GUÍA COMPLETA DE DESARROLLO WEB MODERNO

Fundamentos de JavaScript y Desarrollo de Aplicaciones con Vue.js

INTRODUCCIÓN

Esta guía comprehensive cubre los fundamentos esenciales de JavaScript moderno y las tecnologías de desarrollo web contemporáneas, preparándote para crear aplicaciones web robustas y eficientes.

SECCIÓN 0: FUNDAMENTOS DE JAVASCRIPT

0.1 VARIABLES Y TIPOS DE DATOS

Declaración de Variables

```javascript
// Formas de declarar variables
var variableAntigua = "No recomendada"; // Ámbito de función, puede redeclarse
let variableCambiante = "Puede cambiar"; // Ámbito de bloque
const variableConstante = "No cambia"; // Ámbito de bloque, no puede reasignarse

// Tipos de datos primitivos
let numero = 42; // Number
let texto = "Hola mundo"; // String
let booleano = true; // Boolean
let nulo = null; // Null
let indefinido = undefined; // Undefined
let simbolo = Symbol("descripción"); // Symbol (único)
let bigInt = 1234567890123456789012345678901234567890n; // BigInt
```

0.2 OPERADORES

Operadores Aritméticos y de Comparación

javascript

```javascript
// Operadores aritméticos
let suma = 5 + 3; // 8
let resta = 10 - 4; // 6
let multiplicacion = 3 * 4; // 12
let division = 15 / 3; // 5
let modulo = 10 % 3; // 1 (resto de la división)

// Operadores de comparación
let igual = 5 == "5"; // true (compara valor)
let estrictamenteIgual = 5 === "5"; // false (compara valor y tipo)
let mayorQue = 10 > 5; // true
let mayorOIgual = 10 >= 10; // true

// Operadores lógicos
let y = true && false; // false
let o = true || false; // true
let no = !true; // false
```

0.3 ESTRUCTURAS DE CONTROL

Condicionales y Bucles

javascript

```javascript
// Condicional if-else
if (condicion) {
  // Código si es verdadero
} else if (otraCondicion) {
  // Código si otra condición es verdadera
} else {
  // Código por defecto
}

// Operador ternario
let resultado = condicion ? valorSiVerdadero : valorSiFalso;

// Switch
switch (variable) {
  case valor1:
    // Código para valor1
    break;
  case valor2:
    // Código para valor2
    break;
  default:
    // Código por defecto
}

// Bucles
// For
for (let i = 0; i < 5; i++) {
  // Código que se repite
}

// While
while (condicion) {
  // Código que se repite mientras condicion sea verdadera
}

// Do-While
do {
  // Código que se ejecuta al menos una vez
} while (condicion);
```

0.4 FUNCIONES MODERNAS

Tipos de Funciones

javascript

```javascript
// Función tradicional
function sumar(a, b) {
  return a + b;
}

// Función flecha (Arrow Function)
const sumarFlecha = (a, b) => a + b;

// Función con parámetros por defecto
const saludar = (nombre = "Mundo") => `Hola ${nombre}`;

// Función con número variable de argumentos
const sumarTodos = (...numeros) => {
  return numeros.reduce((total, num) => total + num, 0);
};
```

0.5 MANIPULACIÓN DE ARRAYS

Métodos Modernos de Array

javascript

```javascript
const numeros = [1, 2, 3, 4, 5];

// map: Transformar elementos
const dobles = numeros.map(num => num * 2);
// [2, 4, 6, 8, 10]

// filter: Filtrar elementos
const pares = numeros.filter(num => num % 2 === 0);
// [2, 4]

// reduce: Combinar elementos
const suma = numeros.reduce((total, num) => total + num, 0);
// 15

// Otros métodos útiles
const primerPar = numeros.find(num => num % 2 === 0);
const tienePares = numeros.some(num => num % 2 === 0);
const todosSonPositivos = numeros.every(num => num > 0);
```

0.6 OBJETOS Y DESESTRUCTURACIÓN

Trabajando con Objetos

javascript

```javascript
// Creación de objetos
const persona = {
  nombre: "Juan",
  edad: 30,
  ciudad: "Madrid"
};

// Desestructuración de objetos
const { nombre, edad } = persona;

// Spread operator
const copiaPersona = { ...persona, profesion: "Desarrollador" };

// Desestructuración de arrays
const colores = ["rojo", "verde", "azul"];
const [primerColor, segundoColor] = colores;
```

0.7 PROMESAS Y ASYNC/AWAIT

Manejo de Operaciones Asíncronas

javascript

```javascript
// Promesa básica
const miPromesa = new Promise((resolve, reject) => {
  if (/* condicion */) {
    resolve("Éxito");
  } else {
    reject("Error");
  }
});

// Async/Await
async function obtenerDatos() {
  try {
    const respuesta = await fetch('https://api.ejemplo.com/datos');
    const datos = await respuesta.json();
    return datos;
  } catch (error) {
    console.error("Error al obtener datos:", error);
  }
}
```

SECCIÓN 1: FUNDAMENTOS DE VUE.JS

1.1 REACTIVIDAD: EL CORAZÓN DE VUE.JS

Conceptos Clave de Reactividad

- Sistema que actualiza automáticamente la interfaz cuando cambian los datos
- Vue rastrea las dependencias de los datos
- Actualiza solo los componentes afectados por un cambio

Declaración de Datos

```javascript
// Options API
export default {
  data() {
    return {
      mensaje: 'Hola Vue',
      contador: 0
    }
  }
}

// Composition API (Vue 3)
import { ref, reactive } from 'vue'
export default {
  setup() {
    const mensaje = ref('Hola Vue')
    const estado = reactive({
      contador: 0
    })
    return { mensaje, estado }
  }
}
```

1.2 DIRECTIVAS DE VUE

Directivas Fundamentales

html

```
<!-- v-model: Enlace bidireccional -->
<input v-model="nombreUsuario">

<!-- v-if / v-else / v-show: Renderizado condicional -->
<div v-if="condicion">Mostrar si es verdadero</div>
<div v-else>Mostrar si es falso</div>
<div v-show="condicion">Alterna display con CSS</div>

<!-- v-for: Iteración sobre listas -->
<li v-for="(item, index) in lista" :key="item.id">
  {{ item.nombre }}
</li>

<!-- Eventos y modificadores -->
<button @click.prevent="enviarFormulario">Enviar</button>
```

1.3 COMPONENTES EN VUE

Estructura de un Componente

vue

```
<template>
  <!-- Estructura HTML -->
  <div>{{ titulo }}</div>
</template>

<script>
export default {
  // Lógica del componente
  props: ['titulo'],
  methods: {
    manejarAccion() {
      // Método de componente
    }
  }
}
</script>

<style scoped>
/* Estilos específicos del componente */
</style>
```

Comunicación entre Componentes

vue

```vue
<!-- Props (Padre a Hijo) -->
<hijo-componente :mensaje="mensajePadre" />

<!-- Eventos (Hijo a Padre) -->
<!-- Hijo -->
<button @click="$emit('accion', datos)">Enviar</button>

<!-- Padre -->
<hijo-componente @accion="manejarAccion" />
```

SECCIÓN 2: AG GRID

2.1 CONFIGURACIÓN BÁSICA

javascript

```javascript
const columnDefs = [
  {
    headerName: 'País',
    field: 'name',
    sortable: true,
    filter: true,
    // Renderizado personalizado
    cellRenderer: (params) => {
      return `<img src="${params.data.bandera}" width="30">`
    }
  }
]

const gridOptions = {
  columnDefs: columnDefs,
  rowData: datos,
  pagination: true,
  paginationPageSize: 10
}
```

SECCIÓN 3: TAILWIND CSS

3.1 CLASES UTILITARIAS

```html
<div class="bg-blue-500 text-white p-4 rounded-lg">
  Contenedor con Tailwind
</div>
```

3.2 DISEÑO RESPONSIVE

```html
<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
  <!-- Columnas responsivas -->
</div>
```

PREGUNTAS DE VERIFICACIÓN PROFUNDA

Conceptuales de JavaScript

1. Explica la diferencia entre `var`, `let` y `const`
2. ¿Qué son los tipos de datos primitivos?
3. Describe el funcionamiento de los operadores lógicos
4. Explica el concepto de closures
5. ¿Cómo funcionan las promesas y async/await?

Conceptuales de Vue.js

1. Explica la reactividad en Vue
2. Diferencias entre Options API y Composition API
3. ¿Cómo funcionan las directivas en Vue?
4. Métodos de comunicación entre componentes
5. Explica el ciclo de vida de un componente Vue

Técnicas de Implementación

1. Estrategias para manejo de estado
2. Técnicas de optimización en Vue
3. Implementación de filtros y búsqueda
4. Manejo de eventos complejos
5. Renderizado condicional y sus casos de uso

CONSEJOS FINALES

1. Practica constantemente

2. Construye proyectos reales
3. Mantén el código limpio y organizado
4. Aprende de la comunidad
5. Mantente actualizado con las tecnologías

RECURSOS ADICIONALES

JavaScript

- MDN Web Docs
- JavaScript.info
- Eloquent JavaScript
- You Don't Know JS

Vue.js

- Documentación Oficial
- Vue Mastery
- Frontend Masters

Comunidades

- Stack Overflow
- Dev.to
- Reddit (r/webdev, r/vuejs)

CONCLUSIÓN

Esta guía es tu punto de partida para convertirte en un desarrollador web moderno. Recuerda que la práctica constante y la curiosidad son tus mejores aliados en el aprendizaje.

¡Desarrolla, aprende y diviértete!