In a Frontend System Design (FSD) interview, **IndexedDB** is your answer whenever the interviewer asks: *"How would you build an offline-first application like Google Docs, Figma, or a heavy-duty E-commerce dashboard?"*

It is not just a key-value store; it is a full-blown, transactional, file-system-based database inside the browser.

---

1. Technical Deep Dive: The Architecture

IndexedDB is a **low-level API** for client-side storage of significant amounts of structured data.

- **NoSQL / Document-Based:** It doesn't use SQL. It stores objects (records) in **Object Stores** (similar to tables).
- **Transactional:** Every operation must happen within a transaction. If one step fails, the whole transaction rolls back. This ensures **data integrity**.
- **Asynchronous:** Unlike localStorage, IndexedDB does not block the main thread. It uses "Requests" and "Events" (or Promises in modern wrappers).
- **Capacity:** Huge. It usually allows up to **80% of total disk space** (depending on the browser and OS).

---

2. Key Concepts You Must Mention

To sound like a Senior Engineer, you need to use the correct terminology:

1. **Object Stores:** The buckets where you store data (e.g., a "Users" store, a "Products" store).
2. **Indexes:** You can index specific fields (e.g., email) to search through 10,000 records instantly without iterating through the whole list.
3. **Cursors:** Used to iterate over many records one by one to keep memory usage low.
4. **Key Path:** The unique identifier (Primary Key) for a record.

---

3. Schema Versioning (The Migration Logic)

This is a frequent FSD question: *"How do you update the database structure when you push new code?"*

IndexedDB has a built-in event called onupgradeneeded.

- When you open a database, you provide a version number (e.g., v2).
- If the user's browser has v1, the onupgradeneeded event triggers.

- Inside this event, you can create new Object Stores or modify existing ones.

```
const request = indexedDB.open("MyDatabase", 2); // Version 2

request.onupgradeneeded = (event) => {
  const db = event.target.result;
  if (event.oldVersion < 2) {
    // Perform migration: Create a new store
    db.createObjectStore("settings", { keyPath: "id" });
  }
};
```

---

4. Performance & The "Main Thread" Trap

Even though the API is asynchronous, **the work is still happening on the same disk**.

- **Serialization:** Complex objects are "cloned" using the **Structured Clone Algorithm**. This is faster than JSON.stringify used in localStorage, but it still costs CPU time.
- **Web Workers:** In a high-end design, you should suggest offloading IndexedDB operations to a **Web Worker**. This ensures that even heavy database writes never cause a single frame drop in the UI.

---

5. Storage Limits & Eviction

Unlike localStorage, which just fails, IndexedDB has a more complex relationship with the OS.

- **Persistent vs. Best-Effort:** By default, data is "Best-Effort." If the disk is nearly full, the browser might delete your IndexedDB data to save the OS.
- **Requesting Persistence:** You can use the navigator.storage.persist() API to ask the browser (and the user) for permission to keep the data permanently.

---

6. Comparison Table (The Interview "Cheat Sheet")

Feature	LocalStorage	IndexedDB
Capacity	~5MB	**GBs (Disk limit)**
Data Types	Strings only	**Objects, Blobs, Files, Arrays**
Search	Manual (Iterate all)	**High-speed Indexes**
Thread	Blocking (Sync)	**Non-blocking (Async)**
Complexity	Very Low	**High**

---

7. Most Asked Interview Q&A

Q: "Can I store a 50MB Image file in IndexedDB?"
A: Yes. Unlike localStorage, IndexedDB supports Blobs and Files. You don't even need to convert them to Base64 (which bloats size by 33%). You store the raw file object directly.

Q: "What happens if two tabs try to upgrade the DB version at the same time?"
A: The browser handles this via the onblocked event. The second tab will be "blocked" from opening the database until the first tab is closed or reloaded with the new version.

Q: "Why would I ever use a library like Dexie.js or idb?"
A: The native IndexedDB API is notoriously "clunky" and callback-based. Libraries like Dexie.js provide a Promise-based API and better error handling, making the code much more maintainable in a large-scale system.

Critical Info I didn't want you to miss:

- **Private Mode:** In many browsers (like older Safari/Firefox), IndexedDB is either disabled or wiped immediately when an Incognito/Private window is closed. Always design a fallback!
- **Search Limitations:** IndexedDB doesn't support "Full-Text Search" (like SQL LIKE %term%). You can only search by exact prefix or range. For full-text search, you'd need to build an external index (like Lunr.js).