

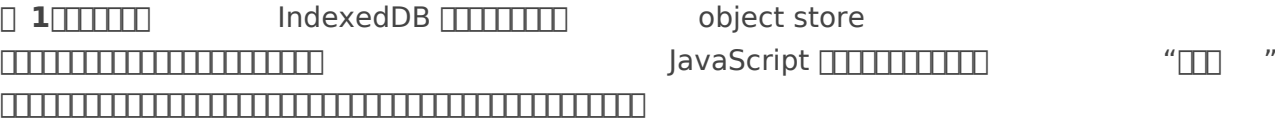
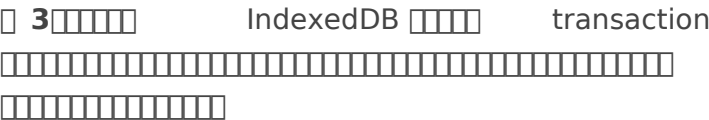
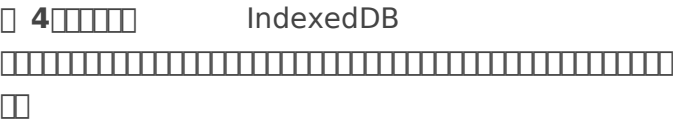
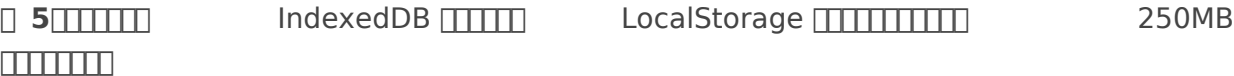
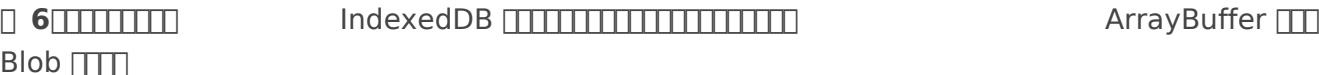
IndexedDB API



- [index](#)
 - [indexName](#)
 - [indexNames](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexName](#)
 - [indexedDB](#) [index](#)
 - [indexedDB.open\(\)](#)
 - [indexedDB.deleteDatabase\(\)](#)
 - [indexedDB.cmp\(\)](#)
 - [IDBRequest](#) [index](#)
 - [IDBDatabase](#) [index](#)
 - [index](#)
 - [index](#)
 - [IDBObjectStore](#) [index](#)
 - [index](#)
 - [index](#)
 - [IDBTransaction](#) [index](#)
 - [IDBIndex](#) [index](#)
 - [IDBCursor](#) [index](#)
 - [IDBKeyRange](#) [index](#)
 - [index](#)
-



IndexedDB



- IDBDatabase

- IDBObjectStore
- IDBIndex
- IDBTransaction
- IDBRequest
- IDBCursor
- IDBKeyRange

1

+ +

IndexedDB

2

object store

3

{ id: 1, text: 'foo' }

id

4

5

error abort complete

[illegible]

--	--	--	--	--

```
indexedDB.open()
```

[illegible][illegible]

IDBRequest []

error ☐ success ☐

upgradeneeded

error

2 success

success

```
var db;
```

```
request.onsuccess = function (event) {
    db = request.result;
    console.log('👉👉👉👉👉');
};
```

request result

3 upgradeneeded

upgradeneeded

```
var db;

request.onupgradeneeded = function (event) {
  db = event.target.result;
}
```


target.result





upgradeneeded




```
request.onupgradeneeded = function(event) {
  db = event.target.result;
  var objectStore = db.createObjectStore('person', { keyPath: 'id' });
}
```


person

id




```
request.onupgradeneeded = function (event) {
  db = event.target.result;
  var objectStore;
  if (!db.objectStoreNames.contains('person')) {
    objectStore = db.createObjectStore('person', { keyPath: 'id' });
  }
}
```

key






{ id: 1, name: ' ' }

id

{ foo: { bar: 'baz' } }

foo.bar


IndexedDB


```
var objectStore = db.createObjectStore(
  'person',
  { autoIncrement: true }
```

```
);
```

```
};
```

```
};
```

```
request.onupgradeneeded = function(event) {  
  db = event.target.result;  
  var objectStore = db.createObjectStore('person', { keyPath: 'id' });  
  objectStore.createIndex('name', 'name', { unique: false });  
  objectStore.createIndex('email', 'email', { unique: true });  
}
```

```
};
```

```
IDBObject.createIndex()
```

```
};
```

```
};
```

```
};
```

```
function add() {  
  var request = db.transaction(['person'], 'readwrite')  
    .objectStore('person')  
    .add({ id: 1, name: '张三', age: 24, email: 'zhangsan@example.com' });  
  
  request.onsuccess = function (event) {  
    console.log('成功');  
  };  
  
  request.onerror = function (event) {  
    console.log('失败');  
  }  
}  
  
add();
```

```
};
```

```
};
```

```
IDBTransaction.objectStore(name)
```

```
};
```

```
add()
```

“ ” “ ”

IDBObjectStore

```
};
```

```
success
```

```
error
```



```
function read() {
  var transaction = db.transaction(['person']);
  var objectStore = transaction.objectStore('person');
  var request = objectStore.get(1);

  request.onerror = function(event) {
    console.log('');
  };

  request.onsuccess = function( event) {
    if (request.result) {
      console.log('Name: ' + request.result.name);
      console.log('Age: ' + request.result.age);
      console.log('Email: ' + request.result.email);
    } else {
      console.log('');
    }
  };
}

read();
```



objectStore.get()



IDBCursor

```
function readAll() {
  var objectStore = db.transaction('person').objectStore('person');

  objectStore.openCursor().onsuccess = function (event) {
    var cursor = event.target.result;

    if (cursor) {
```

```

        console.log('Id: ' + cursor.key);
        console.log('Name: ' + cursor.value.name);
        console.log('Age: ' + cursor.value.age);
        console.log('Email: ' + cursor.value.email);
        cursor.continue();
    } else {
        console.log('');
    }
};
}

readAll();

```

■■■■■■■■■■

openCursor() ■■■■■■■■■■■■

success ■■■

■■■■

■■■■■

IDBObject.put() ■■■

```

function update() {
    var request = db.transaction(['person'], 'readwrite')
        .objectStore('person')
        .put({ id: 1, name: 'li', age: 35, email: 'lisi@example.com' });

    request.onsuccess = function (event) {
        console.log('');
    };

    request.onerror = function (event) {
        console.log('');
    }
}

update();

```

■■■■■

put() ■■■■■■■■

1 ■■■■

■■■■

IDBObjectStore.delete()

```
function remove() {  
  var request = db.transaction(['person'], 'readwrite')  
    .objectStore('person')  
    .delete(1);  
  
  request.onsuccess = function (event) {  
    console.log('');  
  };  
}  
  
remove();
```

name

```
objectStore.createIndex('name', 'name', { unique: false });
```

name

```
var transaction = db.transaction(['person'], 'readonly');  
var store = transaction.objectStore('person');  
var index = store.index('name');  
var request = index.get('');  
  
request.onsuccess = function (e) {  
  var result = e.target.result;  
  if (result) {  
    // ...  
  } else {  
    // ...  
  }  
}
```

indexedDB

indexedDB

indexedDB.open()

indexedDB.open()

IDBOpenDBRequest

```
var openRequest = window.indexedDB.open('test', 1);
```

test 1

open()

0

0

1

4

- **success**
- **error**
- **upgradeneeded**
- **blocked**

upgradeneeded

success

4

```
var openRequest = indexedDB.open('test', 1);
```

```
var db;
```

```
openRequest.onupgradeneeded = function (e) {  
  console.log('Upgrading...');  
}
```

```
openRequest.onsuccess = function (e) {  
  console.log('Success!');  
  db = openRequest.result;  
}
```

```
openRequest.onerror = function (e) {  
  console.log('Error');  
  console.log(e);  
}
```



indexedDB.deleteDatabase()



- success 
- error 

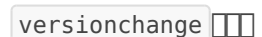
```
var DBDeleteRequest = window.indexedDB.deleteDatabase('demo');

DBDeleteRequest.onerror = function (event) {
    console.log('Error');
};

DBDeleteRequest.onsuccess = function (event) {
    console.log('success');
};
```

```
DBDeleteRequest.onerror = function (event) {
  console.log('Error');
};
```

```
DBDeleteRequest.onsuccess = function (event) {
    console.log('success');
};
```



indexedDB cmp()



```
window.indexedDB.cmp(1, 2) // -1
```



```

window.indexedDB.cmp(1, true) // []
window.indexedDB.cmp({}, {}) // []

```

IDBRequest []

IDBRequest [] indexedDB.open() [] indexedDB.deleteDatabase()

readyState [] pending
done []

success [] error [] result [] error
pending []

IDBRequest []

- IDBRequest.readyState [] pending [] done []
- IDBRequest.result []
- IDBRequest.error []
- IDBRequest.source [] ObjectStore[]
- IDBRequest.transaction [] null []
- IDBRequest.onsuccess [] success []
- IDBRequest.onerror [] error []

IDBOpenDBRequest [] IDBRequest []

- IDBOpenDBRequest.onblocked [] blocked [] upgradeneeded []
- IDBOpenDBRequest.onupgradeneeded [] upgradeneeded []

IDBDatabase []

IDBOpenDBRequest [] result [] IDBDatabase

```

var db;
var DBOpenRequest = window.indexedDB.open('demo', 1);

DBOpenRequest.onerror = function (event) {
  console.log('Error');
};

```

```
};

DBOpenRequest.onsuccess = function(event) {
    db = DBOpenRequest.result;
    // ...
};
```



IDBDatabase []

- IDBDatabase.name []
- IDBDatabase.version []
- IDBDatabase.objectStoreNames [DOMStringList []
object store []
- IDBDatabase.onabort [] abort []
- IDBDatabase.onclose [] close []
- IDBDatabase.onerror [] error []
- IDBDatabase.onversionchange [] upgradeneeded []
- indexedDB.deleteDatabase() []

[] objectStoreNames []
[]
[]

DOMStringList

DOMStringList [] contains

```
if (!db.objectStoreNames.contains('firstOS')) {
    db.createObjectStore('firstOS');
}
```

[]



IDBDatabase []

- IDBDatabase.close() []
- IDBDatabase.createObjectStore()
[] IDBObjectStore
[] versionchange []
- IDBDatabase.deleteObjectStore() [] versionchange
- IDBDatabase.transaction() [] IDBTransaction []

 createObjectStore()

```
var request = window.indexedDB.open('demo', 2);

request.onupgradeneeded = function (event) {
    var db = event.target.result;

    db.onerror = function(event) {
        console.log('error');
    };

    var objectStore = db.createObjectStore('items');

    // ...
};
```

Diagram illustrating a memory layout with four rows of memory cells:

- Row 1: 8 cells, with the last 4 cells labeled `items`.
- Row 2: 24 cells.
- Row 3: 8 cells, with the first 4 cells labeled `objectStoreNames`.
- Row 4: 24 cells, with the first 4 cells labeled `createObjectStore()`.

```
db.createObjectStore('test', { keyPath: 'email' });
db.createObjectStore('test2', { autoIncrement: true });
```

```

graph LR
    Node1["keyPath | email"] --> Node2["null | autoIncrement"]
    Node2 --> Node3["1 | 2"]
    Node3 --> Node4["false | keyPath"]
    Node4 --> Node5["keyPath | autoIncrement"]
    Node5 --> Node6["keyPath | keyPath"]
    Node6 --> Node7["keyPath | keyPath"]
    Node7 --> Node8["keyPath | keyPath"]
  
```

deleteObjectStore()

```
var dbName = 'sampleDB';  
var dbVersion = 2;  
var request = indexedDB.open(dbName, dbVersion);  
  
request.onupgradeneeded = function(e) {  
    var db = request.result;  
    if (e.oldVersion < 1) {  
        db.createObjectStore('store1');
```

```

}

if (e.oldVersion < 2) {
  db.deleteObjectStore('store1');
  db.createObjectStore('store2');
}

// ...
};

```

transaction() IDBTransaction

```
var t = db.transaction(['items'], 'readwrite');
```

transaction()

readonly readwrite readonly

IDBObjectStore

IDBObjectStore object store IDBDatabase.createObjectStore()

IDBDatabase transaction() objectStore() IDBObjectStore

```

db.transaction(['test'], 'readonly')
  .objectStore('test')
  .get(X)
  .onsuccess = function (e) {}

```



IDBObjectStore

- IDBObjectStore.indexNames DOMStringList
- IDBObjectStore.keyPath

- IDBObjectStore.name
- IDBObjectStore.transaction
- IDBObjectStore.autoIncrement



IDBObjectStore

1 IDBObjectStore.add()

IDBObjectStore.add() IDBRequest
 put()

objectStore.add(value, key)

null

add()

```
var db;
var DBOpenRequest = window.indexedDB.open('demo', 1);

DBOpenRequest.onsuccess = function (event) {
  db = DBOpenRequest.result;
  var transaction = db.transaction(['items'], 'readwrite');

  transaction.oncomplete = function (event) {
    console.log('transaction success');
  };

  transaction.onerror = function (event) {
    console.log('transaction error: ' + transaction.error);
  };

  var objectStore = transaction.objectStore('items');
  var objectStoreRequest = objectStore.add({ foo: 1 });

  objectStoreRequest.onsuccess = function (event) {
    console.log('add data success');
  };
};
```

2 IDBObjectStore.put()

```
IDBObjectStore.put()
```

[illegible]IDBRequest 

```
objectStore.put(item, key)
```

[illegible]

3 IDBObjectStore.clear()

IDBObjectStore.clear()

IDBRequest

--	--	--

```
objectStore.clear()
```

--	--	--	--	--	--	--	--	--

4 IDBObjectStore.delete()

```
IDBObjectStore.delete()
```

IDBRequest

```
objectStore.delete(Key)
```

5 IDBObjectStore.count()

IDBObjectStore.count()

IDBRequest

```
IDBObjectStore.count(key)
```

IDBKeyRange

❏ 6❏ IDBObjectStore.getKey()

```
IDBObjectStore.getKey()
```

IDBRequest

--	--	--

```
objectStore.getKey(key)
```

IDBKeyRange

7 IDBObjectStore.get()

IDBObjectStore.get() 

IDBRequest 

```
objectStore.get(key)
```

8 IDBObjectStore.getAll()

IDBObjectStore.getAll() 

IDBRequest 

```
// 省略
objectStore.getAll()

// 指定 IDBKeyRange オブジェクト
objectStore.getAll(query)

// 指定件数
objectStore.getAll(query, count)
```

9 IDBObjectStore.getAllKeys()

IDBObjectStore.getAllKeys() 

IDBRequest 

```
// 省略
objectStore.getAllKeys()

// 指定 IDBKeyRange オブジェクト
objectStore.getAllKeys(query)

// 指定件数
objectStore.getAllKeys(query, count)
```

10 IDBObjectStore.index()

IDBObjectStore.index() 

IDBIndex 

```
objectStore.index(name)
```



```
var t = db.transaction(['people'], 'readonly');
var store = t.objectStore('people');
```

```
var index = store.index('name');
```

```
var request = index.get('foo');
```

```

index()
( foo )
get()
name
name
get()
success
```

11 IDBObjectStore.createIndex()

```
IDBObjectStore.createIndex()
VersionChange
```

```
objectStore.createIndex(indexName, keyPath, objectParameters)
```

- indexName
- keyPath
- objectParameters

- unique true
- multiEntry true

```
person
```

```
var person = {
  name: name,
  email: email,
  created: new Date()
};
```

```
var store = db.createObjectStore('people', { autoIncrement: true });
```

```
store.createIndex('name', 'name', { unique: false });
store.createIndex('email', 'email', { unique: true });
```

```
name
email
```

12 IDBObjectStore.deleteIndex()

IDBObjectStore.deleteIndex() 

VersionChange 

```
objectStore.deleteIndex(indexName)
```

13 IDBObjectStore.openCursor()

IDBObjectStore.openCursor() 

```
IDBObjectStore.openCursor()
```



success  error



```
var t = db.transaction(['test'], 'readonly');
var store = t.objectStore('test');

var cursor = store.openCursor();

cursor.onsuccess = function (event) {
  var res = event.target.result;
  if (res) {
    console.log('Key', res.key);
    console.dir('Data', res.value);
    res.continue();
  }
}
```



target.result 

key 

value 

continue() 



null 

openCursor() 

IDBKeyRange





next 

prev 

nextunique 

prevunique 



14 IDBObjectStore.openKeyCursor()

IDBObjectStore.openKeyCursor() 

IDBObjectStore.openKeyCursor()

IDBTransaction

IDBTransaction

IDBDatabase.transaction()

IDBTransaction

```
var db;
var DBOpenRequest = window.indexedDB.open('demo', 1);

DBOpenRequest.onsuccess = function(event) {
  db = DBOpenRequest.result;
  var transaction = db.transaction(['demo'], 'readwrite');

  transaction.oncomplete = function (event) {
    console.log('transaction success');
  };

  transaction.onerror = function (event) {
    console.log('transaction error: ' + transaction.error);
  };

  var objectStore = transaction.objectStore('demo');
  var objectStoreRequest = objectStore.add({ foo: 1 });

  objectStoreRequest.onsuccess = function (event) {
    console.log('add data success');
  };

};
```

```
var trans1 = db.transaction('foo', 'readwrite');
var trans2 = db.transaction('foo', 'readwrite');
var objectStore2 = trans2.objectStore('foo')
var objectStore1 = trans1.objectStore('foo')
objectStore2.put('2', 'key');
```

```
objectStore1.put('1', 'key');
```



IDBTransaction []

- IDBTransaction.db [] IDBDatabase []
- IDBTransaction.error []
[]
null []
- IDBTransaction.mode [] readonly [] readwrite []
- IDBTransaction.objectStoreNames [] DOMStringList
[]
- IDBTransaction.onabort [] abort []
- IDBTransaction.oncomplete [] complete []
- IDBTransaction.onerror [] error []

IDBTransaction []

- IDBTransaction.abort() []
- IDBTransaction.objectStore(name) [] IDBObjectStore []

IDBIndex []

IDBIndex

[]

IDBIndex []

IDBIndex

[]

IDBObjectStore.index() [] IDBIndex []

```
var transaction = db.transaction(['contactsList'], 'readonly');
var objectStore = transaction.objectStore('contactsList');
var myIndex = objectStore.index('lastName');

myIndex.openCursor().onsuccess = function (event) {
  var cursor = event.target.result;
  if (cursor) {
    var tableRow = document.createElement('tr');
```

```

    tableRow.innerHTML = ' <td>' + cursor.value.id + '</td>'
                        + ' <td>' + cursor.value.lName + '</td>'
                        + ' <td>' + cursor.value.fName + '</td>'
                        + ' <td>' + cursor.value.jTitle + '</td>'
                        + ' <td>' + cursor.value.company + '</td>'
                        + ' <td>' + cursor.value.eMail + '</td>'
                        + ' <td>' + cursor.value.phone + '</td>'
                        + ' <td>' + cursor.value.age + '</td>';

    tableEntry.appendChild(tableRow);

    cursor.continue();
} else {
    console.log('Entries all displayed. ');
}
};

```

IDBIndex

- IDBIndex.name
- IDBIndex.objectStore
- IDBIndex.keyPath
- IDBIndex.multiEntry keyPath true
- IDBIndex.unique

IDBIndex

IDBRequest

- IDBIndex.count()
- IDBIndex.get(key)
- IDBIndex.getKey(key)
- IDBIndex.getAll()

IDBKeyRange

- IDBIndex.getAllKeys()
- IDBIndex.openCursor()
- IDBIndex.openKeyCursor()

IDBCursor

IDBCursor

IDBCursor []

IDBObjectStore []

IDBIndex []

IDBCursor []

IDBObjectStore.openCursor() []

```
var transaction = db.transaction(['rushAlbumList'], 'readonly');
var objectStore = transaction.objectStore('rushAlbumList');

objectStore.openCursor(null, 'next').onsuccess = function(event) {
  var cursor = event.target.result;
  if (cursor) {
    var listItem = document.createElement('li');
    listItem.innerHTML = cursor.value.albumTitle + ', ' + cursor.value.year;
    list.appendChild(listItem);

    console.log(cursor.source);
    cursor.continue();
  } else {
    console.log('Entries all displayed.');
  }
};
```

IDBCursor []

- IDBCursor.source []
- IDBCursor.direction []
[] next
[] nextunique [] prev
[] prevunique
[] IDBObjectStore.openCursor()
- IDBCursor.key []
- IDBCursor.value []
- IDBCursor.primaryKey []
[] objectStore
[] IDBCursor.key [] IDBCursor.key
[]

IDBCursor []

- IDBCursor.advance(n) [] n []
- IDBCursor.continue()
[]
- IDBCursor.continuePrimaryKey() [] key [] primaryKey []
[]

- IDBCursor.delete() 
- 
- IDBCursor.update() 
- 

IDBRequest

IDBRequest

IDBKeyRange

IDBKeyRange  object store



IDBKeyRange 

- IDBKeyRange.lowerBound() 
- IDBKeyRange.upperBound() 
- IDBKeyRange.bound() 
- IDBKeyRange.only() 



```
// All keys ≤ x
var r1 = IDBKeyRange.upperBound(x);

// All keys < x
var r2 = IDBKeyRange.upperBound(x, true);

// All keys ≥ y
var r3 = IDBKeyRange.lowerBound(y);

// All keys > y
var r4 = IDBKeyRange.lowerBound(y, true);

// All keys ≥ x && ≤ y
var r5 = IDBKeyRange.bound(x, y);

// All keys > x && < y
var r6 = IDBKeyRange.bound(x, y, true, true);

// All keys > x && ≤ y
var r7 = IDBKeyRange.bound(x, y, true, false);
```

```
// All keys ≥ x &&< y
var r8 = IDBKeyRange.bound(x, y, false, true);

// The key = z
var r9 = IDBKeyRange.only(z);
```

IDBKeyRange.lowerBound() IDBKeyRange.upperBound() IDBKeyRange.bound()

□□□□ IDBKeyRange □□□□□□□□

- IDBKeyRange.lower 
- IDBKeyRange.lowerOpen 
- IDBKeyRange.upper 
- IDBKeyRange.upperOpen 

IDBKeyRange []

IDBObjectStore □ IDBIndex □□

```
openCursor()
```

```
var t = db.transaction(['people'], 'readonly');
var store = t.objectStore('people');
var index = store.index('name');

var range = IDBKeyRange.bound('B', 'D');

index.openCursor(range).onsuccess = function (e) {
  var cursor = e.target.result;
  if (cursor) {
    console.log(cursor.key + ':');

    for (var field in cursor.value) {
      console.log(cursor.value[field]);
    }

    cursor.continue();
  }
}
```

IDBKeyRange [][][][][]

```
includes(key)
```

[illegible]

```
var keyRangeValue = IDBKeyRange.bound('A', 'K', false, false);
```

```
keyRangeValue.includes('F') // true
```

```
keyRangeValue.includes('W') // false
```



- Raymond Camden, [Working With IndexedDB - Part 1](#)
- Raymond Camden, [Working With IndexedDB - Part 2](#)
- Raymond Camden, [Working With IndexedDB - Part 3](#)
- Tiffany Brown, [An Introduction to IndexedDB](#)
- David Fahlander, [Breaking the Borders of IndexedDB](#)
- TutorialsPoint, [HTML5 - IndexedDB](#)

📄 #2

📄 📄 📄 2018-07-04 08:21:48 UTC

📄 📄 📄 2021-09-01 14:43:33 UTC