# Mongodb

**https://www.npmjs.com/package/mongodb**

**Description**

The official [MongoDB](https://www.mongodb.com/) driver for Node.js. Provides a high-level API on top of [mongodb-core](https://www.npmjs.com/package/mongodb-core) that is meant for end users.

**NOTE: v3.x was recently released with breaking API changes. You can find a list of changes**[here](https://github.com/mongodb/node-mongodb-native/blob/HEAD/CHANGES_3.0.0.md)**.**

**MongoDB Node.JS Driver**

| **what** | **where** |
| --- | --- |
| documentation | <http://mongodb.github.io/node-mongodb-native> |
| api-doc | <http://mongodb.github.io/node-mongodb-native/3.1/api> |
| source | <https://github.com/mongodb/node-mongodb-native> |
| mongodb | [http://www.mongodb.org](http://www.mongodb.org/) |

# Installation

The recommended way to get started using the Node.js 3.0 driver is by using the npm (Node Package Manager) to install the dependency in your project.

## MongoDB Driver

Given that you have created your own project using npm init we install the MongoDB driver and its dependencies by executing the following npm command.

npm install mongodb --save

This will download the MongoDB driver and add a dependency entry in your package.jsonfile.

You can also use the [Yarn](https://yarnpkg.com/en) package manager.

The MongoDB driver depends on several other packages. These are:

* [mongodb-core](https://github.com/mongodb-js/mongodb-core)
* [bson](https://github.com/mongodb/js-bson)
* [kerberos](https://github.com/mongodb-js/kerberos)
* [node-gyp](https://github.com/nodejs/node-gyp)

The kerberos package is a C++ extension that requires a build environment to be installed on your system. You must be able to build Node.js itself in order to compile and install the kerberos module. Furthermore, the kerberos module requires the MIT Kerberos package to correctly compile on UNIX operating systems. Consult your UNIX operation system package manager for what libraries to install.

**Windows already contains the SSPI API used for Kerberos authentication. However, you will need to install a full compiler tool chain using Visual Studio C++ to correctly install the Kerberos extension.**

var mongoClient = require('mongodb').MongoClient;

mongoClient.connect('mongodb://localhost:27017/learning\_mongo',{ useNewUrlParser: true } , function(err, client) {

console.log("Connected successfully to server");

console.log(client);

console.log("err::::"+err);

const db = client.db("learning\_mongo");

const collection = db.collection('tours');

db.collection('tours').find({}).toArray(function(err, docs) {

console.log("docs:::"+JSON.stringify(docs));

});

});

/\* database operation by mongoDb client

db.collection("customer").findOne({}, function(err, result) {

if (err) throw err;

console.log(result);

db.close();

});

db.collection("customer").find({}, function(err, result) {

if (err) throw err;

//console.log(result);

result.toArray(function(err , items){

//console.log("items :::"+JSON.stringify(items));

});

db.close();

});

\*/

db.collection("customer").aggregate([

{$lookup: {from : "cityMaster", localField : "cityId", foreignField : "id" , as: "CityDetail"}}, {$match : {details : {$ne:[]}}}

],function(err,result){

if (err) throw err;

//console.log(result);

result.toArray(function(err , items){

console.log("items :::"+JSON.stringify(items));

});

db.close();

});

**what find provides to the callback is a**[**Cursor**](http://mongodb.github.io/node-mongodb-native/2.0/api/Cursor.html)**, not an array of documents. So if you want your callback to provide results as an array of documents, call**[**toArray**](http://mongodb.github.io/node-mongodb-native/2.0/api/Cursor.html#toArray)**on the cursor to return them:**

collection.find({'\_id':o\_id}, function(err, cursor){

cursor.toArray(callback);

db.close();

});

Note that your function's callback still needs to provide an err parameter so that the caller knows whether the query worked or not.

### module

var mongo **=** require('mongodb');

var Db **=** mongo.Db;

var Server **=** mongo.Server;

var MongoClient **=** mongo.MongoClient;

var ReplSetServers **=** mongo.ReplSetServers;

**...**

# Node.js MongoDB JoinJoin Collections

MongoDB is not a relational database, but you can perform a left outer join by using the $lookup stage.

The $lookup stage lets you specify which collection you want to join with the current collection, and which fields that should match.

Consider you have a "orders" collection and a "products" collection:

### **orders**

[  
  { \_id: 1, product\_id: 154, status: 1 }  
]

### **products**

[  
  { \_id: 154, name: 'Chocolate Heaven' },  
  { \_id: 155, name: 'Tasty Lemons' },  
  { \_id: 156, name: 'Vanilla Dreams' }  
]

### **Example**

Join the matching "products" document(s) to the "orders" collection:

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection('orders').aggregate([  
**{ $lookup:  
       {  
         from: 'products',  
         localField: 'product\_id',  
         foreignField: '\_id',  
         as: 'orderdetails'  
       }  
     }**    ]).toArray(function(err, res) {  
    if (err) throw err;  
    console.log(JSON.stringify(res));  
    db.close();  
  });  
});

[Run example »](https://www.w3schools.com/nodejs/shownodejs_cmd.asp?filename=demo_mongodb_join)

Save the code above in a file called "demo\_mongodb\_join.js" and run the file:

Run "demo\_mongodb\_join.js"

C:\Users\Your Name>node demo\_mongodb\_join.js

Which will give you this result:

[  
  { "\_id": 1, "product\_id": 154, "status": 1, "orderdetails": [  
    { "\_id": 154, "name": "Chocolate Heaven" } ]  
  }  
]