**Node JS**

<https://nodejs.org/en>

Node.js® is an open-source, cross-platform JavaScript runtime environment.

<https://nodejs.org/docs/latest/api/documentation.html>

**V8 Engine**

<https://v8.dev/>

V8 is Google’s open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others. It implements [ECMAScript](https://tc39.es/ecma262/) and [WebAssembly](https://webassembly.github.io/spec/core/), and runs on Windows 7 or later, macOS 10.12+, and Linux systems that use x64, IA-32, ARM, or MIPS processors. V8 can run standalone, or can be embedded into any C++ application.

**Usage and example**

<https://nodejs.org/docs/latest/api/synopsis.html>

**Basics of Node JS**

**Variable declarations**

    let a = 10;

    var b = 10;

    const b = 10;

**Loops**

// for loop

    for (let index = 0; index < array.length; index++) {

        const element = array[index];

    }

    // for in loop

    for(let s in array) {

    }

    // for of loop

    for(let s of string) {

    }

    // while loop

    while (condition) {

    }

    // do while loop

    do {

    }

    while(condition);

    // foreach loop

    array.forEach(element => {

    });

    //map loop

    array.map(element => {

    });

**Conditional Statements**

    // if

    if(condition) {

    }

    // if else

    if(condition) {

    }

    else {

    }

    // if else if else

    if(condition) {

    }

    else if(condition) {

    }

    else {

    }

    // ternary operator

    (condition) ? true : false;

**Import/Export from one file to another**

File\_one.js

    module.exports = {

        x: 12,

        y: 13,

        z: function () {

            return 45;

        }

    }

File\_two.js

    const obj = require('./file\_one');

    console.log(obj);

**Array/string methods**

let str = "Some test string for manipulation";

let str2 = "other string";

// to get each character of string

for(index of str){

    console.log(index);

}

// return length of a given string

console.log('characters count: ' + str.length);

// get 2nd last character

console.log(str.charAt(str.length - 2));

// combine two or more string

console.log(str.concat(str2));

// get index position of given string

console.log(str.indexOf('for'));

// repeat the whole string

console.log(str.repeat(5));

//

console.log(str.replace('test', 'TEST'));

// return string from start index to given index

console.log(str.slice(5, 10));

// convert string to array

console.log(str.split(''));

// convert to lower

console.log(str.toLowerCase());

// convert to upper

console.log(str.toUpperCase());

// remove spaces from start and end

console.log(str.trim());

// remove spaces from left/start

console.log(str.trimLeft());

console.log(str.trimStart());

// remove spaces from end/right

console.log(str.trimEnd());

console.log(str.trimRight());

let arr = [43,23,44,12, 9, 43];

let arr1 = [10, 20, 30,];

console.log("Reverse an array");

console.log(arr.reverse());

let compare\_asc = (a, b) => {

    return a - b;

}

let compare\_desc = (a, b) => {

    return b - a;

}

console.log("Array sort asc");

console.log(arr.sort(compare\_asc));

console.log("Array sort desc");

console.log(arr.sort(compare\_desc));

console.log("Filter an array");

console.log(arr.filter( x => x != 43 ));

// add new value at the end of the array

console.log("Array push");

arr.push(34);

console.log(arr);

// remove value at the end of the array

console.log("Array Pop");

arr.pop();

console.log(arr);

// remove value from start of an array

console.log("Array shift");

arr.shift();

console.log(arr);

// add new value at the begining of an array

console.log("Array unshift");

arr.unshift(50);

console.log(arr);

// convert an array to string

console.log("Array to string");

console.log(arr.toString());

// convert array to given character

console.log("Array join");

console.log(arr.join('-'));

// combine multiple arrays to a single array

console.log("Combine Arrays");

console.log(arr.concat(arr1));

// extract values from start index to given index

console.log("Array Slice");

// slice (start, end);

arr.slice(0,3);

console.log(arr);

// add items in array from specific index and also remove an item or items

console.log("Add Items in array");

arr.splice(2, 1, 51, 52, 53, 54);

console.log(arr);

**Input from user**

    console.log(process.argv);

    node index params

**Promises in Node JS**

    let login = false;

    setTimeout(() => {

        login = true;

    }, 0);

    console.log(login);

    let login = false;

    let newLog = new Promise( (res, rej) => {

        setTimeout(() => {

            login = true;

            return res(login)

        }, 0);

    } )

    newLog.then( login => {

        console.log(login);

    } )

**Core modules**

const os = require('os');

    console.log(os.totalmem() / 1024 / 1024 / 1024);

    console.log(os.freemem() / 1024 / 1024 / 1024);

    console.log(os.homedir());

    console.log(os.hostname());

    console.log(os.machine());

    console.log(os.platform());

    console.log(os.version());

    const path = require('path');

    path.basename('/foo/bar/baz/asdf/quux.html');

    path.extname('index.html');

    path.join(\_\_dirname, 'files');

    const url = require('node:url');

    const myURL = url.parse('https://user:pass@sub.example.com:8080/p/a/t/h?query=string#hash');

    const myURL = new URL('https://example.org/foo#bar');

    console.log(myURL.hash);

    // Prints #bar

    myURL.hash = 'baz';

    console.log(myURL.href);

    const EventEmitter = require('events');

    const event = new EventEmitter();

    event.on("myEvent", function() {

        console.log("My Event Fired");

    });

    event.emit("myEvent");

**Creating HTTP Server**

<https://nodejs.org/en/guides/anatomy-of-an-http-transaction>

    const http = require('http');

    const port = 3000;

    http

    .createServer((request, response) => {

        const { headers, method, url } = request;

        let body = [];

        request

        .on('error', err => {

            console.error(err);

        })

        .on('data', chunk => {

            body.push(chunk);

        })

        .on('end', () => {

            body = Buffer.concat(body).toString();

            console.log(body);

            response.write('<html>');

            response.write('<body>');

            response.write('<h1>Hello, World!</h1>');

            response.write('</body>');

            response.write('</html>');

            response.end();

            // At this point, we have the headers, method, url and body, and can now

            // do whatever we need to in order to respond to this request.

        });

    })

    .listen(port); // Activates this server, listening on port 8080.

**Creating json file**

npm init -y

**Nodemon**

<https://www.npmjs.com/package/nodemon>

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

**Express JS**

<https://expressjs.com/>

Fast, unopinionated, minimalist web framework for Node.js

**Creating server with Express JS**

    const express = require('express');

    const app = new express();

    const port = 3030;

    app.listen( port, () => {

        console.log(`Listen at port: ${port}`);

    } )

**Send / Receive Data**

    app.get('/', (req, res) => {

        let users = [

            {id: 1, name: 'david'},

            {id: 2, name: 'micheal'},

            {id: 3, name: 'johnson'},

            {id: 4, name: 'gravit'},

        ]

        res.send(users)

    });

**Render HTML**

app.get('/', (req, res) => {

        res.send('Hello World!')

});

**working with html files**

app.use(express.static('views'))

app.get('/', (req, res) => {

        // res.send(users);

        file = path.join(\_\_dirname, 'views');

        let readF = readFileSync( `${file}/index.html` );

        res.send(readF.toString());

    });

**Template engine EJS**

<https://www.npmjs.com/package/ejs>

app.set('view engine', 'ejs');

app.get('/', (req, res) => {

    res.render('index');

});

**Extensions For EJS Template Engine Support**

<https://marketplace.visualstudio.com/items?itemName=DigitalBrainstem.javascript-ejs-support>

**Global Middleware**

const firstCall = ( ( req, res, next ) => {

    if(req.query.id == 12) {

        next();

    } else {

        res.send("Invalid Path");

    }

} );

app.use(firstCall);

**Routes Level Middleware**

app.get( '/home', firstCall, (req, res) => {

    res.send('<h1>Home Route</h1>');

} )

**Group Middleware**

const router = express.Router();

router.use(firstCall);

router.get('/about', (req, res) => {

    res.send('<h1>About Route</h1>');

})

router.get('/contact', (req, res) => {

    res.send('<h1>Contact Route</h1>');

})

app.use('/', router);

**Mongo DB Connection**

<https://www.npmjs.com/package/mongodb>

npm install mongodb

const { MongoClient } = require('mongodb');

const url = 'mongodb://localhost:27017/';

const database = 'db\_with\_node\_app';

const client = new MongoClient(url);

async function main() {

    // Use connect method to connect to the server

    await client.connect();

    console.log('Connected successfully to server');

    const db = client.db(database);

    const collection = db.collection('customers');

    const findResult = await collection.find({}).toArray();

    console.log('Found documents =>', findResult);

    // the following code examples can be pasted here...

    return 'done.';

}

main()

  .then(console.log)

  .catch(console.error)

  .finally(() => client.close());

const { MongoClient } = require('mongodb');

const url = 'mongodb://localhost:27017/';

const database = 'db\_with\_node\_app';

const client = new MongoClient(url);

async function dbConnection(coll) {

    // Use connect method to connect to the server

    await client.connect();

    const db = client.db(database);

    return db.collection(coll);

    // const findResult = await collection.find({}).toArray();

    // console.log('Found documents =>', findResult);

    // the following code examples can be pasted here...

    // return 'done.';

}

main('customers').then( (res) => {

    res.find().toArray().then( (data) => {

        console.log(data);

    } );

} )

let main = async () => {

    data = await dbConnection('customers');

    data = await data.find().toArray();

    console.log(data);

}

main();

**API’s**

**GET/POST/PUT/DELETE**

**Mongoose**

<https://mongoosejs.com/>

const express = require('express');

const mongoose = require('mongoose');

const dbURI = 'mongodb://localhost:27017/db\_with\_node\_app';

let dbConnection = () => {

    mongoose.connect(dbURI).then( () => {

    } );

}

let dbConnection = async () => {

    try {

        const conn = await mongoose.connect(dbURI);

    } catch (error) {

        console.log(error);

    }

};

dbConnection();

**Insert Records**

const express = require('express');

const mongoose = require('mongoose');

const dbURI = 'mongodb://localhost:27017/db\_with\_node\_app';

let dbConnection = async () => {

    try {

        await mongoose.connect(dbURI);

        const customerSch = new mongoose.Schema({

            firstName: String,

            lastName: String

        });

        const customerModel = new mongoose.model('customers', customerSch);

        const data = new customerModel({

            firstName: 'from schema',

            lastName: 'again schema',

            email: 'from@em.com'

        });

        let r = await data.save();

        console.log(r);

    } catch (error) {

        console.log(error);

    }

};

dbConnection();

**Schema / Model**

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const CustomerSchema = new Schema({

    firstName: {

        type: String,

        required: true

    },

    lastName: {

        type: String,

        required: true

    },

    email: {

        type: String,

        required: true

    },

    tel: {

        type: String,

        required: true

    },

    createAt: {

        type: Date,

        default: Date.now()

        // required: true

    },

    updateAt: {

        type: Date,

        default: Date.now()

        // required: true

    }

});

module.exports = mongoose.model('Customer', CustomerSchema);

**API’s with mongoose**

const express = require('express');

const dbConnection = require('./config/dbConn');

const Customer = require('./model/customer');

const app = new express();

app.use(express.json());

dbConnection();

app.get('/', async (req, res) => {

    let customers = await Customer.find({});

    res.send(customers);

});

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

    let id = new Customer(req.body);

    let resl = await id.save();

    res.send(resl);

});

app.delete('/customer/:\_id', async(req, res) => {

    let d = await Customer.deleteOne(res.params);

    res.send(d);

});

app.put('/customer/:\_id', async(req, res) => {

    let d = await Customer.updateOne(

        req.params,

        {

            $set: req.body

        }

    );

    res.send(d);

})

app.listen(3000);

**Uploading File**

<https://www.npmjs.com/package/multer>

const express = require('express');

const multer  = require('multer')

const app = new express();

app.use(express.json());

const storage = multer.diskStorage({

    destination: function (req, file, cb) {

      cb(null, 'uploads')

    },

    filename: function (req, file, cb) {

      const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1E9)

      cb(null, file.fieldname + '-' + uniqueSuffix + '.jpg')

    }

})

const upload = multer({ storage: storage })

app.post('/post-file', upload.single('avatar'), function (req, res, next) {

    res.send(req.file);

});

app.listen(3000);

**Express Session**

[**https://www.npmjs.com/package/express-session**](https://www.npmjs.com/package/express-session)

**MySql Connection**

<https://www.npmjs.com/package/mysql>