



# Programación de Servicios y Procesos

Apuntes de clase

[Acceder al curso →](#)

---

Excepto donde se indique lo contrario, el contenido de este sitio ([Apuntes de PSP](#)) creado por Vicente Martínez está bajo una licencia [CC BY-NC-SA 4.0](#) 

---

© 2020 / 2021

# Introduction

VuePress is composed of two parts: a [minimalistic static site generator](#) with a Vue-powered [theming system](#) and [Plugin API](#), and a [default theme](#) optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.



# Programación de Servicios y Procesos

## Tema 3 - Hilos

### Threads y Runnable

---



Apuntes de PSP creados por Vicente Martínez están bajo una licencia [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/) 

## 9. Using Vue in Markdown by Vicente

default theme

- 9. Using Vue in Markdown by Vicente default theme
  - 9.1. Browser API Access Restrictions new!
  - 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

### 9.1. Browser API Access Restrictions new!

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

Pasaremos los datos necesarios para construir un objeto ISBN al constructor de Libro (en nuestro caso una cadena con el isbn) y será este quien lo cree. En el fondo esto es lo que sucede con un tipo valor como `DateTime` por su 'naturaleza'. De esta manera, además, el objeto `Isbn13` que estoy creando desaparecerá junto con el libro.

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: ❤

```

1  # Post Here Code Example
2  function base(a, b)
3  {
4      print "hola";
5  }
```

php

**PHP Example** **Java Title**

```
1      # Post Here Code Example
```

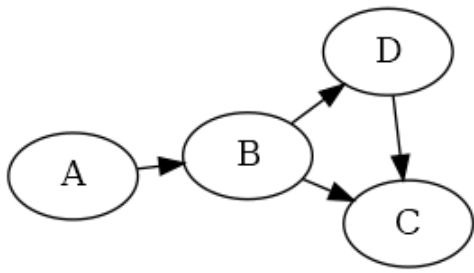
php

```
1      // Post Here Code Examples
```

java

```
1      ---
2      home: true
3      heroImage: /Logo_PSP.png
4      tagline: VuePress & Markdown
5      actionText: Access course →
6      actionLink: /unit1/
7      features:
8      - title: Feature 1 Título
9        details: Feature 1 Description
10     - title: Feature 2 Title
11       details: Feature 2 Description
12     - title: Feature 3 Title
13       details: Feature 3 Description
14     footer: Made by Vicente Martínez with ♥
15     ---
16
```

yaml

**Hola***Cross-Relation*

## 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.

### 9.2.1 Third level heading

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:



# 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

## Licencia

Apuntes de PSP creados por Vicente Martínez están bajo una licencia  
CC BY-NC-SA 4.0 

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:

# 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:

# Introduction

VuePress is composed of two parts: a [minimalistic static site generator](#) with a Vue-powered [theming system](#) and [Plugin API](#) , and a [default theme](#) optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

# 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:

# 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:




# Process and Service Programming

Class notes

[Access lessons →](#)

---

Except where otherwise noted, content on this site ([PSP class notes](#)) by Vicente Martínez is licensed under [CC BY-NC-SA 4.0](#) 

© 2020/2021

# 1 : Introduction to concurrent programming

VuePress is composed of two parts: a [minimalistic static site generator](#) with a Vue-powered [theming system](#) and [Plugin API](#) , and a [default theme](#) optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

## 1.1 Another title

---

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.



# Process and Service Programming

## Unit 9.3 - Multiprogramming

Processes and threads

---



PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0 

## 9. Using Vue in Markdown (print option)

default theme

- 9. Using Vue in Markdown (print option) 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"> <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>

Pasaremos los datos necesarios para construir un objeto ISBN al constructor de Libro (en nuestro caso una cadena con el isbn) y será este quien lo cree. En el fondo esto es lo que sucede con un tipo valor como `Date` por su 'naturaleza'. De esta manera, además, el objeto `Isbn13` que estoy creando desaparecerá junto con el libro.

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: ❤️

```

1  # Post Here Code Example
2  static public void function base(int a, String b)
3  {
4      System.out.println("hola");
5  }
```

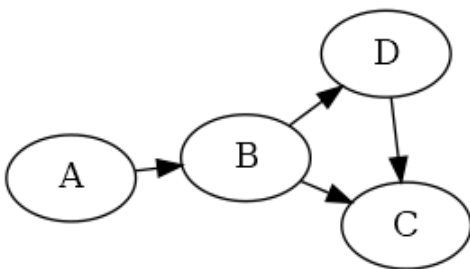
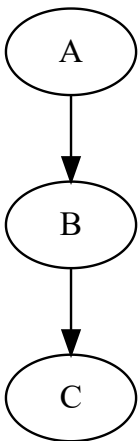
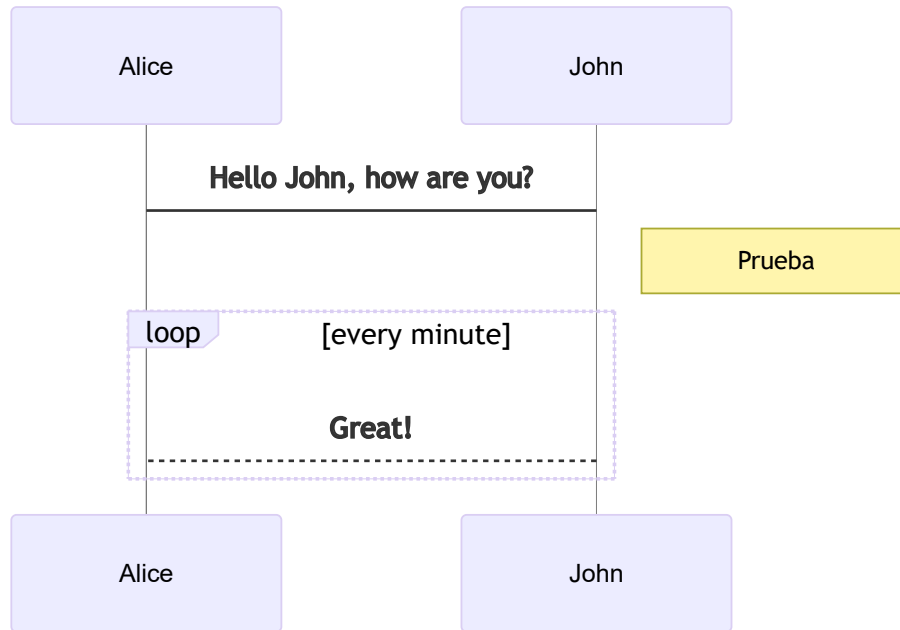
java

```
1 System.out.println("Hola: " + nombre);
```

php

```
1 ---
2 home: true
3 heroImage: /Logo_PSP.png
4 tagline: VuePress & Markdown
5 actionText: Access course →
6 actionLink: /unit1/
7 features:
8 - title: Feature 1 Título
9   details: Feature 1 Description
10 - title: Feature 2 Title
11   details: Feature 2 Description
12 - title: Feature 3 Title
13   details: Feature 3 Description
14 footer: Made by Vicente Martínez with ♥
15 ---
```

yaml



<b>Hola</b>
<i>Cross-Relation</i>

## 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.

### 9.2.1 Third level heading

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:

#### PHP Example

```
1 # Post Here Code Example
```

php

```
1 // Post Here Code Example
```

java

## 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:


This is another component

```
1 ---
2 home: true
3 heroImage: /LogoPSP_en.png
4 tagline: Class notes
5 actionText: Access lessons →
6 actionLink: /unit1/
7 features:
8 - title:
9   details:
10 - title:
```


md

```
11     details:
12 -   title:
13     details:
14 footer: © 2020/2021
15 ---
16
17 <div class="cclicense" align="center">
18   <p xmlns:cc="http://creativecommons.org/ns#" xmlns:dct="http://purl.org/dc/terms/">Exce
19 </div>
```


### Licencia

PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0 

### Licencia

PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0 

### Licencia

PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0 

### ► Licencia

PSP class notes by Vicente Martínez is licensed under CC BY-NC-SA 4.0 

# 9. Using Vue in Markdown by Vicente

default theme

[toc]

## 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"><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>

Pasaremos los datos necesarios para construir un objeto ISBN al constructor de Libro (en nuestro caso una cadena con el isbn) y será este quien lo cree. En el fondo esto es lo que sucede con un tipo valor como `Date` por su 'naturaleza'. De esta manera, además, el objeto `Isbn13` que estoy creando desaparecerá junto con el libro.

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: ♡

```
1 # Post Here Code Example
2 function base(a, b)
3 {
4     print "hola";
5     "123456789012345678901234 567890123456789 01234567890123456789012345 6789012345678 90
6 }
```

```
1 # Post Here Code Example
2 function base(a, b)
3 {
4     print "hola";
5     int i;
6     i=a;
7     b=a*i;
```

```

8
9     return i;
10 }

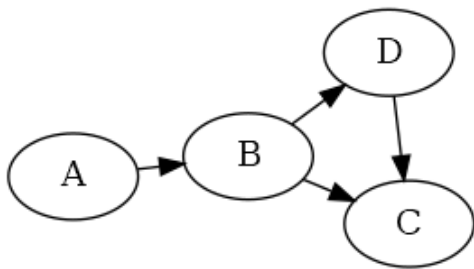
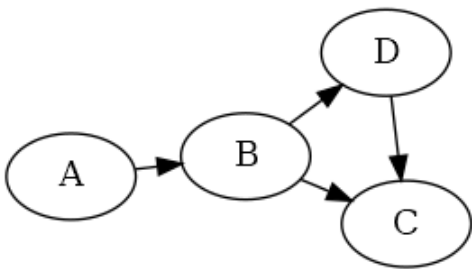
```

@import "../index.md" {as="yaml" code\_block="true" class="line-numbers", highlight="5,8-10,12"}

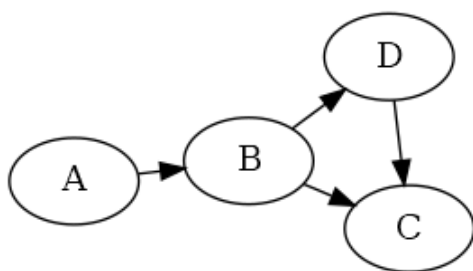
```

1     digraph A {
2         S->X
3         X->Y
4     }

```



*Cross-Relation*



Cross-Relation

## 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.

### 9.2.1 Third level heading

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:



# # PSP - Unit 2 : Processes programming

VuePress is composed of two parts: a [minimalistic static site generator](#) with a Vue-powered [theming system](#) and [Plugin API](#) , and a [default theme](#) optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

# 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.

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

# 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.

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