FullStack Developer

CDN’s, librerías y frameworks JavaScript

¿Qué son los CDN en JavaScript?

CDN (Content Delivery Network): Una red de servidores distribuidos geográficamente que trabajan juntos para proporcionar una entrega rápida de contenido en Internet.

JavaScript y CDN: Utilizar un CDN para alojar archivos JavaScript (como librerías o frameworks) permite que estos sean accesibles más rápidamente y desde múltiples ubicaciones alrededor del mundo.

Beneficios de Usar CDN

Velocidad de Carga: Reduce el tiempo de carga al servir el contenido desde un servidor cercano al usuario.

Disponibilidad: Alta disponibilidad y redundancia, disminuyendo la posibilidad de caídas.

Reducción de Carga en el Servidor: Distribuye la carga de tráfico, permitiendo que tu servidor se enfoque en otras tareas.

Optimización de la Cache: Los CDN suelen tener mecanismos de cache avanzados para servir contenido más rápido.

¿Qué es una Librería en JavaScript?

Librería de JavaScript: Un conjunto de funciones y módulos preescritos que los desarrolladores pueden usar para realizar tareas comunes de programación de manera más eficiente. Las librerías facilitan y agilizan el desarrollo al proporcionar soluciones reutilizables para problemas comunes.

Características de las Librerías

•**Reusabilidad**: Las librerías están diseñadas para ser reutilizadas en diferentes proyectos, reduciendo el esfuerzo necesario para escribir código desde cero.

•**Modularidad**: Suelen estar compuestas de módulos o componentes independientes que se pueden utilizar de manera aislada o en conjunto.

•**Facilidad de Uso**: Proporcionan una interfaz simplificada y funciones predefinidas que facilitan la implementación de funcionalidades complejas.

•**Mantenimiento y Actualización**: Las librerías populares son mantenidas y actualizadas regularmente por comunidades de desarrolladores o empresas, lo que garantiza la corrección de errores y la introducción de mejoras.

Ventajas de Usar Librerías

•**Ahorro de Tiempo**: Permiten a los desarrolladores enfocarse en la lógica de la aplicación en lugar de en detalles de implementación.

•**Reducción de Errores**: Las funciones preescritas y probadas reducen la probabilidad de errores en el código.

•**Estandarización**: Promueven el uso de buenas prácticas y patrones de desarrollo estándar.

Consideraciones al Elegir una Librería

•**Popularidad y Comunidad**: Una librería con una comunidad activa y numerosa suele ser más confiable y tiene más recursos de soporte.

•**Documentación**: Una buena documentación facilita la comprensión y el uso de la librería.

•**Compatibilidad**: Asegúrate de que la librería sea compatible con tu proyecto y con otras herramientas que estés utilizando.

•**Tamaño y Rendimiento**: Considera el impacto en el rendimiento de tu aplicación, especialmente si se trata de una librería grande.

Uso de librerías mediante CDN’s

Animate.css

Es una biblioteca de animaciones CSS listas para usar que puedes integrar fácilmente en tus proyectos web para añadir efectos visuales atractivos a tus elementos HTML.

Características Principales

•**Facilidad de Uso**: Solo necesitas agregar clases CSS a tus elementos HTML para aplicar animaciones.

•**Amplia Variedad de Animaciones**: Ofrece una gran variedad de animaciones, desde simples desvanecidos hasta efectos complejos como rebotes y deslizamientos.

•**Compatibilidad**: Funciona en la mayoría de los navegadores modernos.

•**Customización**: Puedes personalizar la duración, el retraso y otros aspectos de las animaciones usando CSS estándar.

Beneficios

•**Rápida Implementación**: Añadir animaciones a tus proyectos es rápido y sencillo, lo que acelera el proceso de desarrollo.

•**Mejora la Experiencia del Usuario**: Las animaciones pueden hacer que las interfaces de usuario sean más atractivas e intuitivas.

•**Consistencia**: Al usar una biblioteca estándar, garantizas que las animaciones sean consistentes en todos tus proyectos.

Cómo Usar Animate.css

Incluir Animate.css desde un CDN

1. Podes incluir Animate.css en tu proyecto añadiendo un enlace a su archivo CSS alojado en un CDN.

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">
```

2. Aplicar Animaciones a los Elementos HTML

Añade las clases animate__animated y la clase de la animación que desees a tu elemento HTML.

```
<h1 class="animate__animated animate__bounce">CDN'S LIBRERÍAS Y FRAMEWORKS</h1>
```

Lista de Clases de Animaciones en Animate.css

1. **animate__bounce**
2. **animate__flash**
3. **animate__pulse**
4. **animate__rubberBand**
5. **animate__shakeX**
6. **animate__shakeY**
7. **animate__headShake**
8. **animate__swing**
9. **animate__tada**
10. **animate__wobble**
11. **animate__jello**
12. **animate__heartBeat**
13. **animate__bounceIn**
14. **animate__bounceInDown**
15. **animate__bounceInLeft**
16. **animate__bounceInRight**
17. **animate__bounceInUp**
18. **animate__bounceOut**
19. **animate__bounceOutDown**
20. **animate__bounceOutLeft**
21. **animate__bounceOutRight**
22. **animate__bounceOutUp**
23. **animate__fadeIn**
24. **animate__fadeInDown**
25. **animate__fadeInDownBig**
26. **animate__fadeInLeft**
27. **animate__fadeInLeftBig**
28. **animate__fadeInRight**

Gridjs.io

Es una librería de JavaScript ligera y modular para crear tablas interactivas. Grid.js es fácil de usar y se integra bien.

Características Principales

•**Ligereza**: Grid.js es una librería ligera que no sobrecarga tus proyectos.

•**Modularidad**: Permite añadir solo las funcionalidades que necesitas.

•**Integración con Frameworks Modernos**: facilitando su uso en proyectos basados en estos frameworks.

•**Interactividad**: Soporte para paginación, búsqueda, ordenación y personalización de las tablas.

•**Temas Personalizables**: Incluye temas prediseñados que pueden ser personalizados según las necesidades de tu proyecto.

Beneficios

•**Facilidad de Uso**: Configuración sencilla y rápida, ideal para desarrolladores que buscan implementar tablas interactivas sin mucha complejidad.

•**Flexibilidad**: Personalización fácil de las tablas mediante opciones y temas configurables.

•**Compatibilidad**: Funciona bien con proyectos que utilizan diferentes frameworks y librerías.

Cómo Usar Grid.js

Incluir Grid.js desde un CDN

Incluir Grid.js en tu proyecto añadiendo un enlace a su archivo CSS y un script a su archivo JavaScript alojado en un CDN.

```
<link href="https://cdn.jsdelivr.net/npm/gridjs/dist/theme/mermaid.min.css" rel="stylesheet" />
<script src="https://cdn.jsdelivr.net/npm/gridjs/dist/gridjs.umd.js"></script>
```

Crear una Tabla con Grid.js

•**En HTML**:

```
<div class="contenedor_tabla"></div>
<script src="main.js"></script>
```

•**En JS**:

```
const contenedorTabla = document.querySelector('.contenedor_tabla');

new gridjs.Grid({
    columns: ['Nombre', 'Email', 'Número de teléfono'],
    data: [
        ['John', 'john@example.com', '(353) 01 222 3333'],
        ['Mark', 'mark@gmail.com', '(01) 22 888 4444'],
    ]
}).render(contenedorTabla)
```

Mas configuraciónes

```
const contenedorTabla = document.querySelector('.contenedor_tabla');

new gridjs.Grid({
    columns: ['Nombre', 'Email', 'Número de teléfono'],
    data: [
        ['Carlos', 'carlos@example.com', '(51) 01 555 1234'],
        ['María', 'maria@example.com', '(54) 11 678 9101'],
        ['José', 'jose@example.com', '(57) 1 345 6789'],
        ['Ana', 'ana@example.com', '(58) 2 234 5678'],
        ['Luis', 'luis@example.com', '(52) 55 789 0123'],
        ['Carmen', 'carmen@example.com', '(56) 2 890 1234'],
        ['Jorge', 'jorge@example.com', '(53) 7 456 7890'],
        ['Elena', 'elena@example.com', '(51) 1 234 5678'],
        ['Miguel', 'miguel@example.com', '(54) 11 345 6789'],
        ['Lucía', 'lucia@example.com', '(57) 1 678 9101']
    ],
    sort: true,
    search: true,
    pagination: {
        enabled: true,
        limit: 3  // Número de filas por página
    },
}).render(contenedorTabla)
```

Explicación de las Propiedades

1.columns:

- **Descripción**: Define los nombres de las columnas que se mostrarán en la tabla.
- **Ejemplo en el Código**:

```
columns: ['Nombre', 'Email', 'Número de teléfono’]
```

- **Detalles**: En este caso, se definen tres columnas con los nombres "Nombre", "Email" y "Número de teléfono".

2.data:

- **Descripción**: Proporciona los datos que se mostrarán en la tabla. Es un array de arrays, donde cada sub-array representa una fila.
- **Ejemplo en el Código**:

```
data: [ ['Carlos', 'carlos@example.com', '(51) 01 555 1234'], ['María', 'maria@example.com', '(54) 11 678 9101'], // Más filas aquí ]
```

- **Detalles**: Cada sub-array contiene los datos correspondientes a una fila en la tabla, en el orden de las columnas definidas.

3.sort:

- **Descripción**: Habilita la capacidad de ordenar las columnas de la tabla.
- **Ejemplo en el Código**:

```
sort: true
```

- **Detalles**: Al habilitar esta propiedad, los usuarios pueden hacer clic en los encabezados de las columnas para ordenar los datos de manera ascendente o descendente.

4.search:

- **Descripción**: Habilita la funcionalidad de búsqueda en la tabla.
- **Ejemplo en el Código**:

```
search: true
```

- **Detalles**: Al habilitar esta propiedad, se añade un campo de búsqueda que permite a los usuarios filtrar los datos de la tabla.

5.pagination:

- **Descripción**: Configura la paginación de la tabla, permitiendo a los usuarios navegar entre diferentes páginas de datos.
- **Ejemplo en el Código**:

```
pagination: { enabled: true, limit: 3 }
```

- **Detalles**:
  - **enabled**: Si se establece en true, habilita la paginación en la tabla.
  - **limit**: Define el número de filas que se mostrarán por página. En este caso, se muestran 3 filas por página.

6.render(contenedorTabla):

- **Descripción**: Renderiza la tabla dentro del contenedor especificado.
- **Ejemplo en el Código**:

```
.render(contenedorTabla)
```

- **Detalles**: contenedorTabla es un elemento del DOM donde se va a renderizar la tabla. Se selecciona previamente con document.querySelector('.contenedor_tabla').

Type a keyword...

Nombre	Email	Número de teléfono
Carlos	carlos@example.com	(51) 01 555 1234
María	maria@example.com	(54) 11 678 9101
José	jose@example.com	(57) 1 345 6789

Showing **1** to **3** of **10** results

Previous | **1** | 2 | 3 | ... | 4 | Next

SweetAlert

Definición

Es una librería de JavaScript que permite crear alertas visualmente atractivas y personalizables en lugar de las alertas nativas del navegador.

Características Principales

•**Alertas Personalizables**: Permite personalizar el título, el texto, los iconos, los botones y el diseño de las alertas.

•**Variedad de Tipos de Alertas**: Soporta alertas de éxito, error, advertencia, información, y más.

•**Botones Personalizados**: Permite añadir y personalizar botones en las alertas.

•**Promesas**: Las alertas pueden retornar promesas, lo que facilita el manejo de acciones después de que el usuario interactúe con la alerta.

•**Integración Fácil**: Es fácil de integrar y usar en cualquier proyecto web.

Cómo Usar SweetAlert

Definición

Para incluir SweetAlert usamos un enlace a su archivo JavaScript alojado en un CDN.

```
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
```

En HTML:

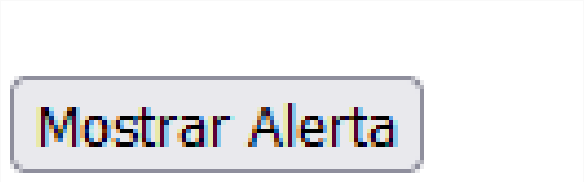
```
    <button id="btn-alert">Mostrar Alerta</button>

    <script src="main.js"></script>
```

En JS:

```
const btnAlert = document.getElementById('btn-alert');

btnAlert.addEventListener('click', function () {
    Swal.fire({
        title: '¡Éxito!',
        text: 'Esta es una alerta de éxito.',
        icon: 'success',
        confirmButtonText: 'Aceptar'
    });
});
```




jose@example.com
(57) 1 345 6789
¡Éxito!
Esta es una alerta de éxito.
Aceptar

Explicación del Ejemplo

1.Incluir SweetAlert:
- Se incluye la librería SweetAlert desde un CDN utilizando una etiqueta <script>.

2.Botón para Mostrar la Alerta:
- Se define un botón en el HTML con el id btn-alert.

3.Manejador de Evento:
- Se añade un manejador de evento click al botón que muestra una alerta cuando se hace clic en él.

4.Configuración de la Alerta:
- Se utiliza Swal.fire para crear y mostrar la alerta. Se configuran el título, el texto, el icono y el texto del botón de confirmación.

Tipos de Alertas en SweetAlert

Éxito:

```
Swal.fire({
    title: '¡Éxito!',
    text: 'Operación realizada con éxito.',
    icon: 'success',
    confirmButtonText: 'Aceptar'
});
```

Error:

```
Swal.fire({
    title: 'Error',
    text: 'Ha ocurrido un error.',
    icon: 'error',
    confirmButtonText: 'Aceptar'
});
```

Advertencia:

```
Swal.fire({
    title: 'Advertencia',
    text: '¿Estás seguro?',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonText: 'Sí, estoy seguro',
    cancelButtonText: 'Cancelar'
});
```

Información:

```
Swal.fire({
    title: 'Información',
    text: 'Este es un mensaje informativo.',
    icon: 'info',
    confirmButtonText: 'Aceptar'
});
```

Alertas con Botones Personalizados y Promesas

SweetAlert también permite configurar botones personalizados y manejar las acciones del usuario utilizando promesas.

```
Swal.fire({
    title: '¿Estás seguro?',
    text: '¡No podrás revertir esto!',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#3085d6',
    cancelButtonColor: '#d33',
    confirmButtonText: 'Sí, eliminarlo',
    cancelButtonText: 'Cancelar'
}).then((result) => {
    if (result.isConfirmed) {
        Swal.fire(
            '¡Eliminado!',
            'Tu archivo ha sido eliminado.',
            'success'
        )
    }
});
```

Chart.js

Definición

Es una librería de JavaScript que permite crear gráficos interactivos en aplicaciones web de manera fácil y flexible.

Características Principales

•**Tipos de Gráficos**: Soporta varios tipos de gráficos como lineales, de barras, radiales, de pastel, de burbuja y mixtos.

•**Interactividad**: Los gráficos son interactivos y pueden responder a eventos del usuario.

•**Personalización**: Ofrece múltiples opciones de personalización para el estilo y comportamiento de los gráficos.

•**Facilidad de Uso**: Es fácil de integrar y utilizar en cualquier proyecto web.

Incluir Chart.js desde un CDN

En HTML

```
<!-- Chart.js -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

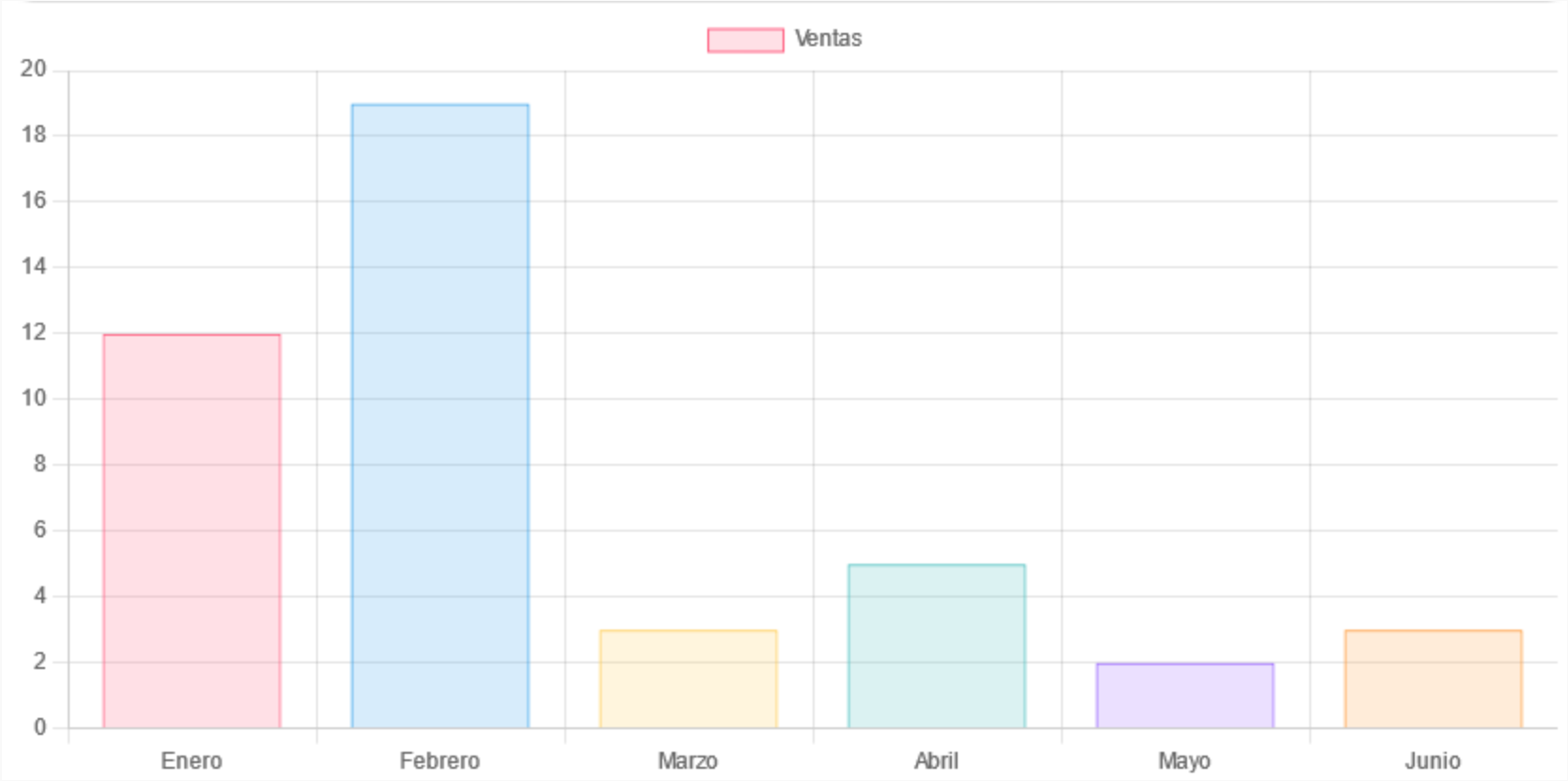
```
    <div>
        <canvas id="myChart"></canvas>
    </div>

    <script src="main.js"></script>
```

En JS

```
const ctx = document.getElementById('myChart').getContext('2d');

const myChart = new Chart(ctx, {
    type: 'bar', // Tipo de gráfico: 'bar', 'line', 'pie', etc.
    data: {
        labels: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio'],
        datasets: [{
            label: 'Ventas',
            data: [12, 19, 3, 5, 2, 3],
            backgroundColor: [
                'rgba(255, 99, 132, 0.2)',
                'rgba(54, 162, 235, 0.2)',
                'rgba(255, 206, 86, 0.2)',
                'rgba(75, 192, 192, 0.2)',
                'rgba(153, 102, 255, 0.2)',
                'rgba(255, 159, 64, 0.2)'
            ],
            borderColor: [
                'rgba(255, 99, 132, 1)',
                'rgba(54, 162, 235, 1)',
                'rgba(255, 206, 86, 1)',
                'rgba(75, 192, 192, 1)',
                'rgba(153, 102, 255, 1)',
                'rgba(255, 159, 64, 1)'
            ],
            borderWidth: 1
        }]
    },
    options: {
        scales: {
            y: {
                beginAtZero: true
            }
        }
    }
});
```


Ventas
20
18
16
14
12
10
8
6
4
2
0
Enero
Febrero
Marzo
Abril
Mayo
Junio

Framework

Un framework es una estructura de soporte sobre la cual un software puede ser construido. Proporciona un conjunto de herramientas, bibliotecas, y mejores prácticas que permiten a los desarrolladores crear aplicaciones de manera estructurada y eficiente. Un framework define el esqueleto o la arquitectura de la aplicación y a menudo controla el flujo del programa, siguiendo el patrón de diseño de "Inversión de Control" (IoC).

Framework vs. Librería

Framework

•**Control de Flujo**: El framework tiene el control del flujo del programa y llama al código del desarrollador en momentos específicos.

•**Estructura**: Proporciona una estructura definida y una arquitectura para construir aplicaciones completas.

•**Uso**: Se utiliza para construir la arquitectura completa de una aplicación, ofreciendo patrones y mejores prácticas.

•**Ejemplo**: Angular, Vue.js, Ember.js, Svelte.

Librería

•**Control de Flujo**: El desarrollador tiene el control del flujo del programa y llama a las funciones de la librería cuando es necesario.

•**Estructura**: Proporciona funciones específicas y módulos que pueden ser usados en diferentes partes de una aplicación.

•**Uso**: Se utiliza para añadir funcionalidades específicas a un proyecto, como manipulación del DOM, manejo de fechas, creación de gráficos, etc.

•**Ejemplo**: Lodash, D3.js, Chart.js, Moment.js.

Resumen de las Diferencias

•**Control de Flujo**: En un framework, el framework tiene el control; en una librería, el desarrollador tiene el control.

•**Estructura y Arquitectura**: Los frameworks proporcionan una estructura completa para aplicaciones, mientras que las librerías proporcionan funciones específicas sin imponer una estructura particular.

•**Funcionalidad**: Los frameworks están diseñados para construir aplicaciones completas, mientras que las librerías se utilizan para realizar tareas específicas dentro de una aplicación.

Estas diferencias determinan cómo se utilizan y cómo se integran en los proyectos de desarrollo de software.

Ejercicio

De las librerías que hemos visto, implementa por lo menos una en tu proyecto, o bien, en un ejercicio de ejemplo que ya realizaste con anterioridad.