

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.

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