', content: 'width=device-width, initial-scale=1' }
    ],
    script: [
      // <script src="https://myawesome-lib.js"></script>
      { src: 'https://awesome-lib.js' }
    ],
    link: [
      // <link rel="stylesheet" href="https://myawesome-lib.css">
      { rel: 'stylesheet', href: 'https://awesome-lib.css' }
    ],
    // please note that this is an area that is likely to change
    style: [
      // <style type="text/css">:root { color: red }</style>
      { children: ':root { color: red }', type: 'text/css' }
    ],
    noscript: [
      // <noscript>JavaScript is required</noscript>
      { children: 'JavaScript is required' }
    ]
  }
}
```

keepalive

- **Type**: `boolean`
- **Default:** `false`

> Default values for KeepAlive configuration between pages.

This can be overridden with `definePageMeta` on an individual page. Only JSON-serializable values are allowed.

See: https://vuejs.org/api/built-in-components.html#keepalive

layoutTransition

- **Type**: `boolean`
- **Default:** `false`

> Default values for layout transitions.

This can be overridden with `definePageMeta` on an individual page. Only JSON-serializable values are allowed.

See: https://vuejs.org/api/built-in-components.html#transition

pageTransition

- **Type**: `boolean`
- **Default:** `false`

> Default values for page transitions.

This can be overridden with `definePageMeta` on an individual page. Only JSON-serializable values are allowed.

See: https://vuejs.org/api/built-in-components.html#transition

rootId

- **Type**: `string`
- **Default:** `"__nuxt"`

> Customize Nuxt root element id.

rootTag

- **Type**: `string`
- **Default:** `"div"`

> Customize Nuxt root element tag.

appConfig

> Additional app configuration

For programmatic usage and type support, you can directly provide app config with this option. It will be merged with `app.config` file as default value.

nuxt

build

> Shared build configuration.

analyze

- **Type**: `boolean`
- **Default:** `false`

> Nuxt uses `webpack-bundle-analyzer` to visualize your bundles and how to optimize them.

Set to `true` to enable bundle analysis, or pass an object with options: for webpack or for vite.

Example:

```
analyze: {
  analyzerMode: 'static'
}
```

templates

- **Type**: `array`

> You can provide your own templates which will be rendered based on Nuxt configuration. This feature is specially useful for using with modules.

Templates are rendered using `lodash/template`.

Example:

```
templates: [
  {
    src: '~/modules/support/plugin.js', // `src` can be absolute or relative
    dst: 'support.js', // `dst` is relative to project `.nuxt` dir
    options: {
      // Options are provided to template as `options` key
      live_chat: false
    }
  }
]
```

transpile

- **Type**: `array`

> If you want to transpile specific dependencies with Babel, you can add them here. Each item in transpile can be a package name, a function, a string or regex object matching the dependency's file name.

You can also use a function to conditionally transpile. The function will receive an object ({ isDev, isServer, isClient, isModern, isLegacy }).

Example:

```
    transpile: [({ isLegacy }) => isLegacy && 'ky']
```

buildDir

- **Type**: `string`
- **Default:** `"/<rootDir>/.nuxt"`

> Define the directory where your built Nuxt files will be placed.

Many tools assume that `.nuxt` is a hidden directory (because it starts with a `.`). If that is a problem, you can use this option to prevent that.

Example:

```
export default {
  buildDir: 'nuxt-build'
}
```

builder

- **Type**: `string`
- **Default:** `"@nuxt/vite-builder"`

> The builder to use for bundling the Vue part of your application.

components

- **Type**: `object`
- **Default**

```
{
```

```
  "dirs": [
    {
      "path": "~/components/global",
      "global": true
    },
    "~/components"
  ]
}
```

> Configure Nuxt component auto-registration.

Any components in the directories configured here can be used throughout your pages, layouts (and other components) without needing to explicitly import them.

See: https://nuxt.com/docs/guide/directory-structure/components

css

- **Type**: `array`

> You can define the CSS files/modules/libraries you want to set globally (included in every page).

Nuxt will automatically guess the file type by its extension and use the appropriate pre-processor. You will still need to install the required loader if you need to use them.

Example:

```
css: [
  // Load a Node.js module directly (here it's a Sass file).
  'bulma',
  // CSS file in the project
  '@/assets/css/main.css',
  // SCSS file in the project
  '@/assets/css/main.scss'
]
```

debug

- **Type**: `boolean`

- **Default:** `false`

> Set to `true` to enable debug mode.

At the moment, it prints out hook names and timings on the server, and logs hook arguments as well in the browser.

dev

- **Type**: `boolean`
- **Default:** `false`

> Whether Nuxt is running in development mode.

Normally, you should not need to set this.

devServer

host

> Dev server listening host

https

- **Type**: `boolean`
- **Default:** `false`

> Whether to enable HTTPS.

Example:

```
export default defineNuxtConfig({
  devServer: {
    https: {
      key: './server.key',
```

```
      cert: './server.crt'
    }
  }
})
```

loadingTemplate

- **Type**: `function`

> Template to show a loading screen

port

- **Type**: `number`
- **Default:** `3000`

> Dev server listening port

url

- **Type**: `string`
- **Default:** `"http://localhost:3000"`

> Listening dev server URL.

This should not be set directly as it will always be overridden by the dev server with the full URL (for module and internal use).

devServerHandlers

- **Type**: `array`

> Nitro development-only server handlers.

See: https://nitro.unjs.io/guide/routing

devtools

- **Type**: `boolean`
- **Default:** `false`

> Enable Nuxt DevTools for development.

This is an experimental feature. Breaking changes for devtools might not reflect on the version of Nuxt.

See: Nuxt DevTools for more information.

dir

> Customize default directory structure used by Nuxt.

It is better to stick with defaults unless needed.

assets

- **Type**: `string`
- **Default:** `"assets"`

> The assets directory (aliased as `~assets` in your build).

layouts

- **Type**: `string`
- **Default:** `"layouts"`

> The layouts directory, each file of which will be auto-registered as a Nuxt layout.

middleware

- **Type**: `string`
- **Default:** `"middleware"`

> The middleware directory, each file of which will be auto-registered as a Nuxt middleware.

modules

- **Type**: `string`
- **Default:** `"modules"`

> The modules directory, each file in which will be auto-registered as a Nuxt module.

pages

- **Type**: `string`
- **Default:** `"pages"`

> The directory which will be processed to auto-generate your application page routes.

plugins

- **Type**: `string`
- **Default:** `"plugins"`

> The plugins directory, each file of which will be auto-registered as a Nuxt plugin.

public

- **Type**: `string`
- **Default:** `"public"`

> The directory containing your static files, which will be directly accessible via the Nuxt server and copied across into your `dist` folder when your app is generated.

static

- **Type**: `string`
- **Default:** `"public"`

experimental

appManifest

- **Type**: `boolean`
- **Default:** `false`

> Use app manifests to respect route rules on client-side.

asyncContext

- **Type**: `boolean`
- **Default:** `false`

> Enable native async context to be accessable for nested composables

See: https://github.com/nuxt/nuxt/pull/20918

asyncEntry

- **Type**: `boolean`
- **Default:** `false`

> Set to true to generate an async entry point for the Vue bundle (for module federation support).

clientFallback

- **Type**: `boolean`
- **Default:** `false`

> Whether to enable the experimental `<NuxtClientFallback>` component for rendering content on the client if there's an error in SSR.

componentIslands

- **Type**: `boolean`
- **Default:** `false`

> Experimental component islands support with and .island.vue files.

configSchema

- **Type**: `boolean`
- **Default:** `true`

> Config schema support

See: https://github.com/nuxt/nuxt/issues/15592

crossOriginPrefetch

- **Type**: `boolean`
- **Default:** `false`

> Enable cross-origin prefetch using the Speculation Rules API.

emitRouteChunkError

- **Type**: `string`
- **Default:** `"automatic"`

> Emit `app:chunkError` hook when there is an error loading vite/webpack chunks.

By default, Nuxt will also perform a hard reload of the new route when a chunk fails to load when navigating to a new route. You can disable automatic handling by setting this to `false`, or handle chunk errors manually by setting it to `manual`.

See: https://github.com/nuxt/nuxt/pull/19038

externalVue

- **Type**: `boolean`
- **Default:** `true`

> Externalize `vue`, `@vue/*` and `vue-router` when building.

See: https://github.com/nuxt/nuxt/issues/13632

headNext

- **Type**: `boolean`
- **Default:** `false`

> Use new experimental head optimisations: - Add the capo.js head plugin in order to render tags in of the head in a more performant way. - Uses the hash hydration plugin to reduce initial hydration

See: https://github.com/nuxt/nuxt/discussions/22632

inlineRouteRules

- **Type**: `boolean`
- **Default:** `false`

> Allow defining `routeRules` directly within your `~/pages` directory using `defineRouteRules`.

Rules are converted (based on the path) and applied for server requests. For example, a rule defined in `~/pages/foo/bar.vue` will be applied to `/foo/bar` requests. A rule in `~/pages/foo/[id].vue` will be

applied to `/foo/**` requests. For more control, such as if you are using a custom `path` or `alias` set in the page's `definePageMeta`, you should set `routeRules` directly within your `nuxt.config`.

inlineSSRStyles

- **Type**: `boolean`
- **Default:** `true`

> Inline styles when rendering HTML (currently vite only).

You can also pass a function that receives the path of a Vue component and returns a boolean indicating whether to inline the styles for that component.

localLayerAliases

- **Type**: `boolean`
- **Default:** `true`

> Resolve `~`, `~~`, `@` and `@@` aliases located within layers with respect to their layer source and root directories.

noScripts

- **Type**: `boolean`
- **Default:** `false`

> Turn off rendering of Nuxt scripts and JS resource hints. You can also disable scripts more granularly within `routeRules`.

noVueServer

- **Type**: `boolean`
- **Default:** `false`

> Disable vue server renderer endpoint within nitro.

payloadExtraction

- **Type**: `boolean`
- **Default:** `true`

> When this option is enabled (by default) payload of pages that are prerendered are extracted

polyfillVueUseHead

- **Type**: `boolean`
- **Default:** `false`

> Whether or not to add a compatibility layer for modules, plugins or user code relying on the old `@vueuse/head` API.

This can be disabled for most Nuxt sites to reduce the client-side bundle by ~0.5kb.

reactivityTransform

- **Type**: `boolean`
- **Default:** `false`

> Enable Vue's reactivity transform

See: https://vuejs.org/guide/extras/reactivity-transform.html

See: https://github.com/vuejs/rfcs/discussions/369#discussioncomment-5059028

renderJsonPayloads

- **Type**: `boolean`

- **Default:** `true`

> Render JSON payloads with support for revivifying complex types.

`respectNoSSRHeader`

- **Type**: `boolean`
- **Default:** `false`

> Allow disabling Nuxt SSR responses by setting the `x-nuxt-no-ssr` header.

`restoreState`

- **Type**: `boolean`
- **Default:** `false`

> Whether to restore Nuxt app state from `sessionStorage` when reloading the page after a chunk error or manual `reloadNuxtApp()` call.

To avoid hydration errors, it will be applied only after the Vue app has been mounted, meaning there may be a flicker on initial load. Consider carefully before enabling this as it can cause unexpected behavior, and consider providing explicit keys to `useState` as auto-generated keys may not match across builds.

`templateRouteInjection`

- **Type**: `boolean`
- **Default:** `true`

> By default the route object returned by the auto-imported `useRoute()` composable is kept in sync with the current page in view in `<NuxtPage>` . This is not true for `vue-router` 's exported `useRoute` or for the default `$route` object available in your Vue templates.

By enabling this option a mixin will be injected to keep the `$route` template object in sync with Nuxt's managed `useRoute()` .

treeshakeClientOnly

- **Type**: `boolean`
- **Default:** `true`

> Tree shakes contents of client-only components from server bundle.

See: https://github.com/nuxt/framework/pull/5750

typedPages

- **Type**: `boolean`
- **Default:** `false`

> Enable the new experimental typed router using unplugin-vue-router.

typescriptBundlerResolution

- **Type**: `boolean`
- **Default:** `false`

> This enables 'Bundler' module resolution mode for TypeScript, which is the recommended setting for frameworks like Nuxt and Vite.

It improves type support when using modern libraries with `exports`. This is only not enabled by default because it could be a breaking change for some projects. See https://github.com/microsoft/TypeScript/pull/51669

viewTransition

- **Type**: `boolean`
- **Default:** `false`

> Enable View Transition API integration with client-side router.

See: https://developer.chrome.com/docs/web-platform/view-transitions

watcher

- **Type**: `string`
- **Default:** `"chokidar-granular"`

> Set an alternative watcher that will be used as the watching service for Nuxt.

Nuxt uses 'chokidar-granular' by default, which will ignore top-level directories (like `node_modules` and `.git`) that are excluded from watching. You can set this instead to `parcel` to use `@parcel/watcher`, which may improve performance in large projects or on Windows platforms. You can also set this to `chokidar` to watch all files in your source directory.

See: https://github.com/paulmillr/chokidar**See**: https://github.com/parcel-bundler/watcher

writeEarlyHints

- **Type**: `boolean`
- **Default:** `false`

> Write early hints when using node server.

Note: nginx does not support 103 Early hints in the current version.

extends

- **Default:** `null`

> Extend project from multiple local or remote sources.

Value should be either a string or array of strings pointing to source directories or config path relative to current config. You can use `github:`, `gitlab:`, `bitbucket:` or `https://` to extend from a remote git repository.

extensions

- **Type**: `array`
- **Default**

```
[
  ".js",
  ".jsx",
  ".mjs",
  ".ts",
  ".tsx",
  ".vue"
]
```

> The extensions that should be resolved by the Nuxt resolver.

generate

`exclude`

- **Type**: `array`

> This option is no longer used. Instead, use `nitro.prerender.ignore`.

`routes`

- **Type**: `array`

> The routes to generate.

If you are using the crawler, this will be only the starting point for route generation. This is often necessary when using dynamic routes. It is preferred to use `nitro.prerender.routes`.

Example:

```
routes: ['/users/1', '/users/2', '/users/3']
```

hooks

- **Default:** `null`

> Hooks are listeners to Nuxt events that are typically used in modules, but are also available in `nuxt.config` .

Internally, hooks follow a naming pattern using colons (e.g., build:done). For ease of configuration, you can also structure them as an hierarchical object in `nuxt.config` (as below).

Example:

```
import fs from 'node:fs'
import path from 'node:path'
export default {
  hooks: {
    build: {
      done(builder) {
        const extraFilePath = path.join(
          builder.nuxt.options.buildDir,
          'extra-file'
        )
        fs.writeFileSync(extraFilePath, 'Something extra')
      }
    }
  }
}
```

ignore

- **Type**: `array`
- **Default**

```
[
  "**/*.stories.{js,cts,mts,ts,jsx,tsx}",
```

```
  "**/*.{spec,test}.{js,cts,mts,ts,jsx,tsx}",
  "**/*.d.{cts,mts,ts}",
  "**/.{pnpm-store,vercel,netlify,output,git,cache,data}",
  ".nuxt/analyze",
  ".nuxt",
  "**/-*.*"
]
```

> More customizable than `ignorePrefix`: all files matching glob patterns specified inside the `ignore` array will be ignored in building.

ignoreOptions

> Pass options directly to `node-ignore` (which is used by Nuxt to ignore files).

See: node-ignore**Example**:

```
ignoreOptions: {
  ignorecase: false
}
```

ignorePrefix

- **Type**: `string`
- **Default:** `"-"`

> Any file in `pages/`, `layouts/`, `middleware/` or `store/` will be ignored during building if its filename starts with the prefix specified by `ignorePrefix`.

imports

> Configure how Nuxt auto-imports composables into your application.

See: Nuxt 3 documentation

dirs

- **Type**: `array`

> An array of custom directories that will be auto-imported. Note that this option will not override the default directories (~/composables, ~/utils).

Example:

```
imports: {
  // Auto-import pinia stores defined in `~/stores`
  dirs: ['stores']
}
```

global

- **Type**: `boolean`
- **Default:** `false`

logLevel

- **Type**: `string`
- **Default:** `"info"`

> Log level when building logs.

Defaults to 'silent' when running in CI or when a TTY is not available. This option is then used as 'silent' in Vite and 'none' in Webpack

modules

- **Type**: `array`

> Modules are Nuxt extensions which can extend its core functionality and add endless integrations.

Each module is either a string (which can refer to a package, or be a path to a file), a tuple with the module as first string and the options as a second object, or an inline module function. Nuxt tries to resolve each item in the modules array using node require path (in `node_modules`) and then will be resolved from project `srcDir` if `~` alias is used.

Note: Modules are executed sequentially so the order is important.

Example:

```
modules: [
  // Using package name
  '@nuxtjs/axios',
  // Relative to your project srcDir
  '~/modules/awesome.js',
  // Providing options
  ['@nuxtjs/google-analytics', { ua: 'X1234567' }],
  // Inline definition
  function () {}
]
```

modulesDir

- **Type**: `array`
- **Default**

```
[
  "/<rootDir>/node_modules",
  "/Users/daniel/code/nuxt.js/packages/schema/node_modules"
]
```

> Used to set the modules directories for path resolving (for example, webpack's `resolveLoading`, `nodeExternals` and `postcss`).

The configuration path is relative to `options.rootDir` (default is current working directory). Setting this field may be necessary if your project is organized as a yarn workspace-styled mono-repository.

Example:

```
export default {
```

```
  modulesDir: ['../../node_modules']
}
```

nitro

> Configuration for Nitro.

See: https://nitro.unjs.io/config/

routeRules

- **Type**: `object`

optimization

> Build time optimization configuration.

asyncTransforms

> Options passed directly to the transformer from `unctx` that preserves async context after `await` .

asyncFunctions

- **Type**: `array`
- **Default**

```
[
  "defineNuxtPlugin",
  "defineNuxtRouteMiddleware"
]
```

objectDefinitions

defineNuxtComponent

- **Type**: `array`
- **Default**

```
[
  "asyncData",
  "setup"
]
```

defineNuxtPlugin

- **Type**: `array`
- **Default**

```
[
  "setup"
]
```

definePageMeta

- **Type**: `array`
- **Default**

```
[
  "middleware",
  "validate"
]
```

keyedComposables

- **Type**: `array`
- **Default**

```
[
  {
    "name": "defineNuxtComponent",
    "argumentLength": 2
  },
  {
    "name": "useState",
    "argumentLength": 2
  },
  {
    "name": "useFetch",
    "argumentLength": 3
  },
  {
    "name": "useAsyncData",
    "argumentLength": 3
  },
  {
    "name": "useLazyAsyncData",
    "argumentLength": 3
  },
  {
    "name": "useLazyFetch",
    "argumentLength": 3
  }
]
```

> Functions to inject a key for.

As long as the number of arguments passed to the function is less than `argumentLength`, an additional magic string will be injected that can be used to deduplicate requests between server and client. You will need to take steps to handle this additional key. The key will be unique based on the location of the function being invoked within the file.

treeShake

> Tree shake code from specific builds.

composables

> Tree shake composables from the server or client builds.

Example:

```
treeShake: { client: { myPackage: ['useServerOnlyComposable'] } }
```

client

- **Type**: `object`
- **Default**

```
{
  "vue": [
    "onServerPrefetch",
    "onRenderTracked",
    "onRenderTriggered"
  ],
  "#app": [
    "definePayloadReducer",
    "definePageMeta"
  ]
}
```

server

- **Type**: `object`
- **Default**

```
{
  "vue": [
    "onBeforeMount",
    "onMounted",
    "onBeforeUpdate",
    "onRenderTracked",
    "onRenderTriggered",
```

```
    "onActivated",
    "onDeactivated",
    "onBeforeUnmount"
  ],
  "#app": [
    "definePayloadReviver",
    "definePageMeta"
  ]
}
```

pages

- **Type**: `boolean`

> Whether to use the vue-router integration in Nuxt 3. If you do not provide a value it will be enabled if you have a `pages/` directory in your source folder.

plugins

- **Type**: `array`

> An array of nuxt app plugins.

Each plugin can be a string (which can be an absolute or relative path to a file). If it ends with `.client` or `.server` then it will be automatically loaded only in the appropriate context. It can also be an object with `src` and `mode` keys.

Note: Plugins are also auto-registered from the `~/plugins` directory and these plugins do not need to be listed in `nuxt.config` unless you need to customize their order. All plugins are deduplicated by their src path.

See: https://nuxt.com/docs/guide/directory-structure/plugins**Example**:

```
plugins: [
  '~/plugins/foo.client.js', // only in client side
  '~/plugins/bar.server.js', // only in server side
  '~/plugins/baz.js', // both client & server
  { src: '~/plugins/both-sides.js' },
  { src: '~/plugins/client-only.js', mode: 'client' }, // only on client side
```

```
  { src: '~/plugins/server-only.js', mode: 'server' } // only on server side
]
```

postcss

plugins

> Options for configuring PostCSS plugins.

https://postcss.org/

autoprefixer

> https://github.com/postcss/autoprefixer

cssnano

- **Type**: `boolean`
- **Default:** `true`

postcss-import

- **Type**: `object`

> https://github.com/postcss/postcss-import

postcss-url

> https://github.com/postcss/postcss-url

rootDir

- **Type**: `string`
- **Default:** `"/<rootDir>"`

> Define the root directory of your application.

This property can be overwritten (for example, running `nuxt ./my-app/` will set the `rootDir` to the absolute path of `./my-app/` from the current/working directory. It is normally not needed to configure this option.

routeRules

> Global route options applied to matching server routes.

Experimental: This is an experimental feature and API may change in the future. **See**: https://nitro.unjs.io/config/#routerules

router

options

> Additional options passed to `vue-router`.

Note: Only JSON serializable options should be passed by nuxt config. For more control, you can use `app/router.options.ts` file.

See: documentation.

runtimeConfig

- **Type**: `object`
- **Default**

```
{
  "public": {},
  "app": {
    "baseURL": "/",
    "buildAssetsDir": "/_nuxt/",
    "cdnURL": ""
  }
}
```

> Runtime config allows passing dynamic config and environment variables to the Nuxt app context.

The value of this object is accessible from server only using `useRuntimeConfig`. It mainly should hold *private* configuration which is not exposed on the frontend. This could include a reference to your API secret tokens. Anything under `public` and `app` will be exposed to the frontend as well. Values are automatically replaced by matching env variables at runtime, e.g. setting an environment variable `NUXT_API_KEY=my-api-key NUXT_PUBLIC_BASE_URL=/foo/` would overwrite the two values in the example below.

Example:

```
export default {
 runtimeConfig: {
    apiKey: '' // Default to an empty string, automatically set at runtime using process.env.NUXT_
    public: {
       baseURL: '' // Exposed to the frontend as well.
    }
  }
}
```

serverDir

- **Type**: `string`
- **Default:** `"/<rootDir>/server"`

> Define the server directory of your Nuxt application, where Nitro routes, middleware and plugins are kept.

If a relative path is specified, it will be relative to your `rootDir`.

serverHandlers

- **Type**: `array`

> Nitro server handlers.

Each handler accepts the following options: - handler: The path to the file defining the handler. - route: The route under which the handler is available. This follows the conventions of https://github.com/unjs/radix3. - method: The HTTP method of requests that should be handled. - middleware: Specifies whether it is a middleware handler. - lazy: Specifies whether to use lazy loading to import the handler.

See: https://nuxt.com/docs/guide/directory-structure/server

Note: Files from `server/api`, `server/middleware` and `server/routes` will be automatically registered by Nuxt.

Example:

```
serverHandlers: [
  { route: '/path/foo/**:name', handler: '~/server/foohandler.ts' }
]
```

sourcemap

- **Type**: `object`
- **Default**

```
{
  "server": true,
  "client": false
}
```

> Whether to generate sourcemaps.

spaLoadingTemplate

- **Default:** `null`

> Boolean or a path to an HTML file with the contents of which will be inserted into any HTML page rendered with `ssr: false` . - If it is unset, it will use `~/app/spa-loading-template.html` if it exists. - If it is false, no SPA loading indicator will be loaded. - If true, Nuxt will look for `~/app/spa-loading-template.html` file or a default Nuxt image will be used.

Some good sources for spinners are SpinKit or SVG Spinners.

Example: ~/app/spa-loading-template.html

```
<!-- https://github.com/barelyhuman/snips/blob/dev/pages/css-loader.md -->
<div class="loader"></div>
<style>
.loader {
  display: block;
  position: fixed;
  z-index: 1031;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 18px;
  height: 18px;
  box-sizing: border-box;
  border: solid 2px transparent;
  border-top-color: #000;
  border-left-color: #000;
  border-bottom-color: #efefef;
  border-right-color: #efefef;
  border-radius: 50%;
  -webkit-animation: loader 400ms linear infinite;
  animation: loader 400ms linear infinite;
}

\@-webkit-keyframes loader {
  0% {
    -webkit-transform: translate(-50%, -50%) rotate(0deg);
  }
  100% {
    -webkit-transform: translate(-50%, -50%) rotate(360deg);
  }
}
\@keyframes loader {
  0% {
    transform: translate(-50%, -50%) rotate(0deg);
```

```
  }
  100% {
    transform: translate(-50%, -50%) rotate(360deg);
  }
}
</style>
```

srcDir

- **Type**: `string`
- **Default:** `"/<rootDir>"`

> Define the source directory of your Nuxt application.

If a relative path is specified, it will be relative to the `rootDir`.

Example:

```
export default {
  srcDir: 'src/'
}
```

This would work with the following folder structure:

```
> -| app/
> ---| node_modules/
> ---| nuxt.config.js
> ---| package.json
> ---| src/
> ------| assets/
> ------| components/
> ------| layouts/
> ------| middleware/
> ------| pages/
> ------| plugins/
> ------| static/
> ------| store/
> ------| server/
> ------| app.config.ts
> ------| app.vue
> ------| error.vue
```

ssr

- **Type**: `boolean`
- **Default:** `true`

> Whether to enable rendering of HTML - either dynamically (in server mode) or at generate time. If set to `false` generated pages will have no content.

telemetry

> Manually disable nuxt telemetry.

See: Nuxt Telemetry for more information.

test

- **Type**: `boolean`
- **Default:** `false`

> Whether your app is being unit tested.

theme

- **Type**: `string`
- **Default:** `null`

> Extend project from a local or remote source.

Value should be a string pointing to source directory or config path relative to current config. You can use `github:` , `gitlab:` , `bitbucket:` or `https://` to extend from a remote git repository.

typescript

> Configuration for Nuxt's TypeScript integration.

builder

- **Default:** `null`

> Which builder types to include for your project.

By default Nuxt infers this based on your `builder` option (defaulting to 'vite') but you can either turn off builder environment types (with `false`) to handle this fully yourself, or opt for a 'shared' option. The 'shared' option is advised for module authors, who will want to support multiple possible builders.

includeWorkspace

- **Type**: `boolean`
- **Default:** `false`

> Include parent workspace in the Nuxt project. Mostly useful for themes and module authors.

shim

- **Type**: `boolean`
- **Default:** `true`

> Generate a `*.vue` shim.

We recommend instead either enabling **Take Over Mode** or adding TypeScript Vue Plugin (Volar)** 👉 [Download].

strict

- **Type**: `boolean`
- **Default:** `true`

> TypeScript comes with certain checks to give you more safety and analysis of your program. Once you've converted your codebase to TypeScript, you can start enabling these checks for greater safety. Read More

tsConfig

- **Type**: `object`
- **Default**

```
{
  "compilerOptions": {}
}
```

> You can extend generated `.nuxt/tsconfig.json` using this option.

typeCheck

- **Type**: `boolean`
- **Default:** `false`

> Enable build-time type checking.

If set to true, this will type check in development. You can restrict this to build-time type checking by setting it to `build`. Requires to install `typescript` and `vue-tsc` as dev dependencies.

See: https://nuxt.com/docs/guide/concepts/typescript

vite

> Configuration that will be passed directly to Vite.

See https://vitejs.dev/config for more information. Please note that not all vite options are supported in Nuxt.

build

assetsDir

- **Type**: `string`
- **Default:** `"_nuxt/"`

emptyOutDir

- **Type**: `boolean`
- **Default:** `false`

clearScreen

- **Type**: `boolean`
- **Default:** `true`

define

- **Type**: `object`
- **Default**

```
{
  "process.dev": false,
  "import.meta.dev": false,
  "process.test": false,
  "import.meta.test": false
}
```

esbuild

jsxFactory

- **Type**: `string`
- **Default:** `"h"`

jsxFragment

- **Type**: `string`
- **Default:** `"Fragment"`

tsconfigRaw

- **Type**: `string`
- **Default:** `"{}"`

mode

- **Type**: `string`
- **Default:** `"production"`

optimizeDeps

exclude

- **Type**: `array`
- **Default**

```
[
  "vue-demi"
]
```

publicDir

- **Type**: `string`
- **Default:** `"/<rootDir>/public"`

resolve

extensions

- **Type**: `array`
- **Default**

```
[
  ".mjs",
  ".js",
  ".ts",
```

```
  ".jsx",
  ".tsx",
  ".json",
  ".vue"
]
```

root

- **Type**: `string`
- **Default:** `"/<rootDir>"`

server

fs

allow

- **Type**: `array`
- **Default**

```
[
  "/<rootDir>/.nuxt",
  "/<rootDir>",
  "/<rootDir>",
  "/<rootDir>",
  "/<rootDir>/node_modules",
  "/Users/daniel/code/nuxt.js/packages/schema/node_modules"
]
```

vue

isProduction

- **Type**: `boolean`
- **Default:** `true`

script

defineModel

- **Type**: `boolean`
- **Default:** `false`

propsDestructure

- **Type**: `boolean`
- **Default:** `false`

template

compilerOptions

- **Type**: `object`

vueJsx

- **Type**: `object`

vue

> Vue.js config

compilerOptions

> Options for the Vue compiler that will be passed at build time.

See: documentation

defineModel

- **Type**: `boolean`
- **Default:** `false`

> Vue Experimental: Enable macro `defineModel`

See: Vue RFC#503

propsDestructure

- **Type**: `boolean`
- **Default:** `false`

> Vue Experimental: Enable reactive destructure for `defineProps`

See: Vue RFC#502

runtimeCompiler

- **Type**: `boolean`
- **Default:** `false`

> Include Vue compiler in runtime bundle.

watch

- **Type**: `array`

> The watch property lets you define patterns that will restart the Nuxt dev server when changed.

It is an array of strings or regular expressions. Strings should be either absolute paths or relative to the `srcDir` (and the `srcDir` of any layers). Regular expressions will be matched against the path relative to the project `srcDir` (and the `srcDir` of any layers).

watchers

> The watchers property lets you overwrite watchers configuration in your `nuxt.config`.

chokidar

> Options to pass directly to `chokidar`.

See: chokidar

ignoreInitial

- **Type**: `boolean`
- **Default:** `true`

rewatchOnRawEvents

> An array of event types, which, when received, will cause the watcher to restart.

webpack

> `watchOptions` to pass directly to webpack.

See: webpack@4 watch options.

aggregateTimeout

- **Type**: `number`
- **Default:** `1000`

webpack

aggressiveCodeRemoval

- **Type**: `boolean`
- **Default:** `false`

> Hard-replaces `typeof process`, `typeof window` and `typeof document` to tree-shake bundle.

analyze

- **Type**: `boolean`
- **Default:** `false`

> Nuxt uses `webpack-bundle-analyzer` to visualize your bundles and how to optimize them.

Set to `true` to enable bundle analysis, or pass an object with options: for webpack or for vite.

Example:

```
analyze: {
  analyzerMode: 'static'
}
```

cssSourceMap

- **Type**: `boolean`
- **Default:** `false`

> Enables CSS source map support (defaults to `true` in development).

devMiddleware

> See webpack-dev-middleware for available options.

stats

- **Type**: `string`
- **Default:** `"none"`

experiments

> Configure webpack experiments

extractCSS

- **Type**: `boolean`
- **Default:** `true`

> Enables Common CSS Extraction.

Using mini-css-extract-plugin under the hood, your CSS will be extracted into separate files, usually one per component. This allows caching your CSS and JavaScript separately.

Example:

```
export default {
  webpack: {
    extractCSS: true,
    // or
```

```
    extractCSS: {
      ignoreOrder: true
    }
  }
}
```

Example:

```
export default {
  webpack: {
    extractCSS: true,
    optimization: {
      splitChunks: {
        cacheGroups: {
          styles: {
            name: 'styles',
            test: /\.(css|vue)$/,
            chunks: 'all',
            enforce: true
          }
        }
      }
    }
  }
}
```

filenames

> Customize bundle filenames.

To understand a bit more about the use of manifests, take a look at this webpack documentation.

Note: Be careful when using non-hashed based filenames in production as most browsers will cache the asset and not detect the changes on first load.

Example:

```
filenames: {
  chunk: ({ isDev }) => (isDev ? '[name].js' : '[id].[contenthash].js')
}
```

app

- **Type**: `function`

chunk

- **Type**: `function`

css

- **Type**: `function`

font

- **Type**: `function`

img

- **Type**: `function`

video

- **Type**: `function`

friendlyErrors

- **Type**: `boolean`
- **Default:** `true`

> Set to `false` to disable the overlay provided by FriendlyErrorsWebpackPlugin.

hotMiddleware

> See webpack-hot-middleware for available options.

loaders

> Customize the options of Nuxt's integrated webpack loaders.

css

esModule

- **Type**: `boolean`
- **Default:** `false`

importLoaders

- **Type**: `number`
- **Default:** `0`

url

filter

- **Type**: `function`

cssModules

esModule

- **Type**: `boolean`
- **Default:** `false`

`importLoaders`

- **Type**: `number`
- **Default:** `0`

`modules`

`localIdentName`

- **Type**: `string`
- **Default:** `"[local]_[hash:base64:5]"`

`url`

`filter`

- **Type**: `function`

`esbuild`

> See https://github.com/esbuild-kit/esbuild-loader

`file`

> See: https://github.com/webpack-contrib/file-loader#options

Default:

```
{ esModule: false }
```

esModule

- **Type**: `boolean`
- **Default:** `false`

fontUrl

> See: https://github.com/webpack-contrib/file-loader#options

Default:

```
{ esModule: false, limit: 1000  }
```

esModule

- **Type**: `boolean`
- **Default:** `false`

limit

- **Type**: `number`
- **Default:** `1000`

imgUrl

> See: https://github.com/webpack-contrib/file-loader#options

Default:

```
{ esModule: false, limit: 1000  }
```

esModule

- **Type**: `boolean`
- **Default:** `false`

limit

- **Type**: `number`
- **Default:** `1000`

less

- **Default**

```
{
  "sourceMap": false
}
```

> See: https://github.com/webpack-contrib/less-loader#options

pugPlain

> See: https://pugjs.org/api/reference.html#options

sass

> See: https://github.com/webpack-contrib/sass-loader#options

Default:

```
{
  sassOptions: {
    indentedSyntax: true
  }
}
```

sassOptions

indentedSyntax

- **Type**: `boolean`
- **Default:** `true`

scss

- **Default**

```
{
  "sourceMap": false
}
```

> See: https://github.com/webpack-contrib/sass-loader#options

stylus

- **Default**

```
{
  "sourceMap": false
}
```

> See: https://github.com/webpack-contrib/stylus-loader#options

vue

> See vue-loader for available options.

`compilerOptions`

- **Type**: `object`

`defineModel`

- **Type**: `boolean`
- **Default:** `false`

`propsDestructure`

- **Type**: `boolean`
- **Default:** `false`

`transformAssetUrls`

`embed`

- **Type**: `string`
- **Default:** `"src"`

`object`

- **Type**: `string`
- **Default:** `"src"`

source

- **Type**: `string`
- **Default:** `"src"`

video

- **Type**: `string`
- **Default:** `"src"`

vueStyle

- **Default**

```
{
  "sourceMap": false
}
```

optimization

> Configure webpack optimization.

minimize

- **Type**: `boolean`
- **Default:** `true`

> Set minimize to `false` to disable all minimizers. (It is disabled in development by default).

minimizer

> You can set minimizer to a customized array of plugins.

`runtimeChunk`

- **Type**: `string`
- **Default:** `"single"`

`splitChunks`

`automaticNameDelimiter`

- **Type**: `string`
- **Default:** `"/"`

`cacheGroups`

`chunks`

- **Type**: `string`
- **Default:** `"all"`

`optimizeCSS`

- **Type**: `boolean`
- **Default:** `false`

> OptimizeCSSAssets plugin options.

Defaults to true when `extractCSS` is enabled.

See: css-minimizer-webpack-plugin documentation.

`plugins`

- **Type**: `array`

> Add webpack plugins.

Example:

```
import webpack from 'webpack'
import { version } from './package.json'
// ...
plugins: [
  new webpack.DefinePlugin({
    'process.VERSION': version
  })
]
```

postcss

> Customize PostCSS Loader. Same options as https://github.com/webpack-contrib/postcss-loader#options

postcssOptions

config

plugins

- **Type**: `object`
- **Default**

```
{
  "postcss-import": {},
  "postcss-url": {},
  "autoprefixer": {},
  "cssnano": true
}
```

profile

- **Type**: `boolean`
- **Default:** `false`

> Enable the profiler in webpackbar.

It is normally enabled by CLI argument `--profile`.

See: webpackbar.



- **Type**: `string`
- **Default:** `"url"`

> The polyfill library to load to provide URL and URLSearchParams.

Defaults to `'url'` (see package).

warningIgnoreFilters

- **Type**: `array`

> Filters to hide build warnings.

workspaceDir

- **Type**: `string`
- **Default:** `"/<rootDir>"`

> Define the workspace directory of your application.

Often this is used when in a monorepo setup. Nuxt will attempt to detect your workspace directory automatically, but you can override it here. It is normally not needed to configure this option.

Edit on Github