VUE

rortegag

Crear una aplicación con Vite

Entorno de desarrollo rápido de Vue3.

```
npm init vite-app <nombre-proyecto>
cd <nombre-proyecto>
npm install
npm run dev
```

Directivas

v-if	Pone en DOM si es verdadero
v-else-if	Como un condicional usual
v-else	Como un condicional usual
v-show	Muestra el valor CSS
v-text	Establece el texto interior
v-html	Establece el HTML interno
v-for	Recorrer un array/obj
v-on or @	Escucha los eventos del DOM
v-bind or :	Atributo de actualizaciones reactivas
v-model	Vinculación bidireccional de datos
v-once	Establece val una vez; nunca actualiza

Renderización de listas

Bucle básico sobre array.

```
<li v-for='item in items' :key='item'>
  {{ item }}
</li>
```

Índice de bucles y pistas.

```
<li v-for='(item, index) in items'>
  {{ index }} : {{ item }}
</li>
```

Valores de bucle en el objeto.

```
<li v-for='obj in objects'>
  {{ obj }}
</li>
```

Vinculación de datos

Vinculación simple.

```
<div v-bind:id='objectID'>...</div>
<!-- ABREVIATURA -->
<div :id='objectID'>...</div>
```

Vinculación bidireccional con datos y entrada.

```
<input v-model='email' />
```

Modificadores de entrada de datos.

.lazy	actualizaciones sobre el evento del cambio
.trim	elimina los espacios en blanco sobrantes

Uso de objetos para vincular clases/estilos.

```
<input :class='{ error: hasError }' />
<input :style='{ margin: space + "px" }' />
```

Sintaxis de plantilla

Opciones de interpolación de texto.

```
<span> {{ msg }} </span>
<span v-text='msg'></span>
```

Configuración de HTML interno.

```
<span v-html='rawHTML'></span>
```

Puede utilizar expresiones JS; NO declaraciones JS

```
SI --> <span> {{ msg.reverse() }} </span>
NO --> <span> {{ let msg = 'hola' }} </span>
```

Renderizado condicional

Añadir/Quitar elemento del DOM con booleano.

```
<div v-if='fecha == hoy'>...</div>
<div v-else-if='!hecho'>...</div>
<div v-else>...</div>
```

Muestra CSS en lugar de editar DOM.

```
<div v-show='fecha == hoy'>...</div>
```

Gestión de eventos

Capturar un evento y llamar a un método.

```
<div v-on:click='count'>Incrementa</div>
<!-- ABREVIATURA -->
<div @click='count'>Incrementa</div>
```

Al método se le pasa un evento DOM nativo.

```
const count = (event) => {
  console.log(event.target)
}
```

Modificadores de eventos (uso: v-on:click.stop).

.stop	Detiene la propagación de eventos
.once	Sólo puede activarse una vez
.prevent	Llama a evt.preventDefault
.self	No enviar si target = child

Vincular datos entre el hijo y el padre

Utiliza v-bind para pasar datos de padre a hijo y emite un evento personalizado para enviar datos de vuelta.

En padre, vincular datos y configurar receptor para actualizar.

```
<custom :msg='s' @update='s = $event' />
```

En hijo, envía de vuelta usando emit(event, data).

```
context.emit('update', 'hola mundo')
```

Slots

Los slots permiten inyectar contenido de un componente padre a un componente hijo.

MÁS

Slots básicos.

Componente hijo (MyButton.Vue).

```
<div>
  Hola Mundo
  <slot></slot>
</div>
```

Componente padre.

```
<my-button>
  Este contenido sustituirá al
  slot
</my-button>
```

Para más información: https://vuejs.org/guide/introduction.html

VUE

rortegag

Slots

Slots con nombre

Útil cuando se tienen varias ranuras. Si no tiene nombre, el nombre es 'default'.

Componente hijo (MyButton.Vue)

```
<div>
  <slot name='top'></slot>
  <slot name='bottom'></slot>
</div>
```

Ranuras de nombre en el componente padre.

```
<my-button>
  <template v-slot:top>// ... </template>
  <template v-slot:bottom>// ... </template>
</my-button>
```

Slots con ámbito

Dar acceso al componente padre a los datos del hijo.

Componente hijo (MyButton.Vue)

```
<div>
  <slot v-bind:post='post'>
    {{post.title}}
  </slot>
</div>
```

El padre tiene acceso a los datos de la entrada MyButton

```
<my-button>
  <template v-slot:default='slotData'>
    {{ post.author }}
  </template>
</my-button>
```

Componentes dinámicos

Cambia el componente renderizado - encuentra un componente registrado con el nombre dado.

```
<component :is='componenteNombre' />
```

Elementos "Keep-Alive"

Almacena una versión en caché de los componentes dinámicos cuando no están visibles. Evita tener que crear un nuevo componente cada vez que se conmuta.

```
<keep-alive>
  <component :is='componenteNombre' />
</keep-alive>
```

API DE COMPOSICIÓN

Todo lo devuelto por setup() se expone a la plantilla.

```
import { ref, reactive } from 'vue'
export default {
  setup(props, context) {
    const val = ref('ejemplo')
    const obj = reactive( {count: 0 } )

    const evtHandler = () => {/*...*/}

    return {
      val, obj, evtHandler
    }
  }
}
```

O puedes usar <script setup>.

```
import { ref, reactive } from'vue'

const props = defineProps({..])
const context = defineEmit(...)

const val = ref('ejemplo')
const obj = reactive( {count: 0 } )

const evtHandler = () => {/*...*/}
```

Propiedades del objeto de contexto setup().

attrs	Tiene los atributos del componente
slots	Tiene slots para los componentes
emit	Función para emitir eventos

Hooks del ciclo de vida de Vue

*beforeCreate	Utiliza setup() en su lugar
*created	Utiliza setup() en su lugar
onBeforeMount	Antes de montar el DOM
onMounted	Se puede acceder al DOM
onBeforeUpdate	Cambios reactivos de los datos
onUpdated	Se ha actualizado el DOM
onBeforeUnmount	Componente todavía completo
onUnmounted	Desmontaje completo

Ejemplo de código Hook de cliclo de vida.

```
import { onMounted } from 'vue'
// ...
setup() {
  onMounted(() => {
    console.log('Componente montado')
  }
}
```

Para más información: https://vuejs.org/guide/introduction.html

VUE

rortegag

Métodos globales de Vue

mount()	Montar el componente en el DOM
forceUpdate()	Forzar nueva renderización
nextTick()	Ejecuta la función de la siguiente actualización
destroy()	Destruir el componente/app

OPCIONES DE LA API DE OBJETOS DE VUE

Si decides no utilizar la API de composición, los componentes tendrán un aspecto similar al de Vue2 con la API de opciones.

data()	Datos reactivos de init
props	Datos visibles por el padre
mixins	Declara mixins
components	Registra los componentes hijos
methods	Conjunto de métodos de Vue
watchers	Vigilar el cambio de los valores
computed	Métodos reactivos en caché

Librerias principales de Vue

vue-cli	Interfaz de línea de comandos
https://cli.vuejs.org/	
vue-router	Maneja el enrutamiento de SPA
https://router.vuejs.org/	
vuex	Biblioteca de gestión del Estado
https://vuex.vuejs.org/	
pinia	Biblioteca de gestión del Estado
https://pinia.vuejs.org/	

Librerias principales de Vue

vue-cli	Interfaz de línea de comandos
https://cli.vuejs.org/	
vue-router	Maneja el enrutamiento de SPA
https://router.vuejs.org/	
vuex	Biblioteca de gestión del Estado
https://vuex.vuejs.org/	
pinia	Biblioteca de gestión del Estado
https://pinia.vuejs.org/	

Propiedades computadas

Una propiedad computada es un valor que se calcula utilizando una o más propiedades.

```
setup() {
  const a = ref(1)
  const b = computed(() => a.value * 2)
  return { a, b }
}
```

watchEffect()

Escucha las dependencias reactivas y ejecuta un método cuando una cambia. También se ejecuta en init.

```
setup() {
  const ejemplo = ref('rortegag.com')

  watchEffect(() => {
    console.log(ejemplo.value)
  })

  return { ejemplo }
}
```

refs de plantillas

Da acceso a los elementos del DOM.

```
// template
<div ref='ejemplo'>Div de ejemplo</div>
// script
setup() {
  const ejemplo = ref('rortegag.com')
  // espera a que se monte el DOM
  onMounted(() => {
    console.log(ejemplo.value)
  })

  return { ejemplo }
}
```

Para más información: https://vuejs.org/guide/introduction.html