**[Patrones Básicos](https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API/Using_IndexedDB" \l "patrones_b%C3%A1sicos)**

El patrón básico que indexedDB propone es:

1. Abrir una base de datos.
2. Crear un objeto de almacenamiento en la base de datos.
3. Iniciar una transacción y hacer una petición para hacer alguna operación de la base de datos, tal como añadir o recuperar datos.
4. Espere a que se complete la operación por la escucha de la clase correcta de eventos DOM .
5. Hacer algo con el resultado (El cual puede ser encontrado en el objeto de la petición).

### [Abriendo una base de datos](https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API/Using_IndexedDB#abriendo_una_base_de_datos)

Iniciamos todo el proceso así:

var request = window.indexedDB.open("MyTestDatabase", 3);

open() retornan unos objetos IDBOpenDBRequest, cuyo resultado, correcto o erróneo, se maneja en un evento. Alguna otra función asincrónica en indexedDB hace lo mismo - Devolver un objeto [IDBRequest](https://developer.mozilla.org/en-US/docs/Web/API/IDBRequest) que disparará un evento con el resultado o el error. El resultado para la función de abrir es una instancia de un IDBDatabase.

El segundo parámetro para el método open es la versión de la base de datos.

Si la base de datos no existe, es creada y se dispara un evento onupgradeneeded de inmediato, permitiéndote proveer una actualización de la estructura e índices en la función que capture dicho evento

**Generando manipuladores**

La primera cosa que usted querrá hacer con la totalidad de las peticiones que usted genera es agregar controladores de éxito y de error:

jsCopy to Clipboard

request.onerror = function (event) {

// Hacer algo con request.errorCode!

};

request.onsuccess = function (event) {

// Hacer algo con request.result!

};

Ahora, asumiendo que el usuario acepta su solicitud para crear una base de datos, y que ha recibido un evento de éxito para activar la devolución de llamada de éxito; ¿Que sigue? La solicitud aquí se generó con una llamada a indexedDB.open (), por lo request.result es una instancia de IDBDatabase, y que sin duda quieren ahorrar para más adelante. Su código podría ser algo como esto:

var db;

var request = indexedDB.open("MyTestDatabase");

request.onerror = function (event) {

alert("Why didn't you allow my web app to use IndexedDB?!");

};

request.onsuccess = function (event) {

db = request.result;

};

**Manejando errores**

Como se mencionó anteriormente, los eventos de error de burbujas. Eventos de error se dirigen a la solicitud que generó el error, entonces el evento se propaga a la operación, y finalmente con el objeto de base de datos. Si desea evitar la adición de controladores de errores a cada solicitud, en su lugar puede añadir un solo controlador de errores en el objeto de base de datos, así:

jsCopy to Clipboard

db.onerror = function (event) {

// Generic error handler for all errors targeted at this database's

// requests!

alert("Database error: " + event.target.errorCode);

};

[**Creación o actualización de la versión de la base de datos**](https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API/Using_IndexedDB#creaci%C3%B3n_o_actualizaci%C3%B3n_de_la_versi%C3%B3n_de_la_base_de_datos)

Cuando se crea una nueva base de datos o se aumenta el número de versión de una base de datos existente (mediante la especificación de un número de versión más alto de lo que hizo antes, en [Cómo abrir una base de datos](https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API/Using_IndexedDB#c%C3%B3mo_abrir_una_base_de_datos)), el evento onupgradeneeded se activará y un objeto [IDBVersionChangeEvent](https://developer.mozilla.org/en-US/docs/Web/API/IDBVersionChangeEvent) será pasado a cualquier controlador de eventos onversionchange establecido en request.result (es decir, db en el ejemplo). En el controlador para el evento upgradeneeded, se debe crear los almacenes de objetos necesarios para esta versión de la base de datos:

// Este evento solamente está implementado en navegadores recientes

request.onupgradeneeded = function (event) {

var db = event.target.result;

// Crea un almacén de objetos (objectStore) para esta base de datos

var objectStore = db.createObjectStore("name", { keyPath: "myKey" });

};

Estas líneas configuran el evento onupgradeneeded del objeto request en IndexedDB. Este evento se ejecuta cuando se crea o actualiza la base de datos por primera vez, o cuando su versión cambia. A continuación, se desglosa el código:

**1. Evento onupgradeneeded**

request.onupgradeneeded = (event) => {

db = event.target.result;

* **request.onupgradeneeded**: Define una función que se ejecuta automáticamente cuando es necesario configurar o actualizar la base de datos.
* **event.target.result**: Contiene la referencia a la base de datos creada o actualizada. Se almacena en la variable db para su uso posterior.

**2. Verificación del almacén de objetos**

if (!db.objectStoreNames.contains("criminales")) {

* **db.objectStoreNames.contains("criminales")**: Comprueba si ya existe un almacén de objetos llamado "criminales" en la base de datos.
* **¿Por qué verificar?**: Si el almacén ya existe, no se vuelve a crear, evitando errores.

**3. Creación del almacén de objetos**

const store = db.createObjectStore("criminales", { keyPath: "id", autoIncrement: true });

* **db.createObjectStore**: Crea un almacén de objetos llamado "criminales".
* **Opciones del almacén**:
  + **keyPath: "id"**: Define que cada objeto almacenado tendrá una clave primaria llamada "id". Esta clave debe ser única para cada registro.
  + **autoIncrement: true**: Genera automáticamente un valor único para la clave primaria cada vez que se agrega un nuevo registro.

**4. Creación de índices**

store.createIndex("criminal", "nombre", { unique: false });

store.createIndex("fecha", "fecha", { unique: false });

* **store.createIndex(nombreIndice, campo, opciones)**:
  + **"criminal" y "fecha"**: Nombres de los índices creados.
  + **"nombre" y "fecha"**: Campos del objeto que se usarán como base para los índices.
  + **{ unique: false }**: Indica que los valores en estos índices no necesitan ser únicos.
* **¿Qué son los índices?**: Son estructuras que mejoran la velocidad de búsqueda para campos específicos. En este caso:
  + **Índice "criminal"** permite buscar criminales rápidamente por su nombre.
  + **Índice "fecha"** permite buscar criminales por la fecha de detención.

**Resumen**

Este código:

1. Configura la base de datos CriminalDB al inicializarla o actualizarla.
2. Verifica si existe un almacén llamado "criminales"; si no, lo crea.
3. Define el almacén con un identificador único (id) que se genera automáticamente.
4. Crea índices ("criminal" y "fecha") para facilitar búsquedas rápidas.

Esto garantiza que la base de datos esté correctamente configurada y lista para almacenar información estructurada sobre criminales.

El código relacionado con const request = indexedDB.open("CriminalDB", 1); y el evento request.onupgradeneeded está profundamente conectado, ya que forman parte del proceso para inicializar y gestionar la base de datos en IndexedDB. A continuación, te explico cómo se relacionan:

**indexedDB.open()**

const request = indexedDB.open("CriminalDB", 1);

1. **indexedDB.open(nombre, version)**:
   * **"CriminalDB"**: Es el nombre de la base de datos que se desea abrir o crear.
   * **1**: Es la versión de la base de datos. Si la versión es mayor que la actual, se activará el evento onupgradeneeded.
2. **Devuelve request**: Este es un objeto de solicitud (IDBOpenDBRequest) que controla el proceso de abrir/crear la base de datos. Proporciona eventos como:
   * **onupgradeneeded**: Se activa si es necesario configurar o actualizar la base de datos.
   * **onsuccess**: Se ejecuta cuando la base de datos se abre con éxito.
   * **onerror**: Se ejecuta si ocurre un error al abrir la base de datos.

**Relación con onupgradeneeded**

request.onupgradeneeded = (event) => {

db = event.target.result;

if (!db.objectStoreNames.contains("criminales")) {

const store = db.createObjectStore("criminales", { keyPath: "id", autoIncrement: true });

store.createIndex("criminal", "nombre", { unique: false });

store.createIndex("fecha", "fecha", { unique: false });

}

};

1. **¿Cuándo se ejecuta?**
   * Si la base de datos no existe (primer acceso).
   * Si la versión especificada (1) es mayor que la versión existente.

En ambos casos, el navegador considera que debe configurar o actualizar la base de datos y activa onupgradeneeded.

1. **Tareas que realiza:**
   * Obtiene acceso a la base de datos a través de event.target.result.
   * Verifica si el almacén "criminales" ya existe con db.objectStoreNames.contains("criminales").
   * Si no existe, crea el almacén y define los índices.
2. **Propósito:**
   * Este paso es obligatorio en IndexedDB para configurar la estructura de la base de datos antes de que se puedan leer o escribir datos.

**Cómo interactúan**

1. **indexedDB.open() inicia el proceso**:
   * Si es la primera vez que se abre "CriminalDB" o la versión es nueva, se dispara onupgradeneeded.
2. **onupgradeneeded configura la estructura**:
   * Se asegura de que la base de datos tenga el almacén ("criminales") y los índices necesarios para almacenar y buscar información eficientemente.
3. **Base de datos lista**:
   * Una vez que onupgradeneeded finaliza, la base de datos queda configurada y lista para usarse en los eventos siguientes, como onsuccess, que maneja la conexión exitosa.

**Flujo completo**

1. indexedDB.open("CriminalDB", 1) se ejecuta.
2. Si es necesario, se dispara onupgradeneeded para configurar la estructura de la base de datos.
3. Cuando la base de datos está lista, onsuccess se activa y permite interactuar con ella.
4. La conexión queda disponible para realizar operaciones como agregar, buscar o eliminar registros.

Este flujo asegura que la base de datos se inicialice correctamente antes de ser utilizada.

No, **no es necesario** usar window. al llamar a indexedDB.open().

**Explicación**

indexedDB es una propiedad global en los navegadores que la soportan, lo que significa que puedes acceder a ella directamente sin anteponer window..

✅ Ambas formas son válidas:

var request = indexedDB.open("MyTestDatabase", 3);

var request = window.indexedDB.open("MyTestDatabase", 3);

**¿Cuándo usar window.?**

El prefijo window. es útil cuando:

1. **Quieres enfatizar que indexedDB es una propiedad global**.
2. **Evitas conflictos con variables locales** que podrían tener el mismo nombre en ciertos contextos.
3. **Sigues un estándar de estilo** donde todas las propiedades globales de window se llaman explícitamente.

**Conclusión**

En la mayoría de los casos, puedes escribir simplemente:

var request = indexedDB.open("MyTestDatabase", 3);

Pero si prefieres mayor claridad o sigues una convención específica, puedes usar window.indexedDB. 🚀

.