**NODE JS**

Node.js is an open source server environment.

Node.js allows you to run JavaScript on the server.

Node.js uses JavaScript on the server

**Node.js uses asynchronous programming!**

A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

Here is how Node.js handles a file request:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request.

Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

What Can Node.js Do?

* Node.js can generate dynamic page content
* Node.js can create, open, read, write, delete, and close files on the server
* Node.js can collect form data
* Node.js can add, delete, modify data in your database

## What is a Node.js File?

* Node.js files contain tasks that will be executed on certain events
* A typical event is someone trying to access a port on the server
* Node.js files must be initiated on the server before having any effect
* Node.js files have extension ".js"

APUNTES RUAN:

creo el Proyecto con [npm init], ejecuto un archivo node con [node nombredelarchivo.js] (ej node index.js). Si quiero que ese archivo se reinicie con cada alteración que hago automáticamente ( en vez de refreshear), puedo usar [node –watch nombredelarchivo.js]

**INTRODUCCION A PACOTES NODE.**

Modulo fs (filesystem) permite interagir con el sist de archivos del OS.

Basicamente: creamos una const fs = require(‘fs). [lo que hicimos es guardar en una variable fs ese comando require(‘fs), asi luego usaremos fs(la variable) para usar los métodos. Por ej:

const fs = require('fs')

const data = fs.readFileSync('arquivo.txt', 'utf-8')

console.log(data)

Aca lo que hicimos es ademas de usar fs.readFileSync(), ese resultado lo almacenamos en la const data y lo mostramos en un console.log

Por ej, para receber dados via terminal usamos el PACOTE:

**npm install readline-sync**

E para receber dados via terminal, usamos a seguinte sintaxe:

const readlineSync = require('readline-sync')

let nomeInserido = readlineSync.question("Qual é o seu nome?")

**Entendé que Node basicamente se maneja por MODULOS.**

## What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

Node.js has a set of built-in modules which you can use without any further installation.

Look at our [Built-in Modules Reference](https://www.w3schools.com/nodejs/ref_modules.asp) for a complete list of modules.

To include a module, use the require() function with the name of the module:

var http = require('http'); //SIEMPRE ES BUENO ‘GUARDAR’ ESA function EN UNA

VARIABLE.

## Create Your Own Modules

You can create your own modules, and easily include them in your applications.

The following example creates a module that returns a date and time object:

exports.myDateTime = function () {  
  return Date();  
};

Use the exports keyword to make properties and methods available outside the module file.

ALGUNOS OTROS MODULOS USADOS COMUNMENTE:

**OPERATING SYSTEM:**

const os = require(“os”)

console.log(“Nome da maquina:”, os.hostname()) y asi varios mas

**MODULO PATH:** para trabajar con caminhos de arq/pastas y manipular estos

const path = require(“path”)

const pasta = “aluno/Desktop/aula\_2”

const arquivo = “script.js”

>>> const camichoCompleto = path.join(pasta, arquivo) y asi varios mas.

**MODULO PROCESS:** fornece informaçoes e controle sobre o processo Node.js em execução

**Sintax >>** console.log(“Variaveis de ambiente:” process.env) las variables de ambiente son informaciones sobre el ambiente en el cual el proceso esta siendo ejecutado. **El primer elemento es el camino para el ejecutable (geralmente “node”) y el Segundo element es el camino para el arquivo node.js que esta siendo ejecutado.**

Ej >> console.log(“ID de processo:”, process.pid) retorna el ID del proceso en el sist operacional

**MODULO dotenv PARA CREACION DE VARIABLES DE AMBIENTE:**

dotenv es una biblioteca para cargar variables de ambiente a partir de archivos de configuracion, donde pueden incluirse senhas de acceso, chaves API y otras informaciones sensibles.

**SIEMPRE HAY QUE INSTALARLO!!**

**npm install dotenv**

y luego crear un archivo **.env** donde guardaremos esas variables que precisamos

ese arquivo **.env** puede tener adentro pro ejemplo:

AULA\_HOJE = “modulos\_node”

MINHA\_SENHA = “javascript123”

FIJATE BIEN LA SINTAX: EN MAYUSCULAS Y LO OTRO ENTRE COMILLAS.

Luego en el arquivo .js necesitamos activarlo:

const dotenv = require(“dotenv”);

dotenv.config()

Con eso ya conseguimos accesar os valores del arquivo **.env** para poder usarlos

Luego por ej puedo usarlos asi:

Const minhaSenha = process.env.MINHA\_SENHA y asi sucesivamente

## What is NPM?

NPM is a package manager for Node.js packages, or modules if you like.

[www.npmjs.com](https://www.npmjs.com/) hosts thousands of free packages to download and use.

The NPM program is installed on your computer when you install Node.js

## What is a Package?

A package in Node.js contains all the files you need for a module.

Modules are JavaScript libraries you can include in your project.

**PARA BAJAR UN PACKAGE, VAS A LA PAGINA DE NODE, BUSCAS EL QUE QUERES BAJAR Y LO INSTALAS CON npm install [nombre\_del\_package]** esto lo haces el el folder donde vas a trabajar.

**INTRO A APIs POR RUAN.**

A veces tenemos que ir a buscar informacion a ciertas APIs, para ello se usa el protocolo http.

El concepto es que tenemos entre cliente/servidor, **requisiciones(req)** y **respuestas(res).**

Basicamente una API gerencia un conjunto de rutinas para que cliente y servidor se puedan comunicar; aquí se definen URLs, métodos(GET,POST, etc), árametros, formatos da datos(JSON, XML, etc) y otros detalles mas.

**MÉTODOS.**

**GET** cliente **pide/recupera** datos del server

**POST** cliente **envía** datos al server

**PUT** cliente **actualiza** datos

**DELETE borra** datos

Cuando recibimos la respuesta del SERVE siempre va a venir acompañada de un STATUS de retorno (100 a 599)

[ver lista de códigos de retorno]

**Normalmente la sintaxis para uma requisitória http es la siguiente**

**Método + url + path**

**GET** [**https://jsonplaceholder.typicode.com/posts**](https://jsonplaceholder.typicode.com/posts)

GET es el método luego en azul la URL y posts es el path

Un dato: cuando mandamos info sensible, no usar GET sino POST que es mas seguro (pues encripta esa info en una sección body).

Otra: ahí, el path es un parámetro para identificar un recurso, pero también puede usarse una **query** para filtrar esos recursos:

GET con path >>> /produtos/{id} trarea el prodycto con determinado ID

GET con query >>> /productos¿precoMin=50precoMax=150 filtra entre esos precios

**POST** se usa para enviar/cadastrar info (sensible) y por tal motivo se pasan esas infos como un JSON en el body de la requisicion:

**PUT** se usa para actualizar los datos de un JSON en el body

**DELETE** se usa para borrar.

**CREACION DE UN SERVIDOR CON EXPRESS.**

PASOS:

INSTALAR EXPRESS: te posicionas en la carpeta donde vas a guardar el proyecto, haces **npm init** e instalas Express con **npm install express**

Una vez instalado ya podes importar el modulo express, guardarlo en una variable para ejecutarlo:

**const express = require(‘express’)**

**const app = express()**

Y LUEGO INICIAS EL SERVIDOR EN PORT 3000, AQUÍ CON UN CALLBACK QUE MANDA UN CONSOLE.LOG AVISANDO QUE ESTA TODO OK

**app.listen(3000, () => console.log(‘O servidor está rodando’))**

EXPLICACION: app (que es express) usa el método .listen para abrir servidor en port 3000 y luego la arrow function manda el mensaje en console.log

**ENTONCES LUEGO LA SINTAXIS PARA USAR EXPRESS ES ASI:**

**app.metodoHTTP(‘/url’, callback)**

EXPLICACION: app(express).un método (GET, POST, etc) luego (el path de la url ej ‘/api’, y el callback va a tener >>> (req, res que es requisicion y respuesta) => { res.status(200).send(‘Hola’)} >>>> eso ultimo (el callback) manda una res con status 200 ok y manda (.send) un mensaje (Hola)

ENTONCES RECAPITULANDO UN POCO TODO ESTO:

Importas express con const express = require(‘express’)

Ejecutas la función const app = express()

Usas esa función **app** con métodos y rutas URL, ej:

**App.get(‘/api’, (req, res) => {**

**Res.status(200).send(‘Ola Mundo!’)**

**})**

Y acordate que siempre al final tenes que tener habilitado el port 3000 para el localhost:

**app.listen(3000, () => console.log(‘O servidor esta rodando’))**

**Luego tb podemos recibir info via rutas con path params, que es esto¿**

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

const aluno = req.**params**.aluno

res.status(200).send(‘Seja bem bindo ${aluno})

})

**EXPLICACION:** app llama a el método put y la ruta va a ser /api mas /:aluno (ese es el **param**), luego tenes el callback con req, res que retorna (guardandola en una variable aluno) la **req.params.aluno (o sea la req del path/param .aluno)** y la **res** devuelve status 200 y un mensaje.

En POSTMAN lo checkeas asi: PUT [http://localhost:3000/api/Ruan[nome](http://localhost:3000/api/Ruan%5bnome) do aluno aca]

**AHORA PARA ACCESAR VIA QUERY PARAMS:**

La sintaxis seria:

app.get(‘/api’, (req, res) => {

console.log(req.query)

res.status(200).send(‘Ola, ${req.query.nome}’)

})

En POSTMAN lo checkeas asi: GET http://localhost:3000/api¿nome=Ruan

EN SOLICITACIONES DE POST, PUT, PATCH QUE ACOSTUMBRAN PRECISAR DE MUCHAS INFOS, ES COMUN PARAS ESA INDO EN EL BODY EN FORMATO JSON.

Primeo instalar el pacote

**npm install body-parser**

luego como siempre, lo importamos:

const bodyParser = require(‘body-parser’)

app.use(bodyParser.json())

y después utilizamos la propiedad **req.body** para accesar:

app.post(‘/api’, (req, res) => {

const body = **req.body**

res,status(200).send(‘Recebemos seus dados ${body.nome}!’)

})

ACA VUELVO A W3 SCHOOLS.