

To run - node index.js

To debugging - node index.js --inspect-brk (OR) node --inspect-brk index.js

Debugg on chrome : chrome://inspect/

Where ever we need to breakpoint – ‘debugger;’

Nodemon – nodemonitoring.

Npm.js ( std repo)

Npm install library-name

Npm install library-name@version ( install a particular version)

2 levels of installation

Globally 🡺 System level install

Locally -> Project level install

Npm install library-name –g/-S/-D ( In terminal)

-g = Global

-S/-D = Local/project level

-S = Dependency

-D= development dependency

To check installation

Windows+r

Appdata-> Romaing->npm 🡪 nodmon

Ajax – Asynronous function

//let, const -. variable types

IIFE – Functions

Closeure functions

Immediate invoke function

Jquery - $.noconflict 🡪

Let is like a block scopr variable

Constant - Can not change.

const a=10;

a=12;

[nodemon] starting `node test1.js`

App startes

C:\Users\Administrator\Desktop\11032019\test1.js:4

a=12;

^

TypeError: Assignment to constant variable.

at Object.<anonymous> (C:\Users\Administrator\Desktop\11032019\test1.js:4:2)

at Module.\_compile (internal/modules/cjs/loader.js:799:30)

at Object.Module.\_extensions..js (internal/modules/cjs/loader.js:810:10)

at Module.load (internal/modules/cjs/loader.js:666:32)

at tryModuleLoad (internal/modules/cjs/loader.js:606:12)

at Function.Module.\_load (internal/modules/cjs/loader.js:598:3)

at Function.Module.runMain (internal/modules/cjs/loader.js:862:12)

at internal/main/run\_main\_module.js:21:11

[nodemon] app crashed - waiting for file changes before starting...

Single value peoperties

//Single value Properties

const name='demo';

const age=21;

const user= {

//name: name,

//age: age

name,

age

}

console.log(user);

console.log('ends.here');

^CTerminate batch job (Y/N)? y

PS C:\Users\Administrator\Desktop\11032019> nodemon test2.js

[nodemon] 1.18.10

[nodemon] to restart at any time, enter `rs`

[nodemon] watching: \*.\*

[nodemon] starting `node test2.js`

{ name: 'demo', age: 21 }

[nodemon] clean exit - waiting for changes before restart

[nodemon] restarting due to changes...

[nodemon] starting `node test2.js`

{ name: 'demo', age: 21 }

[nodemon] clean exit - waiting for changes before restart

[nodemon] restarting due to changes...

[nodemon] starting `node test2.js`

{ name: 'demo', age: 21 }

ends.here

[nodemon] clean exit - waiting for changes before restart

***Destructuring syntax***

/destructuring syntax -- Some js file haveing 100 function if need to user only 2 function we can use this

const name='demo';

const age=21;

const user= { name, age}

const userObj = {

userName: 'demo',

userAge : 21

};

const { userName} = userObj;

console.log(userName);

//console.log(user);

console.log('ends.here');

odemon] restarting due to changes...

demp

ends.here

[nodemon] starting `node test2.js`

demp

ends.here

[nodemon] clean exit - waiting for changes before restart

[nodemon] restarting due to changes...

[nodemon] starting `node test2.js`

demo

ends.here

[nodemon] clean exit - waiting for changes before restart

[nodemon] restarting due to changes...

[nodemon] starting `node test2.js`

demo

ends.here

[nodemon] clean exit

Ajax call 🡺 Goes to servers and re-trive data

Ajax 🡪 server🡪 response

KEYPRESS -> Ajax 🡪 server🡪 response🡪SUGGESTION

SAMSUNG ( 7 REQUEST – 7 RESPONSE ( GOOD CHROME AJAX CALLS))

CALL BACK HELL ??

So many call back, system dosen’t know whic call for whom..

Promise – call back issue resolve

Call back hell -> system doesn’t know which function needs to execute, wherher it is first call or second call.

jaxWithPromise

having then/catch

promise 🡪 once success call is okay, failure call is eliminated from the call back queue

function ajaxWithPromise() {

return new Promise(

function (resolve, reject) {

setTimeout(

function () {

resolve('call successds');

}, 2000

);

setTimeout(

function () {

reject('call fails');

}, 5000

)

}

);

}

ajaxWithPromise()

.then(

function (data) { console.log('success->', data) },

).catch (

function (error) { console.log('failes->', error) }

);

[nodemon] restarting due to changes...

[nodemon] starting `node test4.js`

success-> call successds

[nodemon] clean exit - waiting for changes before restart

$.ajax(

{

Success:function() {},

Faile:function(){}

}

)

$.ajax().then().catch()

Arrow funcctions:

Each js file is one single modules

Modules

//MODULES

const add=(a, b) => {

console.log('sum' , a + b);

}

const sub=(a, b) => {

console.log('sub' , a - b);

}

const mul=(a, b) => {

console.log('mul' , a \* b);

}

const div=(a, b) => {

console.log('div' , a / b);

}

// All the code inside the java file is local to that or private. If you need to use in another js file

//need to do exports operation

module.exports = {

add, sub, mul, div

}

Test.7.JS

//main file

//require ==>commonjs module ( it helps to load library)

//const mathOps=require('./test5'); // everything is object, so test5 is object and mathOps also objects

const { add, div} = require('./test5');

add(2,3);

div(10,2);

*PS C:\Users\Administrator\Desktop\11032019> nodemon test7.js*

*[nodemon] 1.18.10*

*[nodemon] to restart at any time, enter `rs`*

*[nodemon] watching: \*.\**

*[nodemon] starting `node test7.js`*

*sum 5*

*div 5*

*[nodemon] clean exit - waiting for changes before restart*

*That general*

Reduce server load/increase client server usage

Npm initialization:

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Desktop\11032019\new> npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields

and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and

save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (new)

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to C:\Users\Administrator\Desktop\11032019\new\package.json:

{

"name": "new",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

‘’start”: “node index.js”,

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

Is this OK? (yes)

PS C:\Users\Administrator\Desktop\11032019\new>

Npm install lodash –S

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Desktop\11032019\new> npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields

and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and

save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (new)

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to C:\Users\Administrator\Desktop\11032019\new\package.json:

{

"name": "new",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

Is this OK? (yes)

PS C:\Users\Administrator\Desktop\11032019\new>

PS C:\Users\Administrator\Desktop\11032019\new> npm install lodash –S

PS C:\Users\Administrator\Desktop\11032019\new> npm install webpack webpack-cli –D

All the dependcies also will tacks care.

API ---

<https://nodejs.org/dist/latest-v10.x/docs/api/fs.html>

file write:

const fs=require('fs');

//npm run start

fs.appendFile('message.txt', 'data to append', (err) => {

if(err) throw err;

console.log('The "data to append" was appended to the file!');

});

PS C:\Users\Administrator\Desktop\11032019\new> npm run start

> new@1.0.0 start C:\Users\Administrator\Desktop\11032019\new

> node index.js

The "data to append" was appended to the file!

PS C:\Users\Administrator\Desktop\11032019\ne

Npm i yargs

‘i’ is the shorter version of install

Yargs = for command line arguments  
;

const fs=require('fs');

const { argv } = require('yargs');

//node index.js --filename=demo.txt --name=mike --age=21

console.log(argv);

//npm run start

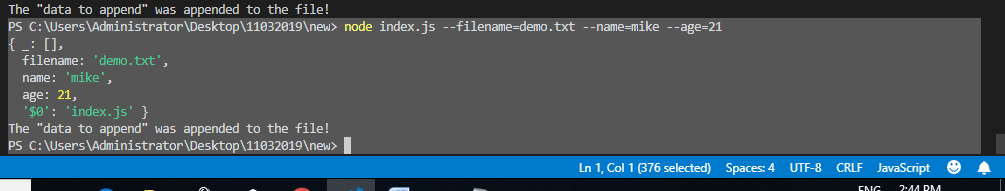
//fs.appendFile('message.txt', 'data to append', (err) => {

fs.appendFile(argv['filename'], 'data to append', (err) => {

if(err) throw err;

console.log('The "data to append" was appended to the file!');

});



Any action is promise/eventemitter

On is keyword for, to add listener.

const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();

myEmitter.on('dbconnected', () => {

console.log('an event occurred!');

});

myEmitter.once('dberror', () => {

console.log('an event occurred here!');

});

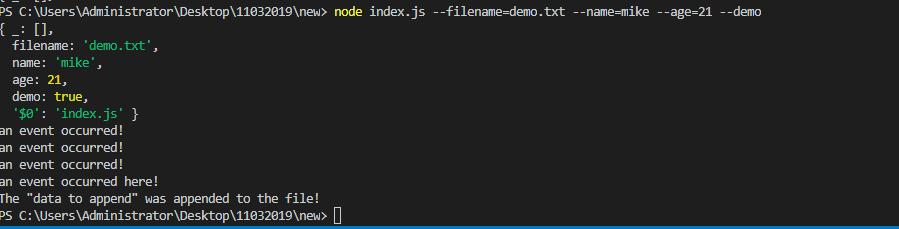
myEmitter.emit('dbconnected');

myEmitter.emit('dbconnected');

myEmitter.emit('dbconnected');

myEmitter.emit('dberror');

myEmitter.emit('dberror');



Nodemon is a example of – on event ( keep listene)

Emit – event occured

On – registered event

Express App

API-APP

src/api – source of the folder.

Config.js 🡪 db connection details

Pac kage.json

Server.js

WEB-APP

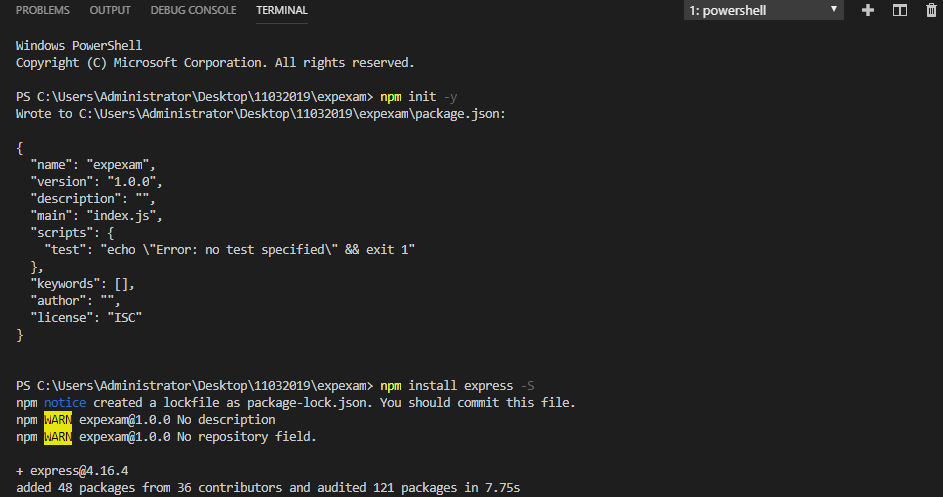
Src

Package.json

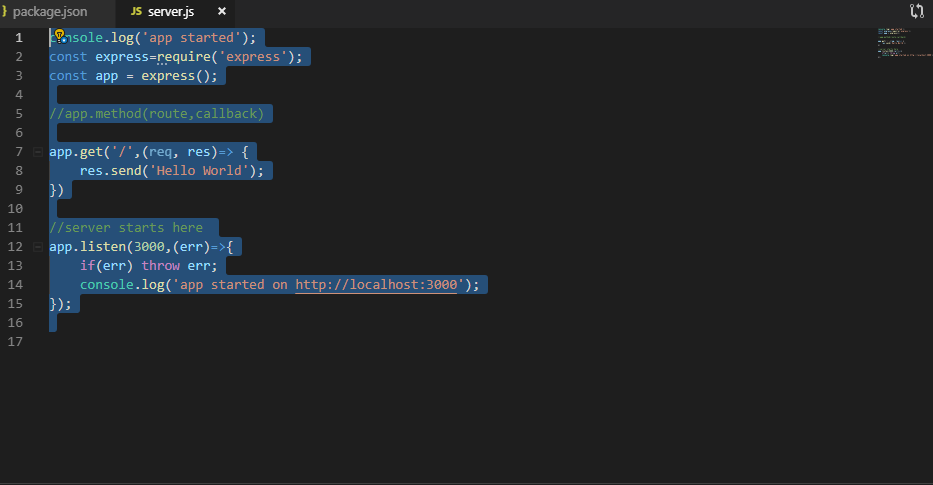
App.js

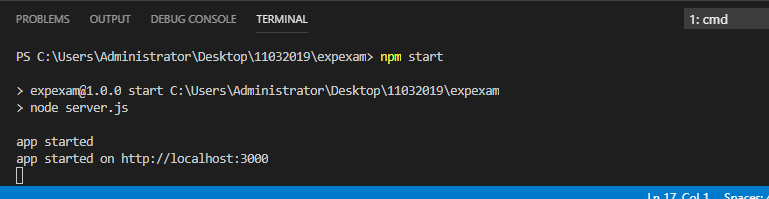
Npm init –y ( to skip questions)

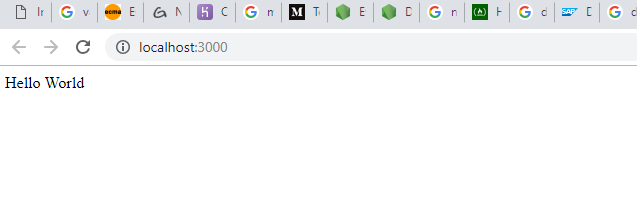
Npm install express –S (dev/prod both)



Typical application like,







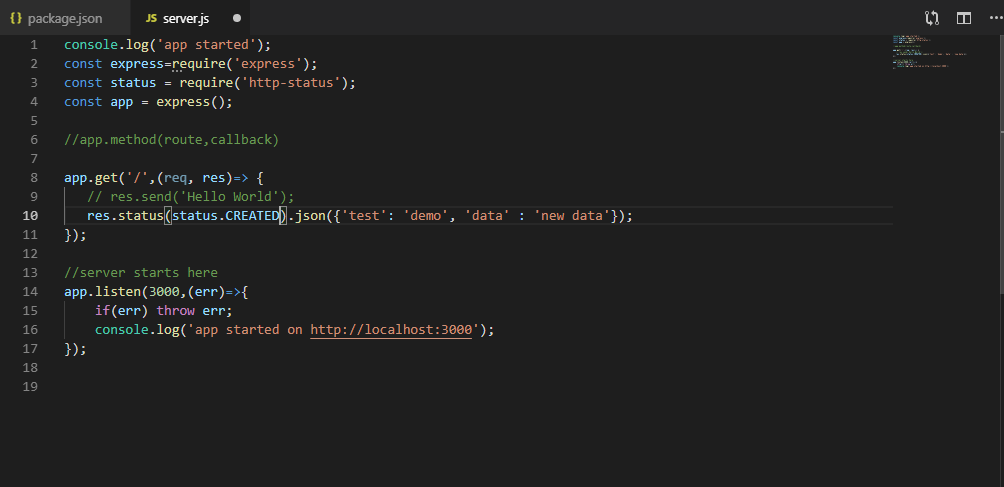
app.get('/',(req, res)=> {

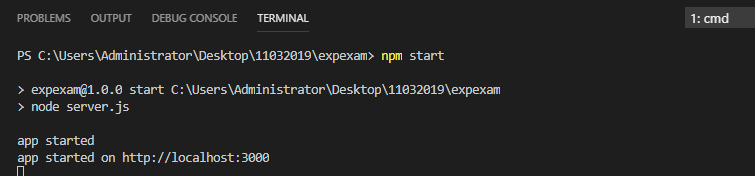
// res.send('Hello World');

res.json({'test': 'demo', 'data' : 'new data'});

});

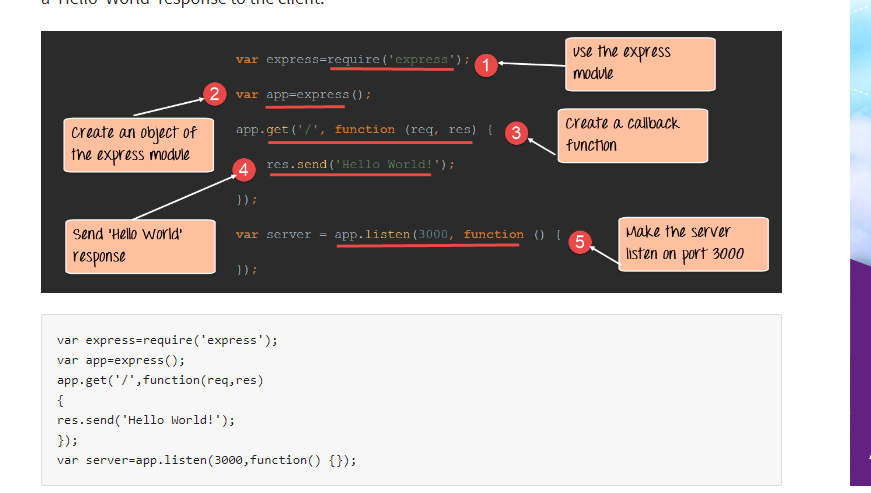
Importing more than one package:





What ever the package, we need to install via “npm install http-status” and do the import using require method.

Const status = require(‘http-status’);



Body-parser used to parse the request message . (npm.js)

URLencode only for form-date

Form date for file upload where dateand file will go ( multi part)

Express- Framework

BodyParser – used parsing request body

Helemet – Used to handle basic web attchs (like xss)

Morgen – logger

C- POST

R- GET

U-PUT

D-DELETE

abc.com/api/employee – GET

abc.com/api/employee/100 – GET

abc.com/api/employee –POST

abc.com/api/employee/100 – PUT

abc.com/api/employee/100 – DELETE

Robo 3t = db connection

Postman

Npm install mongodb

<http://mongodb.github.io/node-mongodb-native/3.1/quick-start/quick-start/>

Mongoose – ODM – Object doument mapping

Sequlize – sql/mysql/sqllite/postgres

Src

Config – application configs

Api – routes definition

Models – schemma definitions

Repository – db helper functions

Server.js – server config file