Javascript #2

(enviar datos al servidor/mostrar error)

1. enviarDatos(event)

function enviarDatos(event) {

event.preventDefault(); // Evitar el envío del formulario por defecto

- **function enviarDatos(event)**: Define una función llamada enviarDatos que recibe un parámetro event. Este parámetro hace referencia al evento que se dispara cuando el formulario se envía (por ejemplo, cuando el usuario hace clic en el botón de enviar).
- **event.preventDefault()**: Llama al método preventDefault() del objeto event para evitar que el navegador ejecute la acción predeterminada del formulario, que sería recargar la página. Esto es útil para manejar el envío de formularios de forma asincrónica (con JavaScript).

const email = document.getElementById('email').value;

const password = document.getElementById('password').value;

- **const email = document.getElementById('email').value**: Obtiene el valor del campo de entrada con el id "email". document.getElementById() busca un elemento con ese id y .value obtiene el texto introducido en el campo.
- **const password = document.getElementById('password').value**: Hace lo mismo con el campo de contraseña, obteniendo el valor introducido por el usuario.

const data = { email, password };

- **const data = { email, password }**: Crea un objeto data que contiene las propiedades email y password, asignándoles los valores que se acaban de obtener de los campos del formulario.

fetch('/logueo', {

method: 'POST',

headers: {

'Content-Type': 'application/json', // Asegúrate de que es JSON

},

body: JSON.stringify(data), // Convertir el objeto en una cadena JSON

})

- **fetch('/logueo', {...})**: Usa la API fetch para hacer una solicitud HTTP al servidor. El primer argumento '/logueo' es la URL a la que se enviará la solicitud.

- **method: 'POST'**: Indica que la solicitud es de tipo POST, es decir, estamos enviando datos al servidor.
- **headers: { 'Content-Type': 'application/json' }**: Establece el tipo de contenido de la solicitud como application/json, lo que le indica al servidor que los datos enviados están en formato JSON.
- **body: JSON.stringify(data)**: Convierte el objeto data (que contiene el email y password) en una cadena JSON antes de enviarlo en el cuerpo de la solicitud.

```
.then(response => {
  if (!response.ok) {
    throw new Error('Credenciales incorrectas');
  }
  return response.text();
})
```

- **.then(response => {...})**: Este método maneja la respuesta del servidor. response es el objeto que representa la respuesta HTTP.
- **if (!response.ok)**: Verifica si la respuesta no fue exitosa. La propiedad response.ok es true si el código de estado de la respuesta está en el rango 200-299 (indicado como éxito).
- **throw new Error('Credenciales incorrectas')**: Si la respuesta no es exitosa, lanza un error con el mensaje 'Credenciales incorrectas'.
- **return response.text()**: Si la respuesta es exitosa, convierte el cuerpo de la respuesta en texto (por lo general, el cuerpo de la respuesta puede ser JSON, texto o HTML, y en este caso lo tratamos como texto).

```
.then(data => {
  window.location.href = '/bienvenido';
})
```

- **.then(data => {...})**: Este método maneja el resultado de la conversión a texto que se realizó en la promesa anterior. En este caso, simplemente redirige al usuario a la página /bienvenido si el inicio de sesión es exitoso.
- **window.location.href = '/bienvenido'**: Redirige al navegador a la URL /bienvenido, lo que generalmente causa que se cargue una nueva página.

```
.catch(error => {
  mostrarErrorModal(error.message);
});
}
```

- **.catch(error => {...})**: Captura cualquier error que se haya lanzado en el bloque anterior. Si algo sale mal (como el lanzamiento de un error por parte de throw new Error()), se maneja en este bloque.
- **mostrarErrorModal(error.message)**: Llama a la función mostrarErrorModal pasando el mensaje del error como argumento. Esto se usa para mostrar el error en un modal en lugar de usar un alert().

---

2. mostrarErrorModal(mensaje)

```
function mostrarErrorModal(mensaje) {
  const modal = document.getElementById('miModal');
  const modalMessage = document.getElementById('modal-message');
  modalMessage.textContent = mensaje;
  modal.style.display = 'block'; // Mostrar el modal
}
```

- **function mostrarErrorModal(mensaje)**: Define una función llamada mostrarErrorModal que recibe un parámetro mensaje, el cual es el mensaje de error a mostrar.
- **const modal = document.getElementById('miModal')**: Obtiene el elemento del modal a través de su id "miModal".
- **const modalMessage = document.getElementById('modal-message')**: Obtiene el elemento dentro del modal donde se mostrará el mensaje de error (con el id "modal-message").
- **modalMessage.textContent = mensaje**: Asigna el mensaje de error al contenido de texto del elemento donde se va a mostrar el error.
- **modal.style.display = 'block'**: Muestra el modal configurando su propiedad display a 'block', lo que hace visible el modal en la página.

---

3. cerrarModal()

```
function cerrarModal() {
  const modal = document.getElementById('miModal');
  modal.style.display = 'none'; // Ocultar el modal
}
```

- **function cerrarModal()**: Define una función llamada cerrarModal que se encarga de ocultar el modal.
- **const modal = document.getElementById('miModal')**: Obtiene el elemento del modal por su id.

- **modal.style.display = 'none'**: Cambia la propiedad display del modal a 'none', lo que lo oculta en la página.

---

4. volver()

```
function volver() {
  window.history.back();
}
```

- **function volver()**: Define una función llamada volver.
- **window.history.back()**: Llama al método back() de la API history para navegar hacia la página anterior en el historial del navegador. Esto es equivalente a presionar el botón de retroceso en el navegador.