Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VI

02 May 2021

UCS 1611 - Internet Programming Lab

Exercise 8: Programs using Node.js

Learning Objective:

1. Write a Node.js program that reads all the greetings from the file greetings.txt, asks the user "What is your name?", then prints a random greeting followed by the given name. Make sure to check for the case where the file doesn't exist! For example, if the greeting is "Hey", then the program will print "Hey, Joe" to the console, then pick some other greeting and do the same until finished. Use Non-blocking I/O. Home, Committee, Call For Papers, Important Dates, Workshops, Registration and Contact.

2. Write a Node.js program that reads all the greetings as before. When all the greetings are loaded, it creates a server listening on port number 8080. On request, it checks for whether there is a name value in the query string. If there isn't, the value of query.name will be undefined. In other words, if you access http://localhost:8080/?name=Mike, then your browser should just display something like "Hello, Mike" when the page loads.

3. Create a web server using node.js which listens for clients request. Once the client request the server, the server returns a web page which contains a list of books and its details in table format.

4. Create a DB with the following details using MongoDB:
   Database Name: PatientDetails
   Table Schema: Name, age, ID, gender, address, marital status, Date of Visit
   Write a node.js program to do the following operations:
   Add, Delete, Update, Search.

Code - Console Greetings:

```javascript
let fs = require("fs");
let rl = require("readline");

let greetings = [];

//Non-blocking file IO
fs.readFile("greetings.txt", (error, data) => {
    if (error) {
        console.error(error);
    } else {
        greetings = data.toString().split("\n");
    }
});

//Interface for the input and output
const readline = rl.createInterface({
    input: process.stdin,
    output: process.stdout
});

//Ask for user response
readline.question("What is your name?\n", name => {
    console.log(`\n${greetings[Math.floor(Math.random() * greetings.length
    )]} ${name}!`);
    readline.close();
});

/*
OUTPUT:

node TerminalGreeting.js
What is your name?
Vishakan

Hi Vishakan!
*/
```

Code - Browser Greetings:

```javascript
let fs = require("fs");
let http = require("http");
let url = require("url");

let greetings = [];

//Non-blocking file IO
fs.readFile("greetings.txt", (error, data) => {
    if (error) {
        console.error(error);
    } else {
        greetings = data.toString().split("\n");
    }
});

//If there is a browser request, serve the request with response
//Listen on port 8080

http
    .createServer((req, res) => {
        let requestQuery = url.parse(req.url, true).query;
        let userName = requestQuery.name;

        res.writeHead(200, { 'Content-Type': 'text/html' });

        if (!userName) {
            res.write(`<h1>Enter your name in the URL as /?name=Mike!</h1>`);
        } else {
            res.write(`<h1>${greetings[Math.floor(Math.random() * greetings.length)]} ${userName}!</h1>`);
        }

        res.end();
    })
    .listen(8080);
```

Code - Greetings.txt:

```
Hello
Hey
Hi
What's up
Welcome
```

Code - Books.js:

```
let fs = require("fs");
let http = require("http");

//If there is a browser request, serve the request with html page
//Listen on port defined under PORT
//Use non-blocking file I/O

const PORT = 8080;

http
    .createServer((req, res) => {
        if (req.url === "/books") {
            fs.readFile("booklist.html", function (error, content) {
                if (error) {
                    res.writeHead(404);
                    res.write("Content Not Found!");
                } else {
                    res.writeHead(200, { "Content-Type": "text/html" });
                    res.write(content);
                }
                res.end();
            });
        } else {
            res.writeHead(200, {"Content-Type" : "text/html"});
            res.write("<h1>Kindly navigate to /books to see your content!"
    );
        }
    })
    .listen(PORT);

console.log("Server listening on port " + PORT);
```

Code - booklist.html:

```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Book Category</title>
    <style>
        body {
            background-color: plum;
            font-family: 'Times New Roman', Times, serif;
        }

        h1 {
            text-align: center;
            color: blueviolet;
            text-decoration: underline;
        }

        table {
            width: 75%;
        }

        table,
        th,
        td {
            border: 2px solid black;
            border-collapse: collapse;
            font-size: 20px;
            padding: 5px;
            text-align: center;
        }

        th {
            font-size: 25px;
        }

        tr:hover {
            background-color: aliceblue;
            color: red;
            font-weight: 250;
        }

        footer {
            position: fixed;
            bottom: 0;
            left: 0;
            border-top: 2px solid black;
```

```
            width: 100%;
            text-align: center;
            font-size: 25px;
        }
    </style>
</head>

<body>
    <h1>BOOKS LIST</h1>
    <hr><br>

    <table align="center">
        <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Genre</th>
            <th>Publisher</th>
        </tr>
        <tr>
            <td>Harry Potter</td>
            <td>J. K. Rowling</td>
            <td>Fantasy</td>
            <td>Bloomsbury</td>
        </tr>
        <tr>
            <td>Percy Jackson</td>
            <td>Rick Riordan</td>
            <td>Fiction</td>
            <td>Disney-Hyperion</td>
        </tr>
        <tr>
            <td>Hunger Games</td>
            <td>Suzanne Collins</td>
            <td>tdriller</td>
            <td>Scholastic</td>
        </tr>
        <tr>
            <td>Divergent</td>
            <td>Veronica Rotd</td>
            <td>Young Adult</td>
            <td>Harper Collins</td>
        </tr>
        <tr>
            <td>Five Kingdoms</td>
            <td>Brandon Mull</td>
            <td>Adventure</td>
            <td>Brandon Mull</td>
        </tr>
    </table>

    <footer>
```

```
        &copy; Page Served from Vishakan's NodeJS Server
    </footer>
</body>

</html>
```

Code - Mongo.js:

```
//Schema: Name, age, ID, gender, address, marital status, Date of Visit

/*
Instructions in Mongo Shell:

Switching to mydb:

use mydb;

Inserting a record:

db.patients.insert({name: "John Doe", age: 45, _id: "P1", gender: "Male",
    address: "177A, Bleecker Street, London, UK", mstatus: "Married", dov:
    new Date(2019, 07, 20)});

Finding a record:

db.patients.find({name: "John Doe"});

*/

/* In this code, we generate a patient with _id: "P2" and perform CRUD on
    it */

/* Dependencies: npm install mongodb */

const mongoClient = require("mongodb").MongoClient;
const url = "mongodb://localhost:27017/mydb";


async function connectToDatabase() {
    //Connecting to the database

    let client;

    try {
        client = await mongoClient.connect(url);
        console.log("\nConnected to database.");
        await client.close();


    } catch (error) {
        console.error(error);
    }
};


```

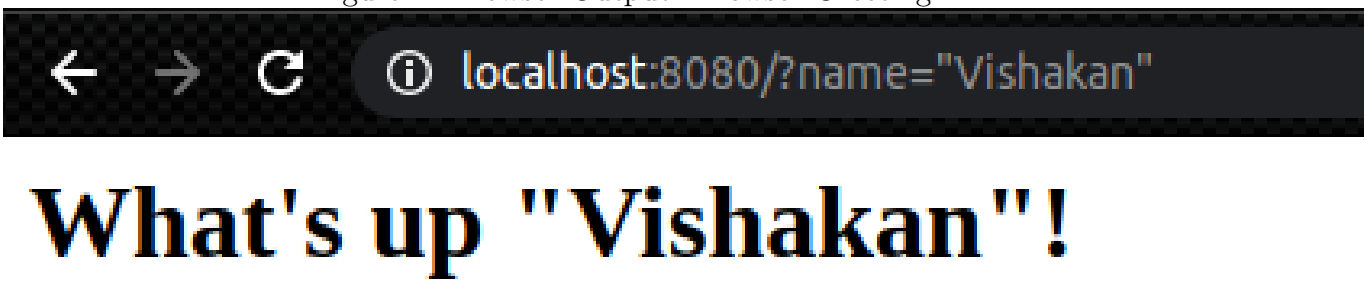
```
async function insertDocument() {
    //Inserting a record to the collection

    let client;

    try {
        client = await mongoClient.connect(url);
        const dbo = client.db("mydb");

        let record = {
            name: "Amy Santiago",
            age: 38,
            _id: "P2",
            gender: "Female",
            address: "275, Sunset Blvd, Brooklyn, NY",
            mstatus: "Married",
            dov: new Date(2020, 12, 27)
        };

        let result = await dbo.collection('patients').insertOne(record);
        console.log("\nDocument inserted successfully.");
        await client.close();

    } catch (error) {
        console.error(error);
    }
};


async function findDocument() {
    //Finding a record from the collection

    let client;

    try {
        client = await mongoClient.connect(url);
        const dbo = client.db("mydb");

        let result = await dbo.collection('patients').findOne({ _id: "P2"
    });

        if (result) {
            console.log("\nDocument found:");
            console.log(result);
        } else {    //document does not exist
            console.log("\nDocument not found.");
        }

        await client.close();

    } catch (error) {
```

```
        console.error(error);
    }
};


async function updateDocument() {
    //Updating a record in the collection

    let client;

    try {
        client = await mongoClient.connect(url);
        const dbo = client.db("mydb");
        let query = { _id: "P2" };
        let newValues = {   //$set only changes the specific fields
            $set: {
                age: 40,
                address: "99, Bush Avenue, Brooklyn, NY"
            }
        };

        let res = await dbo.collection("patients").updateOne(query, newValues);
        console.log("\nDocument updated successfully.");
        await client.close();

    } catch (error) {
        console.error(error);
    }
};


async function deleteDocument() {
    //Deleting a record in the collection

    let client;

    try {
        client = await mongoClient.connect(url);
        const dbo = client.db("mydb");
        let query = {
            _id: "P2"
        };

        let result = await dbo.collection("patients").deleteOne(query);
        console.log("\nDocument deleted successfully.");
        await client.close();

    } catch (error) {
        console.error(error);
    }
```

```
};


//Driver code - Use an IIFE (Immediately Invoked Function Expression)
//To run the async events in a synchronous manner, use async-await logic

(async function driverCode() {
    await connectToDatabase();  //test if connectivity is fine
    await insertDocument();     //insert a new document
    await findDocument();       //find the inserted document
    await updateDocument();     //update the inserted document
    await findDocument();       //find the updated document
    await deleteDocument();     //delete the updated document
    await findDocument();       //find the deleted document
})();

/*
OUTPUT: node --no-warnings Mongo.js

Connected to database.

Document inserted successfully.

Document found:
{
  _id: 'P2',
  name: 'Amy Santiago',
  age: 38,
  gender: 'Female',
  address: '275, Sunset Blvd, Brooklyn, NY',
  mstatus: 'Married',
  dov: 2021-01-26T18:30:00.000Z
}

Document updated successfully.

Document found:
{
  _id: 'P2',
  name: 'Amy Santiago',
  age: 40,
  gender: 'Female',
  address: '99, Bush Avenue, Brooklyn, NY',
  mstatus: 'Married',
  dov: 2021-01-26T18:30:00.000Z
}

Document deleted successfully.

Document not found.
*/
```

Output - Browser Greeting:

Figure 1: Browser Output: Browser Greeting.



Output - Books Table:

Figure 2: Browser Output: Books Table.

BOOKS LIST

Title	Author	Genre	Publisher
Harry Potter	J. K. Rowling	Fantasy	Bloomsbury
Percy Jackson	Rick Riordan	Fiction	Disney-Hyperion
Hunger Games	Suzanne Collins	tdriller	Scholastic
Divergent	Veronica Rotd	Young Adult	Harper Collins
Five Kingdoms	Brandon Mull	Adventure	Brandon Mull

© Page Served from Vishakan's NodeJS Server

Output - Mongo Shell:

Figure 3: Browser Output: Mongo Shell.

```
mongo
→ mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("31ca0282-b4f4-44b8-b4f0-769f6a668e42") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
        2021-05-02T11:59:48.914+05:30: Using the XFS filesystem is strongly recommended with the WiredTiger s
torage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2021-05-02T11:59:53.666+05:30: Access control is not enabled for the database. Read and write access
to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> use mydb;
switched to db mydb
> db.patients.insert({name: "John Doe", age: 45, _id: "P1", gender: "Male", address: "177A, Bleecker Street,
London, UK", mstatus: "Married", dov: new Date(2019, 07, 20)});
WriteResult({ "nInserted" : 1 })
> db.patients.find({_id: "P1"})
{ "_id" : "P1", "name" : "John Doe", "age" : 45, "gender" : "Male", "address" : "177A, Bleecker Street, Londo
n, UK", "mstatus" : "Married", "dov" : ISODate("2019-08-19T18:30:00Z") }
>
```

Learning Outcome:

- From the experiment, I learnt about NodeJS' event-driven architecture.
- I learnt to serve HTML content using NodeJS as a server-side application.
- I learnt to use basic functions available in the **fs** & **http** modules in NodeJS.
- I learnt to read URL queries and respond to a query with HTML content with **writeHead() & write()** methods to serve HTML response.
- I learnt how to serve a static HTML page using NodeJS to the browser.
- I installed and learnt to use **MongoDB** from the shell.
- I learnt to manipulate a MongoDB database from JavaScript using NodeJS' **mongodb** module.
- I understood that these method calls defined by NodeJS for the mongodb moudle are asynchronous, meaning that these operations need not occur one after the other sequentially.
- Thus, I learnt to implement an **async-await** methodology to execute the CRUD operations one by one from NodeJS.
- I was able to successfully perform **CRUD operations** in MongoDB using NodeJS.
- I used an **IIFE**(Immediately Invoked Function Expression) to execute the CRUD driver code.
- I learnt about MongoDB's syntax and methods like insertOne(), insertMany(), findOne(), deleteOne() and updateOne() to perform CRUD operations.
- I understood about collections in MongoDB & the fact that MongoDB is a **NoSQL** database.