Dep Optimization Options

- optimizeDeps.entries
- optimizeDeps.exclude
- optimizeDeps.include
- optimizeDeps.rolldownOptions
- optimizeDeps.esbuildOptions
- optimizeDeps.force
- optimizeDeps.noDiscovery
- optimizeDeps.holdUntilCrawlEnd
- optimizeDeps.disabled
- optimizeDeps.needsInterop

- **Related:** Dependency Pre-Bundling

Unless noted, the options in this section are only applied to the dependency optimizer, which is only used in dev.

optimizeDeps.entries

- **Type:** `string | string[]`

By default, Vite will crawl all your `.html` files to detect dependencies that need to be pre-bundled (ignoring `node_modules`, `build.outDir`, `__tests__` and `coverage`). If the top-level `input` or `build.rolldownOptions.input` is specified, Vite will crawl those entry points instead.

If neither of these fit your needs, you can specify custom entries using this option - the value should be a `tinyglobby` pattern or array of patterns that are relative from Vite project root. This will overwrite default entries inference. Only `node_modules` and `build.outDir` folders will be ignored by default when `optimizeDeps.entries` is explicitly defined. If other folders need to be ignored, you can use an ignore pattern as part of the entries list, marked with an initial `!`. `node_modules` will not be ignored for patterns that explicitly include the string `node_modules`.

optimizeDeps.exclude

- **Type:** `string[]`

Dependencies to exclude from pre-bundling.

:::warning CommonJS CommonJS dependencies should not be excluded from optimization. If an ESM dependency is excluded from optimization, but has a nested CommonJS dependency, the CommonJS dependency should be added to `optimizeDeps.include`. Example:

```
import { defineConfig } from 'vite'
// ---cut---
export default defineConfig({
  optimizeDeps: {
    include: ['esm-dep > cjs-dep'],
  },
})
```

:::

optimizeDeps.include

- **Type:** `string[]`

By default, linked packages not inside `node_modules` are not pre-bundled. Use this option to force a linked package to be pre-bundled.

Experimental: If you're using a library with many deep imports, you can also specify a trailing glob pattern to pre-bundle all deep imports at once. This will avoid constantly pre-bundling whenever a new deep import is used. Give Feedback. For example:

```
import { defineConfig } from 'vite'
// ---cut---
export default defineConfig({
  optimizeDeps: {
    include: ['my-lib/components/**/*.vue'],
  },
})
```

optimizeDeps.rolldownOptions

- **Type:** `Omit<RolldownOptions, 'input' | 'logLevel' | 'output'> & { output?: Omit<RolldownOutputOptions, 'format' | 'sourcemap' | 'dir' | 'banner'> }`

Options to pass to Rolldown during the dep scanning and optimization.

Certain options are omitted since changing them would not be compatible with Vite's dep optimization.

- `plugins` are merged with Vite's dep plugin

optimizeDeps.esbuildOptions

- **Type:** `Omit<EsbuildBuildOptions, 'bundle' | 'entryPoints' | 'external' | 'write' | 'watch' | 'outdir' | 'outfile' | 'outbase' | 'outExtension' | 'metafile'>`
- **Deprecated**

This option is converted to `optimizeDeps.rolldownOptions` internally. Use `optimizeDeps.rolldownOptions` instead.

optimizeDeps.force

- **Type:** `boolean`

Set to `true` to force dependency pre-bundling, ignoring previously cached optimized dependencies.

optimizeDeps.noDiscovery

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

When set to `true`, automatic dependency discovery will be disabled and only dependencies listed in `optimizeDeps.include` will be optimized. CJS-only dependencies must be present in `optimizeDeps.include` during dev.

optimizeDeps.holdUntilCrawlEnd

- **Experimental:** Give Feedback
- **Type:** `boolean`
- **Default:** `true`

When enabled, it will hold the first optimized deps results until all static imports are crawled on cold start. This avoids the need for full-page reloads when new dependencies are discovered and they trigger the generation of new common chunks. If all dependencies are found by the scanner plus the explicitly defined ones in `include`, it is better to disable this option to let the browser process more requests in parallel.

optimizeDeps.disabled

- **Deprecated**
- **Experimental:** Give Feedback
- **Type:** `boolean | 'build' | 'dev'`
- **Default:** `'build'`

This option is deprecated. As of Vite 5.1, pre-bundling of dependencies during build have been removed. Setting `optimizeDeps.disabled` to `true` or `'dev'` disables the optimizer, and configured to `false` or `'build'` leaves the optimizer during dev enabled.

To disable the optimizer completely, use `optimizeDeps.noDiscovery: true` to disallow automatic discovery of dependencies and leave `optimizeDeps.include` undefined or empty.

:::warning Optimizing dependencies during build time was an **experimental** feature. Projects trying out this strategy also removed `@rollup/plugin-commonjs` using `build.commonjsOptions: { include: [] }`. If you did so, a warning will guide you to re-enable it to support CJS only packages while bundling. :::

optimizeDeps.needsInterop

- **Experimental**
- **Type:** `string[]`

Forces ESM interop when importing these dependencies. Vite is able to properly detect when a dependency needs interop, so this option isn't generally needed. However, different combinations of dependencies could cause some of them to be prebundled differently. Adding these packages to `needsInterop` can speed up cold start by avoiding full-page reloads. You'll receive a warning if this

is the case for one of your dependencies, suggesting to add the package name to this array in your config.

□□□ #2
□ □□□ □□□ 2026-10-31 13:13:28 UTC
□ □□□ □□□ 2026-10-31 13:13:29 UTC