

UD 7

Almacenamiento en cliente

DESARROLLO WEB EN ENTORNO CLIENTE

Técnico de Grado Superior Desarrollo de Aplicaciones Web

2024-25

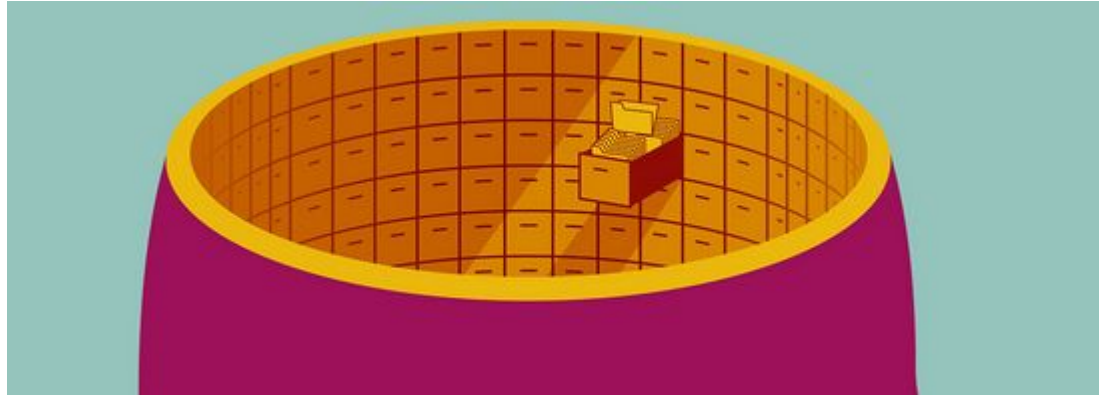
Contenidos

Almacenamiento de datos en lado cliente

- Almacenamiento Web.
- La especificación Web Storage de la W3C.
 - LocalStorage.
 - SessionStorage.
- Bases de datos en entorno cliente.
 - IndexedDB
- JsStore
 - Instalación
 - Conexión.
 - Creación de base de datos y tablas.
 - CRUD sobre una tabla.

Introducción

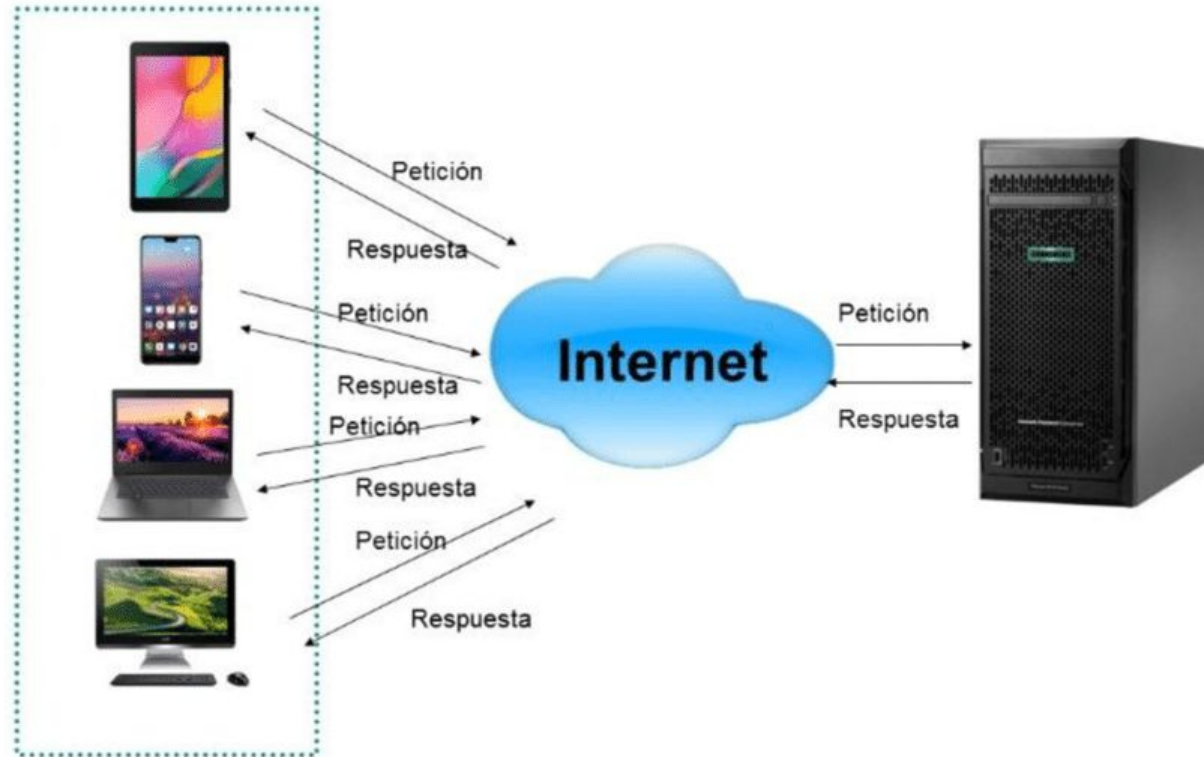
En el desarrollo de aplicaciones web surge frecuentemente la necesidad de almacenar datos en el navegador del usuario.



Ya, ya, se puede almacenar cualquier cosa en el servidor pero se trata de que las aplicaciones sean más **rápidas, interactivas y eficientes**.

Para ello había que buscar una manera de **reducir la necesidad de realizar constantes solicitudes al servidor**.

Almacenamiento web



Cliente:

Los datos se almacenan en el navegador del usuario.

Ejemplo: Preferencias del usuario.

Servidor:

Los datos se almacenan en un servidor remoto.

Almacenamiento web

- **Cookies (1994)**

Introducidas por Netscape, permitieron almacenar pequeños fragmentos de datos en el cliente. Su capacidad es limitada (alrededor de 4 KB) y su uso puede afectar el rendimiento, ya que **se envían con cada solicitud HTTP**.

- **Web Storage (HTML5) (2009)**

La especificación de W3C introduce *localStorage* y *sessionStorage* para ofrecer un mecanismo más eficiente y seguro para almacenar datos en el navegador **sin enviarlos al servidor**.

- **IndexedDB (2011)**

Diseñada para almacenar grandes volúmenes de datos estructurados, es una base de datos **NoSQL** embebida **en el navegador**.

- **JsStore (2017)**

Librería basada en IndexedDB que simplifica su uso mediante una sintaxis tipo SQL.

Almacenamiento web

Tecnología	Capacidad aprox.	Persistencia	Envío al servidor
Cookies	4 KB	Hasta su expiración	Sí
localStorage	5-10 MB	Permanente	No
sessionStorage	5-10 MB	Hasta cerrar la pestaña	No
IndexedDB	Ilimitada	Permanente	No

La especificación web storage de la W3C

mdn web docs

References

Learn

Plus

Curriculum

Blog

Tools

Theme

Log in

Sign up for free

Tecnología para desarrolladores web > Referencia de la API Web > API de almacenamiento web

Español

Esta página ha sido traducida del inglés por la comunidad. Aprende más y únete a la comunidad de MDN Web Docs.

Filter

API de almacenamiento web

Guías

Usando la API de almacenamiento web

Interfaces

Storage

StorageEvent (inglés)

Propiedades

Window.sessionStorage

Window.localStorage

Eventos

Window: storage (inglés)

API de almacenamiento web

La **API de almacenamiento web** proporciona los mecanismos mediante los cuales el navegador puede almacenar información de tipo clave/valor, de una forma mucho más intuitiva que utilizando cookies.

Almacenamiento web, conceptos y uso

Los dos mecanismos en el almacenamiento web son los siguientes:

- sessionStorage** mantiene un área de almacenamiento separada para cada origen que está disponible mientras dure la sesión de la página (mientras el navegador esté abierto, incluyendo recargas de página y restablecimientos).
- localStorage** hace lo mismo, pero persiste incluso cuando el navegador se cierre y se reabra.

Estos mecanismos están disponibles mediante las propiedades `Window.sessionStorage` y `Window.localStorage` (dicho con más precisión, en navegadores con soporte, el objeto `Window` implementa los objetos `WindowLocalStorage` y `WindowSessionStorage`, en los cuales se basan las propiedades `localStorage` y `sessionStorage`). Al invocar uno de éstos, se creará una instancia del objeto `Storage`, a través del cual los datos pueden ser creados, recuperados y eliminados. `sessionStorage` y `localStorage` utilizan un objeto de almacenamiento diferente según su origen — funcionan y son controlados por separado.

En este artículo

Almacenamiento web, conceptos y uso

Interfaces de almacenamiento web

Ejemplos

Especificaciones

Compatibilidad con navegadores

Navegación privada / Modo incógnito

Ver también

Usando la API de almacenamiento web

La API de almacenamiento web proporciona los mecanismos mediante los cuales el navegador puede almacenar información de tipo clave/valor, de una forma mucho más intuitiva que utilizando cookies.

Este artículo proporciona una guía general de cómo usar esta tecnología.

Conceptos básicos

Los objetos de almacenamiento son simples almacenes de clave/valor, similares a objetos, pero que permanecen intactos cuando la página se recarga. Las claves y los valores siempre son cadenas de texto (fíjate que las claves con enteros se convierten automáticamente a cadenas, tal y como lo hacen los objetos). Puedes acceder a estos valores como un objeto, o con los métodos `Storage.getItem()` y `Storage.setItem()`. Estas tres líneas modifican el valor de `colorSetting` de la misma manera:

```
JS

localStorage.colorSetting = "#a4509b";
localStorage["colorSetting"] = "#a4509b";
localStorage.setItem("colorSetting", "#a4509b");
```

La especificación web storage de la W3C

LocalStorage

Permite almacenar datos que persisten en el navegador
(incluso después de cerrarlo)

Almacena datos en formato clave / valor

```
// Guardar un dato
localStorage.setItem("nombre", "Juan");

// Recuperar un dato
let nombre = localStorage.getItem("nombre");

// Eliminar un dato
localStorage.removeItem("nombre");

// Limpiar todo el almacenamiento
localStorage.clear();
```


La especificación web storage de la W3C

SessionStorage

Los datos persisten solo durante la sesión del navegador
(mientras la pestaña está abierta)

Almacena datos en formato clave / valor

```
// Guardar un dato
sessionStorage.setItem("tema", "oscuro");

// Recuperar el dato
let tema = sessionStorage.getItem("tema");

// Eliminar un dato
sessionStorage.removeItem("tema");

// Limpiar todo
sessionStorage.clear();
```

WEB STORAGE

This example is designed to demonstrate usage of the [W3C Web Storage API](#). It should work as far back as IE8. Choose custom background colours, logos and fonts from the below drop down menus, then try closing and reopening the page — you will find that your choices are remembered, as they are stored using Web Storage. You can also visit the [storage event output](#) (opens in new tab). Open this, change some values in the index page, then look at the events page — you'll see the storage changes reported.



Choose background color:

FF170F

Choose font style:

Sans-serif

Choose image:

Crocodile

<https://mdn.github.io/dom-examples/web-storage/>

La especificación web storage de la W3C

- `localStorage`: persiste datos más allá de la sesión.
- `sessionStorage`: persiste datos mientras la pestaña está abierta.
- APIs comunes: `setItem`, `getItem`, `removeItem`, `clear`.

Reto:

Crear una aplicación que almacene el nombre del usuario en `localStorage` y lo muestre al recargar.

Para ello:

1. Diseña un formulario con un campo de texto y un botón.
2. Al enviar el formulario, guarda el nombre en `localStorage`.
3. Si el nombre ya está almacenado, muéstralo en la página al cargar.
4. Encuentra en las DevTools la información que estás almacenando.

La especificación web storage de la W3C

El almacenaje está vinculado al origen
a dominio/protocolo/puerto



The screenshot shows a web browser window with the title "Guardo tu nombre en localStorage" and the text "Hola, Fulano!". Below the page content, the developer tools are open, and the "Almacenamiento" (Storage) tab is selected. The "Almacenamiento local" (Local Storage) section is expanded, showing a table of data for the origin "http://127.0.0.1:3000". The table has two columns: "Key" and "Value". A single entry is visible with the key "nombre" and the value "Fulano".

Key	Value
nombre	Fulano

Bases de datos en entorno cliente

¿Podría ser posible disponer de bases de datos en el cliente que permitan almacenar grandes volúmenes de datos estructurados?

¡Sí!

IndexedDB es una opción nativa

y

JsStore una de las librerías que simplifica su uso.



Bases de datos en entorno cliente

IndexedDB

Filter

IndexedDB

▼ Guías

Usando IndexedDB

Basic_Concepts_Behind_IndexedDB (Inglés)

Browser_storage_limits_and_eviction_criteria (Inglés)

Checking when a deadline is due (Inglés)

▼ Interfaces

IDBCursor

IDBCursorWithValue (Inglés)

IDBDatabase

IDBFactory (Inglés)

IDBIndex (Inglés)

IDBKeyRange (Inglés)

IDBObjectStore

IDBOpenDBRequest (Inglés)

IDBRequest (Inglés)

IDBTransaction (Inglés)


IDBVersionChangeEvent (Inglés)

▼ Propiedades

window.indexedDB

WorkerGlobalScope.indexedDB (Inglés)

IndexedDB

 **Experimental:** Esta es una tecnología experimental. Comprueba la Tabla de compatibilidad de navegadores cuidadosamente antes de usarla en producción.


IndexedDB es una API de bajo nivel que ofrece almacenamiento en el cliente de cantidades significativas de datos estructurados, incluyendo archivos y blobs. Para búsquedas de alto rendimiento en esos datos usa índices. Mientras DOM Storage es útil para el almacenamiento de pequeñas cantidades de datos, no es útil para almacenar grandes cantidades de datos estructurados. IndexedDB proporciona una solución.

Esta página es básicamente el punto de entrada para la descripción técnica de los objetos de la API. Si necesita algo elemental, debería consultar Conceptos básicos acerca de IndexedDB (Inglés). Para más detalles, vea Usando IndexedDB.

IndexedDB provee APIs separados para un acceso síncrono o asíncrono. El API síncrono está destinado a ser usado únicamente dentro de Web Workers, pero no será implementado aún por cualquier navegador. El API asíncrono trabaja con o sin Web Workers.

API Asíncrono

Los métodos del API Asíncrono, retornan sin bloquear el hilo de llamada. Para obtener un acceso asíncrono a la base de datos, use open (Inglés)() en el atributo indexedDB de un objeto window. Este método retorna un objeto IDBRequest (IDBOpenDBRequest); operaciones asíncronas se comunicarán con la aplicación que llama, disparando eventos en los objetos IDBRequest.

 **Nota:** El objeto `indexedDB` se prefija en las versiones antiguas de los navegadores (propiedad `mozIndexedDB` para Gecko < 16, `webkitIndexedDB` en Chrome, y `msIndexedDB` en IE 10).

En este artículo

API Asíncrono

Límites de almacenamiento

Ejemplo

Ver también

Bases de datos en entorno cliente

IndexedDB

¿Qué es esto de NoSQL?

Es una base de datos NoSQL incorporada en los navegadores modernos.

Ventajas:

- Capacidad para almacenar datos complejos (objetos, arrays, etc.)
- Diseñada para aplicaciones offline

Operaciones básicas:

- Abrir una conexión a la base de datos
- Crear una *transacción* para realizar operaciones (lectura/escritura)
- Usar `objectStore` para guardar y recuperar datos

Bases de datos en entorno cliente

IndexedDB

```
// Abrir base de datos
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 transaccion = db.transaction("usuarios", "readwrite");
  let store = transaccion.objectStore("usuarios");
  store.add({ id: 1, nombre: "Juan" });
};
```


Bases de datos en entorno cliente

IndexedDB

Reto:

Crear una base de datos en IndexedDB para almacenar información de tareas.

Para ello:

1. Crea una base de datos llamada "TareasDB".
2. Define un almacén de objetos "tareas" con clave primaria `id`.
3. Agrega alguna tarea de ejemplo al abrir la base de datos.
4. Recupera y muestra las tareas en consola.
5. ¿Quieres continuar?
 1. Almacena tareas de forma interactiva, con un campo de texto y un botón (o Enter)
 2. Recupera las tareas almacenadas y muéstralas en la página

Bases de datos en entorno cliente

IndexedDB JsStore

Es una librería que facilita el uso de IndexedDB mediante una sintaxis del estilo SQL.



Ventajas:

- Fácil de usar para desarrolladores familiarizados con SQL.
- Simplifica operaciones comunes como crear bases de datos, tablas y realizar consultas *CRUD*.

¿Qué es esto de CRUD?

Bases de datos en entorno cliente

IndexedDB - JsStore - Instalación



Installation

Let's install jsstore using npm.

```
npm i jsstore
```

Según la documentación oficial, obteniendo las librerías necesarias en nuestro proyecto

Podemos usar alternativamente un CDN

```
<script src="https://cdn.jsdelivr.net/npm/jsstore@4.9.0/dist/jsstore.min.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/jsstore@4.9.0/dist/jsstore.worker.min.js"></script>
```

- jsstore.min.js es la biblioteca principal que contiene la API de JsStore y se ejecuta en el hilo principal de JavaScript. Proporciona los métodos `connection.insert()`, `connection.select()`, etc.
- jsstore.worker.min.js es el *web worker* de JsStoreSe y ejecuta en un hilo separado del hilo principal, desde el que realiza todas las operaciones pesadas de la base de datos. Así se evita que las operaciones de IndexedDB bloqueen la interfaz de usuario.

Bases de datos en entorno cliente

IndexedDB - JsStore - Conexión

Connection

In JsStore, the Connection class serves as the primary interface for querying the single database. It can be initialized with or without a web worker file, offering flexibility in handling database operations either directly on the main thread or in the background with a worker for improved performance.

For more information, refer to the [Connection](#) page.

```
import { Connection } from "jsstore";

const getWorkerPath = () => {
  // return dev build when env is development
  if (process.env.NODE_ENV === "development") {
    return require("file-loader?name=scripts/[name].[hash].js!jsstore/dist/jsstore.worker.js");
  } else {
    // return prod build when env is production
    return require("file-loader?name=scripts/[name].[hash].js!jsstore/dist/jsstore.worker.min.js");
  }
};

const workerPath = getWorkerPath().default;

// The connection will be used to execute indexeddb queries
const connection = new JsStore.Connection(new Worker(workerPath));
```

```
const connection = new JsStore.Connection();
```

Bases de datos en entorno cliente

IndexedDB - JsStore - Base de datos y tablas

```
const dbName = 'MiTienda';

const tblProducto = {
  name: 'Productos',
  columns: {
    id: { primaryKey: true, autoIncrement: true },
    nombre: { notNull: true, dataType: 'string' },
    precio: { notNull: true, dataType: 'number' },
    categoria: { dataType: 'string' },
    fechaCreacion: { dataType: 'date_time', default: new Date() }
  }
};

const database = {
  name: dbName,
  tables: [tblProducto]
};
```

Bases de datos en entorno cliente

IndexedDB - JsStore - Base de datos y tablas

```
async function initDatabase() {  
  try {  
    await connection.initDb(database);  
    console.log('Base de datos creada exitosamente');  
  } catch (error) {  
    console.error('Error al crear la base de datos:', error);  
  }  
}
```

Bases de datos en entorno cliente

IndexedDB - JsStore - CRUD sobre una tabla

```
async function insertData() {
  try {
    const productos = [
      {
        nombre: 'Laptop',
        precio: 999.99,
        categoria: 'Electrónicos',
      },
      {
        nombre: 'Smartphone',
        precio: 599.99,
        categoria: 'Electrónicos',
      }
    ];

    const insertCount = await connection.insert({
      into: 'Productos',
      values: productos
    });

    console.log(`${insertCount} productos insertados`);
    alert(`${insertCount} productos insertados exitosamente`);
  } catch (error) {
    console.error('Error al insertar datos:', error);
  }
}
```

```
async function getData() {
  try {
    const results = await connection.select({
      from: 'Productos',
      where: {
        categoria: 'Electrónicos'
      }
    });

    const resultsDiv = document.getElementById('results');
    resultsDiv.innerHTML = '<h3>Productos encontrados:</h3>' +
      results.map(product =>
        `<p>Nombre: ${product.nombre}, Precio: ${product.precio}</p>`
      ).join('');

    console.log('Productos recuperados:', results);
  } catch (error) {
    console.error('Error al obtener datos:', error);
  }
}
```

Bases de datos en entorno cliente

Otras librerías



Beware. This specification is no longer in active maintenance and the Web Applications Working Group does not intend to maintain it further.

Web SQL Database

W3C Working Group Note 18 November 2010

La API de Web SQL Database, que permite almacenar datos de forma estructurada en la computadora del usuario (basada internamente en el motor de base de datos SQLite), se presentó en abril de 2009 y se abandonó en noviembre de 2010.

Bases de datos en entorno cliente

Otras librerías



The Database that Syncs!

PouchDB is an open-source JavaScript database inspired by **Apache CouchDB** that is designed to run well within the browser.

PouchDB was created to help web developers build applications that work as well offline as they do online.

It enables applications to store data locally while offline, then synchronize it with CouchDB and compatible servers when the application is back online, keeping the user's data in sync no matter where they next login.

LOCALFORAGE

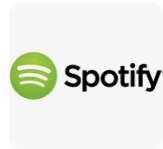
Offline storage, improved.

localForage is a JavaScript library that improves the offline experience of your web app by using an asynchronous data store with a simple, `localStorage`-like API. It allows developers to [store many types of data](#) instead of just strings.

localForage includes a localStorage-backed fallback store for browsers with no IndexedDB or WebSQL support. Asynchronous storage is available in the current versions of all major browsers: Chrome, Firefox, IE, and Safari (including Safari Mobile).

localForage offers a callback API as well as support for the [ES6 Promises API](#), so you can use whichever you prefer.

Almacenamiento web y la web de hoy



Necesitamos funcionar con conexiones inestables... incluso sin conexión!

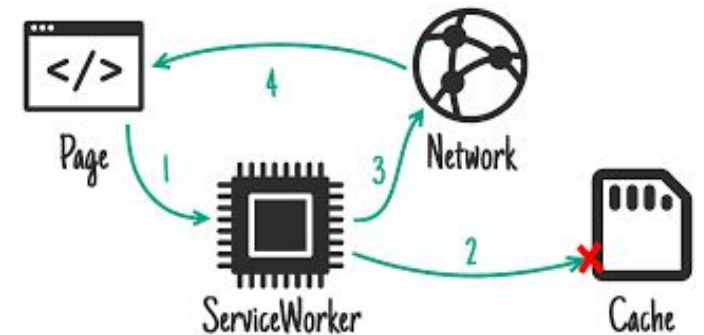
El almacenamiento local es clave para facilitar esta funcionalidad.

Google Drive utiliza IndexedDB para mantener una copia local de los documentos recientes y así permite la edición offline y sincronización posterior

Notion aprovecha una combinación de localStorage y IndexedDB para mantener el estado de la interfaz y los datos del usuario

...

Se complementa con más tecnologías: Caché API, service workers, etc.



La combinación de estas tecnologías posibilita el desarrollo de las...

Aplicaciones Web Progresivas

su funcionalidad incluye trabajar sin conexión, notificaciones push y acceso al hardware del dispositivo, lo que permite crear experiencias de usuario similares a las aplicaciones nativas en dispositivos móviles y de escritorio.

