**Fundamentales Vue I**

**Instructor**: Abner Saavedra

**Fecha**: sept. de 2024

**Email**: [ingenieroabnersaavedra@gmail.com](mailto:ingenieroabnersaavedra@gmail.com)

**GracoSoft** Centro Empresarial Plaza Madrid, piso 9, oficinas 9-7 a la 9-10

# Agenda

## La instancia de la aplicación

## Parámetros de Ruta

## Controladores en Express.js

## Middleware de Rutas

## Rutas Anidadas y Modularización

## Ejemplo Completo

**La instancia de la aplicación​**

Cada aplicación Vue comienza creando una nueva instancia de aplicación con la función createApp:

import { createApp } from 'vue'

const app = createApp({

/\* root component options \*/

})

**El componente raíz**[​](https://vuejs.org/guide/essentials/application.html#the-root-component)

El objeto al que estamos pasando es, de hecho, un componente. Cada aplicación requiere un "componente raíz" que puede contener otros componentes como sus elementos secundarios.createApp

Si está utilizando componentes de un solo archivo, normalmente importamos el componente raíz de otro archivo:

Si bien muchos ejemplos de este curso solo necesitan un solo componente, la mayoría de las aplicaciones reales se organizan en un árbol de componentes anidados y reutilizables. Por ejemplo, el árbol de componentes de una aplicación Todo podría tener el siguiente aspecto:

App (root component)

├─ TodoList

│ └─ TodoItem

│ ├─ TodoDeleteButton

│ └─ TodoEditButton

└─ TodoFooter

├─ TodoClearButton

└─ TodoStatistics

En secciones posteriores del curso, discutiremos cómo definir y componer varios componentes juntos. Antes de eso, nos centraremos en lo que sucede dentro de un solo componente.

**Montaje de la aplicación**[​](https://vuejs.org/guide/essentials/application.html#mounting-the-app)

Una instancia de aplicación no representará nada hasta que se llame a su método. Espera un argumento "contenedor", que puede ser un elemento DOM real o una cadena de selección: .mount()

Html:

<div id="app"></div>

Js:

app.mount('#app')

El contenido del componente raíz de la aplicación se representará dentro del elemento contenedor. El elemento contenedor en sí no se considera parte de la aplicación.

Siempre se debe llamar al método después de que se hayan realizado todas las configuraciones de la aplicación y los registros de recursos. Tenga en cuenta también que su valor devuelto, a diferencia de los métodos de registro de activos, es la instancia del componente raíz en lugar de la instancia de la aplicación. .mount()

**Plantilla de componente raíz en DOM​**

La plantilla para el componente raíz suele formar parte del propio componente, pero también es posible proporcionar la plantilla por separado escribiéndola directamente dentro del contenedor de montaje:

<div id="app">

<button @click="count++">{{ count }}</button>

</div>

import { createApp } from 'vue'

const app = createApp({

data() {

return {

count: 0

}

}

})

app.mount('#app')

Vue usará automáticamente el contenedor como plantilla si el componente raíz aún no tiene una opción.innerHTMLtemplate

Las plantillas In-DOM se utilizan a menudo en aplicaciones que utilizan Vue sin un paso de compilación. También se pueden utilizar junto con marcos del lado del servidor, donde el servidor puede generar dinámicamente la plantilla raíz.

**Configuraciones de la aplicación**

La instancia de la aplicación expone un objeto que nos permite configurar algunas opciones a nivel de aplicación, por ejemplo, definir un controlador de errores a nivel de aplicación que capture errores de todos los componentes descendientes: .config

app.config.errorHandler = (err) => {

/\* handle error \*/

}

La instancia de la aplicación también proporciona algunos métodos para registrar recursos con ámbito de aplicación. Por ejemplo, registrar un componente:

app.component('TodoDeleteButton', TodoDeleteButton)

Esto hace que esté disponible para su uso en cualquier lugar de nuestra aplicación. Discutiremos el registro de componentes y otros tipos de activos en secciones posteriores de la guía. También puede examinar la lista completa de API de instancia de aplicación en su referencia de API. TodoDeleteButton

¡Asegúrese de aplicar todas las configuraciones de la aplicación antes de montar la aplicación!

**Varias instancias de aplicación**[​](https://vuejs.org/guide/essentials/application.html#multiple-application-instances)

No está limitado a una sola instancia de aplicación en la misma página. La API permite que varias aplicaciones de Vue coexistan en la misma página, cada una con su propio alcance para la configuración y los activos globales: createApp

const app1 = createApp({

/\* ... \*/

})

app1.mount('#container-1')

const app2 = createApp({

/\* ... \*/

})

app2.mount('#container-2')

Si está utilizando Vue para mejorar el HTML renderizado por el servidor y solo necesita Vue para controlar partes específicas de una página grande, evite montar una sola instancia de aplicación Vue en toda la página. En su lugar, cree varias instancias de aplicaciones pequeñas y móntelas en los elementos de los que son responsables.

**Sintaxis de la plantilla**

Vue utiliza una sintaxis de plantilla basada en HTML que le permite vincular declarativamente el DOM renderizado a los datos de la instancia del componente subyacente. Todas las plantillas de Vue son HTML sintácticamente válidos que pueden ser analizadas por navegadores que cumplan con las especificaciones y analizadores de HTML.

Bajo el capó, Vue compila las plantillas en código JavaScript altamente optimizado. Combinado con el sistema de reactividad, Vue puede determinar de manera inteligente el número mínimo de componentes para volver a renderizar y aplicar la cantidad mínima de manipulaciones del DOM cuando cambia el estado de la aplicación.

Si está familiarizado con los conceptos de Virtual DOM y prefiere la potencia bruta de JavaScript, también puede escribir directamente funciones de renderizado en lugar de plantillas, con soporte opcional para JSX. Sin embargo, tenga en cuenta que no disfrutan del mismo nivel de optimizaciones en tiempo de compilación que las plantillas.

**Interpolación de texto**

La forma más básica de enlace de datos es la interpolación de texto mediante la sintaxis "Bigote" (llaves dobles):

<span>Message: {{ msg }}</span>

La etiqueta bigote se reemplazará por el valor de la propiedad [de la instancia del componente correspondiente](https://vuejs.org/guide/essentials/reactivity-fundamentals.html#declaring-reactive-state). También se actualizará cada vez que cambie la propiedad. msg msg

**GRACOSOFT ES EXCELENCIA EDUCATIVA**