Production Deployment

Development vs. Production

During development, Vue provides a number of features to improve the development experience:

- Warning for common errors and pitfalls
- Props / events validation
- Reactivity debugging hooks
- Devtools integration

However, these features become useless in production. Some of the warning checks can also incur a small amount of performance overhead. When deploying to production, we should drop all the unused, development-only code branches for smaller payload size and better performance.

Without Build Tools

If you are using Vue without a build tool by loading it from a CDN or self-hosted script, make sure to use the production build (dist files that end in `.prod.js`) when deploying to production. Production builds are pre-minified with all development-only code branches removed.

- If using global build (accessing via the `Vue` global): use `vue.global.prod.js`.
- If using ESM build (accessing via native ESM imports): use `vue.esm-browser.prod.js`.

Consult the dist file guide for more details.

With Build Tools

If using a custom setup, make sure that:

1. `vue` resolves to `vue.runtime.esm-bundler.js`.
2. The compile time feature flags are properly configured.
3. `process.env.NODE_ENV` is replaced with `"production"` during build.

Additional references:

- Vite production build guide
- Vite deployment guide
- Vue CLI deployment guide

Tracking Runtime Errors

The app-level error handler can be used to report errors to tracking services:

```js
import { createApp } from 'vue'

const app = createApp(...)

app.config.errorHandler = (err, instance, info) => {
  // report error to tracking services
}
```

Services such as Sentry and Bugsnag also provide official integrations for Vue.

Edit this page on GitHub

Previous
Server-Side Rendering (SSR)

Next
Performance