IndexedDB

<script>

    let solicitud = indexedDB.open("MiBaseDatos", 1);

    solicitud.onupgradeneeded = (e) => {

        let db = e.target.result;

        db.createObjectStore("usuarios", {keyPath: "id"});

    };

    solicitud.onsuccess = (e) => {

        let db = e.target.result;

        let transaction = db.transaction("usuarios" , "readwrite");

        let store = transaction.objectStore("usuarios");

        store.add({id: 1, nombre: "pepe"});

    };

</script>

Claro, vamos a analizar este código paso a paso:

**1. Apertura de la base de datos**

let solicitud = indexedDB.open("MiBaseDatos", 1);

Aquí se está utilizando **IndexedDB**, una base de datos en el navegador para almacenar información estructurada. La función indexedDB.open("MiBaseDatos", 1) intenta abrir una base de datos llamada "MiBaseDatos" con la versión **1**. Si la base de datos no existe, la creará.

**2. Evento onupgradeneeded**

solicitud.onupgradeneeded = (e) => {

let db = e.target.result;

db.createObjectStore("usuarios", {keyPath: "id"});

};

Este evento **se activa solo si**:

* La base de datos no existía antes y se está creando por primera vez.
* Se está actualizando a una nueva versión (cambiando el número de versión en indexedDB.open()).

Dentro del evento:

1. let db = e.target.result; obtiene el objeto de la base de datos.
2. db.createObjectStore("usuarios", {keyPath: "id"}); crea un almacén de objetos (object store) llamado "usuarios" y define que el **campo clave** (keyPath) será "id".
   * Esto significa que cada objeto almacenado en "usuarios" debe tener un campo "id" único.

**3. Evento onsuccess**

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios" , "readwrite");

let store = transaction.objectStore("usuarios");

store.add({id: 1, nombre: "pepe"});

};

Este evento se ejecuta cuando la base de datos se abre con éxito.

Dentro del evento:

1. let db = e.target.result; obtiene la referencia a la base de datos.
2. let transaction = db.transaction("usuarios", "readwrite");
   * Se inicia una **transacción** sobre el almacén "usuarios", con el modo "readwrite" (lectura y escritura).
3. let store = transaction.objectStore("usuarios");
   * Se obtiene el almacén de objetos "usuarios", donde se guardarán los datos.
4. store.add({id: 1, nombre: "pepe"});
   * Se agrega un objeto con id: 1 y nombre: "pepe" a la base de datos.

**Resumen**

Este código:

1. **Abre o crea** una base de datos llamada "MiBaseDatos" con versión 1.
2. **Si la base de datos es nueva o la versión cambia**, se crea un almacén "usuarios" con el campo clave "id".
3. **Cuando la base de datos está lista**, se inicia una transacción para agregar el usuario {id: 1, nombre: "pepe"} al almacén "usuarios".

¿Necesitas que agregue más detalles o un ejemplo práctico para visualizar los datos? 😊

Para realizar operaciones de **creación, modificación, eliminación y listado de usuarios** en IndexedDB, debes usar transacciones y el almacén de objetos (objectStore). A continuación, te explico cómo hacer cada operación en detalle.

**1. Crear o Agregar un Usuario**

Para agregar un nuevo usuario, utilizamos el método .add(), pero si queremos **insertar o actualizar** un usuario, usamos .put().

**Ejemplo: Agregar un Usuario**

function agregarUsuario(id, nombre) {

let solicitud = indexedDB.open("MiBaseDatos", 1);

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios", "readwrite");

let store = transaction.objectStore("usuarios");

let usuario = { id: id, nombre: nombre };

let request = store.add(usuario); // Agrega un nuevo usuario

request.onsuccess = () => console.log("Usuario agregado con éxito");

request.onerror = () => console.log("Error: El usuario ya existe");

};

}

agregarUsuario(2, "Juan"); // Agrega un usuario con ID 2

💡 **Nota:** Si el usuario ya existe, el .add() generará un error.

**2. Modificar un Usuario**

Para modificar un usuario, usamos el método .put(), que **inserta o actualiza** un objeto en la base de datos.

**Ejemplo: Modificar un Usuario**

function modificarUsuario(id, nuevoNombre) {

let solicitud = indexedDB.open("MiBaseDatos", 1);

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios", "readwrite");

let store = transaction.objectStore("usuarios");

let usuarioActualizado = { id: id, nombre: nuevoNombre };

let request = store.put(usuarioActualizado); // Inserta o actualiza el usuario

request.onsuccess = () => console.log("Usuario modificado con éxito");

request.onerror = () => console.log("Error al modificar usuario");

};

}

modificarUsuario(2, "Juan Pérez"); // Modifica el usuario con ID 2

💡 **Nota:** Si el usuario no existe, .put() lo creará automáticamente.

**3. Eliminar un Usuario**

Para eliminar un usuario, usamos .delete(id) con su clave primaria.

**Ejemplo: Eliminar un Usuario**

function eliminarUsuario(id) {

let solicitud = indexedDB.open("MiBaseDatos", 1);

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios", "readwrite");

let store = transaction.objectStore("usuarios");

let request = store.delete(id); // Elimina el usuario por su ID

request.onsuccess = () => console.log("Usuario eliminado con éxito");

request.onerror = () => console.log("Error al eliminar usuario");

};

}

eliminarUsuario(2); // Elimina el usuario con ID 2

💡 **Nota:** Si el usuario no existe, no ocurre un error, simplemente no hace nada.

**4. Listar Todos los Usuarios**

Para obtener todos los usuarios almacenados, usamos .openCursor() o .getAll().

**Ejemplo 1: Listar Usuarios con openCursor()**

function listarUsuarios() {

let solicitud = indexedDB.open("MiBaseDatos", 1);

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios", "readonly");

let store = transaction.objectStore("usuarios");

let request = store.openCursor(); // Cursor para recorrer los registros

request.onsuccess = (e) => {

let cursor = e.target.result;

if (cursor) {

console.log("ID:", cursor.value.id, "Nombre:", cursor.value.nombre);

cursor.continue(); // Continúa con el siguiente registro

} else {

console.log("Fin de la lista de usuarios");

}

};

request.onerror = () => console.log("Error al listar usuarios");

};

}

listarUsuarios();

💡 **Nota:** .openCursor() es útil si quieres procesar los datos uno por uno.

**Ejemplo 2: Listar Usuarios con getAll()**

function listarUsuariosRapido() {

let solicitud = indexedDB.open("MiBaseDatos", 1);

solicitud.onsuccess = (e) => {

let db = e.target.result;

let transaction = db.transaction("usuarios", "readonly");

let store = transaction.objectStore("usuarios");

let request = store.getAll(); // Obtiene todos los usuarios

request.onsuccess = (e) => console.log("Usuarios:", e.target.result);

request.onerror = () => console.log("Error al obtener usuarios");

};

}

listarUsuariosRapido();

💡 **Nota:** getAll() devuelve un **array con todos los usuarios** y es más rápido.

**Resumen de Operaciones**

| **Acción** | **Método en objectStore** |
| --- | --- |
| Agregar usuario | .add(objeto) |
| Modificar usuario | .put(objeto) |
| Eliminar usuario | .delete(id) |
| Listar usuarios | .openCursor() o .getAll() |

**Conclusión**

Este código permite manejar usuarios en IndexedDB de manera eficiente. Puedes:

* **Agregar usuarios** con .add()
* **Modificar usuarios** con .put()
* **Eliminar usuarios** con .delete()
* **Listar usuarios** con .openCursor() o .getAll()

Si necesitas algo más, dime. 😊 🚀

Para eliminar una base de datos en **IndexedDB**, se usa el método indexedDB.deleteDatabase(nombre). A continuación, te explico cómo hacerlo con un ejemplo.

**Ejemplo: Eliminar una base de datos**

function eliminarBaseDeDatos(nombreBD) {

let solicitud = indexedDB.deleteDatabase(nombreBD);

solicitud.onsuccess = () => console.log(`Base de datos "${nombreBD}" eliminada con éxito.`);

solicitud.onerror = () => console.log(`Error al eliminar la base de datos "${nombreBD}".`);

solicitud.onblocked = () => console.log(`La eliminación de "${nombreBD}" está bloqueada. Cierra todas las pestañas abiertas que la usen.`);

}

// Llamada a la función para eliminar la base de datos

eliminarBaseDeDatos("MiBaseDatos");

**Explicación del código**

1. indexedDB.deleteDatabase(nombreBD);
   * Inicia el proceso de eliminación de la base de datos especificada.
2. **Manejo de eventos**:
   * .onsuccess: Se ejecuta si la base de datos se elimina con éxito.
   * .onerror: Se ejecuta si hay un error en la eliminación.
   * .onblocked: Se activa si la base de datos está en uso por otra pestaña o proceso y no se puede eliminar hasta que se cierre.

**Consideraciones importantes**

* **Si la base de datos está abierta en otra pestaña, la eliminación puede bloquearse** (onblocked).
  + Para solucionar esto, **cierra todas las pestañas que usen la base de datos** y vuelve a intentarlo.
* **Una vez eliminada, la base de datos no se puede recuperar.**
  + Asegúrate de que no necesitas los datos antes de eliminarla.

Si necesitas más ayuda, dime. 🚀😊