Vue.js que incluye una explicación general del framework, ejemplos básicos y referencias documentales que los alumnos pueden consultar para profundizar.

¿Qué es Vue.js?

Vue.js es un **framework progresivo de JavaScript** utilizado para construir interfaces de usuario (UI). A diferencia de otros frameworks como Angular o React, Vue está diseñado para ser adoptado de manera incremental. Esto significa que puedes empezar con funcionalidades básicas, como integrar Vue en una página HTML simple, y luego ampliar la aplicación a algo más complejo.

Características clave de Vue.js:

1. **Reactividad**: Vue proporciona un sistema de reactividad sencillo y eficiente que facilita la vinculación entre datos y el DOM.
2. **Componentes**: Vue permite crear componentes reutilizables, que son piezas de UI que encapsulan HTML, CSS y JavaScript.
3. **Enfoque progresivo**: Puedes usar Vue en una parte pequeña de tu proyecto y luego expandirlo sin necesidad de reescribir todo el código.
4. **Facilidad de uso**: Es fácil de aprender para quienes ya tienen conocimientos de HTML, CSS y JavaScript, lo que lo hace ideal para estudiantes que están comenzando con el desarrollo de aplicaciones web.

Instalación de Vue.js

Hay dos formas principales de usar Vue.js:

1. **Directamente desde un CDN**: Para proyectos pequeños o demostraciones.
2. **Instalación mediante Node.js y npm**: Para proyectos más grandes y aplicaciones más avanzadas.

Usar Vue.js con CDN

Una manera fácil de empezar es incluir Vue.js directamente en un archivo HTML a través de un CDN. Aquí hay un ejemplo básico:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vue.js Hola Mundo</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
</head>
<body>
  <div id="app">
    <h1>{{ message }}</h1>
  </div>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        message: 'Hola Mundo desde Vue.js'
      }
    });
  </script>
</body>
</html>
```

En este ejemplo, se está utilizando la versión de Vue 2. Vue enlaza el dato message del objeto data directamente en el DOM con el uso de {{ message }}, lo que se conoce como **Interpolación**.

Usar Vue.js con Node.js y npm

Si deseas crear una aplicación más compleja, es recomendable instalar Vue con npm. Estos son los pasos para empezar:

1**. Instala Node.js** si aún no lo tienes. (Esto ya lo hemos cubierto en temas anteriores)
2. **Crea un proyecto con Vue CLI**:

```
npm install -g @vue/cli
vue create my-project
cd my-project
npm run serve
```

Esto configurará un entorno de desarrollo completo para Vue.js con herramientas modernas como Webpack, lo que te permite construir aplicaciones avanzadas.

Componentes en Vue.js

Vue permite crear **componentes** que encapsulan código HTML, CSS y JavaScript, haciendo el código más modular y reutilizable.

Aquí tienes un ejemplo básico de un componente en Vue.js:

```
Vue.component('saludo', {
```

```
  template: '<h1>{{ saludo }}</h1>',
  data: function() {
    return {
      saludo: '¡Hola desde un componente Vue!'
    }
  }
});

var app = new Vue({
  el: '#app'
});
```

Y en el archivo HTML:

```
<div id="app">
  <saludo></saludo>
</div>
```

Este componente llamado saludo puede ser reutilizado en cualquier parte del proyecto simplemente agregando la etiqueta <saludo></saludo>.

Reactividad y Directivas

La reactividad es una de las principales características de Vue. En Vue, los datos reactivos son los que actualizan automáticamente el DOM cuando cambian.

Por ejemplo, puedes usar la directiva v-model para vincular el valor de un campo de entrada con un dato:

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

```
  <input v-model="message">
</div>

<script>
  var app = new Vue({
    el: '#app',
    data: {
      message: 'Escribe algo aquí...'
    }
  });
</script>
```

Cuando escribes algo en el campo de entrada, el contenido del párrafo se actualiza automáticamente.

Ciclo de Vida de un Componente

Vue tiene un ciclo de vida muy claro para cada componente, lo que permite ejecutar código en varios puntos clave del ciclo de vida. Algunas de las etapas más importantes son:

- **created**: Cuando el componente ha sido creado.
- **mounted**: Cuando el componente ha sido montado en el DOM.
- **updated**: Cuando los datos del componente han cambiado.
- **destroyed**: Cuando el componente ha sido destruido.

Por ejemplo, puedes agregar un “log” al cargar un componente:

```
new Vue({
  el: '#app',
  data: {
```

```
    message: 'Hola!'
  },
  created: function() {
    console.log('Componente creado');
  },
  mounted: function() {
    console.log('Componente montado');
  }
});
```

Manejo de Eventos

En Vue, puedes manejar eventos de usuario de manera simple usando la directiva v-on. Aquí hay un ejemplo de cómo manejar el evento click:

```
<div id="app">
  <button v-on:click="mostrarAlerta">Haz clic aquí</button>
</div>

<script>
  var app = new Vue({
    el: '#app',
    methods: {
      mostrarAlerta: function() {
        alert('¡Botón clickeado!');
      }
    }
  });
</script>
```

Recursos adicionales para aprender Vue.js

Aquí tienes algunas referencias documentales que pueden ayudar a tus estudiantes a profundizar en Vue.js:

1. **Documentación oficial de Vue.js**:
   - https://vuejs.org/
   - La mejor fuente para aprender Vue, con guías detalladas y una API completa.
2. **Guía Vue en español**:
   - https://es.vuejs.org/
   - Versión en español del sitio oficial.
   - **Vue Mastery (Cursos interactivos)**: https://www.vuemastery.com/
   - Cursos gratuitos y premium para aprender Vue.js.
3. **Curso de Vue.js en YouTube (Ejemplos Prácticos)**:
   - Vue.js Crash Course 2021 - YouTube
   - Un curso introductorio muy completo para principiantes.
4. **Repositorio GitHub de Vue.js**:
   - https://github.com/vuejs/vue
   - Código fuente del proyecto Vue.js, ideal para los estudiantes que quieran explorar cómo está construido el framework.

Conclusión

Vue.js es una excelente opción para el desarrollo de aplicaciones web modernas. Su enfoque progresivo y fácil de aprender lo convierte en una gran herramienta para enseñar a estudiantes cómo construir interfaces de usuario reactivas y componentes reutilizables. Esta introducción debería ayudar a que tus alumnos comprendan los fundamentos y comiencen a experimentar con Vue en sus propios proyectos.