

9. Using Vue in Markdown by Vicente default theme

- 9. Using Vue in Markdown by Vicente default theme
 - new! 9.1. Browser API Access Restrictions
 - 9.2. Browser API Access Restrictions
 - 9.3. Browser API Access Restrictions
 - 9.4. Browser API Access Restrictions
 - 9.5. Browser API Access Restrictions
 - 9.6. Browser API Access Restrictions
 - 9.7. Browser API Access Restrictions

new! 9.1. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#) [🔗]. In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

java.lang.ProcessBuilder	java.lang.Process
<ul style="list-style-type: none">- directory: File- redirectErrorStream: boolean+ ProcessBuilder(command: String[])+ ProcessBuilder(command: List<String>)+ command(): List+ command(command: List<String>): ProcessBuilder+ command(command: String[]): ProcessBuilder+ directory(): File+ directory(directory: File): ProcessBuilder+ environment(): Map+ redirectErrorStream(): boolean+ redirectErrorStream(redirectErrorStream: boolean): ProcessBuilder+ start(): Process	<ul style="list-style-type: none">+ destroy()+ exitValue(): int+ getErrorStream(): InputStream+ getInputStream(): InputStream+ getOutputStream(): OutputStream+ waitFor(): int

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PHP Example	Java Title	
1	# Post Here Code Example	php
1	// Post Here Code Example	java

9.2. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#) . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

PRUEBA

This is a tip

9.3. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#) . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.4. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#) . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.5. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#) . In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.6. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#). In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

9.7. Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the [universal code requirements](#). In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component: