Table of Contents >

Legacy Composition API

Nuxt Bridge provides access to Composition API syntax. It is specifically designed to be aligned with Nuxt 3. Because of this, there are a few extra steps to take when enabling Nuxt Bridge, if you have been using the Composition API previously.

Remove Modules

- Remove `@vue/composition-api` from your dependencies.
- Remove `@nuxtjs/composition-api` from your dependencies (and from your modules in `nuxt.config`).

Using `@vue/composition-api`

If you have been using just `@vue/composition-api` and not `@nuxtjs/composition-api` , then things are very straightforward.

1. First, remove the plugin where you are manually registering the Composition API. Nuxt Bridge will handle this for you.

   ```diff
   - import Vue from 'vue'
   - import VueCompositionApi from '@vue/composition-api'
   -
   - Vue.use(VueCompositionApi)
   ```

2. Otherwise, there is nothing you need to do. However, if you want, you can remove your explicit imports from `@vue/composition-api` and rely on Nuxt Bridge auto-importing them for you.

Migrating from `@nuxtjs/composition-api`

Nuxt Bridge implements the Composition API slightly differently from `@nuxtjs/composition-api` and provides different composables (designed to be aligned with the composables that Nuxt 3 provides).

Because some composables have been removed and don't yet have a replacement, this will be a slightly more complicated process.

Remove `@nuxtjs/composition-api/module` from your buildModules

You don't have to immediately update your imports yet - Nuxt Bridge will automatically provide a 'shim' for most imports you currently have, to give you time to migrate to the new, Nuxt 3-compatible composables, with the following exceptions:

- `withContext` has been removed. See below.
- `useStatic` has been removed. There is no current replacement. Feel free to raise a discussion if you have a use case for this.
- `reqRef` and `reqSsrRef` , which were deprecated, have now been removed entirely. Follow the instructions below regarding ssrRef to replace this.

Set `bridge.capi`

```
import { defineNuxtConfig } from '@nuxt/bridge'

export default defineNuxtConfig({
  bridge: {
    capi: true,
    nitro: false // If migration to Nitro is complete, set to true
  }
})
```

For each other composable you are using from `@nuxtjs/composition-api` , follow the steps below.

useFetch

`$fetchState` and `$fetch` have been removed.

```
const {
- $fetch,
- $fetchState,
+ fetch,
+ fetchState,
} = useFetch(() => { posts.value = await $fetch('/api/posts') })
```

defineNuxtMiddleware

This was a type-helper stub function that is now removed.

Remove the `defineNuxtMiddleware` wrapper:

```
- import { defineNuxtMiddleware } from '@nuxtjs/composition-api`
- export default defineNuxtMiddleware((ctx) => {})
+ export default (ctx) => {}
```

For typescript support, you can use `@nuxt/types` :

```
import type { Middleware } from '@nuxt/types'

export default <Middleware> function (ctx) { }
```

defineNuxtPlugin

This was a type-helper stub function that is now removed.

You may also keep using Nuxt 2-style plugins, by removing the function (as with defineNuxtMiddleware).

Remove the `defineNuxtPlugin` wrapper:

```
- import { defineNuxtPlugin } from '@nuxtjs/composition-api'
```

```diff
- export default defineNuxtPlugin((ctx, inject) => {})
+ export default (ctx, inject) => {}
```

For typescript support, you can use `@nuxt/types` :

```ts
import type { Plugin } from '@nuxt/types'

export default <Plugin> function (ctx, inject) {}
```

`useRouter` and `useRoute`

Nuxt Bridge provides direct replacements for these composables via `useRouter` and `useRoute` .

The only key difference is that `useRoute` no longer returns a computed property.

```diff
- import { useRouter, useRoute } from '@nuxtjs/composition-api'

  const router = useRouter()
  const route = useRoute()

- console.log(route.value.path)
+ console.log(route.path)
```

Edit on Github