Node.js - MongoDB

We’ll be using the official mongodb [npm](https://flaviocopes.com/npm/) package. If you already have a Node.js project you are working on, install it using

npm install mongodb

If you start from scratch, create a new folder with your [terminal](https://flaviocopes.com/macos-terminal/) and run npm init to start up a new Node.js project, and then run the npm install mongodb command.

**Connecting to MongoDB**

You require the mongodb package and you get the MongoClient object from it.

const mongo = require('mongodb').MongoClient

Create a URL to the MongoDB server. If you use MongoDB locally, the URL will be something like mongodb://localhost:27017, as 27017 is the default port.

const url = 'mongodb://localhost:27017'

Then use the mongo.connect() method to get the reference to the MongoDB instance client:

mongo.connect(url, (err, client) => {

if (err) {

console.error(err)

return

}

//...

})

Now you can select a database using the client.db() method:

const db = client.db('kennel')

**Create and get a collection**

You can get a collection by using the db.collection() method. If the collection does not exist yet, it’s created.

const collection = db.collection('dogs')

**Insert data into a collection a Document**

Add to app.js the following function which uses the insertOne() method to add an object dogs collection.

collection.insertOne({name: 'Roger'}, (err, result) => {

})

You can add multiple items using insertMany(), passing an array as the first parameter:

collection.insertMany([{name: 'Togo'}, {name: 'Syd'}], (err, result) => {

})

**Find all documents**

Use the find() method on the collection to get all the documents added to the collection:

collection.find().toArray((err, items) => {

console.log(items)

})

**Find a specific document**

Pass an object to the find() method to filter the collection based on what you need to retrieve:

collection.find({name: 'Togo'}).toArray((err, items) => {

console.log(items)

})

If you know you are going to get one element, you can skip the toArray() conversion of the cursor by calling findOne():

collection.findOne({name: 'Togo'}, (err, item) => {

console.log(item)

})

**Update an existing document**

Use the updateOne() method to update a document:

collection.updateOne({name: 'Togo'}, {'$set': {'name': 'Togo2'}}, (err, item) => {

console.log(item)

})

**Delete a document**

Use the deleteOne() method to delete a document:

collection.deleteOne({name: 'Togo'}, (err, item) => {

console.log(item)

})

**Closing the connection**

Once you are done with the operations you can call the close() method on the client object:

client.close()

**Use promises or async/await**

I posted all those examples using the [callback](https://flaviocopes.com/javascript-callbacks/) syntax. This API supports [promises](https://flaviocopes.com/javascript-promises/) (and [async/await](https://flaviocopes.com/javascript-async-await/)) as well.

For example this

collection.findOne({name: 'Togo'}, (err, item) => {

console.log(item)

})

Can be used with promises:

collection.findOne({name: 'Togo'})

.then(item => {

console.log(item)

})

.catch(err => {

console.error(err)

})

or async/await:

const find = async () => {

try {

const item = await collection.findOne({name: 'Togo'})

} catch(err => {

console.error(err)

})

}

find()