Node

Table of Contents

[Node 5](#_Toc27232782)

[Install node.js 5](#_Toc27232783)

[code editor 5](#_Toc27232784)

[Start 5](#_Toc27232785)

[Modules 5](#_Toc27232786)

[Export a module 5](#_Toc27232787)

[Load a module 5](#_Toc27232788)

[Docs 5](#_Toc27232789)

[Events module 6](#_Toc27232790)

[UserData.js 6](#_Toc27232791)

[View 6](#_Toc27232792)

[NPM – Node Package Manager 6](#_Toc27232793)

[package.json 6](#_Toc27232794)

[install package 6](#_Toc27232795)

[update packages 6](#_Toc27232796)

[uninstall package 6](#_Toc27232797)

[install dev package 6](#_Toc27232798)

[Use package 6](#_Toc27232799)

[list installed packages 7](#_Toc27232800)

[Rest 7](#_Toc27232801)

[Express 7](#_Toc27232802)

[Create 7](#_Toc27232803)

[Select all 7](#_Toc27232804)

[Select one 8](#_Toc27232805)

[Update 8](#_Toc27232806)

[Delete 8](#_Toc27232807)

[Starting the listener 8](#_Toc27232808)

[Express 8](#_Toc27232809)

[If body is JSON it will be parse into req.body 8](#_Toc27232810)

[If request has form it will be parse into req.body 8](#_Toc27232811)

[expose static files 8](#_Toc27232812)

[Custom middleware on global 9](#_Toc27232813)

[Custom middleware on route 9](#_Toc27232814)

[Environment 9](#_Toc27232815)

[setting the environemnt 9](#_Toc27232816)

[Configutation 9](#_Toc27232817)

[Debugging 9](#_Toc27232818)

[Templating Engines 9](#_Toc27232819)

[Pug 10](#_Toc27232820)

[viewName.pug 10](#_Toc27232821)

[Asynchronous JavaScript 10](#_Toc27232822)

[Callback 10](#_Toc27232823)

[Promises 10](#_Toc27232824)

[Multiple Promises 11](#_Toc27232825)

[Async / Await 11](#_Toc27232826)

[MongoDB 11](#_Toc27232827)

[Connection 11](#_Toc27232828)

[Schema 11](#_Toc27232829)

[Data Types 11](#_Toc27232830)

[Model 12](#_Toc27232831)

[Crud 12](#_Toc27232832)

[Save 12](#_Toc27232833)

[Select 12](#_Toc27232834)

[Comparison 12](#_Toc27232835)

[Scalar 12](#_Toc27232836)

[Update – query first 12](#_Toc27232837)

[Update – update first 13](#_Toc27232838)

[Remove 13](#_Toc27232839)

[Validation 13](#_Toc27232840)

[String 13](#_Toc27232841)

[Number 13](#_Toc27232842)

[Date 13](#_Toc27232843)

[Get & Set 13](#_Toc27232844)

[Based on other property 14](#_Toc27232845)

[Custom validation 14](#_Toc27232846)

[Validation with callback 14](#_Toc27232847)

[Modeling Relationships between Connected Data 14](#_Toc27232848)

[Referenced document 14](#_Toc27232849)

[Embedded documents 14](#_Toc27232850)

[Transactions 15](#_Toc27232851)

[Object ID 15](#_Toc27232852)

[Parse 15](#_Toc27232853)

[Validation 15](#_Toc27232854)

[joi validation 15](#_Toc27232855)

[Authentication and Authorization 15](#_Toc27232856)

[Password validation 15](#_Toc27232857)

[Hash 16](#_Toc27232858)

[JSON Web Token 16](#_Toc27232859)

[Implementation 16](#_Toc27232860)

[./config/default.js 16](#_Toc27232861)

[Validate jwt in index.js 16](#_Toc27232862)

[Login router 16](#_Toc27232863)

[Authorization middleware 16](#_Toc27232864)

[Authentication middleware 17](#_Toc27232865)

[Errors 17](#_Toc27232866)

[Errors 17](#_Toc27232867)

[in index.js 17](#_Toc27232868)

[Winston 17](#_Toc27232869)

[Transports 18](#_Toc27232870)

[Unit Testing 18](#_Toc27232871)

[Tools 18](#_Toc27232872)

[Jest 18](#_Toc27232873)

[Test sample 19](#_Toc27232874)

[Test exception 19](#_Toc27232875)

# Node

## Install node.js

<https://nodejs.org>

## code editor

Visual Studio Code

<http://code.visualstudio.com>

## Start

node index.js // to start the app

nodemon index.js // to start the auto mon app

# Modules

## Export a module

module.exports.functionName = functionName;

or

module.exports = functionName;

## Load a module

require(“./moduleFileName”);

functionName(“”);

const moduleName = require(“./moduleFileName”);

moduleName.functionName(“”);

or

moduleName(“”);

# Docs

<https://nodejs.org/en/docs/>

const fs = require(“fs”);

const files = fs.readdirSync(“./”);

[“git”, …]

const files = fs.readdir(“./”, function(err, files) {

if (err) console.log(“Error”, err);

else console.log(files);

});

[“git”, …]

# Events module

## UserData.js

const EventEmmitter = require("events");

class UserData extends EventEmmitter {

setData() {

this.emit("onUserChanged", { id: 1, url: "http://..." });

}

}

module.exports = UserData;

## View

const UserData = require("./UserData");

const userData = new UserData();

userData.on("onUserChanged", eventArg => {

console.log("handleUserChanged", eventArg);

});

userData.setData();

# NPM – Node Package Manager

<https://www.npmjs.com>

## package.json

npm init [--yes]

## install package

npm i <packageName>

npm i // update all packages in package,json file

## update packages

npm update

## uninstall package

npm un <packageName>

## install dev package

npm i jshint –save-dev

## Use package

var \_ = require(“underscore”);

var result = \_.contains([1, 2, 3], 2);

## list installed packages

npm list

├─┬ mongoose@5.7.14

│ ├── bson@1.1.3

│ ├── kareem@2.3.1

│ ├─┬ mongodb@3.3.5

│ │ ├── bson@1.1.3 deduped

│ │ ├─┬ require\_optional@1.0.1

│ │ │ ├── resolve-from@2.0.0

│ │ │ └── semver@5.7.1

│ │ ├── safe-buffer@5.1.2 deduped

│ │ └─┬ saslprep@1.0.3

│ │ └─┬ sparse-bitfield@3.0.3

│ │ └── memory-pager@1.5.0

│ ├── mongoose-legacy-pluralize@1.0.2

│ ├── mpath@0.6.0

│ ├─┬ mquery@3.2.2

│ │ ├── bluebird@3.5.1

│ │ ├─┬ debug@3.1.0

│ │ │ └── ms@2.0.0

│ │ ├── regexp-clone@1.0.0 deduped

│ │ ├── safe-buffer@5.1.2 deduped

│ │ └── sliced@1.0.1 deduped

│ ├── ms@2.1.2

│ ├── regexp-clone@1.0.0

│ ├── safe-buffer@5.1.2

│ ├── sift@7.0.1

│ └── sliced@1.0.1

└── underscore@1.9.1

# Rest

## Express

const express = require(‘express’);

const app = express();

## Create

app.post(‘/api/courses’, (req, res) => {

// Create the course and return the course object

resn.send(course);

});

## Select all

app.get(‘/api/courses’, (req, res) => {

// To read query string parameters (?sortBy=name)

const sortBy = req.query.sortBy;

// Return the courses

res.send(courses);

});

## Select one

app.get(‘/api/courses/:id/:month’, (req, res) => {

const courseId = req.params.id;

const month = req.params.month;

// If not found, return 404

res.status(404).send(‘Course not found.’);

// Else, return the course object

res.send(course);

});

## Update

app.put(‘/api/courses/:id’, (req, res) => {

// If course not found, return 404, otherwise update it

// and return the updated object.

});

## Delete

app.delete(‘/api/courses/:id’, (req, res) => {

// If course not found, return 404, otherwise delete it

// and return the deleted object.

});

## Starting the listener

app.listen(3000, () => console.log(‘Listening…’));

const port = process.env.PORT || 3000;

app.listen(port);

# Express

## If body is JSON it will be parse into req.body

app.use(express.json());

## If request has form it will be parse into req.body

app.use(express.urlencoded());

## expose static files

add files under ./public

like ./public/readme.txt and it will be http://www.domain.com/readme.txt

app.use(express.static());

## Custom middleware on global

app.use(function(req, res, next)) {

// …

next();

}

## Custom middleware on route

app.use(‘/api/admin’, function(req, res, next)) {

// …

next();

}

# Environment

app.get(‘env’); // returns development, production, etc

## setting the environemnt

export NODE\_ENV=production

# Configutation

npm i config

in ./config/default.json // default config values

in ./config/development.json // override the default config values

in ./config/production.json // override the default config values

# Debugging

npm I debug

const debug = require(“debug”)(“debugLevel”);

debug(“log message”);

$export DEBUG=debugLevel // log only this level

$export DEBUG=\* // log all

$export DEBUG= // stopping logging

# Templating Engines

Pug

Mustache

EJS

## Pug

npm I pug

app.set(“view engine”, “pug”);

./views // all views go here

./views/index.pug

## viewName.pug

html

head

title= title

body

h1= message

in a route

res.render(“viewName”, { title: “My Title”, message: “My Message” });

# Asynchronous JavaScript

## Callback

console.log('Before');

getUser(1, (user) => {

console.log("User", user);

});

console.log('After');

function getUser(id, callback) {

setTimeout(() => {

console.log("Reading a user from a database...");

callback({ id: id, name: "userName" });

}, 2000);

}

output:

// Before

// After

// Reading a user from a database...

// { id: id, name: "userName" }

## Promises

const p = new Promise((resolve, reject) => {

setTimeout(() => {

// do somthing

resolve(1);

reject(new Error("Message"));

}, 2000);

});

p

.then(result => console.log(result))

.catch(err => console.log('Error', err.message));

## Multiple Promises

Promise.all([p1, p2])

.then(result => console.log(“execute when all promises are finished”))

.catch(err => console.log('Error', err.message));

Promise.race([p1, p2])

.then(result => console.log(“execute when first promises is finished”))

.catch(err => console.log('Error', err.message));

## Async / Await

await function displayCommits() {

try {

const user = await getUser(1);

console.log("done");

}

Catch (err) {

Console.log(“Error”, err.message);

}

}

# MongoDB

## Connection

const mongoose = require(‘mongoose’);

mongoose.connect(‘mongodb://localhost/playground')

.then(() => console.log(‘Connected…’))

.catch(err => console.error(‘Connection failed…’));

## Schema

const courseSchema = new mongoose.Schema({

name: String,

author: String,

tags: [ String ],

date: { type: Date, default: Date.now },

isPublished: { type: Boolean, default: false }

});

## Data Types

String, Number, Date, Buffer (for storing binary data), Boolean and ObjectID

## Model

const Course = mongoose.model(‘Course’, courseSchema);

const course = new Course( {} );

## Crud

### Save

course = await course.save();

### Select

const courses = await Course

.find({ author: ‘Mosh’, isPublished: true }) // where and

.and( [ { author: ‘Mosh’}, { isPublished: true } ] ) // where and (same as find)

.or( [ { author: ‘Mosh’}, { isPublished: true } ] ) // where or

.skip(10)

.limit(10)

.sort({ name: 1, price: -1 })

.select({ name: 1, price: 1 }); // what columns to return

### Comparison

eq == equal

ne != not equal

gt > greater than

gte >= greater than or equal

lt < less than

lte <= less than or equal to

in

nin not in

const courses = await Course

.find({ price: 10 }) // == 10

.find({ price: { $eq: 10 } }) // == 10

.find({ price: { $gte: 10, $lte: 20 } }) // <= 20 && >= 10

.find({ price: { $in: [10, 15, 20] } }) // in (10, 15, 20)

### Scalar

const courses = await Course

.find({ author: ‘Mosh’, isPublished: true }) // where and

.count()

### Update – query first

const course = await Course.findById(id);

if (!course) return;

course.set({ name: ‘…’ });

course.save();

### Update – update first

const result = await Course.update({ \_id: id }, { $set: { name: "…" } });

const result = await Course.findByIdAndUpdate({ \_id: id }, { $set: { name: "…" } }, { new: true });

### Remove

const count = await Course.deleteOne({ \_id: id });

const count = await Course.deleteMany({ \_id: id });

const course = await Course.findByIdAndRemove(id);

## Validation

new mongoose.Schema({

name: { type: String, required: true }

})

### String

name: {

type: String,

required: true,

unique: true,

match: /regex/, // validation by regex

enum: [“web”, “mobile”, “network”], // validation to enum

lowercase: true, // convert string to lowercase

uppercase: true, // convert string to uppercase

trim: true // trim padding from string

}

### Number

price: {

type: Number

required: true,

min: 10,

max: 100

}

### Date

birthday: {

type: Date

required: true,

min: 10,

max: 100

}

### Get & Set

price: {

type: String

get: v => Math.Round(v),

set: v => Math.Round(v)

}

### Based on other property

**Price is required if isPublished is true**

isPublished: Boolean,

price: {

type: Number,

required: function() { return this.isPublished; }

}

### Custom validation

tags: [

type: Array,

validate: {

validator: function(v) { return v && v.length > 0; },

message: ‘A course should have at least 1 tag.’

}

]

### Validation with callback

tags: [

type: Array,

validate: {

isAsync: true

validator: function(v, callback) {

// Do the validation, when the result is ready, call the callback

callback(isValid); // return a Boolean value isValid

}

}

]

## Modeling Relationships between Connected Data

### Referenced document

const courseSchema = new mongoose.Schema({

author: {

type: mongoose.Schema.Types.ObjectId,

ref: ‘Author’

}

})

### Embedded documents

const courseSchema = new mongoose.Schema({

author: {

type: new mongoose.Schema({

name: String,

bio: String

})

}

})

## Transactions

npm i fawn

const Fawn = require(“fawn”);

Fawn.init(mongoose);

try {

await new Fawn.Task()

.save(‘rentals’, newRental)

.update(‘movies’, { \_id: movie.\_id }, { $inc: numberInStock: -1 }})

.run();

}

catch (ex) {

// At this point, all operations are automatically rolled back

}

## Object ID

They consist of 12 bytes:

- 4 bytes: timestamp

- 3 bytes: machine identifier

- 2 bytes: process identifier

- 3 bytes: counter

### Parse

const id = new mongoose.Type.ObjectId();

id.getTimestamp();

### Validation

mongoose.Types.ObjectID.isValid(id);

### joi validation

const Joi = require(“joi”);

Joi.objectId = npm I joi-objectid(Joi);

customerId: Joi.objectId().required()

# Authentication and Authorization

## Password validation

npm i joy-password-complexity

## Hash

npm i bcrypt

const salt = await bcrypt.genSalt(10);

const hashed = await bcrypt.hash(‘1234’, salt);

const isValid = await bcrypt.compare(‘1234’, hashed);

## JSON Web Token

npm i jsonwebtoken

const jwt = require(‘jsonwebtoken’);

const token = jwt.sign({ \_id: user.\_id}, ‘privateKey’);

Save the private key in the environment variables ([Environment](#_Environment))

## Implementation

### ./config/default.js

{

"jwtPrivateKey": "vidly\_jwtPrivateKey"

}

### Validate jwt in index.js

if (!config.get('jwtPrivateKey')) {

console.error('FATAL ERROR: jwtPrivateKey is not defined.');

process.exit(1);

}

### Login router

router.post('/', async (req, res) => {

const { error } = validate(req.body);

if (error) return res.status(400).send(error.details[0].message);

let user = await User.findOne({ email: req.body.email });

if (!user) return res.status(400).send('Invalid email or password.');

const validPassword = await bcrypt.compare(req.body.password, user.password);

if (!validPassword) return res.status(400).send('Invalid email or password.');

const token = user.generateAuthToken();

res.send(token);

});

### Authorization middleware

const jwt = require('jsonwebtoken');

const config = require('config');

module.exports = function (req, res, next) {

const token = req.header('x-auth-token');

if (!token) return res.status(401).send('Access denied. No token provided.');

try {

const decoded = jwt.verify(token, config.get('jwtPrivateKey'));

req.user = decoded;

next();

}

catch (ex) {

res.status(400).send('Invalid token.');

}

}

### Authentication middleware

module.exports = function (req, res, next) {

// 401 Unauthorized

// 403 Forbidden

if (!req.user.isAdmin) return res.status(403).send('Access denied.');

next();

}

# Errors

## Errors

npm i express-async-errors

### in index.js

require(“express-async-errors”);

app.use(function(err, req, res, next) {

// Log the exception and return a friendly error to the client.

res.status(500).send(‘Something failed’);

});

## Winston

npm i Winston;

const winston = require('winston');

module.exports = function(err, req, res, next){

winston.error(err.message, err);

// error

// warn

// info

// verbose

// debug

// silly

res.status(500).send('Something failed.');

}

### Transports

Console, File and Http, MongoDB, CouchDB, Redis and Loggly

# Unit Testing

## Tools

* Jasmine
* Mocha, Chai, Sinon
* Jest

## Jest

npm i jest –-save-dev

Run jest with —coverage flag to get a code coverage report.

// Equality

expect(…).toBe();

expect(…).toEqual();

// Truthiness

expect(…).toBeDefined();

expect(…).toBeNull();

expect(…).toBeTruthy();

expect(…).toBeFalsy();

// Numbers

expect(…).toBeGreaterThan();

expect(…).toBeGreaterThanOrEqual();

expect(…).toBeLessThan();

expect(…).toBeLessThanOrEqual();

// Strings

expect(…).toMatch(/regularExp/);

// Arrays

expect(…).toContain();

// Objects

expect(…).toBe(); // check for the equality of object references

expect(…).toEqual(); // check for the equality of properties

expect(…).toMatchObject();

// Exceptions

expect(() => { someCode }).toThrow();

## Test sample

describe(“absolute”, () => {

it(“should be positive if negative value”, () => {

expect(value).toBe(1);

}

});

## Test exception

describe(“registerUser”, () => {

it(“should throw if username is falsy”, () => {

const args = [null, undefined, NaN, “”, 0, false];

args.foreach(arg => {

expect(() => { lib.registerUser(arg) }).toThrow();

});

});

});

## Continuously Running Tests

Modify package.json to allow jest to monitor file changes

“scripts”: {

“test”: “jest –watchAll”

},

# Integration Tests

npm i supertest –save-dev