

INFO3139 Lab 5

Rev 1.0

General JavaScript Topic #4 – IIFE (Immediately Invoked Function Expressions)

An IIFE is a JavaScript function that runs as soon as it is defined. It typically takes this format:

```
(function () {  
    statements  
})();
```

According to MDN It is a design pattern which is also known as a Self-Executing Anonymous Function and contains two major parts:

- The first is the anonymous function with its scope enclosed within the Grouping Operator () (the first and 3rd last parenthesis in this case). This prevents accessing variables within the IIFE idiom as well as polluting the global scope.
- The second part creates the immediately invoked function expression () through which the JavaScript engine will directly interpret the function.

Redo an example we looked at last class as an IIFE. Create a new folder called **week3\class1** and place a file called **iifeexample.js** in it with this code:

```
// Load the got module  
import got from "got";  
  
(async () => {  
    // IIFE  
    // Lets try to make a HTTP GET request to GOC's website and get some transfer info in JSON.  
    const transferPaymentURL =  
        "http://www.infrastructure.gc.ca/alt-format/opendata/transfer-program-programmes-de-transfert-bil.json";  
  
    // Create a currency formatter.  
    const formatter = new Intl.NumberFormat("en-US", {  
        style: "currency",  
        currency: "USD",  
        minimumFractionDigits: 0,  
    });  
  
    try {  
        // grab the remote json using got  
        const transferPayments = await got(transferPaymentURL).json();  
  
        // strip out the Ontario amount  
        let ont = transferPayments.ccbf.on["2022-2023"];  
  
        // format to currency  
        ont = formatter.format(ont);  
  
        // dump to the console using template literal  
        console.log(`Ontario's transfer amount for 2022-2023 was ${ont}`);  
    } catch (error) {  
        console.log(error);  
        //=> 'Internal server error ...'  
    }  
})(); // IIFE
```

Then execute the new function in a terminal window to confirm its working like it did before:

```
PS E:\winter2023\info3139\programming\nodeexercises\week3\class2> node iife_example
Ontario's transfer amount for 2022-2023 was $853,621,164
_
```

Now back to today's material...

Case Study# 1 – Some Preliminary Work

Some of the code in the lab from last class (lab4) can be used in the upcoming case study. As such, we can start doing some of the plumbing for the case study today:

1. Create a new folder under nodeexercises called **project1**
2. Create **.env** and **config.js** files to utilize the following constants:

```
GOALERTS=http://data.international.gc.ca/travel-voyage/index-alpha-eng.json
ISOCOUNTRIES=https://raw.githubusercontent.com/elauersen/info3139/master/isocountries.json
```

3. Create a new JavaScript module file called **utilities.js**
 - o Add the `getJSONFROMWWWPromise` from lab4 and add an import for `got` and an export for the method.
4. Create a new JavaScript file called **project1_setup.js**, create an async method that loads the ISO Countries into an array using the utilities module and then loads the GOALERTS information into a variable (`alertJson` shown below).

You can't dump out the number of nodes using the `length` method like we will for countries. The reason is, the alert file data is **not in an array**. You can however, use the following to determine the number of different countries in the **data** property (assumes you've loaded the actual json in a variable `alertJson`).

```
Object.keys(alertJson.data).length
```

From StackOverflow: *Object.keys(obj).length; Works by internally iterating over the keys to compute a temporary array and returns its length.*

5. For now call the async method and produce some count output like this when issuing **node project1_setup**:

```
PS E:\winter2023\info3139\programming\nodeexercises\project1> node project1_setup
Retrieved Alert JSON from remote web site.
Retrieved Country JSON from remote web site.
There are 230 alerts and 249 countries
_
```


We'll come back to this project next class once we get our mongo database setup.

MongoDB with Atlas

6. Start out by registering for a free account on a web service called MongoDB Atlas at:

<https://www.mongodb.com/cloud/atlas/signup>

Get started free
No credit card required.

 Sign up with Google

or

Your Company (optional)
Fanshawe College

Your Work Email
elauersen@fanshaweonline.ca

First Name
Evan

Last Name
Lauersen

Password

☒ I agree to the terms of service and privacy policy.


Get started free

Included with


- ✓ 512 MB of Storage
- ✓ Shared RAM
- ✓ Highly available automated patching


Additionally, you get the following when you create a new cluster:

- ✓ 10 GB or more of storage
- ✓ Dedicated RAM
- ✓ Performance optimized
- ✓ Backups & point-in-time recovery
- ✓ Enterprise security
- ✓ Key management
- ✓ Database audit logs
- ✓ Global Clusters


 **MongoDB.**

Great, now verify your email




 **MongoDB.**

Email Address Verification



Verify Email

Email successfully verified!



Continue

Welcome to Atlas! 🌱

Tell us a few things about yourself and your project.



What is your goal today?

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

- ☐ Explore what I can build
- ☒ Build a new application
- ☐ Migrate an existing application
- ☐ Learn MongoDB

What type of application are you building?

Mobile

What is your preferred language?

We'll use this to customize code samples and content we share with you. You can always change this later.

JS JavaScript

Finish

less

development and
loads with variable
configuration

or the operations you

scale seamlessly to
workload

security and

reate

Starting at
1M reads

ADVANCED

Dedicated

For production applications with
sophisticated workload requirements.
Advanced configuration controls.

- ✓ Network isolation and fine-grained access controls
- ✓ On-demand performance advice
- ✓ Multi-region and multi-cloud options available

Create

Starting at
\$0.08/hr*

FREE

Shared

For learning and exploring MongoDB
in a cloud environment. Basic
configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at
FREE

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ^



★ Recommended region ⓘ  Dedicated tier region ⓘ

NORTH AMERICA

 N. Virginia (us-east-1) ★

 Oregon (us-west-2) ★

 Ohio (us-east-2) ★ 

 N. California (us-west-1) 

 Montreal (ca-central-1) ★ 

EUROPE

 Frankfurt (eu-central-1)

 Paris (eu-west-3) ★

 Ireland (eu-west-1) ★

 Stockholm (eu-north-1) ★

 London (eu-west-2) ★ 

AUSTRALIA

 Sydney (ap-southeast-2) ★

ASIA

 Tokyo (ap-northeast-1) ★

 Singapore (ap-southeast-1) ★

 Hong Kong (ap-east-1) ★

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)

Create Cluster

FANSHAWE COLLEGE > PROJECT 0

Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

✓ How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password. You can manage existing users via the [Database Access Page](#).

Username

elauersen

Password 

.....

 Autogenerate Secure Password

 Copy

Create User

This password contains special characters which will be URL-encoded.

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

IP Address

Description

Enter IP Address

Enter description

Add My Current IP Address

Add Entry

IP Access List

Description

192.24.47.227/32

My IP Address

EDIT

REMOVE

ned provisioning.



Finish and Close

Congratulations on setting up access rules!

You will now be able to connect to your deployments. You can continue to add and update access rules in [Database Access](#) and [Network Access](#).

- ☒ Hide Quickstart guide in the navigation. You can visit [Project Settings](#) to access it in the future.

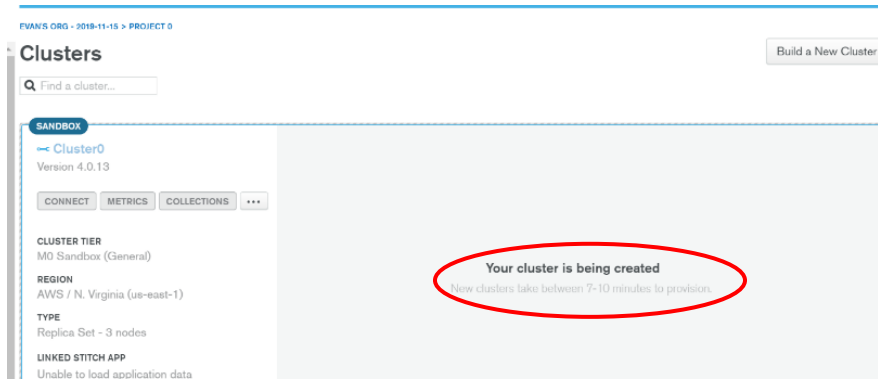
Go to Databases

Introducing Termination Protection

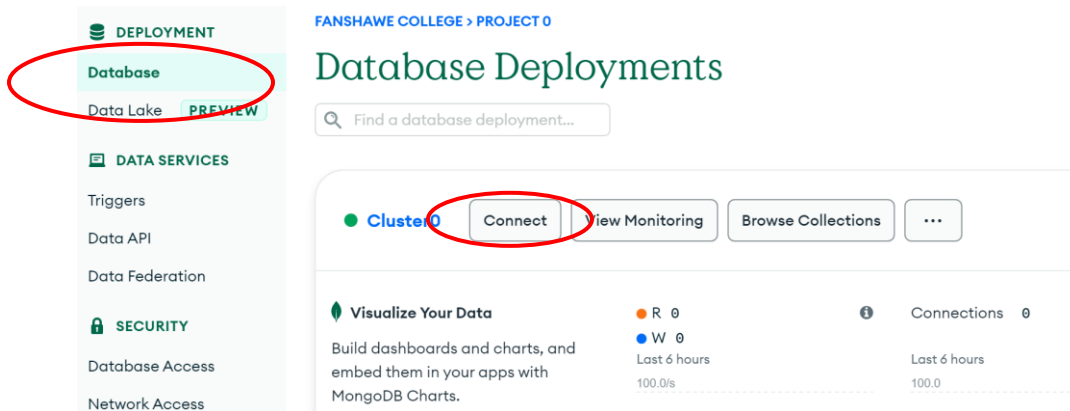
Now, you can prevent any user from accidentally deleting a cluster.

Setup

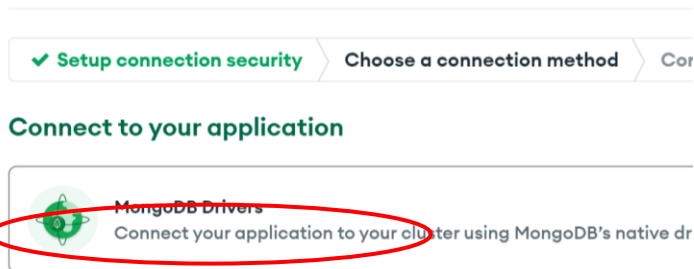
Close



7. Obtain the connection string for your application (replace the <password> string with your actual password, if your password has special characters in it, you need to use the hex value e.g. s@mepassword would be s%40mepassword)



Connect to Cluster0



Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

1 Select your driver and version

DRIVER: Node.js | VERSION: 4.1 or later

2 Add your connection string into your application code

☐ Include full driver code example

mongodb+srv://elauersen:<password>@cluster0.tnt4ujv.mongodb.net/?retryWrites=true&w=majority

Replace <password> with the password for the elauersen user. Ensure any option params are URL encoded.

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back | Close

8. Edit a new `week3\class2\.env` file and paste the connection string and add a database constant in it like the following:

```
DBURL= mongodb+srv://yourusername...:.....Rest of your connection string goes here
DB=info3139db
```

9. Setup up a `config.js` (feel free to change the names to your liking) to match up to the `.env` file e.g.:

```
const dotenv = require('dotenv');
dotenv.config();
module.exports = {
  atlas: process.env.DBURL,
  appdb: process.env.DB
};
```

10. Install the **mongodb** node driver in the **nodeexercises** folder:

```
PS E:\winter2023\info3139\programming\nodeexercises> npm i mongodb
added 94 packages, and audited 134 packages in 41s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
"dependencies": {
  "dotenv": "^16.0.3",
  "got": "^12.5.2",
  "mongodb": "^4.11.0",
  "yargs": "^17.6.0"
}
```


Notes from the Homework Reading on MongoDB

- Before we get to coding, review some Mongo terms, do a comparison with relational database terms in this table:

Relational Database Term	MongoDB Term
Database	Database
Table	Collection
Row	Document
Column	Field

Interesting side note, MongoDB use to use the same V8 engine as Node for its JavaScript processor, not anymore. Now it uses **SpiderMonkey** (developed by the Mozilla Group same as Firefox).

Creating and Using the info3139db database

11. Create a new file called week3\class2**db_routines.js** with the following code:

```
import { MongoClient } from "mongodb";
import * as cfg from "../config.js";
let db;

const getDBInstance = async () => {
  if (db) {
    console.log("using established connection");
    return db;
  }

  try {
    const client = new MongoClient(cfg.atlas, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log("establishing new connection to Atlas");
    const conn = await client.connect();
    db = conn.db(cfg.appdb);
  } catch (err) {
    console.log(err);
  }

  return db;
};

export { getDBInstance };
```

12. Create another file called **mongo_exercise1.js** with the following:

```
import { getDBInstance } from './db_routines.js';

const connectDb = async () => {
  try {
    let db = await getDBInstance();
    console.log(`established connection for ${db.databaseName} on Atlas`);
  } catch (err) {
    console.log(err);
  } finally {
    process.exit(); // we don't need to disconnect, connection pooled
  }
};

connectDb();
```

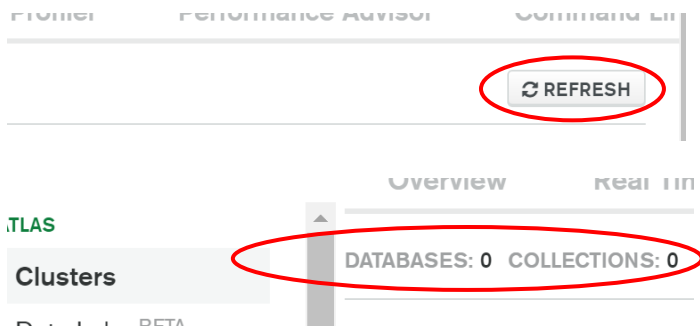
13. Sign-in to Atlas and look at the databases available (none is shown below):



14. Execute the **mongo_exercise1.js** file now, which results in our program connecting to Atlas using the **.dotenv/config** connection string value:

```
PS E:\winter2023\info3139\programming\nodeexercises\week3\class2> node mongo_exercise1
establishing new connection to Atlas
established connection for info3139db on Atlas
```

15. However, if we refresh the Atlas page using the refresh button, note we see **no new database has been established**, the database **is not created** until we actually add something to it.



16. Edit the **db_routines.js** and add the following methods and **update the export** accordingly:

```
const addOne = (db, coll, doc) => db.collection(coll).insertOne(doc);
const count = (db, coll) => db.collection(coll).countDocuments();
```

17. Add a new file called week4\mongo_exercise2.js with the following code:

```
import * as dbRtns from "../db_routines.js";

const addADocument = async () => {
  try {
    let db = await dbRtns.getDBInstance();
    console.log(`established connection for ${db.databaseName} on Atlas`);
    let results = await dbRtns.addOne(db, "testcollection", {
      property1: "somedata",
    });
    let count = await dbRtns.count(db, "testcollection");
    results.insertedId
      ? console.log(
          `added new document to testcollection, there are currently ${count} documents in the testcollection`
        )
      : console.log("document not added");
  } catch (err) {
    console.log(err);
  } finally {
    process.exit(); // we don't need to disconnect, connection pooled
  }
};

addADocument();
```

18. Then execute the new mongo_exercise program 2 times:

```
PS E:\winter2023\info3139\programming\nodeexercises\week3\class2> node mongo_exercise2
establishing new connection to Atlas
established connection for info3139db on Atlas
added new document to testcollection, there are currently 1 documents in the testcollection
PS E:\winter2023\info3139\programming\nodeexercises\week3\class2> node mongo_exercise2
establishing new connection to Atlas
established connection for info3139db on Atlas
added new document to testcollection, there are currently 2 documents in the testcollection
```

19. Again, refresh the Atlas console once more:

 REFRESH

info3139db.testcollection

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 92B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Index

FILTER { field: 'value' }

QUERY RESULTS: 1-2 OF 2

_id: ObjectId('6369181d621d74d0fb05cf19')
property1: "somedata"

_id: ObjectId('6369182292cde628edf2a4e2')
property1: "somedata"

And we see now we have a new database called info3139db, a new collection called **testcollection** and 2 new documents as well

Lab 5 – 2.5%

Part A - (2%)

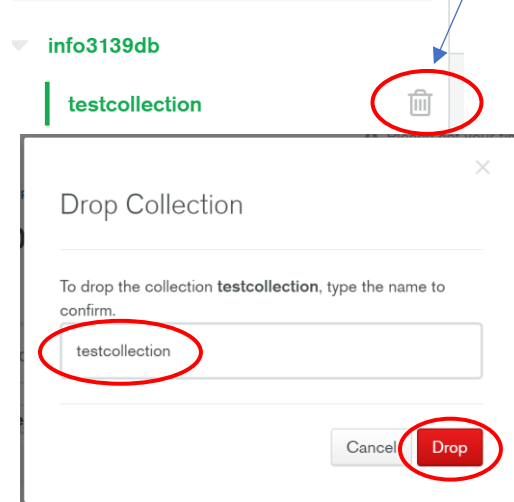
1. Start the project1 setup work described in steps 1-5. Submit the following in a **single Word document**:
 - Screen shot of project1_setup executing
 - Source code for the utilities.js and project1_setup.js files
2. Then update the code in the addADocument method (step 18) and replace somedata with **your name** and then re-execute mongo_exercise2
 - take a screenshot of the new document in your testcollection in your atlas database and include it in the **same word doc**

Lab 4 Part B - (.5%) - Lab 4 Theory Quiz on FOL - 10 questions

Review Lab 5

1. What does the acronym IIFE stand for?
2. Why would we use one?
3. What version of MongoDB did we utilize?
4. What public cloud provider is used with the default sandbox option?
5. What database construct in Mongo is analogous to a row in a relational database?
6. What database construct in Mongo is analogous to a table in a relational database?
7. What JavaScript engine does Mongo use?
8. When manually adding a document to the collection, what are you passing to Atlas?
9. What are we installing with this command: `npm i mongodb`?
10. T/F – a promise is returned with this db routines code:
`db.collection(coll).insertOne(doc);`
11. What syntax do we use to connect to a database with the native driver?
12. What property of the returned variable from the dbRtns.addOne method did we check to ensure the add operation worked?

After capturing the lab screen shot, using the Atlas console **delete** the **testcollection** collection as it is no longer needed:



Which results back to the original display:

DATABASES: 0 COLLECTIONS: 0