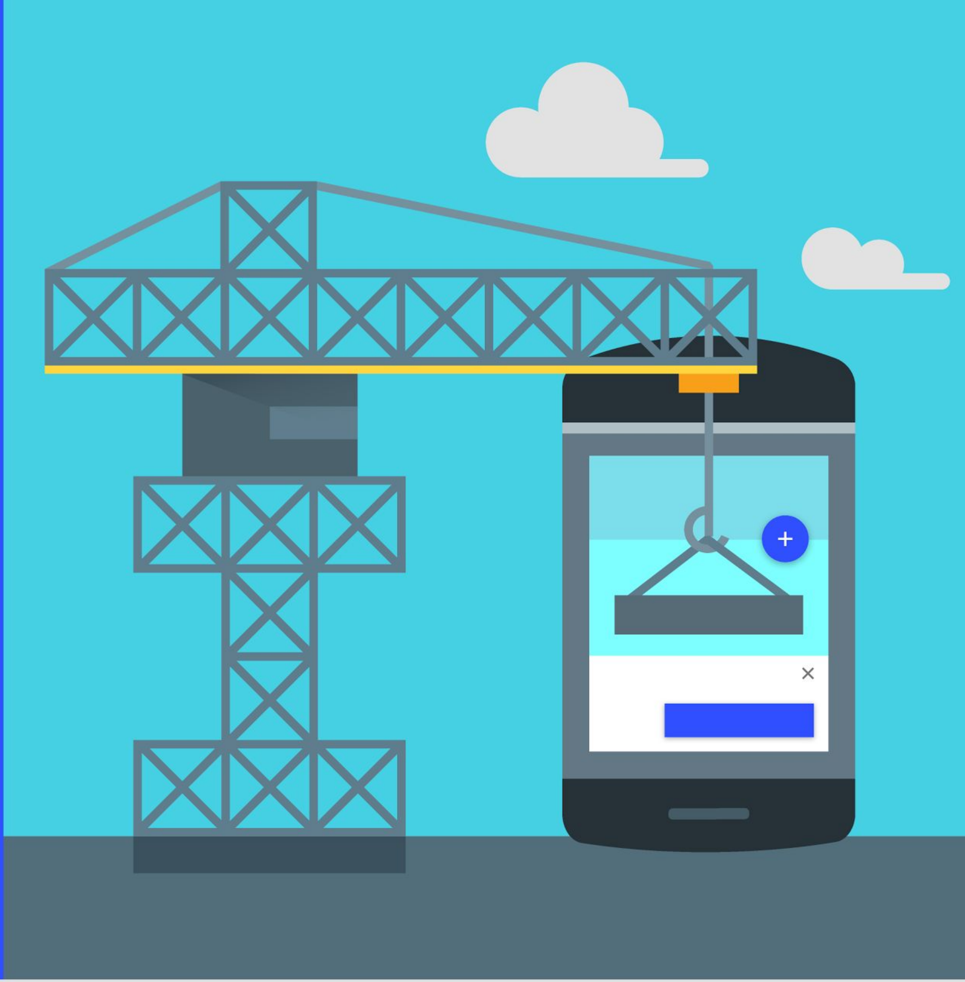
IndexedDB



What you will learn

- What is IndexedDB?
- Where is it supported
- How you can store and retrieve data in it
- How you can run queries against it
- Options for when it’s not supported

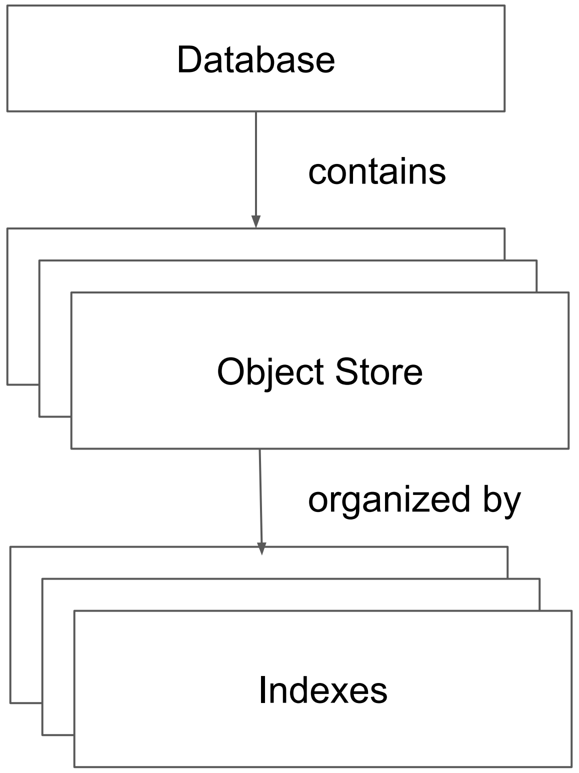
Prerequisites: Javascript, Promises

What is IndexedDB?

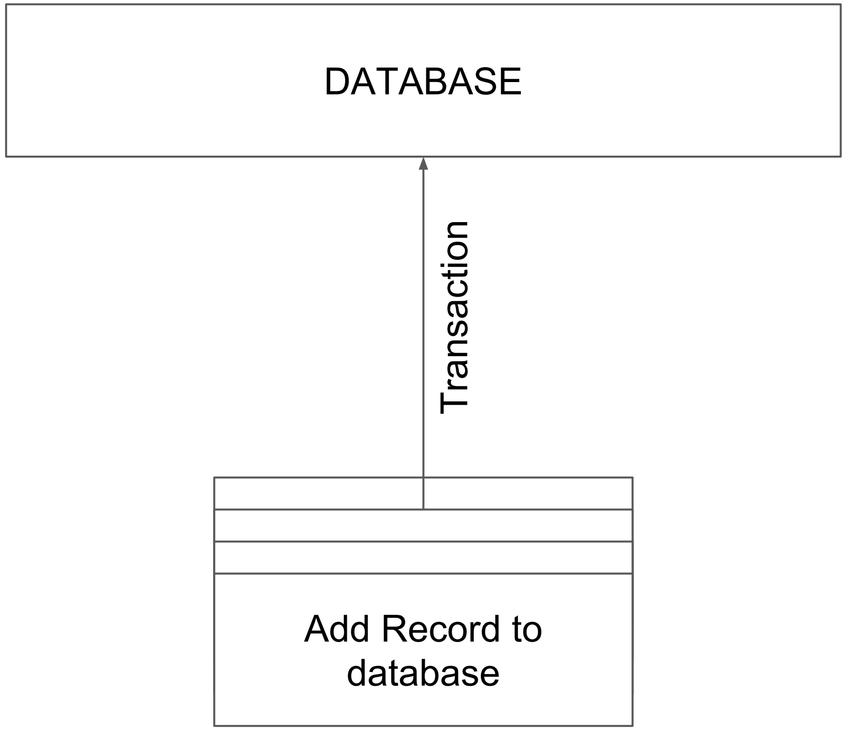
IndexedDB provides an **object store** in the browser

- Stores: JS objects, files, blobs, etc.
- Searchable
    - Uses indexes to control sorting in search results
- Transactional
- *Not a relational database*

IndexedDB Structure



IndexedDB Terminology

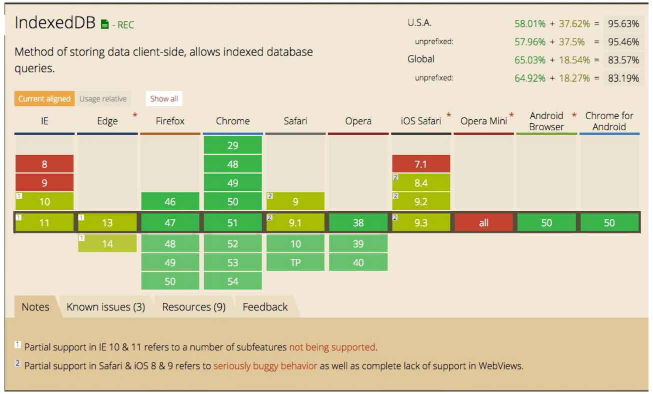


Browser Support

84% support globally (caniuse.com July ‘16)

Caveats:

- Not in “mini” browsers
- Known bugs in Safari



IndexedDB - REC

Method of storing data client-side, allows indexed database queries.

U.S.A.	58.01%	+ 37.62%	=	95.63%
unprefixed:	57.96%	+ 37.5%	=	95.46%
Global	65.03%	+ 18.54%	=	83.57%
unprefixed:	64.92%	+ 18.27%	=	83.19%

Current aligned · Usage relative · Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
8			48			7.1			
9			49			[2] 8.4			
[1] 10		46	50	[2] 9		[2] 9.2			
[1] 11	[1] 13	47	51	[2] 9.1	38	[2] 9.3	all	50	50
	[1] 14	48	52	10	39				
		49	53	TP	40				
		50	54						

Notes · Known issues (3) · Resources (9) · Feedback

[1] Partial support in IE 10 & 11 refers to a number of subfeatures not being supported.

[2] Partial support in Safari & iOS 8 & 9 refers to seriously buggy behavior as well as complete lack of support in WebViews.

Polyfills and Libraries

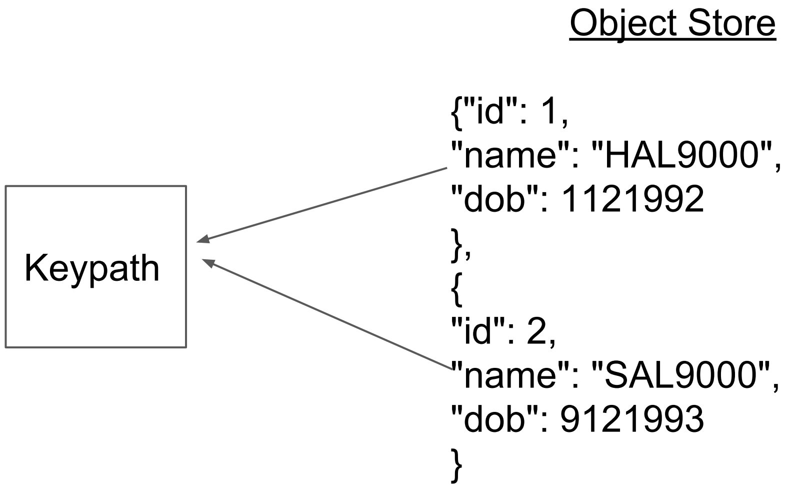
- IndexedDB Shim
- IndexedDB Promised
- Localforage
- Dexie.js
- Taffydb

Sources:
https://github.com/bebraw/jswiki/wiki/Storage-libraries
https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills

Check for IndexedDB Support

```
if (!(‘indexedDB’ in window)) { return; }
```

Creating DB Overview



```
{"id": 1,
"name": "HAL9000",
"dob": 1121992
},
{
"id": 2,
"name": "SAL9000",
"dob": 9121993
}
```

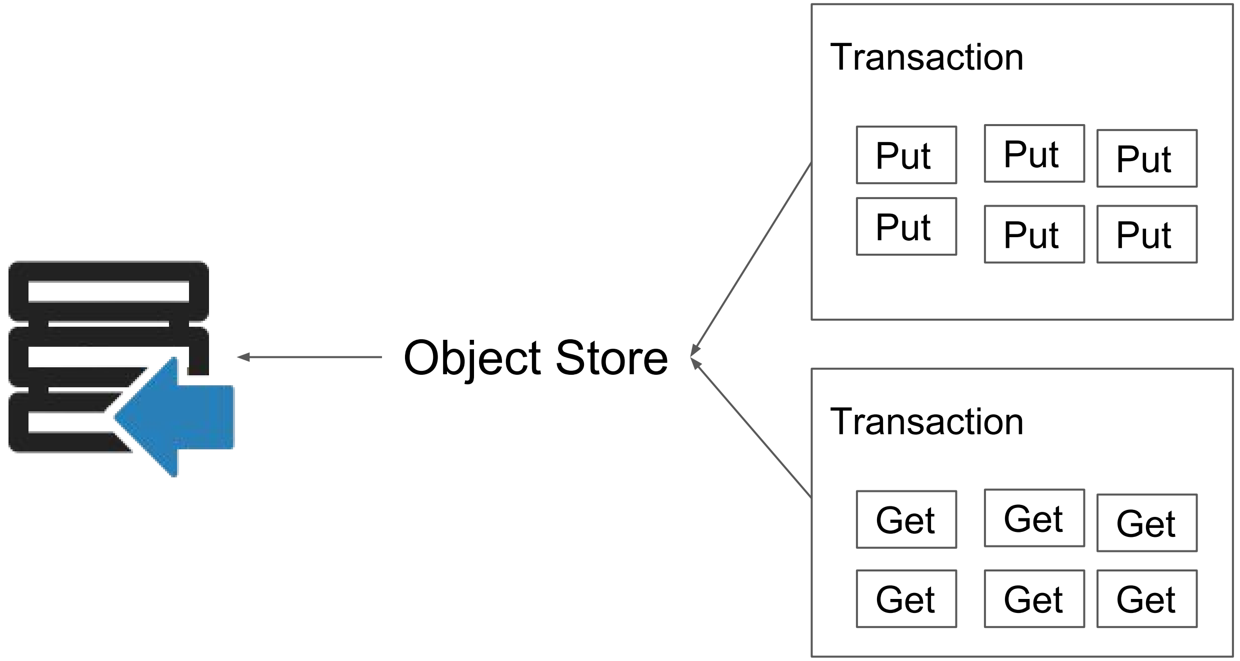
IndexedDB Example: Creating DB

```
function createDB() {
  if (!(‘indexedDB’ in window )) { return; }
  var request = indexedDB.open('library', 1);
  var db;
  request.onupgradeneeded = function(event) {
    var db = event.target.result;
    var store = db.createObjectStore('books', {keyPath: 'isbn'});
    var titleIndex = store.createIndex('by_title', 'title',
{unique: true});
    var authorIndex = store.createIndex('by_author', 'author');
};
```

IndexedDB Example: Creating DB

```
    request.onsuccess = function(event) {
      db = event.target.result;
    };
    request.onerror = function(error) {
      console.log('Could not open database: ' + error);
    };
  }
```

IndexedDB example: Populating DB



Icon from Benjamin STAWARZ **License:** Creative Commons (Attribution 3.0 Unported)

IndexedDB example: Populating DB

```
function addData() {
 if (!window.indexedDB) { return; }

  var request = indexedDB.open('library', 1);
  var db;
  var transaction;
```

IndexedDB example: Populating DB

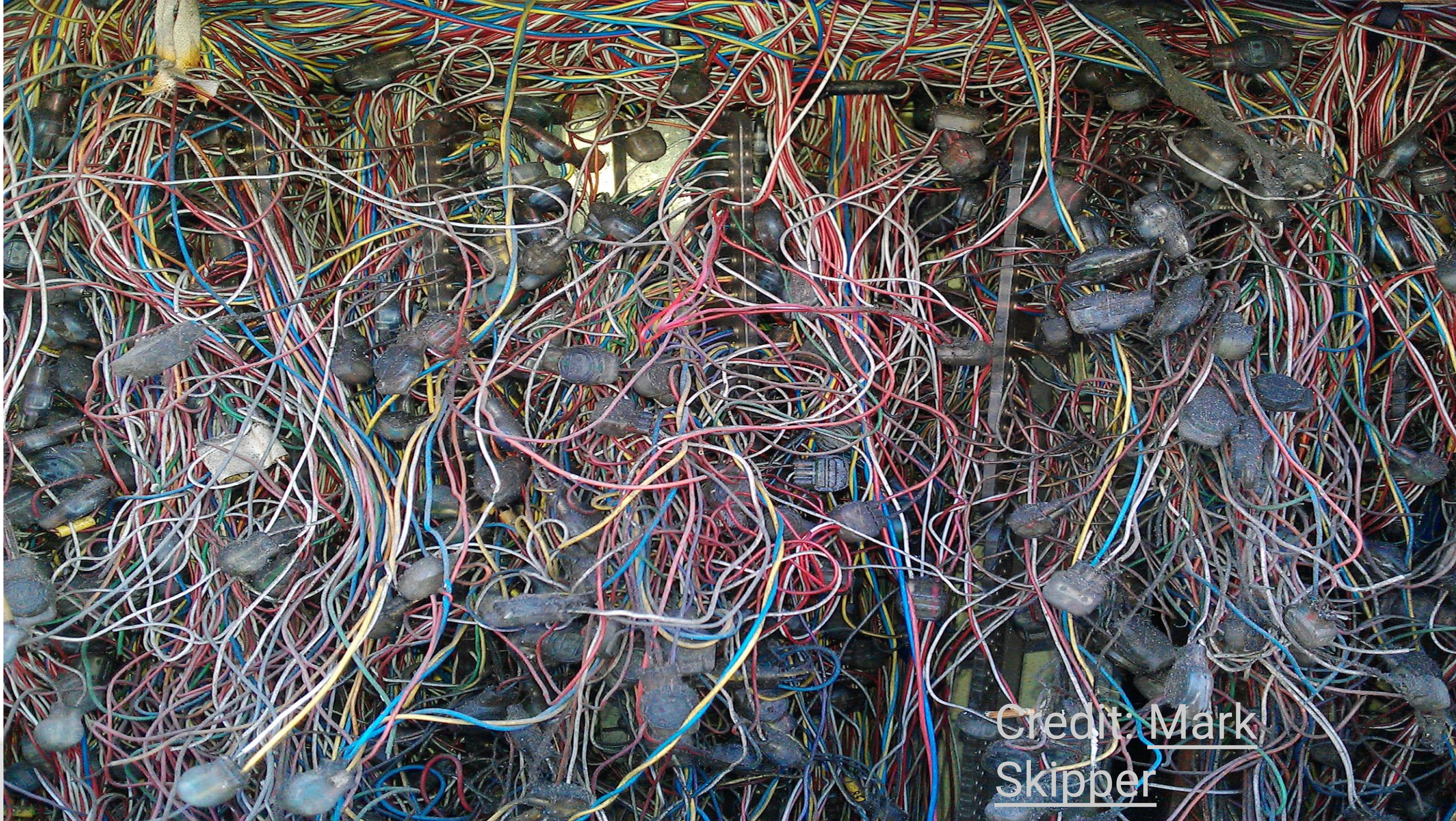
```
request.onerror = function(event) {console.log('Failed');};
request.onsuccess = function(event) {
  db = request.result;
  transaction = db.transaction('books', 'readwrite');
  transaction.oncomplete = function(event) {console.log('Completed');};
      transaction.onerror = function(event) {console.log('Failed');};
```

IndexedDB example: Populating DB

```
var objectStore = transaction.objectStore('books');
var objectStoreRequest;
var BOOK_DATA = [
  {title: 'Bam Bam and me', author: 'Betty', isbn: 19543},
  {title: 'Quarry Memories', author: 'Fred', isbn: 123456},
];
```

IndexedDB example: Populating DB

```
      objectStoreRequest.onsuccess = function(event) {
        console.log('Transaction successful');
      };
      objectStoreRequest.onerror = function(error) {
        console.log('Unable to add book: ', error);
      };
    };
  }
```

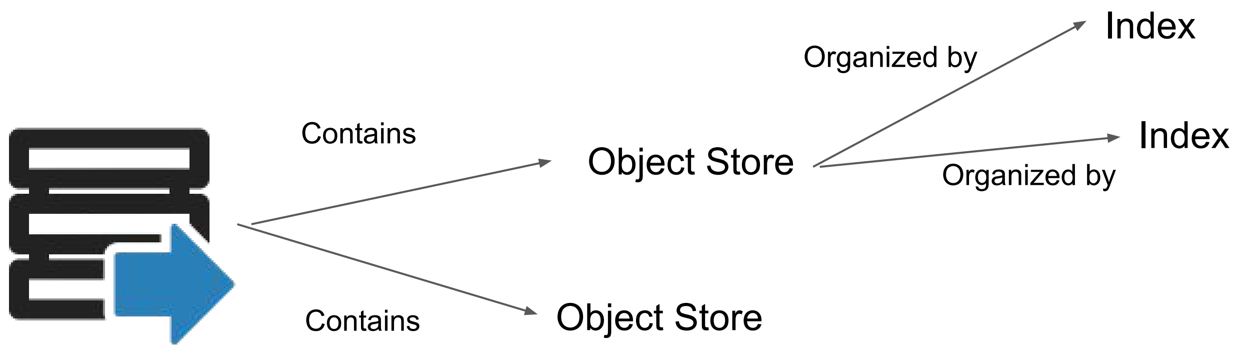

Credit: Mark Skipper

Opening a database

```
idb.open(name, version, {upgradeCallback})
```

```
var dbPromise = idb.open('test-db1', 1);
```

Creating Object Stores



Icon from Benjamin STAWARZ **License:** Creative Commons (Attribution 3.0 Unported)

Creating Object Stores

```
  var dbPromise = idb.open('test-db2', 1,
function(upgradeDb) {
    console.log('making a new object store');
    if (!upgradeDb.objectStoreNames.contains('firstOS')) {
      upgradeDb.createObjectStore('firstOS');
    }
  });
};
```

Ways to Create Primary Keys

1. `upgradeDb.createObjectStore(“people”, {keyPath: “email”});`

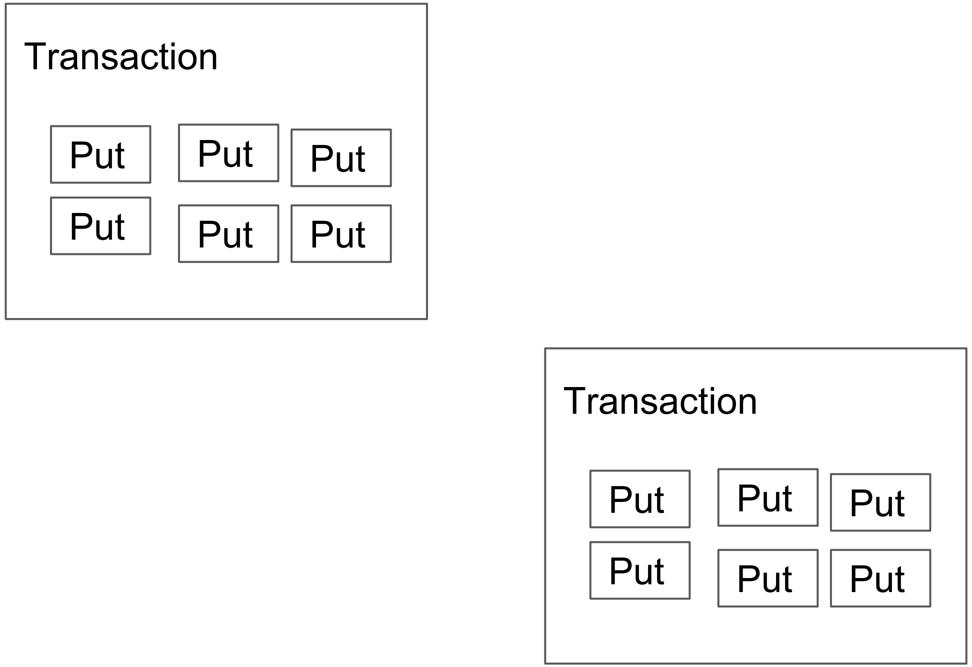
2. `upgradeDb.createObjectStore(“notes”, {autoIncrement:true});`

3. `upgradeDb.createObjectStore(“logs”, {keyPath: “id”, autoIncrement:true});`

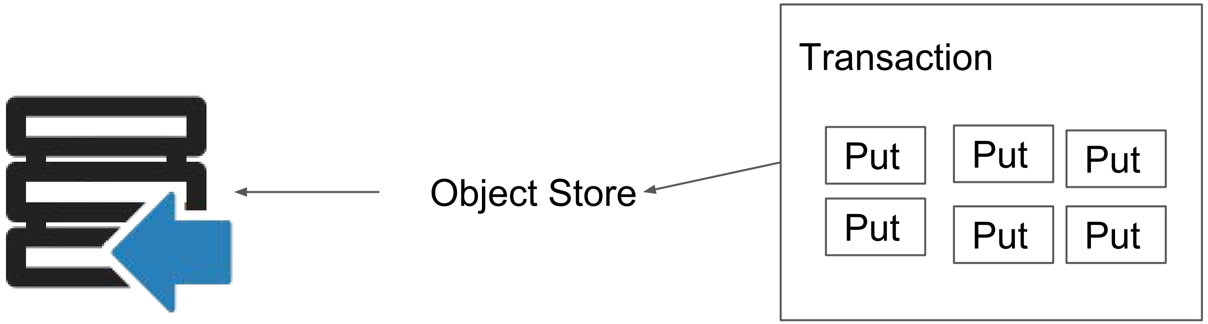
Defining Indexes

```
 var dbPromise = idb.open('test-db4', 1, function(upgradeDb) {
    if (!upgradeDb.objectStoreNames.contains('people')) {
      var peopleOS = upgradeDb.createObjectStore('people',
{keyPath: 'email'});
      peopleOS.createIndex('gender', 'gender', {unique: false});
      peopleOS.createIndex('ssn', 'ssn', {unique: true});
    }
  });
```

Transactions



Adding Data



Icon from Benjamin STAWARZ **License:** Creative Commons (Attribution 3.0 Unported)

Adding Data: Creating The Stores

```
var dbPromise = idb.open('test-db5', 1, function(upgradeDb) {
    if (!upgradeDb.objectStoreNames.contains('store')) {
      var storeOS = upgradeDb.createObjectStore('store', {keyPath:
'name'});
    }
    if (!upgradeDb.objectStoreNames.contains('notes')) {
      var notesOS = upgradeDb.createObjectStore('notes',
{autoIncrement: true});
    }
  });
```

Add Data: Defining the data

```
function addItem() {
  dbPromise.then(function(db) {
    var tx = db.transaction(['store'], 'readwrite');
    var store = tx.objectStore('store');
    var item = {name: name};
```

Add Data: Adding the data

```
    store.add(item);
    return tx.complete;
  }).then(function() {
    console.log('added item to the store os!');
  });
}
```

Reading Data

```
 dbPromise.then(function(db) {
  var tx = db.transaction(['store'], 'readonly');
  var store = tx.objectStore('store');
  return store.get(key);
});
```

Update Data

```
dbPromise.then(function(db) {
  var tx = db.transaction(['store'], 'readwrite');
  var store = tx.objectStore('store');
  var item = {name: name
  store.put(item);
  return tx.complete;
}).then(function() {
  console.log('item updated!');
});
```

Deleting data

```
dbPromise.then(function(db) {
  var tx = db.transaction(['store'], 'readwrite');
  var store = tx.objectStore('store');
  store.delete(key);
  return tx.complete;
}).then(function() {
  console.log('Item deleted');
});
```

Retrieving multiple records

```
dbPromise.then(function(db) {
  var tx = db.transaction(['store'], 'readonly');
  var store = tx.objectStore('store');
  return store.getAll();
}).then(function(items) {
  console.log('Items by name:', items);
});
```

Using Cursors to Retrieve Data

```
someObjectStore.openCursor(optionalKeyRange,
optionalDirection);
```

Get All Records

```
dbPromise.then(function(db) {
  // open transaction on db object and open store on transaction
  return store.openCursor();
}).then(function showItems(cursor) {
  if (!cursor) {return;}
  for (var field in cursor.value) {console.log(cursor.value[field]);}
  return cursor.continue().then(showItems);
})
```

Working with ranges and indexes

```
IDBKeyRange.lowerBound(indexKey);

IDBKeyRange.upperBound(indexKey);

IDBKeyRange.bound(lowerIndexKey, upperIndexKey);
```

Using Exclusive Ranges

```
IDBKeyRange.lowerBound(indexKey, false);


IDBKeyRange.upperBound(indexKey, false);


IDBKeyRange.bound(lowerIndexKey, upperIndexKey,
false);
```

Ranges Example (part 1)

```
function searchItems() {
   var lower = document.getElementById('lower').value;
   var upper = document.getElementById('upper').value;
   if (lower == '' && upper == '') {return;}
   var range;
   if (lower != '' && upper != '') {
     range = IDBKeyRange.bound(lower, upper);
   } else if (lower == '') {
     range = IDBKeyRange.upperBound(upper);
   } else {
     range = IDBKeyRange.lowerBound(lower);
   }
```

Ranges Example (part 2)

```
dbPromise.then(function(db) {
  var tx = db.transaction(['store'], 'readonly');
  var store = tx.objectStore('store');
  var index = store.index('price');
  return index.openCursor(range);
}).then(function showRange(cursor) {
  if (!cursor) {return;}
  for (var field in cursor.value) {console.log(cursor.value[field]);  }
  return cursor.continue().then(showRange);
}).then(function() {console.log('Transaction complete');});
```

Versioning Databases

```
idb.open(dbName, 2, function(upgradeDb) { }
```

Versioning Databases

```
var dbPromise = idb.open('test-db7', 2, function(upgradeDb) {
  switch (upgradeDb.oldVersion) {
   case 0:
     upgradeDb.createObjectStore('store', {keyPath: 'name'});
   case 1:
     var storeOS = upgradeDb.transaction.objectStore('store');
     storeOS.createIndex('price', 'price');
});
```