**NOTE: This is also called HW\_4\_Ch\_16\_Ex.docx**

**MongoDB and Node**

**There are basically two ways to work with MongoDB on Node --- thru the official MongoClient or thru Mongoose.  
  
Mongoose** is an API in which you have a design/model for your particular database.

**MongoClient** provides a complete connection to all the Mongo functions.

Which you will use depends on your needs.

In addition to that, you need to ask if you are using a local version of Mongo (as we did in CS221) or the Atlas (cloud) version.

**Let's start with the local version and the Mongo Client.**

* Node has a package mongodb which establishes communication between a MongoDB server and a Node server. The name of the connection is MongoClient.
* By convention (there are different conventions for each major DB) Mongo uses port 27017.
* So you need **npm install mongodb –save**
* The next information is from <https://www.npmjs.com/package/mongodb> (scroll down to "Diagnosing on Windows").
  + You will want to make sure that your paths for node is on your mongodb path
  + Your MongoDB should have been previous installed (or do it now).
  + If you followed the instructions for MongoDB Version 4.4 this fall your data folder will be in **C:\Program Files\MongoDB\Server\4.4\data\**  and the path to that will be on your Mongo server path. For older versions your data might be at **C:\data**I believe that even with version 4.4 you can change to C:\data by starting **mongod** andsaying  **mongod --dbpath=/data**
  + Once you have done all this, your app can say  
     **const MongoClient = require('mongodb').MongoClient;**
  + You can make sure your connection works with the following app:

const MongoClient = require('mongodb').MongoClient;  
const assert = require('assert');  
  
// Connection URL  
const url = 'mongodb://localhost:27017';

// Database Name  
const dbName = 'myproject';  
const client = new MongoClient(url);

// Use connect method to connect to the server  
client.connect(function(err) {  
 assert.equal(null, err);  
 console.log('Connected successfully to server');

const db = client.db(dbName);  
 client.close();  
});

* Typically you will code something like  
    
   **const collection = db.collection('**myCollection');   
    
  to simplify your code --- but you are basically now going to do classic MongoDB scripting.
* Another good source of instruction is <https://attacomsian.com/blog/nodejs-mongodb-local-connection> (which has info about both Mongoose and MongoClient)  
    
  Again the basic steps inside your app are:  
   **const** **MongoClient = require('mongodb').MongoClient;  
   const url = 'mongodb://127.0.0.1:27017';** //127.0.0.1 is localhost  
    
   **const dbName = '**whateverDB\_YouWantToUse**';  
   MongoClient.connect(url, function(err, client){  
   if (err) {** some error handling and exit  
   **} else {  
   console.log('Connected to server and db ' + dbName);  
   const db = client.db(dbName);  
   // now carry on db operations – see the two sits for examples  
   client.close()  
   });**

<https://flaviocopes.com/node-mongodb/> has similar examples.

**Next let's look at a local MongoDB database and Mongoose.**

* Of course you will need to **npm install mongoose**
* In your app your will define a url which includes the name of the db you want to use  
  **const mongoose = require('mongoose');  
  const url = 'mongodb://127.0.0.1:27017//**myDB**';**
* **Now you are ready to connect or to createCollection – but you also need a model for your database.**
* <https://attacomsian.com/blog/nodejs-getting-started-with-mongoose>shows an example of writing a schema (which will be compiled into a model)  
  <https://www.npmjs.com/package/mongoose> has other examples.

**If you are using MongoDB with Atlas**<https://docs.mongodb.com/drivers/node/current/> has the official API documentiation.

The Fundamentals section has information about how to set up a connection. This assumes that you have already set up an Atlas database, which you can do at <https://www.mongodb.com/nodejs-database> There is also sample code there for you to try. You will see that the functions as specified as being asynch, but otherwise it looks very much like the mongo code from CS 221.

Appendix – remember from CS 221

You open a terminal window and migrate to C:\Program Files\MongoDB\Server\bin

In the terminal window you type **mongod**

At the folder C:\Program Files\MongoDB\Server\bin in your File Explorer you hold the shift key & right click to get a PowerShell window and when it opens you type **./mongo**

You can now say **show dbs** and after you choose one **show collections**

**Setting up:**

Using VS Code, in the folder where I have all my Node applications I typed:

mkdir forMongo  
cd forMongo  
mkdir checkConn  
I then copied the code on p.3 into app.js in the checkConn folder  
npm install mongdb  
npm init  
node app

AND it worked! (I got a message that I'm using an old version of mongo – which I knew.)

**Code for checkConn app.js**const MongoClient = require('mongodb').MongoClient;  
const assert = require('assert');

// Connection URL  
const url = 'mongodb://localhost:27017';

// Database Name  
const dbName = 'myproject'; //or whatever  
const client = new MongoClient(url);  
// Use connect method to connect to the server  
client.connect(function(err) {  
 assert.equal(null, err);  
 console.log('Connected successfully to server');  
 const db = client.db(dbName);  
 client.close();  
});

**Code for SeePlayers2 app.js**

const MongoClient = require('mongodb').MongoClient;  
const assert = require('assert');

// Connection URL  
const url = 'mongodb://localhost:27017';

// Database Name  
const dbName = 'penguins';  
const client = new MongoClient(url, {useNewUrlParser: true,  
 useUnifiedTopology: true});  
 //I need to pass these options in b/c running mongodb v3.6- not nec. with 4.4

// Use connect method to connect to the server  
client.connect(function(err) {  
 //assert.equal(null, err); equal is deprecaated  
 console.log('Connected successfully to server');  
  
 const db = client.db(dbName);

console.log('Now try to find all players');

const findDocuments = function(db, callback) {  
 // Get the players collection  
 const collection = db.collection('players');

// Find some documents  
 collection.find({}).toArray(function(err, docs) {  
 //assert.equal(err, null);  
 console.log('Found the following records');  
 console.log(docs);  
 callback(docs);  
 });  
 };  
 findDocuments(db, function(){client.close()});  
});

**Other references:**<https://www.mongodb.com/developer/quickstart/node-crud-tutorial/> (This actually starts at <https://www.mongodb.com/blog/post/quick-start-nodejs-mongodb-how-to-get-connected-to-your-database> about how to connect.) This is mostly about how to CRUD in mongo (which you learned in CS221.) It does use asynch functions (which makes a lot of sense if you are CRUDing remotely.)

<https://www.guru99.com/node-js-mongodb.html> This is a very clear tutorial which includes connecting to the the DB and displaying the results with Express.

<https://www.youtube.com/watch?v=bxsemcrY4gQ> This is part of the Simplilearn tutorials on Node.js The part on Mongo starts at minute 2:24 and it shows both how to use Atlas (how to set up an account on Atlas with a user name and password, and then save the connection string to put into your Express app) and then, starting at minute 6:51, how to use mongoose. The information (again, which you know from CS221) and exmaples follow the information about connecting. (That starts at minute 19:26)

NOTE: Mongoose, unlike mongo, starts with a *schema* – that is your DB will have a *model* – that is it will have a *class* of documents, each of which follows the *schema.* You can find an introduction at <https://mongoosejs.com/docs/> or from the video above.   
  
If you have collection thing then mongoose will look in the collection thing**s** (the plural of the name of your model.) That is you created a Schema (define an object with the key-value pairs, and you can have options for the values, such as whether or not they are required) and then you create a model with a name (e. g. thing) which identifies the collection, and with the name of the schema you will be using for that collection.

The collection (as specified in the model) will have the usual MongoDB methods such as find() etc.

The mongoose methods such as those used to store new documents, retrieve documents, etc. are all asynchronous (return a promise), so they have a then() method which takes whatever is returned and handles it, displays it, etc.   
  
 app. get('some\_route', (req, res) => {  
 myModel.find()  
 .then( (result) => {res.send(result)})  
 });