Introducción a Componentes Web y Vue Js

Framework Vue JS

{desafío} latam_

Describir los aspectos fundamentales de un framework orientado a componentes para el desarrollo de una aplicación Front-End.

- 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

¿Qué aprenderás en esta sesión?

- *Describe el rol de un framework orientado a componentes reconociendo sus beneficios para el desarrollo de una aplicación Front-End.*
- *Describe las características principales del framework Vue.js para el desarrollo de aplicaciones Front-End.*

{desafío} latam_

¿Qué es el SFC?

¿Cuál es la sintaxis que se debe seguir para interpolar variables del estado en el template?

/* El Objeto Data */

El Objeto Data

Vue trabaja con un concepto llamado **estado**, el cuál se entiende como un banco de variables reactivas que pueden ser utilizadas en cualquier parte del componente. Para crear un estado local en el componente debemos agregar el método **data()** en el objeto que se está exportando en la etiqueta script.

```
<script>
export default {
  name: "App",
  data() {
    return {
      titulo: "Vue Js es el mejor framework de JavaScript 😎",
    };
  },
};
</script>
```

Con esta modificación estamos agregando una variable **titulo** en el componente **App.**

El método **data** debe retornar un objeto, cuyas propiedades serán las variables del estado local.

El Objeto Data

interpolaciones en el template

Las variables del estado podemos interpolarlas dentro del template al incluirlas dentro de doble llaves: {{ }}

```
<template>
  <div id="app">
    <h1>{{ titulo }}</h1>
  </div>
</template>
```

El Objeto Data

Si vemos el navegador podremos apreciar como ahora el título muestra el valor de la variable **titulo**.

De esta manera podemos almacenar en el estado **toda la información variable** que necesitemos incluir en el template.

Ejercicio guiado

"Interpolemos nuestro nombre en el componente App"

Interpolación de variables

Pongamos a prueba nuevamente la interpolación completando la frase:
“Hola, me llamo {{ nombre }}”

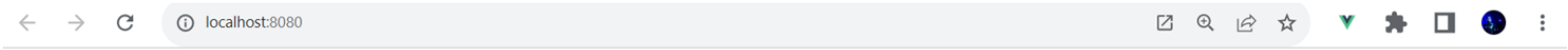
Para esto actualiza el template y el estado de la siguiente manera:

```
<template>
  <div id="app">
    <p>Hola, me llamo {{ nombre }}</p>
  </div>
</template>
```

```
<script>
export default {
  name: "App",
  data() {
    return {
      nombre: "Evan You",
    };
  },
};
</script>
```

Interpolación de variables

Ahora si visualizamos el navegador, podremos ver lo siguiente:



Hola, me llamo Evan You

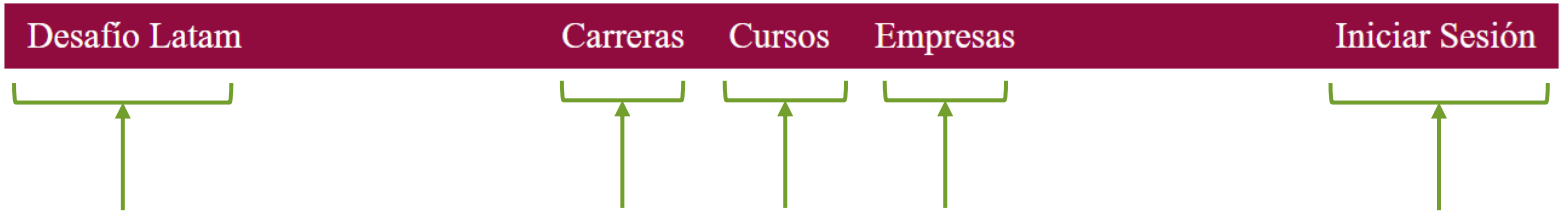
De esta misma manera podemos crear templates personalizados y rellenarlos con información diferente.

Ejercicio guiado

“Navbar interpolado"

Interpolación de variables

Sigamos poniendo en práctica la interpolación asignando el contenido textual en un navbar.



Cada una de estas divisiones deberán ser una variable en el estado diferente.

Interpolación de variables

Para esto escribamos nuestro **template, script** y **style**

```
<template>
 <div>
   <nav>
     <span>{{ nombreDelNegocio }}</span>
     <div class="opciones">
       <span>{{ opcion1 }}</span>
       <span>{{ opcion2 }}</span>
       <span>{{ opcion3 }}</span>
     </div>

     <div>
       <span>{{ inicioDeSesion }}</span>
     </div>
   </nav>
 </div>
</template>
```

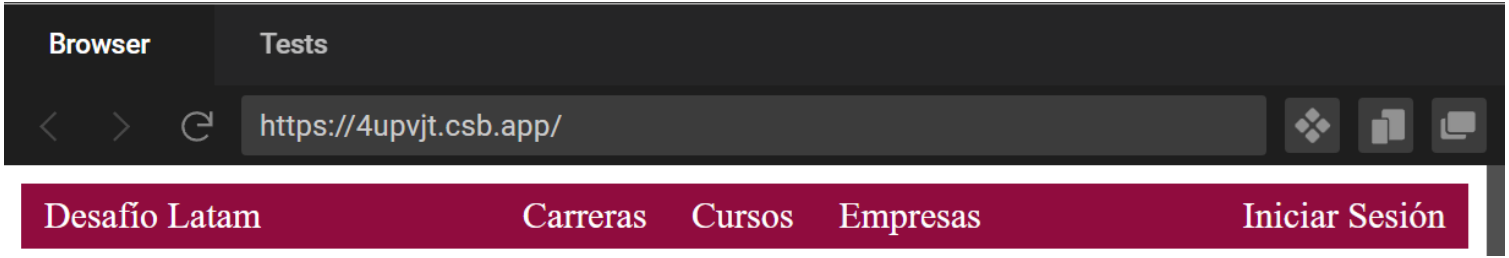
```
<script>
export default {
 name: "App",
 data() {
   return {
     nombreDelNegocio: "Desafío Latam",
     opcion1: "Carreras",
     opcion2: "Cursos",
     opcion3: "Empresas",
     inicioDeSesion: "Iniciar Sesión",
   };
 },
};
</script>
```

```
<style>
nav {
 background: #900c3e;
 color: white;
 padding: 5px 10px;
 display: flex;
 justify-content: space-between;
}

.opciones {
 flex-basis: 200px;
 display: flex;
 justify-content: space-between;
}
</style>
```

{desafío} latam_

Interpolación de variables

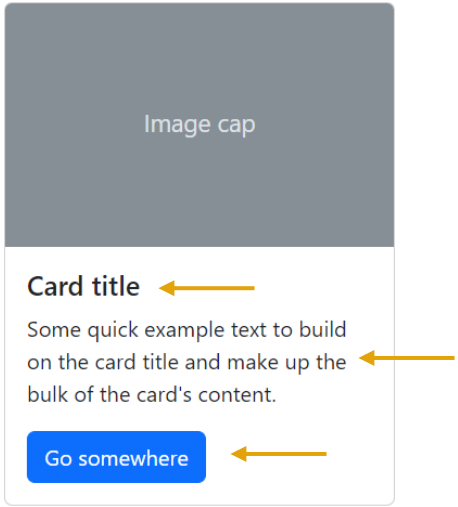
Nuestro resultado entonces será nuestro navbar planteado.



Ejercicio propuesto

Card interpolada

Replica el siguiente componente el contenido textual:



{desafío} latam_

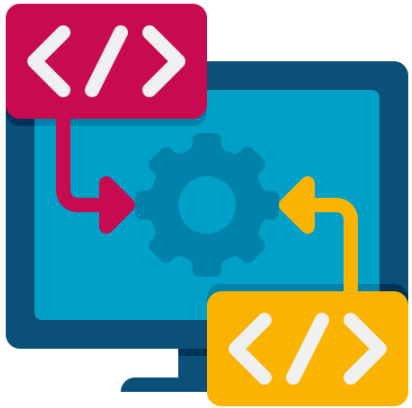
/* Framework */

Framework progresivo

- Nos permite escalar el tamaño del proyecto a medida que vayamos necesitamos más funcionalidades.

- Vue Js es entonces un framework progresivo porque a pesar que ya nos provee de varias herramientas, **está preparado para integrar diferentes plugins compatibles** que nos extienden sus capacidades.

- *Más adelante conoceremos librerías como Vue Router y Vuex que nos permitirá crear aplicaciones frontend profesionales.*



Beneficios de utilizar un framework

El tiempo que le podemos dedicar a una tarea depende de varios factores como nuestra destreza, conocimiento de la tecnología y las facilidades que ésta nos provea para cumplir un objetivo determinado.

Si comparamos el desarrollo de una tarea con JavaScript puro vs el uso de Vue, notaremos que el uso de frameworks tiene una ventaja en **tiempos** importante por las diversas herramientas que contiene para tareas.

Beneficios de utilizar un framework

Por ejemplo, una de las tareas más comunes del desarrollo es la **creación de un sistema de gestión de recursos**.

A este sistema se le conoce como **CRUD** (**C**reate **R**ead **U**pdate **D**elete).

La creación de estos sistemas con JavaScript desde 0, puede significar la “reconstrucción de la rueda”, mientras que los frameworks ya están pre dispuestos a estas situaciones y nos traen un conjunto de herramientas que nos faciliten las tareas.

/* Otras librerías o frameworks orientadas a componentes */

Otras librerías o frameworks

Vue Js es uno de varios frameworks y librerías dedicadas al desarrollo web.

Solo en el mundo de JavaScript podemos encontrarnos con los siguientes:

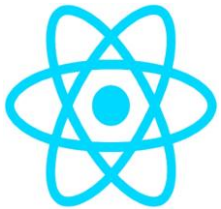
Todos estos están basados en componentes al igual que Vue pero **su forma de trabajar es diferente**.

/* Comparación entre React, Angular y Vue */

Comparación entre Angular, React y Vue

A pesar de que los 3 están basados en componentes y manejan varios conceptos en común, sus sintaxis son muy diferentes.

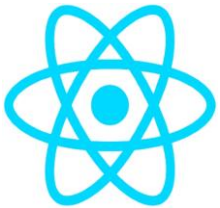
Revisemos cada uno:



Angular

```angular
import { Component } from "@angular/core";

@Component({
  selector: "app-root",
  templateUrl: "./app.component.html",
  styleUrls: ["./app.component.css"]
})
export class AppComponent {
  title = "CodeSandbox";
}
```

React



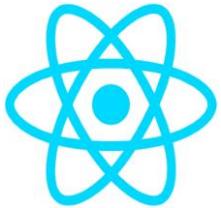
```jsx
import "./styles.css";

export default function App() {
  return (
    <div className="App">
      <h1>Hello CodeSandbox</h1>
      <h2>Start editing to see some magic happen!</h2>
    </div>
  );
}
```

Vue

```vue
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png" width="25%">
    <HelloWorld msg="Hello Vue in CodeSandbox!"/>
  </div>
</template>

<script>
import HelloWorld from "./components/HelloWorld";

export default {
  name: "App",
  components: {
    HelloWorld
  }
};
</script>

<style>
#app {
  font-family: "Avenir", Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

Vue

Tabla comparativa



Usa JavaScript	Usa TypeScript	Usa JavaScript
Es un framework que puede crecer progresivamente	Es un framework que contiene por defecto muchas herramientas	Es una librería que puede integrarse a proyectos ya creados
Maneja el concepto de estado	No maneja el concepto de estado	Maneja el concepto de estado
La comunidad lo mantiene	Google lo mantiene	Meta(facebook) lo mantiene

- No es necesario aprender todos los frameworks para empezar a trabajar.
- Cada librería tiene un estilo particular.
- Lo ideal es trabajar en el que te haga sentir más cómodo.

Próxima sesión...

- *Guía de ejercicios.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales