Comparación: JavaScript Puro vs Vue.js

1. Manipulación del Estado

JavaScript Puro

javascript

```javascript
// Variables para el estado de la aplicación
let mensajeEvento = '';
let numCopias = 0;
let historialAcciones = [];

// Necesitamos una función para actualizar la UI manualmente
function actualizarUI() {
  // Actualizar mensaje
  if (mensajeEvento) {
    mensajeDiv.textContent = mensajeEvento;
    mensajeDiv.style.display = 'block';
  } else {
    mensajeDiv.style.display = 'none';
  }

  // Actualizar contador
  contadorCopias.textContent = numCopias;

  // Actualizar historial
  actualizarHistorial();
}
```

Vue.js

javascript

```javascript
// Estado centralizado en data()
data() {
  return {
    mensajeActivo: false,
    mensajeTexto: '',
    tipoMensaje: 'copia',
    contadores: {
      copias: 0,
      cortes: 0,
      pegados: 0
    },
    historial: []
  }
}
// No necesitamos una función para actualizar la UI,
// Vue lo hace automáticamente cuando cambian los datos
```

2. Selección de Elementos DOM

JavaScript Puro

javascript

```javascript
// Debemos seleccionar cada elemento del DOM manualmente
const areaTexto = document.getElementById('area-texto');
const mensajeDiv = document.getElementById('mensaje');
const btnLimpiar = document.getElementById('btn-limpiar');
const contadorCopias = document.getElementById('contador-copias');
```

Vue.js

javascript

```javascript
// No necesitamos seleccionar elementos DOM,
// Vue maneja esto automáticamente
// Accedemos directamente a las propiedades reactivas
```

3. Manejo de Eventos

JavaScript Puro

javascript

```javascript
// Añadir listeners de eventos manualmente
areaTexto.addEventListener('copy', detectarCopia);
areaTexto.addEventListener('cut', detectarCorte);
btnLimpiar.addEventListener('click', limpiarMensaje);
```

Vue.js

html

```html
<!-- Eventos declarados directamente en el template -->
<textarea
  v-model="textoArea"
  @copy="detectarCopia"
  @cut="detectarCorte"
  @paste="detectarPegado"
></textarea>
<button @click="limpiarMensaje">Limpiar mensaje</button>
```

4. Renderizado Condicional

JavaScript Puro

javascript

```javascript
// Mostrar/ocultar elementos modificando sus estilos
if (mensajeEvento) {
  mensajeDiv.textContent = mensajeEvento;
  mensajeDiv.style.display = 'block';
} else {
  mensajeDiv.style.display = 'none';
}
```

Vue.js

html

```html
<!-- Usar v-if para renderizado condicional -->
<div
  v-if="mensajeActivo"
  :class="['mensaje', `tipo-${tipoMensaje}`]"
>
  {{ mensajeTexto }}
</div>
```

5. Enlace de Datos (Data Binding)

JavaScript Puro

```javascript
// Actualizar manualmente el contenido de los elementos
mensajeDiv.textContent = mensajeEvento;
contadorCopias.textContent = numCopias;

// Para formularios necesitamos eventos adicionales
areaTexto.addEventListener('input', function(e) {
  textoDelArea = e.target.value;
});
```

Vue.js

```html
<!-- Enlace automático de datos -->
<div>{{ mensajeTexto }}</div>
<span>{{ contadores.copias }}</span>

<!-- Enlace bidireccional con v-model -->
<textarea v-model="textoArea"></textarea>
```

6. Renderizado de Listas

JavaScript Puro

```javascript
// Limpiar y recrear elementos de lista manualmente
function actualizarHistorial() {
  listaHistorial.innerHTML = '';

  historialAcciones.forEach(function(accion) {
    const item = document.createElement('div');
    item.className = 'historial-item';
    item.textContent = accion;
    listaHistorial.appendChild(item);
  });
}
```

Vue.js

html

```
<!-- Directiva v-for para renderizado de listas -->
<div
  v-for="(accion, index) in historialLimitado"
  :key="index"
  class="historial-item"
>
  {{ accion }}
</div>
```

7. Clases y Estilos Dinámicos

JavaScript Puro

javascript

```
// Cambiar clases manualmente
function actualizarClaseMensaje(tipo) {
  mensajeDiv.className = 'mensaje';
  if (tipo === 'copia') mensajeDiv.classList.add('tipo-copia');
  if (tipo === 'corte') mensajeDiv.classList.add('tipo-corte');
  if (tipo === 'pegado') mensajeDiv.classList.add('tipo-pegado');
}
```

Vue.js

html

```
<!-- Enlace de clases y estilos dinámicos -->
<div
  v-if="mensajeActivo"
  :class="['mensaje', `tipo-${tipoMensaje}`]"
>
  {{ mensajeTexto }}
</div>
```

8. Propiedades Computadas

JavaScript Puro

javascript

```javascript
// Necesitamos recalcular y actualizar manualmente cada vez
function obtenerHistorialLimitado() {
  return historialAcciones.slice(0, 5);
}

// Y llamarlo cada vez que cambia historialAcciones
```

Vue.js

javascript

```javascript
// Las propiedades computadas se actualizan automáticamente cuando sus dependencias cambi
computed: {
  historialLimitado() {
    return this.historial.slice(0, 5);
  }
}
```

9. Ciclo de Vida de la Aplicación

JavaScript Puro

javascript

```javascript
// Esperar a que el DOM esté cargado antes de inicializar
document.addEventListener('DOMContentLoaded', function() {
  // Inicialización de la aplicación
  // Selección de elementos
  // Configuración de event listeners
});
```

Vue.js

javascript

```javascript
// Vue maneja el ciclo de vida automáticamente
// La aplicación se inicia cuando createApp().mount() se ejecuta
const { createApp } = Vue;
createApp({
  // Configuración de la aplicación
}).mount('#app');

// También hay hooks de ciclo de vida
// mounted() {
//   // Se ejecuta cuando la app está montada en el DOM
// }
```

10. Organización del Código

JavaScript Puro

javascript

```javascript
// El código a menudo se mezcla: selección de elementos,
// manipulación del DOM, lógica de negocio, etc.

// Funciones separadas pero sin estructura clara
function detectarCopia() { /* ... */ }
function actualizarUI() { /* ... */ }
function limpiarMensaje() { /* ... */ }
```

Vue.js

javascript

```javascript
// Separación clara del código por responsabilidad

// Template (Vista)
// <template>...</template>

// Lógica (Controlador/Modelo)
// data() { ... }       // Estado
// computed: { ... }    // Valores derivados
// methods: { ... }     // Comportamiento

// El código está naturalmente organizado
```

Analogías para Explicar las Diferencias

JavaScript Puro vs Vue.js

Analogía de la Pizzería:

- **JavaScript Puro** es como hacer una pizza completamente a mano: mezclar la masa, preparar la salsa, rallar el queso, hornear. Tienes control total sobre cada paso, pero requiere más trabajo.
- **Vue.js** es como usar ingredientes pre-preparados de calidad: masa ya estirada, salsa lista, queso rallado. Sigues haciendo la pizza, pero te concentras en la creatividad y el sabor, no en las tareas repetitivas.

Analogía del Control Remoto:

- En **JavaScript Puro**, tienes que levantarte, caminar hasta el televisor y presionar botones físicos para cambiar el canal, ajustar el volumen, etc.
- Con **Vue.js**, tienes un control remoto que te permite hacer esos cambios sin moverte. El control remoto (Vue) maneja la complejidad de comunicarse con el televisor (DOM).

Analogía de la Construcción:

- **JavaScript Puro** es como construir una casa ladrillo por ladrillo, mezclando tu propio cemento, cortando cada tabla.
- **Vue.js** es como usar secciones prefabricadas bien diseñadas. Sigues construyendo la casa, pero mucho del trabajo repetitivo ya está hecho.

Cuándo Usar Cada Enfoque

Usa JavaScript Puro cuando:

- Estás aprendiendo los fundamentos
- Trabajas en un proyecto muy pequeño
- Necesitas máximo control y mínima sobrecarga
- Estás añadiendo interactividad a una página existente
- Tienes que dar soporte a navegadores muy antiguos

Usa Vue.js cuando:

- Trabajas en aplicaciones de mediana a gran escala
- Necesitas organizar tu código en componentes reutilizables
- Quieres una forma más declarativa de programar
- Necesitas mantener sincronizados la UI y los datos
- Trabajas en equipo y necesitas patrones consistentes