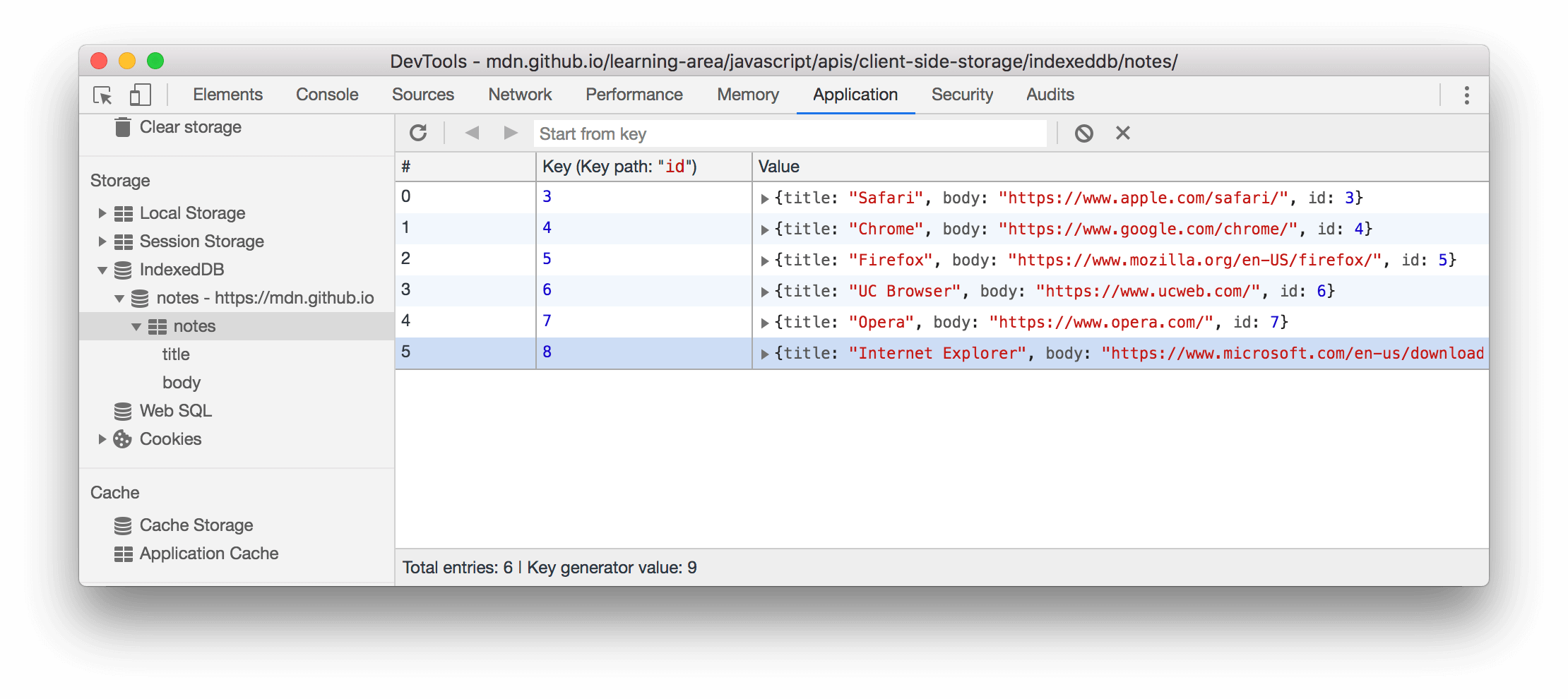
IndexedDB

IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs. This API uses indexes to enable high-performance searches of this data. While Web Storage is useful for storing smaller amounts of data, it is less useful for storing larger amounts of structured data. IndexedDB provides a solution. This is the main landing page for MDN's IndexedDB coverage — here we provide links to the full API reference and usage guides, browser support details, and some explanation of key concepts.

IndexedDB is a transactional database system, like an SQL-based RDBMS. However, unlike SQL-based RDBMSes, which use fixed-column tables, IndexedDB is a JavaScript-based object-oriented database. IndexedDB lets you store and retrieve objects that are indexed with a key; any objects supported by the structured clone algorithm can be stored. You need to specify the database schema, open a connection to your database, and then retrieve and update data within a series of transactions.

Operations performed using IndexedDB are done asynchronously, so as not to block applications.

What is IndexedDB used for?

IndexedDB is a large-scale, NoSQL storage system. It lets you store just about anything in the user's browser. In addition to the usual search, get, and put actions, IndexedDB also supports transactions

Is IndexedDB better than localStorage?

If you want to store structured data on the client side, IndexedDB is the better choice, especially since localStorage isn't built to store sensitive information. But if you're storing a simple, small amount of key-value pair data, use localStorage

Does IndexedDB work offline?

IndexedDB is provided by the browser and thus does not need internet for performing CRUD (Create Read Update Delete) operations.

Is it safe to store data in IndexedDB?

The short answer is IndexedDB is vulnerable to malware and physical takeover attacks. It's better than many options because cryptography is done outside the browser execution environment, but it's not totally secure

IndexedDB with golang

Package indexeddb implements the IndexedDB domain.

**Package name**:go-indexeddb

**Tutorial**: go get github.com/hack-pad/go-indexeddb@latest

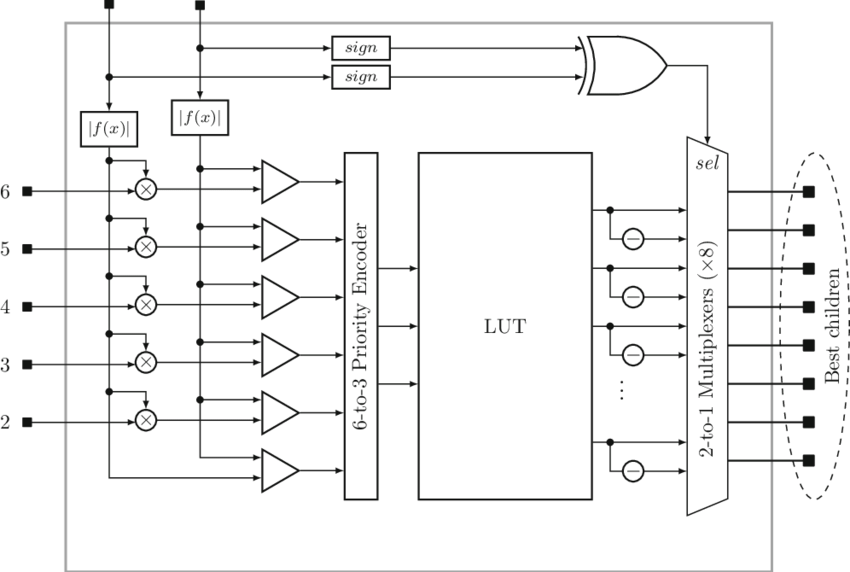
import "github.com/hack-pad/go-indexeddb/idb"

**Useful links**: <https://github.com/hack-pad/go-indexeddb>

<https://pkg.go.dev/github.com/mafredri/cdp/protocol/indexeddb>

XOR

XOR gate (sometimes EOR, or EXOR and pronounced as Exclusive OR) is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd. An XOR gate implements an exclusive or ( ↮\nleftrightarrow) from mathematical logic; that is, a true output results if one, and only one, of the inputs to the gate is true. If both inputs are false (0/LOW) or both are true, a false output results. XOR represents the inequality function, i.e., the output is true if the inputs are not alike otherwise the output is false. A way to remember XOR is "must have one or the other but not both". An XOR gate may serve as a "programmable inverter" in which one input determines whether to invert the other input, or to simply pass it along with no change. Hence it functions as a inverter (a NOT gate) which may be activated or deactivated by a switch.[1][2] XOR can also be viewed as addition modulo 2. As a result, XOR gates are used to implement binary addition in computers. A half adder consists of an XOR gate and an AND gate. The gate is also used in subtractors and comparators.



What is bitwise exclusive OR XOR?

The bitwise exclusive OR operator (in EBCDIC, the ‸ symbol is represented by the ¬ symbol) compares each bit of its first operand to the corresponding bit of the second operand. If both bits are 1 's or both bits are 0 's, the corresponding bit of the result is set to 0 .

What is the relation between bitwise XOR and OR?

Bitwise operations basically perform the namesake operation (OR, XOR, or AND) on every single bit in the two operands. For each operation, you have a left input, and a right input. XOR stands for eXclusive OR, and returns 1 if the left OR the right input is 1, but not if they are both 1.

Is ⊕ commutative is it associative?

Through the use of Boolean algebra, show that the XOR operator ⊕ is both commutative and associative.

Is XOR a commutative operator?

Hence (S, XOR) is a group. In fact it is an Abelian group because we showed above that XOR is also commutative. Two groups are said to be isomorphic if there is a one-to-one mapping between the elements of the sets that preserves the operation.