Seção 10 Javascript – Node, Npm, Express e MongoDB(NodeJs)

* Aula 01 Modulos(1)

const nome = 'Luiz'

const sobrenome = 'Miranda'

const falaNome = () => {

    console.log(nome + ' '+sobrenome);

}

module.exports.nome = nome

module.exports.sobrenome = sobrenome

module.exports.falaNome = falaNome

console.log(module.exports);

para exportar variáveis com node

const nome = 'Luiz'

const sobrenome = 'Miranda'

const falaNome = () => {

    console.log(nome + ' '+sobrenome);

}

// module.exports.nome = nome

// module.exports.sobrenome = sobrenome

// module.exports.falaNome = falaNome

exports.nome = nome

exports.sobrenome = sobrenome

exports.falaNome = falaNome

this.qualquerCoisa = 'O que eu quiser exportar'

console.log(exports);

aqui temos uma forma reduzida desse export

exports.NOME = nome

exports.SOBRENOMEQUALQUER = sobrenome

a variável do exports não precisa ter o mesmo nome, ela ira receber o valor que esta recebendo.

Agora vamos importar esses códigos em outro arquivo:

const mod1 = require('./mod1')

em outro arquivo (app.js), vamos fazer o import do código do arquivo(mod1).

Nesse caso nos não ultilizamos o import e sim o require(‘caminho\_do\_arquivo’), para fazer a importação

const mod1 = require('./mod1')

console.log(mod1);

mod1.falaNome()

const falaNome = require('./mod1').falaNome;

console.log(falaNome());

aqui estou fazendo a importação direto da função falaNome, na variável falaNome

const { nome, sobrenome, falaNome} = require('./mod1')

console.log(nome, sobrenome);

console.log(falaNome());

aqui estamos fazendo a destruct do objeto, e pegando so as variáveis que queremos

module.exports = {

    nome, sobrenome, Pessoa

}

Você pode exportar varias variáveis juntas caso deseje

const mod1 = require('./mod1')

const { Pessoa } = require('./mod1')

const p1 = new Pessoa('Luis')

const p2 = new mod1.Pessoa('Jão')

console.log(p1);console.log(p2);console.log(mod1.nome);

aqui estamos importando todos os export de mod1 e também uma destruct que so recebe a classe Pessoa de mod1

quando se usa o objeto de module.exports, ele sobre escreve oque foi exportado anteriomente

module.exports = {

    nome, sobrenome, Pessoa

}

Em outro arquivo:

console.log(mod1);

{ nome: 'Luiz', sobrenome: 'Miranda', Pessoa: [class Pessoa] }

Isso eo que foi exportado

exports.nome = nome

exports.sobrenome = sobrenome

exports.falaNome = falaNome

exports.qualquer = qualquer

// module.exports = {

//     nome, sobrenome, Pessoa

// }

Quando tiramos o module.exports

No outro arquivo recememos:

console.log(mod1);

{

  nome: 'Luiz', sobrenome: 'Miranda',: [Function: falaNome],

  qualquer: 'Qualquer coisa'

}

* Aula 02 Modulos (2)

module.exports = class Animal {

    constructor(nome){

        this.nome = nome

    }

    latir() {

        console.log(`${this.nome} está fazendo au au`);

    }

}

Arquivo mod

const Cachorro = require('../B/C/D/mod')

module.exports = Cachorro

arquivo mod2

const Cachorro = require('./Z/mod2')

const c1 = new Cachorro('Dog')

c1.latir()

console.log(Cachorro);

arquivo app, que recebe de mod2, que por sua vez recebe de mod

console.log(\_\_filename);  // nome do arquivo atual, caminho

absoluto

console.log(\_\_dirname);   // nome da pasta atual, caminho absoluto

const path = require('path')

console.log(\_\_filename);  // nome do arquivo atual

console.log(\_\_dirname);   // nome da pasta atual

console.log(path.resolve(\_\_dirname, '..','..','nome\_da\_pasta\_para\_ir'));

aqui ele esta voltando duas pastas e entrando no nome da pasta passado

* Aula 03 NPM – Node Package Manager

Vamos iniciar uma nova npm

Na pasta que deseja realizar o projeto, abra o terminal e digite:

Npm init -y

Feito ira gerar um arquivo package.json, na sua pasta principal do projeto, contendo sua configuração

Agora vamos instalar o express:

npm install express

sera instalado no arquivo package.json, em dependencies o express na sua ultima versão

para atualizar seus pacotes instalados, basta colocar o código:

npm update

para mover o express para a parte de desenvolvimento:

npm install express --save-dev

agora seus express esta em devDependencies

para voltar para dependencies

npm install express --save-prod

agora seu express está em dependencies

para escolher a versão que deseja trabalhar:

npm install express@2.1.0

versão 2. 1. 0

major minor patch

patch = correção de bug

minor = adicionado novo recurso

major = vou quebrar a compatibilidade com coisas que tem nas versões anteriores

E - tira a atualização do pacote

^ - atualiza tanto a minor quanto a pach, ele não atualiza o major

~ - mexe só na versão pacth

express@4.x – instala o pacote express na versão 4 em sua ultima versão

npm update – atualizar a versão

npm unistall express – desistala o pacote

npm ls - lista os pacotes instalados pelos pacotes

npm ls -depth=0 - lista somente os pacotes instalados por mim

npm ls -depth=1 - lista os pacotes instalados com um de profundidade