

**PREGRADO**



UNIDAD 2 | FRONTEND WEB APPLICATIONS

# **VUE FUNDAMENTALS**

Al finalizar la unidad, el estudiante desarrolla y comunica una solución de aplicación frontend integrada con RESTful web services con una Arquitectura Orientada a Servicios en un ambiente de desarrollo ágil e inclusivo

---

# AGENDA

OBJECTS & FUNCTIONS

PROTOTYPES

CLASSES



# ¿Qué es Vue?

Es un framework de JavaScript, orientado al desarrollo de interfaces de usuario.

Se basa en tecnologías standard como HTML, CSS, JavaScript.

Ofrece modelos de programación:

- Declarative
- Component-based

# ¿Qué es Vue?

Es un framework de JavaScript, orientado al desarrollo de interfaces de usuario.

Se basa en tecnologías standard como HTML, CSS, JavaScript.

Ofrece modelos de programación:

- Declarative
- Component-based

# Declarative rendering and reactivity

Vue proporciona extensiones al lenguaje HTML, con una sintaxis de templates que permite describir de forma declarativa HTML output en base a estados de JavaScript.

Vue automaticamente sigue los cambios de estado de JavaScript y de forma eficiente actualiza el DOM cuando se producen cambios.

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

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

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

# Single-File component

En la mayoría de casos, se crea Vue componets usando un formato al estilo HTML llamado Single-File Component (conocido como \*.vue, abreviado SFC).

SFC encapsula la lógica del componente (JavaScript), el template (HTML) y estilos (CSS) en un solo archivo.

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  }
}
</script>
```

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

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

# API Styles

Dos estilos:

Options API.

Composition API.



# Options API

Permite definir la lógica del componente usando un objeto con opciones como *data*, *methods* y *mounted*.

Las propiedades que se definen en las opciones son expuestas en *this* dentro de funciones.

```
<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 listeners 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>
```

# Composition API

Se define la lógica del componente usando funciones de API importadas.

Se usa con `<script setup>`.

El atributo setup es un *hint* que hace que Vue realice ajustes en compile-time que permitan utilizar Composition API con menos código.

Por ejemplo, imports y top-level variables/functions declaradas en `<script setup>` se utilizan directamente en el template.

```
<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>
```

# Creating a Vue project

Requisitos:

Acceso a command line

Instalar Node.js version  $\geq 15.0$

Para crear un proyecto soportado por build-tools, se ejecuta el siguiente comando (> es el OS prompt):

```
> npm init vue@latest
```

# Creating a Vue project

Con el comando anterior se instala y ejecuta *create-vue*, el scaffolding tool oficial.

- ✓ 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 Cypress for both Unit and End-to-End testing? ... [No](#) / Yes
- ✓ Add ESLint for code quality? ... [No](#) / Yes
- ✓ Add Prettier for code formatting? ... [No](#) / Yes

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

# Install dependencies and run

Ejecutar estos comandos para ubicarse en el directorio del proyecto, instalar las dependencias y ejecutarlo.

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

# Lifecycle Hooks

Algunos hooks:

Setup

onBeforeMounted

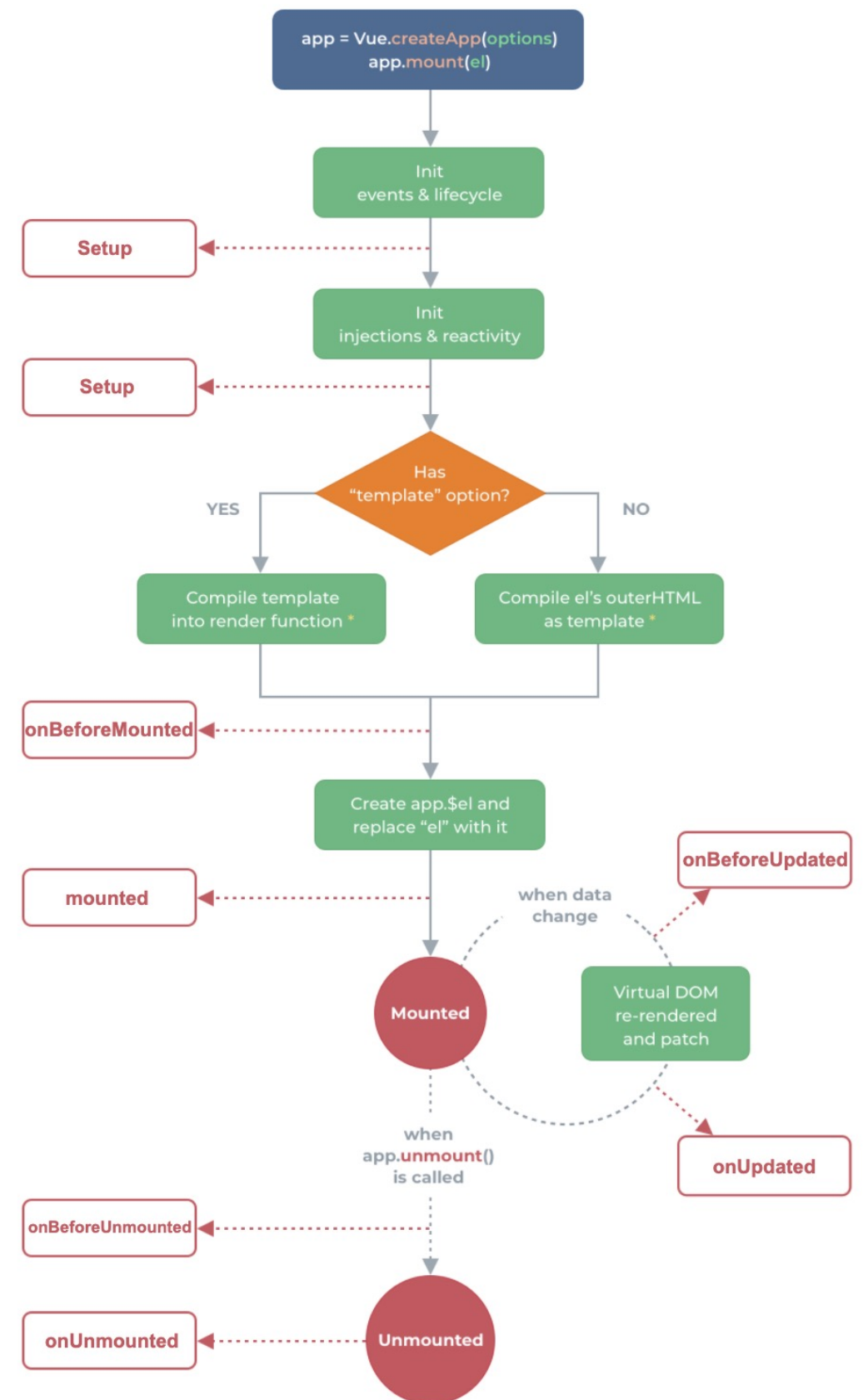
Mounted

onBeforeUpdated

onUpdated

onBeforeUnmounted

onUnmounted



---

# RESUMEN

Recordemos

Qué es Vue

Qué modelos de programación soporta

Cómo se crea un proyecto.

Qué lifecycle hooks incluye.



---

# REFERENCIAS

Para profundizar

<https://vuejs.org>





# PREGRADO

## Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería



### UPC

Universidad Peruana  
de Ciencias Aplicadas

Prolongación Primavera 2390,  
Monterrico, Santiago de Surco  
Lima 33 - Perú  
T 511 313 3333  
<https://www.upc.edu.pe>

***exígete, innova***