

# Manejo de eventos y reutilización de componentes

Reutilización de componentes

***Implementar interacción en los elementos de una interfaz web utilizando manejo de eventos Vue para dar solución a un requerimiento y componentes reutilizables utilizando el framework Vue para el desarrollo de una aplicación web mantenible en el tiempo.***

- Unidad 1: Introducción a Componentes Web y Vue Js
- Unidad 2: Binding de formularios
- Unidad 3: Templates y rendering en Vue
- Unidad 4: Manejo de eventos y reutilización de componentes
- Unidad 5: Consumo de datos desde una API REST



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Describe el concepto de reutilización de componentes distinguiendo sus beneficios para el desarrollo de una aplicación web mantenible.*

# ¿Cómo se asignan eventos en Vue Js?



¿Qué son y para qué  
sirven los modificadores  
de eventos y teclas?

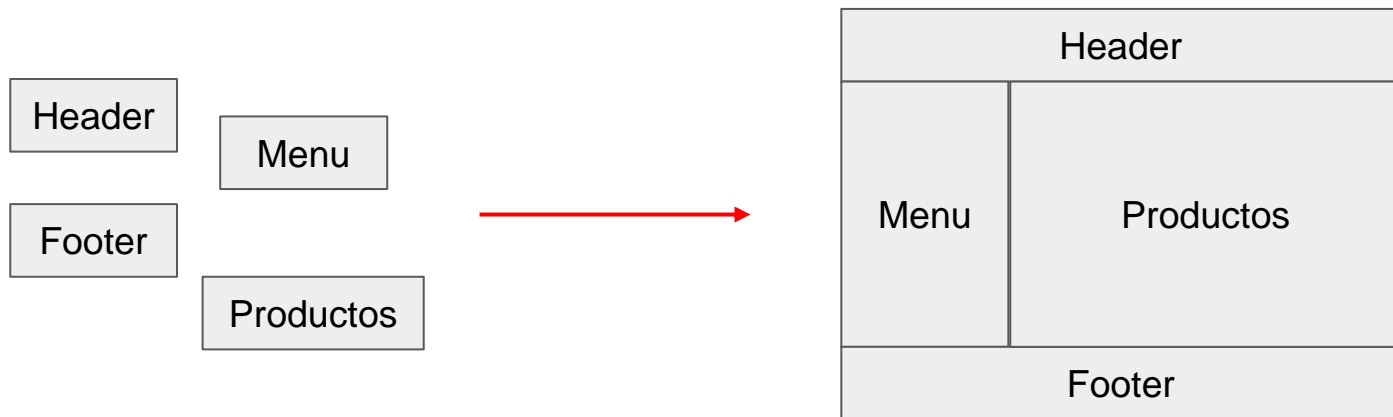


**/\* Componentes \*/**

# Componentes

## *modularización*

En Vue podemos construir nuestras aplicaciones como si se tratara de un rompecabezas, donde cada pieza es un componente importado, a esto se conoce como **modularización**.



# Ejercicio guiado 1





# Componentes

## modularización

1. En la carpeta componentes, crea un componente **Header.vue** que contengan lo siguiente

```
<template>
  <div>
    <header>
      <h1>Componentes en Vue js</h1>
    </header>
  </div>
</template>
```

```
<style scoped>
header {
  background: darkred;
  color: white;
  padding: 5px;
  text-align: center;
}
</style>
```

El atributo **scoped** nos permite que los estilos que definamos en esta etiqueta sean estilos solo y exclusivamente para este componente.

# Componentes

## modularización

2. Ahora modifica el componente **App.vue** para que importe el componente Header

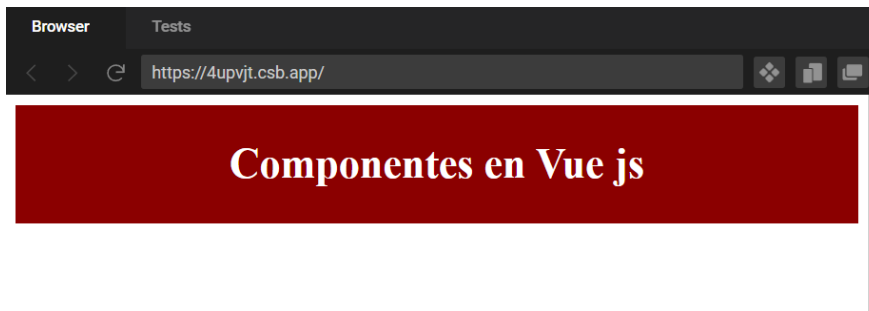
```
<template>
  <div>
    <Header />
  </div>
</template>

<script>
import Header from "../components/Header.vue";

export default {
  components: {
    Header,
  },
};
</script>
```

{desafío}  
latam\_

Con el atributo **components** debemos agregar todos los componentes que necesitemos usar en el template



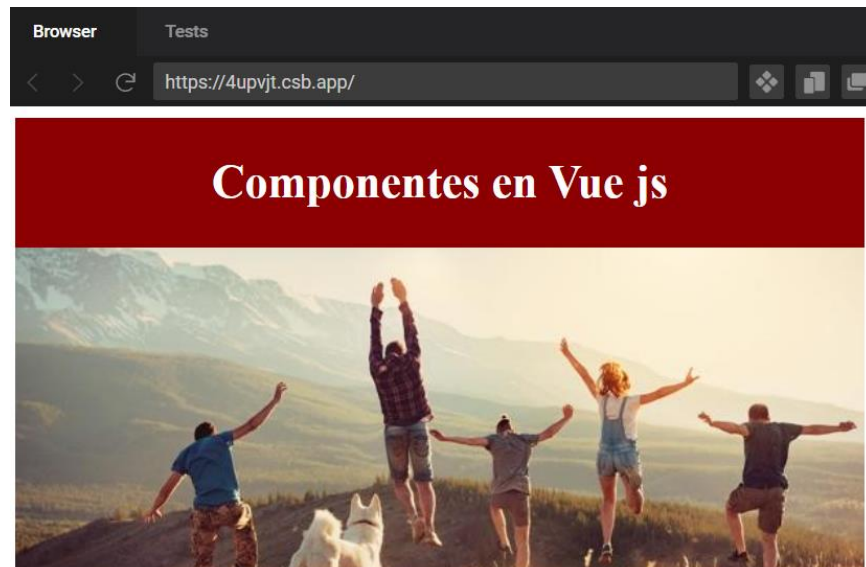
# Ejercicio propuesto 1



# Componentes

## *modularización*

- Agrega un Hero Section a la aplicación creando un componente **Hero.vue** y agregandolo al componente **App.vue**

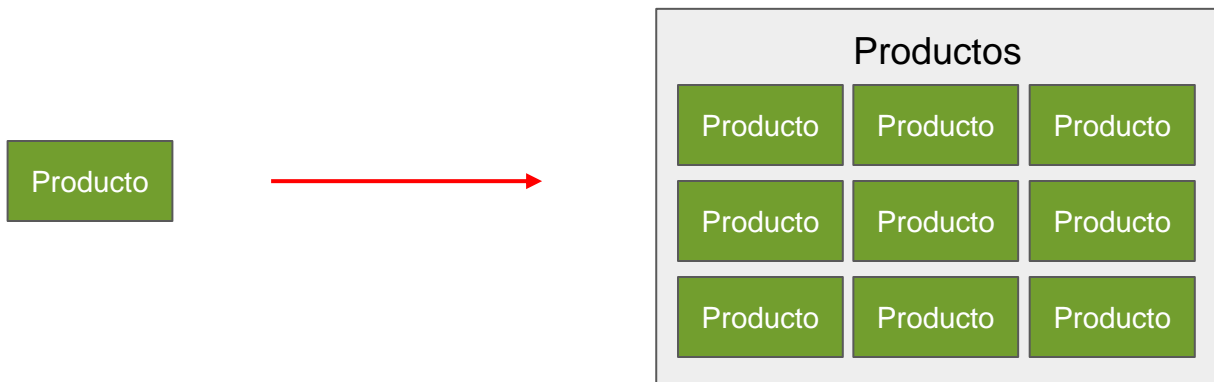


# **/\* Reutilización de Componentes \*/**

# Componentes

## Reutilización

Podemos utilizar la práctica de los componentes y mezclarla con la directiva **v-for** para reutilizar un mismo componente las **n** veces que necesitemos.



## Ejercicio guiado 2



# Componentes

## Reutilización

1. Crea un nuevo componente llamado **Persona.vue** que muestre lo siguiente. Debes agregar este componente al componente **App.vue** para visualizarlo en el navegador.

```
<template>
  <div class="persona">
    <div class="avatar"></div>
    <h2>Evan You</h2>
    <p>Creador de Vue Js</p>
  </div>
</template>
```

```
<style scoped>
  .persona {
    margin-top: 10px;
    text-align: center;
    width: fit-content;
  }

  .avatar {
    background-image:
      url("https://avatars.githubusercontent.com/u/499550?v=4");
    background-size: cover;
    background-position: center;
    height: 200px;
    width: 200px;
    border-radius: 50%;
  }
</style>
```



**Evan You**

Creador de Vue Js



# Componentes

## Reutilización

2. En el componente App, crea una variable en el estado que sea igual a un arreglo **personas** con 3 elementos

```
data() {  
  return {  
    personas: [null, null, null],  
  };  
},
```

Por ahora los elementos serán **null**, más adelante serán objetos con los datos de la persona.

# Componentes

## Reutilización

3. En el template de **App.vue** renderiza dinámicamente el componente **Persona.vue** iterando en el arreglo **personas**.

```
<template>
  <div>
    <Header />

    <section>
      <div v-for="persona in personas">
        <Persona />
      </div>
    </section>

  </div>
</template>
```

Componentes en Vue.js



Evan You

Creador de Vue.js



Evan You

Creador de Vue.js



Evan You

Creador de Vue.js

# Componentes

## Reutilización

4. Acomoda un poco la estética creando una grilla con CSS grid en el componente **App.vue**

```
<style>
.personas {
  width: 80%;
  margin: 30px auto;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
}
</style>
```

### Componentes en Vue js



Evan You

Creador de Vue Js



Evan You

Creador de Vue Js



Evan You

Creador de Vue Js

De esta manera reutilizamos un mismo componente usando la renderización dinámica del v-for.

**`/* Props */`**

# Componentes

## *props*

Ahora que ya sabemos cómo importar componentes y renderizarlos dinámicamente, podemos avanzar al paso de **props**.

Esta práctica consiste en entregar propiedades o datos a un componente hijo directamente en el template.

Ejemplo.vue

Hola, me llamo {{nombre}}



<Ejemplo :nombre="Steve" />



Esta es una **prop**



Ejemplo.vue

Hola, me llamo **Steve**

## Ejercicio guiado 3



# Componentes

## *props*

Continuando con los ejercicios guiados anteriores...

1. Identifiquemos qué valores del componente **Persona.vue** podemos convertir en **props**



← Foto

Nombre → **Evan You**

Creador de Vue Js ← Descripción


# Componentes

## props

2. Modifiquemos el template del componente **Persona.vue** y agreguemos un script para que reciba los valores dinámicamente mediante props

```
<template>
  <div class="persona">
    <div
      :style="{ backgroundImage: `url(${foto})` }"
      class="avatar">
    </div>
    <h2>{{ nombre }}</h2>
    <p>{{ descripcion }}</p>
  </div>
</template>
```

```
<script>
export default {
  props: ["foto", "nombre", "descripcion"],
};
</script>
```



Los componentes hijos deben declarar las props que esperan recibir de esta manera, a través de un atributo **props** cuyo valor es un arreglo de strings con los nombres de cada prop.



# Componentes

## *props*

3. En el componente **App.vue** modifiquemos el arreglo **personas** para que contenga la información de 3 personas diferentes.

Utiliza el siguiente formato

```
{  
  foto: String,  
  nombre: String,  
  descripcion: String  
}
```



```
personas: [  
  {  
    foto: "https://avatars.githubusercontent.com/u/499550?v=4",  
    nombre: "Evan You",  
    descripcion: "Creador de Vue Js",  
  },  
  ...  
],
```

# Componentes

## props

4. Finalmente modifiquemos en el template la renderización dinámica del **v-for** para suministrar las diferentes props al componente **Persona**

```
<div v-for="persona in personas">
  <Persona
    :foto="persona.foto"
    :nombre="persona.nombre"
    :descripcion="persona.descripcion"
  />
</div>
```

### Componentes en Vue.js



**Brendan Eich**

Creador de JavaScript



**Douglas Crockford**

Creador del JSON



**Evan You**

Creador de Vue.js

## Ejercicio propuesto 2



# Componentes

## Reutilización

- Crea una galería de productos con instrumentos musicales usando la renderización dinámica y el paso de props

### Instrumentos Musicales



**Batería electrónica**

\$649.900



**Guitarra eléctrica**

\$429.900



**Teclado**

499.900

¿Qué te parece  
lo aprendido?



**Comparte con tus  
compañeros qué casos de  
uso de renderización  
dinámica de componente  
conoces en la web**





## Próxima sesión...

- *Implementa una aplicación web utilizando comunicación entre componentes padres e hijos para resolver un problema determinado.*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

