Fundamentos VueJS

Instructor: Abner Saavedra
Fecha: sept. de 2024
Email: ingenieroabnersaavedra@gmail.com
GracoSoft Centro Empresarial Plaza Madrid, piso 9, oficinas 9-7 a la 9-10

Agenda

- ¿Qué es Vue?
- ¿El marco progresista?
- Componentes de un solo archivo
- Estilos API
  - API de Opciones
  - API de composición
- ¿Cuál elegir?
- Recomendaciones
- Mi primer proyecto
  - Andamiaje oficial Vue
  - Consejos adicionales para desarrollo Vue
  - Compilación de aplicación
- Uso de Vue desde CDN
- Uso de la compilación global
- Uso de la compilación del módulo ES
- Habilitación de la importación de mapas
- División de los módulos

¿Qué es Vue?

Vue (pronunciado /vjuː/, como view) es un marco de JavaScript para crear interfaces de usuario. Se basa en HTML, CSS y JavaScript estándar y proporciona un modelo de programación declarativo basado en componentes que lo ayuda a desarrollar de manera eficiente interfaces de usuario de cualquier complejidad.

- Vue es un marco (Framework) de JavaScript para construir interfaces de usuario.
- Se basa en HTML, CSS y JavaScript estándar, y proporciona un modelo de programación declarativo y basado en componentes que lo ayuda a desarrollar interfaces de usuario de manera eficiente, ya sea simple o compleja.

He aquí un ejemplo mínimo:

```
js
import { createApp, ref } from 'vue'

createApp({
  setup() {
    return {
      count: ref(0)
    }
  }
}).mount('#app')
```

```
plantilla
<div id="app">
  <button @click="count++">
    Count is: {{ count }}
  </button>
</div>
Resultado

El conteo es: 0
```

El ejemplo anterior demuestra las dos características principales de Vue:

- **Representación declarativa** : Vue extiende el HTML estándar con una sintaxis de plantilla que nos permite describir de forma declarativa la salida HTML en función del estado de JavaScript.
- **Reactividad** : Vue rastrea automáticamente los cambios de estado de JavaScript y actualiza eficientemente el DOM cuando ocurren cambios.

Es posible que ya tengas preguntas, no te preocupes. Trataremos cada pequeño detalle más adelante.

El marco progresista

Vue es un marco y un ecosistema que cubre la mayoría de las características comunes necesarias **en el desarrollo de frontend**. Pero la web es extremadamente diversa: las cosas que creamos en la web pueden variar drásticamente en forma y escala. Con eso en mente, Vue está diseñado para ser flexible y adoptarse de manera incremental. Según su caso de uso, **Vue se puede utilizar de diferentes maneras**:

- Mejorar HTML estático sin un paso de compilación
- Incrustar como componentes web en cualquier página
- Solicitud de una sola página (SPA)
- Representación fullstack/del lado del servidor (SSR)
- Jamstack / Generación de sitios estáticos (SSG)
- Orientado a computadoras de escritorio, dispositivos móviles, WebGL e incluso terminales

Si estos conceptos te intimidan, no te preocupes. Este curso solo requerirá conocimientos básicos de HTML y JavaScript, y deberías poder seguirlos sin ser un experto en ninguno de ellos.

Componentes de un solo archivo

En la mayoría de los proyectos de Vue habilitados para herramientas de compilación, creamos componentes de Vue utilizando un formato de archivo similar a HTML llamado **componente de archivo único** (también conocido como *.vuearchivos, abreviado como **SFC**). Un SFC de Vue, como sugiere el nombre, encapsula la lógica del componente (JavaScript), la plantilla (HTML) y los estilos (CSS) en un solo archivo. Aquí está el ejemplo anterior, escrito en formato SFC:

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

SFC es una característica definitoria de Vue y es la forma recomendada de crear componentes Vue si su caso de uso justifica una configuración de compilación. Puede obtener más información sobre el cómo y el por qué de SFC en su sección dedicada, pero por ahora, solo sepa que Vue se encargará de toda la configuración de las herramientas de compilación por usted.

Estilos API

Los componentes de Vue se pueden crear en dos estilos de API diferentes: API de opciones y API de composición.

API de Opciones

Con la API de opciones, definimos la lógica de un componente mediante un objeto de opciones como data, methods y mounted.
Las propiedades definidas por las opciones se exponen en thisfunciones internas, que apuntan a la instancia del componente:

```
<script>
export default {
  // Properties returned from data() become reactive state
  // and will be exposed on `this`.
  data() {
    return {
      count: 0
    }
  },

  // Methods are functions that mutate state and trigger updates.
  // They can be bound as event handlers in templates.
  methods: {
    increment() {
      this.count++
    }
  },

  // Lifecycle hooks are called at different stages
  // of a component's lifecycle.
  // This function will be called when the component is mounted.
  mounted() {
    console.log(`The initial count is ${this.count}.`)
  }
}
</script>
<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

API de composición

Con Composition API, definimos la lógica de un componente mediante funciones API importadas. En los SFC, Composition API se usa normalmente con <script setup>. El setup atributo es una sugerencia que hace que Vue realice transformaciones en tiempo de compilación que nos permiten usar Composition API con menos código repetitivo. Por ejemplo, las importaciones y las variables/funciones de nivel superior declaradas en <script setup>se pueden usar directamente en la plantilla.
Aquí está el mismo componente, con exactamente la misma plantilla, pero usando Composition API y <script setup>en su lugar:

```
<script setup>
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)

// functions that mutate state and trigger updates
function increment() {
  count.value++
}

// lifecycle hooks
onMounted(() => {
  console.log(`The initial count is ${count.value}.`)
})
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

¿Cuál elegir?

Ambos estilos de API son totalmente capaces de cubrir casos de uso comunes. Son diferentes interfaces impulsadas por exactamente el mismo sistema subyacente. De hecho, **la API de opciones se implementa sobre la API de composición**. Los conceptos y conocimientos fundamentales sobre Vue se comparten en los dos estilos.

La API de opciones se centra en el concepto de una "**instancia de componente**" (como se ve en el ejemplo), que generalmente se alinea mejor con un modelo mental basado en clases para usuarios que provienen de entornos de lenguaje OOP. También es más amigable para los principiantes al abstraer los detalles de reactividad y hacer cumplir la organización del código a través de grupos de opciones. this

La API de composición se centra en declarar variables de estado reactivo directamente en el ámbito de una función y componer el estado de varias funciones juntas para controlar la complejidad. Es más libre y requiere una comprensión de cómo funciona la reactividad en Vue para ser utilizada de manera efectiva. A cambio, su flexibilidad permite patrones más potentes para organizar y reutilizar la lógica.

Recomendaciones

Si eres nuevo en Vue, aquí tienes la recomendación general:

Para fines de aprendizaje, elija el estilo que le parezca más fácil de entender. Una vez más, la mayoría de los conceptos básicos son compartidos entre los dos estilos. Siempre puedes elegir el otro estilo más tarde.

Para uso en producción:

Vaya con la **API de opciones** si no está utilizando herramientas de compilación, o planea usar Vue principalmente en escenarios de baja complejidad, por ejemplo, mejora progresiva.

Opta por la **API de composición** + los componentes de un solo archivo si planeas crear aplicaciones completas con Vue.

Creación de una aplicación Vue

Prerrequisitos

- Familiaridad con la línea de comandos
- Instalar Node.js versión 18.3 o superior

En esta sección, presentaremos cómo crear un andamiaje para una aplicación de una sola página de Vue en su máquina local. El proyecto creado utilizará una configuración de compilación basada en Vite y nos permitirá usar Vue Single-File Components (SFC). Asegúrese de tener instalada una versión actualizada de Node.js y de que su directorio de trabajo actual es aquel en el que tiene la intención de crear un proyecto. Ejecute el siguiente comando en la línea de comandos (sin el signo): $

```
$ npm create vue@latest
```

Este comando instalará y ejecutará create-vue, la herramienta oficial de andamiaje del proyecto Vue. Se le presentarán indicaciones para varias funciones opcionales, como TypeScript y soporte de pruebas:

```
✔ Project name: … <your-project-name>
✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit testing? … No / Yes
✔ Add an End-to-End Testing Solution? … No / Cypress / Nightwatch / Playwright
✔ Add ESLint for code quality? … No / Yes
✔ Add Prettier for code formatting? … No / Yes
✔ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in ./<your-project-name>...
Done.
```

Si no está seguro de una opción, simplemente elija presionando enter por ahora. Una vez creado el proyecto, siga las instrucciones para instalar dependencias e iniciar el servidor de desarrollo: No

```
$ cd <your-project-name>
$ npm install
$ npm run dev
```

¡Ahora deberías tener tu primer proyecto de Vue en ejecución! Tenga en cuenta que los componentes de ejemplo del proyecto generado se escriben mediante la API de composición y , en lugar de la API de opciones.

Estos son algunos consejos adicionales:<script setup>

- La configuración recomendada del IDE es Visual Studio Code + Vue - Official extension. Si utilizas otros editores, echa un vistazo a la sección de soporte del IDE.
- Más detalles de las herramientas, incluida la integración con los marcos de backend, se analizan en la Guía de herramientas.
- Para obtener más información sobre la herramienta de compilación subyacente Vite, consulte los documentos de Vite.
- Si eliges usar TypeScript, consulta la Guía de uso de TypeScript.

Compilación de la aplicación

Cuando esté listo para enviar la aplicación a producción, ejecute lo siguiente:

```
$ npm run build
```

De este modo, se creará una compilación lista para producción de la aplicación en el directorio del proyecto. Echa un vistazo a la Guía de implementación de producción para obtener más información sobre cómo enviar tu aplicación a producción. ./dist

Uso de Vue desde CDN

Puede usar Vue directamente desde una CDN a través de una etiqueta de script:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

Aquí estamos usando **unpkg**, pero también puedes usar cualquier CDN que sirva paquetes npm, por ejemplo **jsdelivr** o **cdnjs**. Por supuesto, también puede descargar este archivo y servirlo usted mismo.

Cuando se usa Vue desde una CDN, no hay un "paso de compilación" involucrado. Esto hace que la configuración sea mucho más sencilla y es adecuada para mejorar el HTML estático o integrarse con un marco de backend. Sin embargo, no podrá usar la sintaxis de componente de archivo único (SFC).

Uso de la compilación global

El enlace anterior carga la *compilación global* de Vue, donde todas las API de nivel superior se exponen como propiedades en el objeto global. A continuación, se muestra un ejemplo completo con la compilación global:Vue

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
  const { createApp, ref } = Vue

  createApp({
    setup() {
      const message = ref('Hello vue!')
      return {
        message
      }
    }
  }).mount('#app')
 </script>
```

Uso de la compilación del módulo ES

A lo largo del resto de la documentación, utilizaremos principalmente la sintaxis de los módulos ES. La mayoría de los navegadores modernos ahora soportan módulos ES de forma nativa, por lo que podemos usar Vue desde una CDN a través de módulos ES nativos como este:

```
<div id="app">{{ message }}</div>

<script type="module">

  import { createApp, ref } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'
  createApp({
    setup() {
      const message = ref('Hello Vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

Tenga en cuenta que estamos usando , y la URL de la CDN importada apunta a la compilación de módulos ES de Vue en su lugar.

```
<script type="module">
```

Habilitación de la importación de mapas

En el ejemplo anterior, estamos importando desde la URL completa de la CDN, pero en el resto de curso verás un código como este:

```
import { createApp } from 'vue'
```

Podemos enseñarle al navegador dónde ubicar la importación mediante el uso de Importar Mapas: vue

```
<script type="importmap">
  {
    "imports": {
      "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
    }
  }
</script>

<div id="app">{{ message }}</div>

<script type="module">
  import { createApp, ref } from 'vue'

  createApp({
    setup() {
      const message = ref('Hello Vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

También puede agregar entradas para otras dependencias al mapa de importación, pero asegúrese de que apunten a la versión de los módulos ES de la biblioteca que desea utilizar.

Compatibilidad con el navegador de mapas de importación

Importar mapas es una función relativamente nueva del navegador. Asegúrese de utilizar un navegador dentro de su rango de soporte. En particular, solo es compatible con Safari 16.4+.

Notas sobre el uso de la producción

Los ejemplos hasta ahora utilizan la compilación de desarrollo de Vue: si tiene la intención de usar Vue desde una CDN en producción, asegúrese de consultar la Guía de implementación de producción.

Si bien es posible usar Vue sin un sistema de compilación, un enfoque alternativo a considerar es usar vuejs/petite-vue que podría adaptarse mejor al contexto en el que se podría usar jquery/jquery (en el pasado) o alpinejs/alpine (en el presente).

División de los módulos

A medida que profundicemos en la guía, es posible que necesitemos dividir nuestro código en archivos JavaScript separados para que sean más fáciles de administrar. Por ejemplo:

```
<!-- index.html -->
<div id="app"></div>

<script type="module">
  import { createApp } from 'vue'
  import MyComponent from './my-component.js'

  createApp(MyComponent).mount('#app')
</script>
```

```
// my-component.js
import { ref } from 'vue'
export default {
  setup() {
    const count = ref(0)
    return { count }
  },
  template: `<div>Count is: {{ count }}</div>`
}
```

Si abre directamente lo anterior **en su navegador**, encontrará que arroja un error porque los módulos ES no pueden funcionar sobre el protocolo, que es el protocolo que usa el navegador cuando abre un archivo local.index.htmlfile://

Por razones de seguridad, los módulos ES solo pueden funcionar a través del protocolo, que es lo que utilizan los navegadores al abrir páginas en la web. Para que los módulos ES funcionen en nuestra máquina local, necesitamos servirlos a través del protocolo, con un servidor HTTP local. http://index.htmlhttp://

Para iniciar un servidor HTTP local, primero asegúrese de tener Node.js instalado, luego ejecútelo desde la línea de comandos en el mismo directorio donde se encuentra su archivo HTML. También puede utilizar cualquier otro servidor HTTP que pueda servir archivos estáticos con los tipos MIME correctos. npx serve

Es posible que haya notado que la plantilla del componente importado está insertada como una cadena de JavaScript. Si usa VS Code, puede instalar la extensión es6-string-html y anteponer un comentario a las cadenas para obtener un resaltado de sintaxis. /*html*/

Bibliografía

https://vuejs.org/guide/introduction.html#what-is-vue

https://bluuweb.github.io/vue-udemy/30-01-fundamentos/#objetivos

GRACOSOFT ES EXCELENCIA EDUCATIVA