Node JS – DM

(git repository: github.com/deep-mm/Node-JS)

1. Make an **new folder** and open terminal in it

2. Run in terminal: *npm install --save express body-parser* (with sudo if using mac)

3. Make a **file**: *server.js* in the root project folder

4. Make a **folder**: *server* -> Inside server: Make a **folder**: *routes*
   |-> Inside routes make 2 **files**:
   - *api.js*
   - *index.js*

5. Make a **folder**: *public* -> Inside this create a **file**: *index.html*

6. Contents of *server.js:*

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const api = require('./server/routes/api');
const index = require('./server/routes/index');
const port = 3000;

const app = express();

app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());
app.use('/api', api);
app.use('/',index);

app.listen(port, function(){
    console.log("Server running on localhost:" + port);
});
```

7. Contents of *api.js & index.js* initially:

```
const express = require('express');
const router = express.Router();
const path = require('path');

router.get('/',function(req,res){
    res.send('Module Works');
});

module.exports = router;
```

8. Handling 404 Errors: At the end of each JS file put this

```
router.get('*', function(req,res) {
    res.send('Error 404, Page not found');
});
```

Also can use this,

```
app.use(function(err, req, res, next) {
    res.status(404);
    res.send("Oops, something went wrong. This is error 404. Please try again
later.")
 });
```

9. Load html file for a route:

```
router.get('/(any-path)',function(req,res){
    res.sendFile(path.join(__dirname+'public/(any-file).html'));
});
```

10. Use of **JSON API**: In api.js write this:

```
router.get('/getObjects', function(req,res){
    object = {name: 'Deep', age: '22', mobile: '9999999999'};
    res.json(object);
});
```

11. Cookie-Parser

Run this in the terminal: *npm install cookie-parser* (sudo if using mac)

Add in **server.js:**

```
   var cookieParser = require('cookie-parser')
   app.use(cookieParser())
```

Now in **index.js**:

To **store** cookies:

```
router.get('/settingCookie', function(req,res){
    res.cookie('name', 'express').send('cookie set');
})
```

To **get** cookies:

```
router.get('/checkingCookies', function(req,res){\
    res.send(req.cookies);
})
```

12. File Operations

```
var fs = require('fs');
var urlToCheck = '/Users/aa';
```

a. Check if path is valid or not and we have permission to use it

```
fs.access(urlToCheck, fs.constants.X_OK, (err) => {
    if (err) {
        console.log("%s doesn't exist", urlToCheck);
    } else {
        console.log('can execute %s', urlToCheck);
    }
});
```

```
fs.constants.R_OK | fs.constants.W_OK

Use these to write different functions to find read/write permission
```

b. Read & count lines in a text file

Create a **text file in public folder** with any sample content

In **index.js** file:

```
const readline = require('readline');
var urlToCheck = '/Users/deepmehta/Deep/Projects/GITHUB/Web Development/Node
JS/public/textFile.txt';

var str = '';
var linescount = 0;
var lineReader = readline.createInterface({
    input: fs.createReadStream(urlToCheck)
});

lineReader.on('line', function (line) {
    linescount++;
    console.log('Line from file:', line);
    str = str + line;
    str = str + '\n';
});

router.get('/',function(req,res){
    res.sendFile(path.join(url+'public/index.html'));
});
```

13. Building **own module** to authenticate email:

In server/routes create a new **file**: *authentication.js*

In **authentication.js** write this:

```
function authenticate(email){
    if(email.length>10){
        return true;
    }
    else{
        return false;
    }
}

module.exports.authenticate = authenticate;
```

Now, in **index.js:**

```
var logger = require('./authentication');
router.get('/authenticate/:email',function(req,res){
    var email = req.params.email;
    if(logger.authenticate(email)){
        res.send('Authentication Success');
    }
    else{
        res.send('Authentication Failed');
    }
})
```

14. Take **input** from user using **CMD**

In **server.js**

```
var standard_input = process.stdin;
standard_input.setEncoding('utf-8');
standard_input.on('data', function (data) {

    if(data === 'exit\n'){
        console.log("User input complete, program exit.");
        process.exit();
    }else
    {
        console.log('User Input Data : ' + data);
    }
});
```

15. Using **multer** to upload files

Run the following command in **terminal**: *npm install --save multer*

In **index.js**

```
const multer = require('multer');
var storage = multer.diskStorage({
    destination: function (req, file, cb) {
      cb(null, 'uploads')
    },
    filename: function (req, file, cb) {
      cb(null, file.fieldname + '-' + Date.now())
    }
  });

var upload = multer({ storage: storage });
```

```
router.post('/uploadfile', upload.single('myFile'), (req, res, next) => {
    const file = req.file
    if (!file) {
      const error = new Error('Please upload a file')
      error.httpStatusCode = 400
      return next(error)
    }
      res.send(file)

});

router.post('/uploadmultiple', upload.array('myFiles', 12), (req, res, next) =>
{
    const files = req.files
    if (!files) {
      const error = new Error('Please choose files')
      error.httpStatusCode = 400
      return next(error)
    }

      res.send(files)

});
```

In **index.html**

```
    <h2>All Type of files</h2>
        <form action="/uploadfile" enctype="multipart/form-data" method="POST">
            <input type="file" name="myFile" />
            <input type="submit" value="Upload a file"/>
         </form>

         <form action="/uploadmultiple"  enctype="multipart/form-data"
method="POST">
           Select images: <input type="file" name="myFiles" multiple>
           <input type="submit" value="Upload your files"/>
         </form>

    <h2>Only image files</h2>
         <form action="/uploadfile" enctype="multipart/form-data"
method="POST">
            <input type="file" name="myFile" accept="image/*"/>
            <input type="submit" value="Upload a file"/>
         </form>
```

16. **Reverse proxy** using express JS

Run the following in **terminal**: *npm install --save http-proxy*

Create 2 **files** in root folder: *proxy-server1.js* & *proxy-server2.js*

In both files: (change the port to 3002 for proxy-server2)

```
const express = require('express');
const bodyParser = require('body-parser');
const port = 3001;

const app = express();

app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());

app.listen(port, function(){
    console.log("Server running on localhost:" + port);
});
```

In **server.js**

```
var httpProxy = require('http-proxy');
var apiProxy = httpProxy.createProxyServer();
```

```
var serverOne = 'http://localhost:3001',
    ServerTwo = 'http://localhost:3002';

app.all("/app1/*", function(req, res) {
    console.log('redirecting to Server1');
    apiProxy.web(req, res, {target: serverOne});
});

app.all("/app2/*", function(req, res) {
    console.log('redirecting to Server2');
    apiProxy.web(req, res, {target: ServerTwo});
});
```

Now run all 3 servers at same time, and run localhost/app1 and see the console output

17. **TLS**: demonstrate the use of Server and client application using node.js

Make a **new file** in the root folder: *tls.js*

Run following code in your terminal: *npm install socket.io*

Add following to **tls.js**:

```
const express = require('express');
const app = express();
const port = 4000;
const server = app.listen(port,console.log("Socket.io Hello World server
started!"));
const io = require('socket.io')(server);

io.on('connection', (socket) => {
    socket.on('message-from-client-to-server', (msg) => {
        console.log(msg);
    })
    socket.emit('message-from-server-to-client', 'From Server');
});
```

Make a **new file** *client.html* in public directory:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Hello World with Socket.io</title>
  </head>
  <body>
    <script src="https://cdn.socket.io/socket.io-1.4.5.js"></script>
    <script>
      var socket = io("http://localhost:4000");
      socket.on("message-from-server-to-client", function(msg) {
          document.getElementById('message').innerHTML = msg;
      });
      socket.emit('message-from-client-to-server', 'From Client');
    </script>
    <p>Socket.io Hello World client started!</p>
    <p id="message"></p>
  </body>
</html>
```

Add this to **index.js** file:

```
router.get('/client',function(req,res){
    res.sendFile(path.join(url+'public/client.html'));
});
```

18. Create the worker **child process** in the main process that handles the load across multiple cores.

Create a **two new files** in root: *master.js*
slave.js

In **master.js**:

```
const fs = require('fs');
const child_process = require('child_process');

for(var i = 0; i < 3; i++) {
var workerProcess = child_process.exec('node slave.js ' + i, function(error,
stdout, stderr) {
if(error) {
    console.log(error.stack);
}
console.log("stdout: " + stdout);
console.log("stderr: " + stderr);
});

workerProcess.on('exit', function(code) {
console.log("Child process exitted with exit code: " + code);
});
}
```

In **slave.js**:

```
console.log("Child process " + process.argv[2] + " says Hello!!");
```

Run master.js in **terminal**: *node master*

Connecting Mongo DB with Node JS:

1. Run in **terminal**: *npm install –save mongoose*
2. Run in **terminal**: *mongod* //To start mongoDB server
3. Create a new **folder** inside server: *models*
4. Inside models create a new **file**: *student.js*

   Inside **student.js**:

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const studentSchema = new Schema({
    name: String,
    age: String,
    college: String
});

module.exports = mongoose.model('student', studentSchema, 'students');
```

5. In **api.js** write these statements:

```
const url = "mongodb://localhost:27017/students"
mongoose.Promise = global.Promise;

mongoose.connect(url,function(err,db){
    if(err){
        console.log("Error "+err);
    }
    else{
        console.log("Connected on url "+url);
    }
});
```

6. **CRUD** Operations to perform on database

   a. **Create:**

```
router.post('/student', function(req,res){
    console.log("Posting a document");
    var newStudent = new Student();
    newStudent.name = req.body.name;
    newStudent.age = req.body.age;
    newStudent.college = req.body.college;
    console.log(req.body.name);
    newStudent.save(function(err, insertedDoc){
        if(err){
            console.log("Error "+err);
        }
        else{
            res.json(insertedDoc);
        }
    })
});
```

   b. **Read:**

```
router.get('/students',function(req,res){
    console.log('Get request for all students');
    Student.find({})
    .exec(function(err, students){
        if(err){
            console.log("Error "+err);
        }
        else{
            res.json(students);
        }
    });
});

router.get('/student/:id',function(req,res){
    console.log('Get request for a single student');
    const id = req.params.id;
    Student.findById(id)
    .exec(function(err, student){
        if(err){
            console.log("Error "+err);
        }
        else{
            res.json(student);
        }
    });
});
```

c. Update

```
router.put('/students/:id',function(req,res){
    console.log('Update a student');
    const id = req.params.id;
    Student.findByIdAndUpdate(id, {
        $set: {name:req.body.name, age: req.body.age, college:
req.body.description}
    },
    {
        new: true
    },
    function(err,updateDoc){
        if(err){
            console.log("Error "+err);
        }
        else{
            res.json(updateDoc);
        }
    })
});
```

d. Delete

```
router.delete('/student/:id',function(req,res){
    console.log('Delete a single student');
    const id = req.params.id;
    Student.findByIdAndDelete(id, function(err,deletedDoc){
        if(err){
            console.log("Error "+err);
        }
        else{
            res.json(deletedDoc);
        }
    })
});
```

Hosting Node JS project to firebase:

1. Run in **terminal**: *npm install -g firebase-tools*

2. Run in **terminal**: *firebase login*
   |-> Enter your firebase login credentials

3. Run in terminal: *firebase init*
   | -> Select Functions & Hosting
   | -> Select your project
   | -> Select language as JavaScript
   | -> Yes for ESLint & dependencies
   | -> Public directory = public
   | -> Single page app = No
   | -> Overwrite index.html & 404.html = No

4. Put all data of **server.js** in **functions/index.js**

5. Integrate **package.json** present outside with **functions/package.json**

6. Run in **terminal**: *firebase deploy*

7. Your site is hosted and URL is available