Node notes

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Node fundamental modules:

- File system module(fs) : allows access to the system files
  https://nodejs.org/dist/latest-v11.x/docs/api/fs.html#fs_fs_appendfile_path_data_options_callback
- Validator : npm module : validation of input
  https://www.npmjs.com/package/validator
- Nodemon : install global , it restarts automatically after saving changes.
- Debugger: add debugger to your code wherever you want to debug and run node inspect yourcode.js
- Request : make http requests. : api services : darksky.net for weather and mapbox for geolocation
- Express : creating servers
- Path : path.join(paths…) exp below
- Handlebars : templating service (npm I hbs)
- Heroku : install Heroku-cli
- Mongoose
- Validator
- Bcryptjs: to hash password
- Jsonwebtoken : use login tokens
- Npm multer : file upload
- Sharp : modify pictures
- Npm env-cmd : environment variables

To get arguments form user :

```
console.log(process.argv)
```

Using Yargs to process user input:

```
yargs.command({
    command: 'add',
    description: 'add a note',
    builder: {
        title: {
            describe: 'title of note',
            demandOption:true, //required argument
            type:'string'
        },
        body: {
            describe:'note body',
            demandOption:true,
        }
    },
    handler: function (argv) { //function that handles this argument
        console.log('adding note with title: ',argv.title,' and body : ',argv.body);
    }
})
```

```
How to use filter to scan an array or json file
var newnotes=oldnotes.filter(n => n.title !== title);
```

HTTP Request

```
var username = 'username',

    password = 'password',

    url = 'http://' + username + ':' + password + '@some.server.com';

request({url: url}, function (error, response, body) {

   // Do more stuff with 'body' here

});
```

Callbacks :

```
const geocode = (address, callback) => {
    setTimeout(()=>{
        const data = {
            lat: 0,
            long: 0
        }
        callback(data); // call the callback function and pass data
    },2000)
}
//call geocode function and use arrow function to replace the callback
geocode('calgary',(data)=>{
console.log(data);
})
```

Promises :

```
const doWorkpromise = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve([1, 4, 7]);
        // reject('error msg');
    }, 2000)
})
doWorkpromise.then((result) => {
    console.log('success', result);
}).catch((error) => {
    console.log('error', error);
})
```

Promises chaining

```
user.updateOne({ _id: '5c86c7a365ad9d3f0435c85d' },{age:10}).then((res) => {
    console.log(res);
    return user.count({age:0}).then((count)=>{
        console.log(count);
    })
}).catch((e)=>{
    console.log(e);})
```

async await :async always returns a promise

```
const doWork=async()=>{
const sum= await add(1,50);
const sum2=await add(sum,5);
return sum2
}
```

Creating a server using Express.js: npm install express –save

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

var app = express();
app.use(express.static(__dirname + '/public'));
app.get('/', (req, res) => {
    res.send('<h1>Hi</h1>hello express!');

});
app.get('/aboutme', (req, res) => {
    res.send({
        name: 'hayouni',
        likes: ['biking', 'hiking']
    })
})
app.listen(3000, () => {
    console.log('server is up');
});
```

How to use path

```
const path=require ('path');
console.log(path.join(__dirname,'..','public'));
```

How to serve a static html file using express:

```
const publicPath=path.join(__dirname,'..','public');
app.use('/',express.static(publicPath));
```

how to use handlebars to serve dynamic content

```
// set up handlebars
app.set('view engine', 'hbs');
app.set('views', viewsPAth);
app.get('/about',(req,res)=>{
    res.render('about',{
        name:'hayouni',
        job:'software eng'
    })
})
```
How to use partials :

```
const partialsPath=path.join(__dirname,'../templates/partials')
hbs.registerPartials(partialsPath);
```
how to allow nodemon to restart when hbs is changed

```
C:\Users\Me\Desktop\Node.js\web-server\src>nodemon node app.js -e js,hbs
```
How to read input from the url :

```
app.get('/weather', (req, res) => {
if (!req.query.location) {
        return res.send('No Location was found');
    }
…//OR if no ? is used in url
app.get('/users/:id', (req, res) => {
    console.log(req.params);
})
```

Git hub essential commands:

```
Git init
Git add .
Git commit -m “first”
Git status
Git remote add origin https//…   // only first time
Git push -f origin master        // only first time
Git push origin master  // every other time
Git pull origin master  //pull changes
```
SSH key for safe connect to github and heroku

```
ssh-keygen -t rsa -b 4096 -C "hayouni.com"  // to generate key
$ eval $(ssh-agent -s)   // to check agent pid number
$ ssh-add ~/.ssh/id_rsa  // add the key
cat ~/.ssh/id_rsa.pub   // to see the ssh code and add to the gihust server
$ ssh -T git@github.com  // to check if you are authenticated by github
```
How to use Heroku : start a new terminal and type the following

```
Heroku -v
Heroku login
Heroku keys:add //to add ssh key
Heroku create nameOfYourApplication
Then go to package.json , change in “scripts” to “start”:”node src/app.js”
In app.js  const port = process.env.PORT||3000;  // to add heroku port

Git push Heroku master
```

Mongodb

How to start mongodb server from terminal

```
mongodb\bin\mongod --dbpath=\Users\Me\Desktop\Node.js\mongodb-data
```

or C:\Program Files\MongoDB\Server\4.0\bin>mongod.exe --dbpath /users/Me/Documents/mongo-data and then in another cmd : mongo.exe

use Robo 3T as mongodb Gui

```
mongodb\bin\mongod --dbpath=\Users\Me\Desktop\Node.js\mongodb-data
or use R MongoClient.connect(url, (err, client) => {

        if (err) {
            return console.log('unable to connect to db ');
        }
        console.log("connected successfully to server");
        const db = client.db(dbName);
        db.collection('tasks').insertMany(tasks, (err, result) => {
            if (err) {
                return console.log('unable to add tasks');
            }
            console.log(result.ops);
        })
        client.close();
    })
```

Mongoose:

```

const dbName = 'task-manager-api';
const url = 'mongodb://localhost:27017/' + dbName;

mongoose.connect(url, { useNewUrlParser: true, useCreateIndex: true }, (err, client) => {
    if (err) {
        return console.log('unable to connect');
    }
    console.log('connected successfully to database using mongoose');

})

const Task = mongoose.model('Task', {
    description: { type: String },
    completed: { type: Boolean }
});

const me = new User({ name: 'hayouni', age: 25 });
const task=new Task({description:'clean the house',completed:false});
task.save().then((res) => {
    console.log(res);
}).catch((err) => {
    console.log(err);
})
```

How to validate mongoose model:

```
const User = mongoose.model('User', {
    name: {
        type: String,
        trim: true
    },
    email: {
        type: String,
        required: true,
        validate(email) {
            if (!validator.isEmail(email)) {
                throw new Error('unvalid email');
            }
        }
    },
    age: {
        type: Number,
        default: 0,
        validate(value) {
            if (value < 0) {
                throw new Error('age must be positive number');
            }
        }
    },
    password: {
        required: true,
        type: String,
        validate(value) {
            if (!validator.isLength(value, [{ min: 6 }])) {
                throw new Error('password must be greater than 6 characters');
            }
            if (validator.isIn(value.toLowerCase(), ['password'])) {
                throw new Error('password, cant be a password');
            }
        }
    }
})
```

Express Middleware

```
const auth= async(req,res,next)=>{
    console.log('auth middleware');
    next();
}
module.exports=auth
```
how to use the middleware before routing :

```
//get all users
const auth = require('../middleware/auth')

router.get('/users',auth,(req, res) => {
    user.find({}).then((users) => {
        res.send(users);
    }).catch((err) => {
        res.status(400);
        res.send(err);
    })
})
```

How to upload picture//

The client side

```
  // Upload picture
        const file = new FormData()
        const filedata = document.querySelector('#Article-Picture').files[0]
        file.append('ArticlePicture', filedata)
        console.log(file)

        fetch('/uploadPic', {
                method: "POST", // *GET, POST, PUT, DELETE, etc.

                headers: {
                    'Accept': 'application/json'
                    // "Content-Type": "application/x-www-form-urlencoded",
                },
                referrer: "no-referrer", // no-referrer, *client
                body: file // body data type must match "Content-Type" header
            }).then((res) => {
                console.log(res)
                return res.json()
            })
```
Server side

```
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, './public/ArticlesImages/')
    },
    filename: function (req, file, cb) {
        cb(null, Date.now() + file.originalname)
    },
    fileFilter(req, file, cb) {
        if (!file.originalname.match(/\.(jpg|jpeg|png)$/)) {
            return cb(new Error('Please uplaod and image or video'))
        }
        cb(undefined, true)
    }
})
const upload = multer({
    storage: storage
})
app.post('/uploadPic', upload.single('ArticlePicture'), (req, res) => {
    console.log(req.body)
    try {
        console.log(req.file)
        res.send({
            failure: false,
            message: ' uploaded',
            filename: req.file.originalname,
            filepath: req.file.path,
            file: req.file
        })
    } catch (E) {
        console.log(E)
        res.send({
            failure: true,
            message: 'Failed to upload pic'
        })
    }
})
```

Ignore git

Create .gitignore and includes files u want to ignore inside

```
node_modules/
```

generate ssh key

ssh-keygen -t rsa -b 4096 -C “email address” then ssh-add ~/.ssh/id_rsa

To add the ssh to github use the following command and copy it from cmd to github:

```
$ cat ~/.ssh/id_rsa.pub
```

To deploy app to Heroku u must add start to scripts at package.json :

```
"scripts": {
    "start":"node app.js",
    "test": "echo \"Error: no test specified\" && exit 1",
    "ex": "node src/app.js",
    "dev": "nodemon src/app.js"
  }
```

And change port to :

```
const port=process.env.PORT||3000
app.listen(port, () => {
    console.log('Server is up on port 3000.')
})
```

Define Environment variable

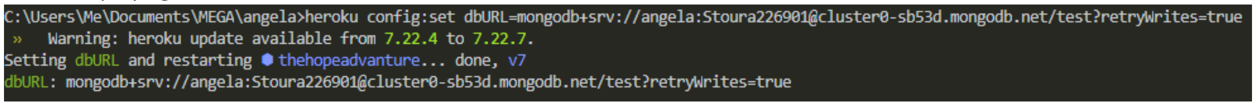
- Create config file and add it to .gitignore
- Install npm env-cmd to populate variables
- Update the dev script as following :

```
"dev": "env-cmd ./config/dev.env nodemon src/app.js "
```

```
const dbURL=process.env.dbURL
```

production database (deploy app with database)

- Create account at mongodb atlas
- Choose free tier option and create cluster
- Wait until the cluster is created and click connect
- Add IP address(0.0.0.0/0) and create db user (username and pw)
- Connect to mongodb compass `mongodb+srv://username:<password>@cluster0-sb53d.mongodb.net/admin`
- Before deploying to Heroku set environment variable :



```
C:\Users\Me\Documents\MEGA\angela>heroku config:set dbURL=mongodb+srv://angela:Stoura226901@cluster0-sb53d.mongodb.net/test?retryWrites=true
 »   Warning: heroku update available from 7.22.4 to 7.22.7.
Setting dbURL and restarting ⬢ thehopeadvanture... done, v7
dbURL: mongodb+srv://angela:Stoura226901@cluster0-sb53d.mongodb.net/test?retryWrites=true
```

Encrypt password (use bcrypt to hash your password)

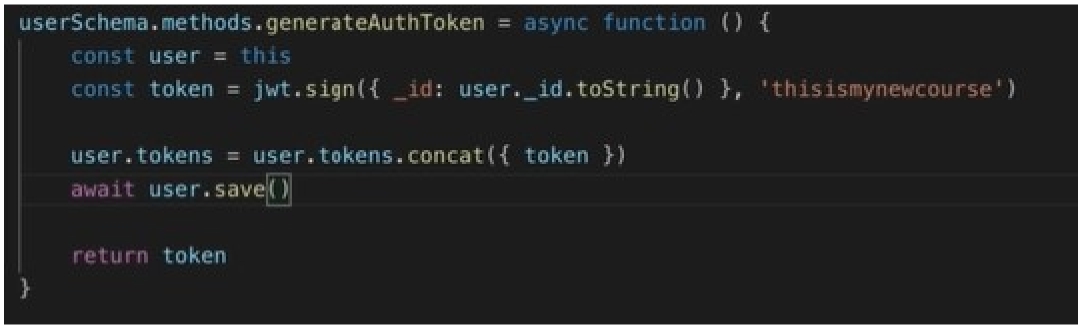
https://www.npmjs.com/package/bcrypt

Mongoose middleware

```
var schema = new Schema(..);
userschema.pre('save', function(next) {
const user =this
// do something with user like hash password
  next();
});
```

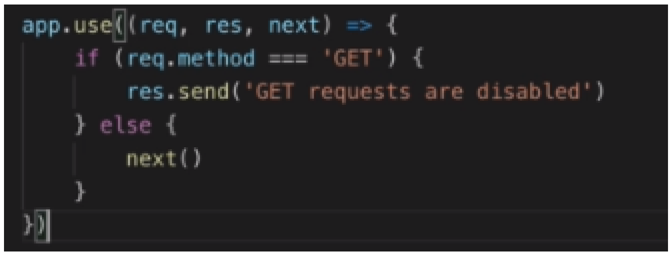
Authentiaction tokens use **jsonwebtoken npm**

add a method to generate authtokens in mongoose model(usermodel)



```
userSchema.methods.generateAuthToken = async function () {
    const user = this
    const token = jwt.sign({ _id: user._id.toString() }, 'thisismynewcourse')

    user.tokens = user.tokens.concat({ token })
    await user.save()

    return token
}
```

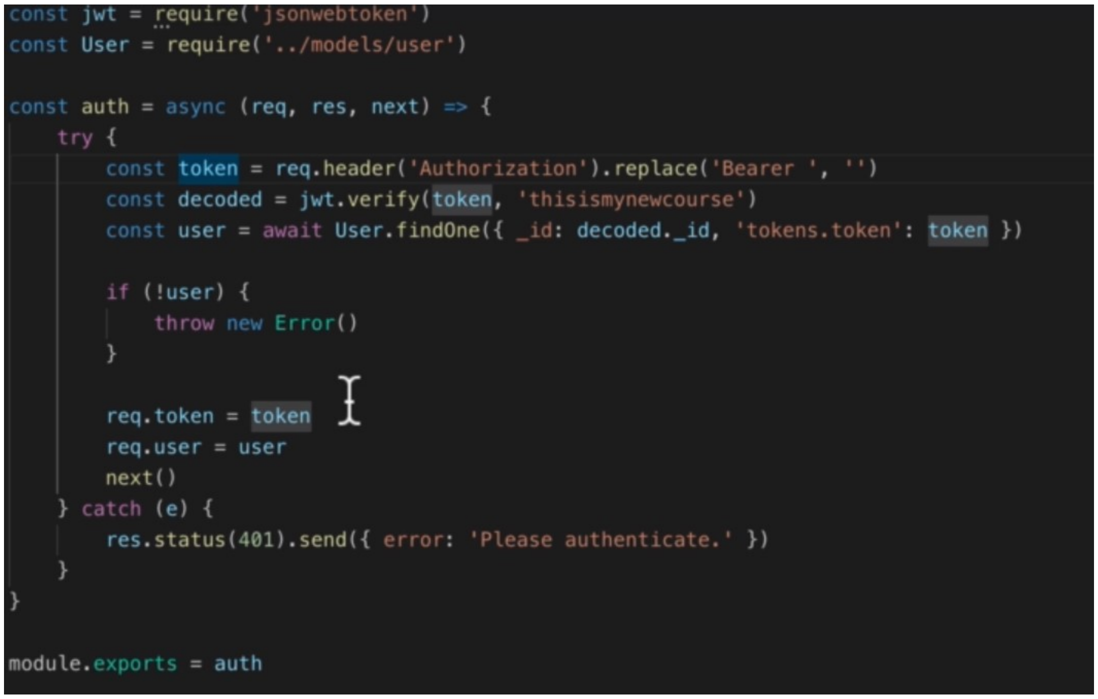
Express middleware



```
app.use((req, res, next) => {
    if (req.method === 'GET') {
        res.send('GET requests are disabled')
    } else {
        next()
    }
})
```

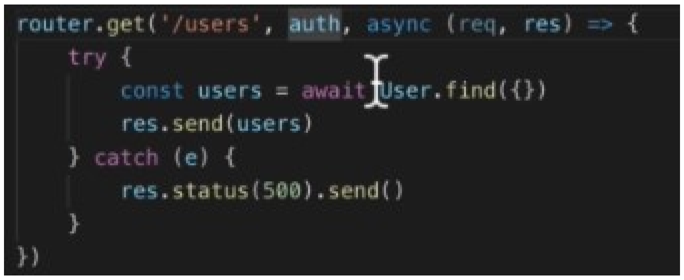
Use express created middleware to authenticate

- Create a folder called middleware and create the middleware file (auth.js)



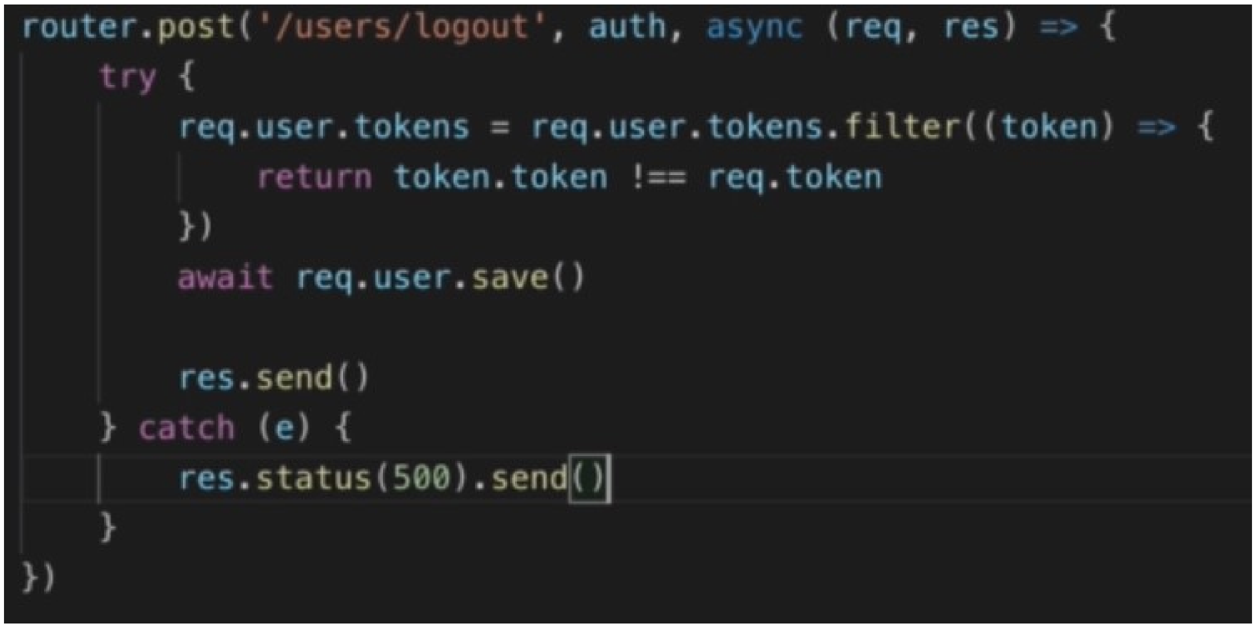
```
const jwt = require('jsonwebtoken')
const User = require('../models/user')

const auth = async (req, res, next) => {
    try {
        const token = req.header('Authorization').replace('Bearer ', '')
        const decoded = jwt.verify(token, 'thisismynewcourse')
        const user = await User.findOne({ _id: decoded._id, 'tokens.token': token })

        if (!user) {
            throw new Error()
        }

        req.token = token
        req.user = user
        next()
    } catch (e) {
        res.status(401).send({ error: 'Please authenticate.' })
    }
}

module.exports = auth
```

- Load the middleware using : auth=require('../middleware/auth')
- Use the middleware



```
router.get('/users', auth, async (req, res) => {
    try {
        const users = await User.find({})
        res.send(users)
    } catch (e) {
        res.status(500).send()
    }
})
```

How to log aout and destroy token:



```
router.post('/users/logout', auth, async (req, res) => {
    try {
        req.user.tokens = req.user.tokens.filter((token) => {
            return token.token !== req.token
        })
        await req.user.save()

        res.send()
    } catch (e) {
        res.status(500).send()
    }
})
```

How to render image from buffer

<img src="data:image/jpg;base64,the buffer of the image>