

---

## Setup

To use this , include one of the ydn-db minified js files from download page in your HTML page.

- User Guide
- API Reference
- Demo applications
- Example applications
- Release notes
- Download

## Supported browsers

This library can be used in almost any browser.

- Chrome 4+
- Firefox 3+
- IE 6 (userData), IE7+ (localStorage), IE10+ desktop/mobile (indexeddb)
- Safari 3.1+ desktop/mobile (webkit)
- Android browser 2.1+ (webkit), 4+ (indexeddb)
- Android web client, iOS web client (webkit)
- Opera 10+ (webkit), Opera 15+ (indexeddb)

## Features

- Unified data access layer on IndexedDB, WebDatabase and WebStorage storage mechanisms.
- Support *all* features of asynchronous IndexedDB API.
- The implementation of WebSQL supports schema reflection, untyped column key, composite index, multiEntry and IndexedDB-like aborting and implicit commit transaction. localStorage use on-memory AVL tree index for key range query and its performance is in-par with database.
- Support for on-the-fly database schema generation, IndexedDB-style versioned schema migration and advance schema-centric (auto-version) migration by reflecting on the existing schema.
- Well tested closure library module including 234 unit test functions in addition to qunit end-to-end test to validate library API specification.
- Advance transaction workflow and managed request (meaning you will never ever see an InvalidStateError).
- Designed for high performance index query (only).

- 
- Customized log messages, improper usage protection and guided error messages on dev distribution.
  - Basic support for high level query using SQL.
  - Full text search (via ydn-db-text module).
  - Client-server Synchronization (via ydn-db-sync module).
  - We adopt strict javascript coding patterns for performance and robustness: no global; no eval; no error globbing; parameterized query; all public methods and constructors are strongly typed; this is this; and coding errors throw errors.

## Examples

### Schema definition

```
1 var schema = {
2   stores: [{
3     name: 'people',
4     indexes: [{
5       name: 'age'
6     }, {
7       name: 'age, name',
8       keyPath: ['age', 'name']
9     }]
10  }]
11 }
12 db = new ydn.db.Storage('db-name', schema);
```

If the database exists, it will be opened and updated with the given schema if necessary. In doing so, object stores and indexes will be created or deleted.

### Simple usage

Simple usage for opening, storing and retrieving by a primary key `id1`.

```
1 db = new ydn.db.Storage('db-name', schema);
2 db.put('people', {name: 'John', age: 10, sex: 'Male'}, 'id1');
3 db.get('people', 'id1').done(function(record) {
4   console.log(record);
5 });
```

---

## Query

The following snippet shows querying from the `people` object store using index `age` by a key range bounded by 25. The result will be sorted by `age`.

```
1 var q = db.from('people').where('age', '>=', 25);
2 var limit = 10;
3 q.list(limit).done(function(objs) {
4   console.log(objs);
5 });
```

Sorting using an index with filtering on another index.

```
1 var q = db.from('people').where('age', '=', 25);
2 q.order('name').list().done(function(objs) {
3   console.log(objs);
4 });
```

Note that the above sort query requires a compound index [ `'age'`, `'name'` ].

## Transaction

By default, database requests are executed in separate transactions and executed in order. The following code snippet shows running all database requests in a single transaction.

```
1 var req = db.run(function update_prop (run_db) {
2   run_db.get('player', 1).done(function(data) {
3     data.health += 10;
4     run_db.put('player', data).done(function(key) {
5       if (data.health > 100) {
6         req.abort();
7       }
8     });
9   });
10 }, ['player'], 'readwrite');
11 req.then(function() {
12   console.log('updated.');
13 }, function(e) {
14   console.log('transaction aborted');
15 });
```

## Events

`ydns.db.Storage` dispatch events for connection and error. Additionally modification of records events can be installed by defining in schema.

---

Data heavy query should be execute after database connection is established by listening `ready` event.

```
1 db.onReady(function (err) {
2   if (err) {
3     console.error(err);
4     return;
5   }
6   // heavy database operations should start from this.
7 });
```

## Library developer guide

If you haven't try Closure Tools before, setup can be time consuming and painful. I recommend to read Michael Bolin book's Closure: The Definitive Guide. A good understanding of closure coding pattern is necessary to understand and follow this library codes.

Apache ant is used to build javascript compiler. ydn-base repo build.xml defines compiler and others tools setting. You must change according to your local machine setting. Specifically check property values of `closure-library.dir` and `closure-compiler.dir`, which point to respective directories.

Downloads the following three repos a directory.

```
1 svn checkout http://closure-library.googlecode.com/svn/trunk/
2 git clone git@bitbucket.org:ytkyaw/yn-db.git
3 git clone https://bitbucket.org/ytkyaw/yn-base.git
```

that should create three directories for closure-library, ydn-base and ydn-db.

Run local apache (recommended) or a static server on that directory.

Go to ydn-db folder and run `ant deps` and `ant ydn-base.deps` to generate closure dependency tree.

Use HTML files in the /test folder for getting started. These files are also used for debug development.

Note: we use master track version of closure tools. Compiling with pre-build jar may encounter compile error.

Note: precompile files are built by using custom compiler to strip debug messages. See detail on ydn-base/tools/strip\_debug.txt.

Additional features requires the following optional repos.

- 
1. Full text search <https://github.com/yathit/ydn-db-fulltext.git>
  2. Dependency for ydn-db-fulltext <https://github.com/yathit/fullproof>
  3. Dependency for ydn-db-fulltext <https://github.com/yathit/natural>
  4. Synchronization <https://bitbucket.org/ytkyaw/ydn-db-sync> (private)

## Testing

You should be able to run `/ydn-db/test/all-test.html` or run tests individually. Since all tests are async, disable run the 'in parallel' check box. These test files are for basic testing and debugging.

The coverage test is performed by JsTestDriver test. Notice that `ant gen-alltest-js` generates `jsTestDriver.conf` to prepare the testing configuration.

```
1 java -jar JsTestDriver.jar --tests all
```

End-to-end testing for distribution can be found in the `test/qunit` folder as well as online [qunit test kits] (<http://dev.yathit.com/index/demos.html>).

## Deployment

For update bower, create a tag

```
1 git tag -a v1.3.3 -m "bug fixes"
```

and push the tag to github

```
1 git push origin master v1.3.3
```

## Contributing

Sending pull requests is easiest. For large or architectural changes, please email one of the authors in the source code.

We follow the Google JavaScript Style Guide. All commits on the master branch must pass the most stringent compilation settings and pass all unit tests.

A few coding dialects we follow:

- Preferred variable naming is `like_this`, `notLikeThis`. Function names `areLikeThis`.
- Assume native types (boolean, number, string) are not nullable. If a nullable type is used, it is different from `undefined`. Using `undefined` for missing values in native type is encourage over `null`.

---

## Library design

- Library API should be similar to IndexedDB API and use exact terminology and concept in the IndexedDB specification. So that, people who already familiar with it can pick up immediately as well as go forward with native API.
- Simple operations should be easy to use as well as optimized for it. Also impose user to use efficient methods while making inefficient ways very difficult or impossible.
- For complex query, helper utility functions and classes will be provided. Storage class has deep understanding about these helper classes and do optimization behind the sense.
- Memory efficient and must not use buffer memory. If buffer is used, it must be explicit. Memory leak is unacceptable.
- Provide extensive error and log message in debug mode, spare no expense since we will strip them in production binary. Error and exception should be thrown as soon as possible, preferable before async callback.
- Since this API is very simple, fallback to WebSQL and WebStorage should be straight forward. This library design have no consideration for these storage mechanisms.

## Bug reports

Please file an issue for bug reports describing how we could reproduce your problem. We will try address any subtle problems, memory and speed performance issues, and even extending the features of the IndexedDB API.

You may also ask questions in Stackoverflow #ydn-db with ydb-db hash, or follow us on Twitter @yathit.

## License

Licensed under the Apache License, Version 2.0