


# 9.5 Clase en vivo - Profesores

|  |      |
|--|------|
|  Status | Done |
|--|------|

## Unidad 9: Uso Avanzado de JavaScript

### Repaso Unidad 9

**Duración:** 5 minutos

**Consigna:** Repasar lo visto en la Unidad 9

### Ejemplo en vivo

**Duración:** 15 minutos

Realicemos un ejemplo integrando Sweet Alert con tres botones, y que cada uno de ellos controle alguna acción/función JS específica dentro de nuestro código. Vamos a trabajar de manera grupal en la aplicación de librerías dentro del proyecto. Aprovechen el espacio grupal para compartir pantalla, saldar dudas y nutrir sus proyectos.

```
// Función para mostrar la alerta con SweetAlert
function mostrarAlerta() {
  Swal.fire({
    title: 'Elige una acción',
    text: 'Selecciona un botón para ejecutar una función es|
    icon: 'question',
    showCancelButton: true,
    showDenyButton: true,
    confirmButtonText: 'Acción 1',
    denyButtonText: 'Acción 2',
    cancelButtonText: 'Acción 3'
  }).then((result) => {
    if (result.isConfirmed) {
```

```

        // Acción 1: Saludar al usuario
        accionUno();
    } else if (result.isDenied) {
        // Acción 2: Mostrar la hora actual
        accionDos();
    } else if (result.dismiss === Swal.DismissReason.cancel) {
        // Acción 3: Cambiar el color de fondo
        accionTres();
    }
});
}

// Función 1: Saludar al usuario
function accionUno() {
    Swal.fire('¡Hola! Esta es la acción 1.');
```

```

}
```

```

// Función 2: Mostrar la hora actual
function accionDos() {
    const horaActual = new Date().toLocaleTimeString();
    Swal.fire(`La hora actual es: ${horaActual}`);
}
```

```

// Función 3: Cambiar el color de fondo
function accionTres() {
    document.body.style.backgroundColor = '#ffcccc';
    Swal.fire('El fondo ha cambiado de color.');
```

```

}
```

## Actividad: Implementación y Uso de Librerías

### Instrucciones para el Profesor:

- Muestra cómo acceder a la documentación de **Sweet Alert** y buscar diferentes opciones de personalización.

- Realiza un ejemplo en vivo donde cambies el ícono, el texto y uses un temporizador para cerrar la alerta automáticamente.

#### Posible Resolución:

```
btn.addEventListener('click', () => {
  Swal.fire({
    title: '¡Guardado!',
    text: 'Tu trabajo ha sido guardado exitosamente.',
    icon: 'success',
    showConfirmButton: false,
    timer: 1500
  });
});
```

## Actividad: Integración con Fetch y Promesas

#### Instrucciones para el Profesor:

- Explica cómo funciona `fetch` y cómo se integran las promesas en JavaScript.
- Muestra en vivo cómo construir la lógica para que **Sweet Alert** controle la ejecución de la petición `fetch`.

#### Posible Resolución:

```
const URL = "https://jsonplaceholder.typicode.com/posts";

async function cargarPublicaciones() {
  const resp = await fetch(URL);
  const posts = await resp.json();
  console.table(posts);
}

btn.addEventListener('click', () => {
  Swal.fire({
    title: '¿Descargar publicaciones del servidor?',
    icon: 'question',
```

```

        showCancelButton: true,
        confirmButtonText: 'Sí ver posts',
        cancelButtonText: 'No por ahora'
    }).then((result) => {
        if (result.isConfirmed) {
            cargarPublicaciones();
        }
    });
});

```

## Find the Bug: Debugging en Vivo

### Instrucciones para el Profesor:

- Presenta el código con los errores y guía a los estudiantes a través de la identificación y corrección de los mismos.
- Demuestra en vivo cómo depurar el código y hacer que funcione correctamente.

### Posible Resolución:

```

const URL = "https://jsonplaceholder.typicode.com/posts";

async function cargarPublicaciones() {
    const resp = await fetch(URL);
    const posts = await resp.json();
    console.table(posts); // Error corregido
}

btn.addEventListener('click', () => {
    Swal.fire({ // Error corregido
        title: '¿Descargar publicaciones del servidor?',
        icon: 'question',
        showCancelButton: true,
        confirmButtonText: 'Sí ver posts',
        cancelButtonText: 'No por ahora'
    });
});

```

```
    }).then((result) => {  
        if (result.isConfirmed) {  
            cargarPublicaciones();  
        }  
    });  
});
```

## Preclase - 30 minutos previos

### Repaso de Conceptos Clave (10 minutos)

- **Promesas en JavaScript:** Revisión rápida de qué son las promesas, su estructura básica ( `resolve` , `reject` ) y cómo ayudan a manejar operaciones asíncronicas de manera más ordenada.
- **Uso de `fetch` con Promesas:** Explicación de cómo realizar peticiones HTTP utilizando `fetch` y cómo manejar las respuestas con promesas.
- **Librerías en JavaScript:** Introducción a las librerías, cómo funcionan como herramientas preconstruidas que facilitan el desarrollo, y ejemplos comunes como Sweet Alert y Luxon.
- **Implementación de Librerías:** Métodos de incorporación de librerías al proyecto, ya sea mediante descarga de archivos o mediante un CDN, y la importancia de usar archivos minificados.

### Espacio para Preguntas y Respuestas (15 minutos)

- Abrir el espacio para que los estudiantes planteen preguntas sobre promesas, manejo de peticiones con `fetch` , y la implementación de librerías.
- Resolver dudas con ejemplos en vivo si es necesario, como cómo manejar errores en promesas o cómo integrar una librería como Sweet Alert en un proyecto.
- Si no hay preguntas, sugerir temas comunes como la diferencia entre `then()` y `catch()` en promesas, o cómo elegir la librería adecuada para un proyecto específico.

### Demostraciones Prácticas (Opcional, dependiendo de las preguntas)

- Realizar demostraciones en vivo si las preguntas lo requieren, por ejemplo, mostrar cómo crear una alerta personalizada con Sweet Alert y vincularla a un evento `click`.
- Ejemplos posibles: Uso de Sweet Alert para crear un cuadro de diálogo de confirmación antes de realizar una petición `fetch`, o cómo formatear fechas usando Luxon.