

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

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: