Experimental Features

The Nuxt experimental features can be enabled in the Nuxt configuration file. Internally, Nuxt uses `@nuxt/schema` to define these experimental features. You can refer to the API documentation or the source code for more information.

> 💡 Note that these features are experimental and could be removed or modified in the future.

asyncContext

Enable native async context to be accessable for nested composables in Nuxt and in Nitro. See full explanation in #20918.

nuxt.config.ts

```ts
export defineNuxtConfig({ experimental: { asyncContext: true } })
```

asyncEntry

Enables generation of an async entry point for the Vue bundle, aiding module federation support.

nuxt.config.ts

```ts
export defineNuxtConfig({ experimental: { asyncEntry: true } })
```

reactivityTransform

Enables Vue's reactivity transform. Note that this feature has been marked as deprecated in Vue 3.3 and is planned to be removed from core in Vue 3.4.

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { reactivityTransform: true } })
```

👉 Read more in Docs > Getting Started > Configuration #enabling Experimental Vue Features.

externalVue

Externalizes `vue`, `@vue/*` and `vue-router` when building. *Enabled by default.*

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { externalVue: true } })
```

⚠️ This feature will likely be removed in Nuxt 3.6.

treeshakeClientOnly

Tree shakes contents of client-only components from server bundle. *Enabled by default.*

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { treeshakeClientOnly: true } })
```

emitRouteChunkError

Emits `app:chunkError` hook when there is an error loading vite/webpack chunks. Default behavior is to perform a hard reload of the new route when a chunk fails to load. You can disable automatic handling by setting this to `false`, or handle chunk errors manually by setting it to `manual`.

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { emitRouteChunkError: 'automatic' } }) // or 'manual' or
```

restoreState

Allows Nuxt app state to be restored 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.

nuxt.config.ts

```ts
export defineNuxtConfig({ experimental: { restoreState: true } })
```

inlineSSRStyles

Inlines styles when rendering HTML. This is currently available only when using Vite. 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.

nuxt.config.ts

```ts
export defineNuxtConfig({ experimental: { inlineSSRStyles: true } }) // or a function to determine
```

noScripts

Disables rendering of Nuxt scripts and JS resource hints. Can also be configured granularly within `routeRules`.

nuxt.config.ts

```ts
export defineNuxtConfig({ experimental: { noScripts: true } })
```

renderJsonPayloads

Allows rendering of JSON payloads with support for revivifying complex types. *Enabled by default.*

```ts
export defineNuxtConfig({ experimental: { renderJsonPayloads: true } })
```
nuxt.config.ts

noVueServer

Disables Vue server renderer endpoint within Nitro.

```ts
export defineNuxtConfig({ experimental: { noVueServer: true } })
```
nuxt.config.ts

payloadExtraction

Enables extraction of payloads of pages generated with `nuxt generate` .

```ts
export defineNuxtConfig({ experimental: { payloadExtraction: true } })
```
nuxt.config.ts

clientFallback

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

```ts
export defineNuxtConfig({ experimental: { clientFallback: true } })
```
nuxt.config.ts

crossOriginPrefetch

Enables cross-origin prefetch using the Speculation Rules API.

```ts
export defineNuxtConfig({ experimental: { crossOriginPrefetch: true } })
```
nuxt.config.ts

viewTransition

Enables View Transition API integration with client-side router.

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { viewTransition: true } })
```

👉 Read more in Docs > Getting Started > Transitions #view Transitions API Experimental.

writeEarlyHints

Enables writing of early hints when using node server.

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { writeEarlyHints: true } })
```

componentIslands

Enables experimental component islands support with `<NuxtIsland>` and `.island.vue` files.

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { componentIslands: true } })
```

👉 Read more in Docs > Guide > Directory Structure > Components #server Components.

You can follow the server components roadmap on GitHub.

configSchema

Enables config schema support. *Enabled by default.*

```ts [nuxt.config.ts]
export defineNuxtConfig({ experimental: { configSchema: true } })
```

polyfillVueUseHead

Adds a compatibility layer for modules, plugins, or user code relying on the old `@vueuse/head` API.

```nuxt.config.ts
export defineNuxtConfig({ experimental: { polyfillVueUseHead: false } })
```

respectNoSSRHeader

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

```nuxt.config.ts
export defineNuxtConfig({ experimental: { respectNoSSRHeader: false } })
```

localLayerAliases

Resolve `~`, `~~`, `@` and `@@` aliases located within layers with respect to their layer source and root directories. *Enabled by default.*

```nuxt.config.ts
export defineNuxtConfig({ experimental: { localLayerAliases: true } })
```

typedPages

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

```nuxt.config.ts
export defineNuxtConfig({ experimental: { typedPages: false } })
```

Out of the box, this will enable typed usage of `navigateTo`, `<NuxtLink>`, `router.push()` and more. You can even get typed params within a page by using `const route = useRoute('route-name')`.

watcher

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.

nuxt.config.ts

```ts
export defineNuxtConfig({
    experimental: {
        watcher: 'chokidar-granular' // 'chokidar' or 'parcel' are also options
    }
})
```

Edit on Github