

## Conteúdo

### ***Módulo 1: Introdução ao Node.JS***

- Primeiro Projeto com NodeJS - Mensagem de Saudação
- Projeto - Cores no Console.log
- Projeto - Server embutido com node.JS
- Projeto - Servidor embutido carregando página index.html

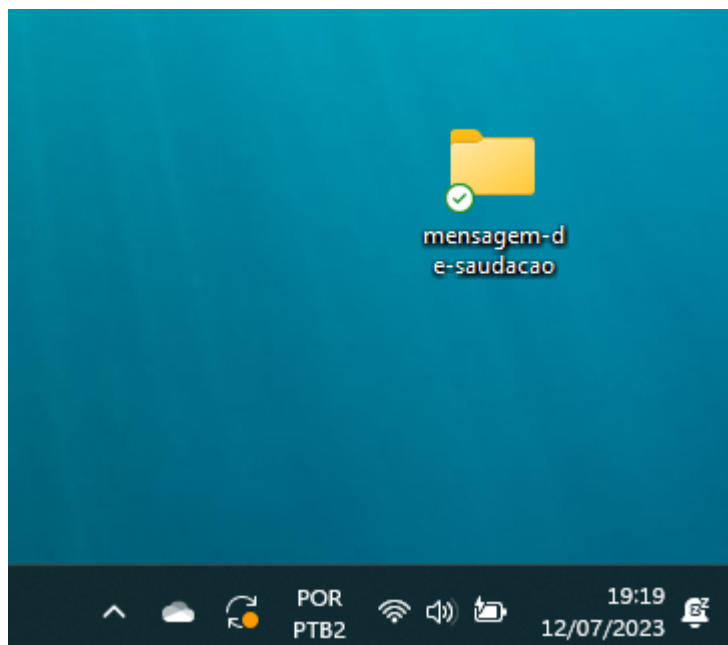
## PRIMEIRO PROJETO EM NODEJS - MENSAGEM DE SAUDAÇÃO

Nesta aula criaremos uma aplicação simples, que retorna uma saudação de acordo com a hora do dia. Vale ressaltar que este não é um exemplo de uma aplicação back-end, é apenas uma introdução ao Node.js.

O exemplo que veremos é um programa que retorna uma mensagem de saudação de acordo com a hora do dia.

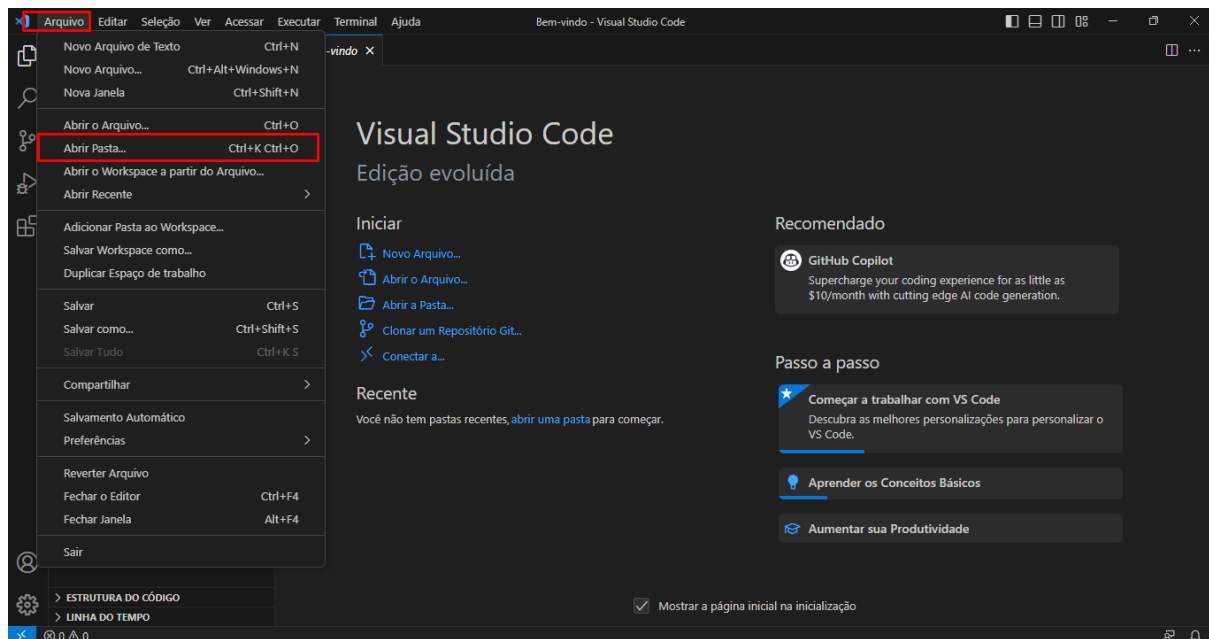
Esta não é uma aplicação **back-end**, mas sim uma aplicação de linha de comando. Estaremos executando um simples código Javascript através do **Node.js**.

Primeiro passo, criaremos uma pasta na área de trabalho ou em meus documentos em seu computador com o nome do projeto:

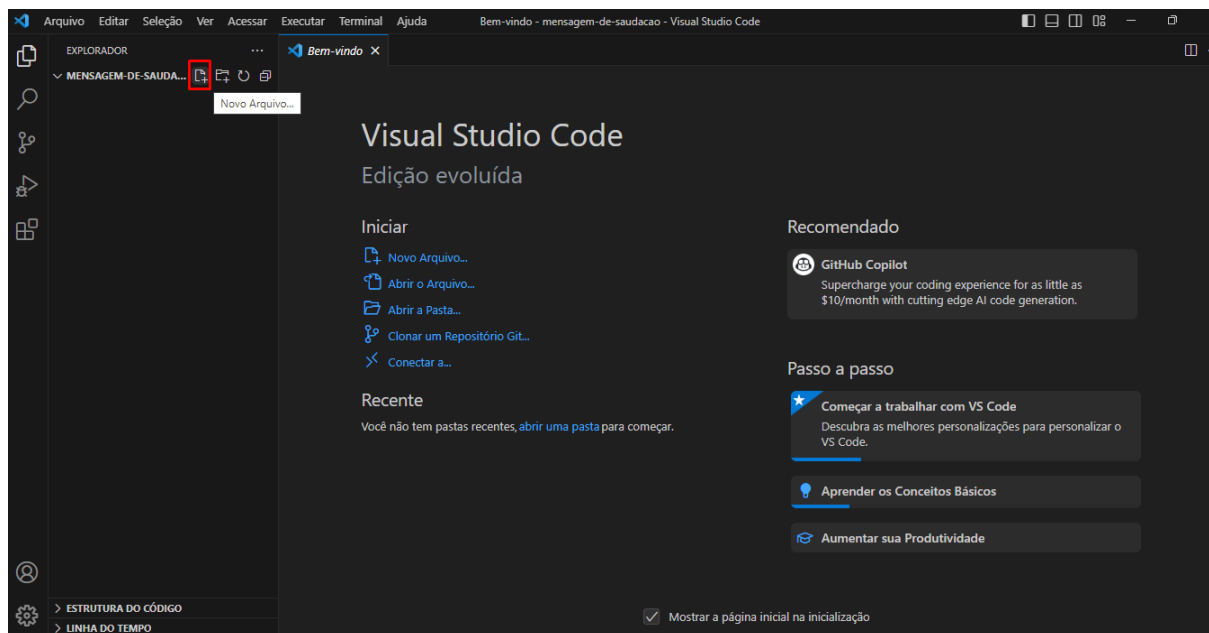


Abrimos a pasta através do Vscode:

# Unisenai



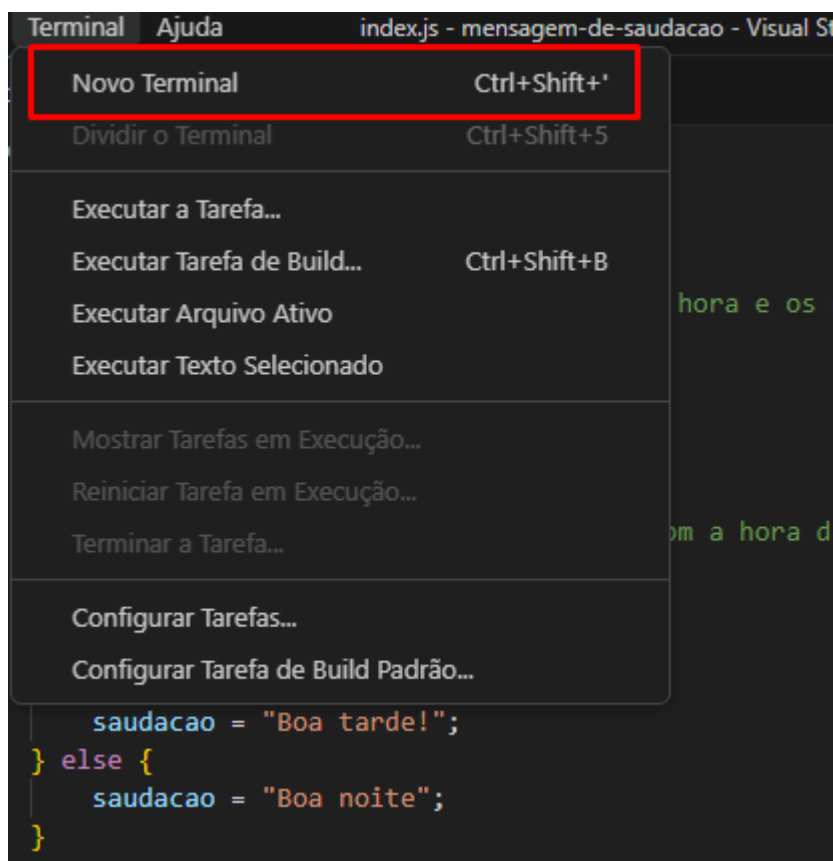
Criamos um arquivo chamado index.js



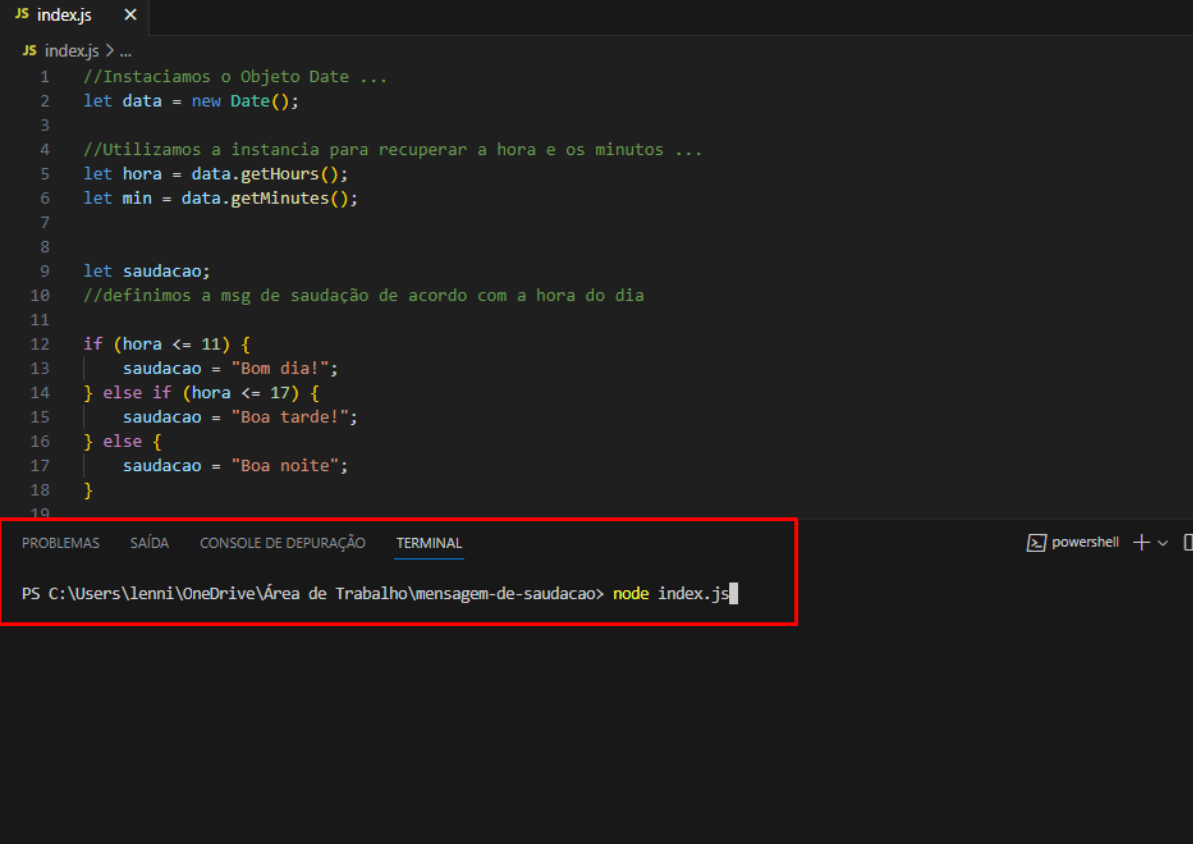
Agora vamos para a implementação do projeto.

# Unisenai

Pronto, agora vamos aprender como executar o nosso projeto.



Executamos o comando Node seguido do nome do arquivo index.js que criamos dentro da pasta do projeto no nosso terminal:



```
JS index.js x
JS index.js > ...
1 //Instanciamos o Objeto Date ...
2 let data = new Date();
3
4 //Utilizamos a instancia para recuperar a hora e os minutos ...
5 let hora = data.getHours();
6 let min = data.getMinutes();
7
8
9 let saudacao;
10 //definimos a msg de saudação de acordo com a hora do dia
11
12 if (hora <= 11) {
13     saudacao = "Bom dia!";
14 } else if (hora <= 17) {
15     saudacao = "Boa tarde!";
16 } else {
17     saudacao = "Boa noite";
18 }
19

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
PS C:\Users\lenni\OneDrive\Área de Trabalho\mensagem-de-saudacao> node index.js
```

Este é o resultado da execução da Aplicação:

# Unisenai

```
JS index.js X
JS index.js > ...
6   let min = data.getMinutes();
7
8
9   let saudacao;
10  //definimos a msg de saudação de acordo com a hora do dia
11
12  if (hora <= 11) {
13      saudacao = "Bom dia!";
14  } else if (hora <= 17) {
15      saudacao = "Boa tarde!";
16  } else {
17      saudacao = "Boa noite";
18  }
19
20  console.log('Olá! ' + saudacao);
21  console.log('Agora são ' + hora + ' horas e ' + min + ' minutos.');
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL powershell + -

```
PS C:\Users\lenni\OneDrive\Área de Trabalho\mensagem-de-saudacao> node index.js
Olá! Boa noite
Agora são 19 horas e 44 minutos.
PS C:\Users\lenni\OneDrive\Área de Trabalho\mensagem-de-saudacao>
```

Dessa forma nós implementamos e executamos nosso primeiro projeto com Node.js.

Vamos continuar praticando? vamos precisar da lib cli-color como já vimos anteriormente.

nossa Atividade ficará dessa forma:

```
$ node index.js
Orange text on dark gray background
Mensagem verde;
Mensagem red;
Mensagem blue;
Mensagem yellow;
red blue red
red plain blue
Underlined red text on white background.
Text in red
Error!
Warning
Notice
First Name | Last Name | Age
John       | Doe       | 34
Martha     | Smith    | 20
Jan        | Kowalski  | 30
```

Primeiro passo:

Crie uma pasta com nome 'ManipulandoCoresConsole'

e vamos instalar o pacote npm com o comando 'npm i', para podermos criar nosso package.json e após isso podermos instalar nossa lib cli-color com o comando:

```
npm install cli-color
```

A documentação pode ser encontrada aqui :

<https://www.npmjs.com/package/cli-color>

basta criarmos um arquivo agora chamado, index.js

e dentro dele iniciarmos a chamada da lib e a sequência de cores, conforme exemplo abaixo:

Usage:

```
var clc = require("cli-color");
```

Output colored text:

```
console.log(clc.red("Text in red"));
```

Styles can be mixed:

```
console.log(clc.red.bgWhite.underline("Underlined red text on white background."));
```

Styled text can be mixed with unstyled:

```
console.log(clc.red("red") + " plain " + clc.blue("blue"));
```

Styled text can be nested:

```
console.log(clc.red("red " + clc.blue("blue") + " red"));
```

Como podem ver essa lib ela faz com que coloquemos cores no console.log de formas diferentes.

Então iniciaremos colocando a chamada da lib e as cores, que irão representar erro no sistema, aviso de segurança(warning) e notificações (notice).

```
var clc = require("cli-color");
var error = clc.red.bold;
var warn = clc.yellow;
var notice = clc.blue;
var clc = require("cli-color");

var msg = clc.xterm(202).bgXterm(236);

console.log(msg("Orange text on dark gray background"));

console.log(clc.green("Mensagem verde;"));
console.log(clc.red("Mensagem red;"));
console.log(clc.blue("Mensagem blue;"));
console.log(clc.yellow("Mensagem yellow;"));
console.log(clc.red("red " + clc.blue("blue") + " red"));
console.log(clc.red("red") + " plain " + clc.blue("blue"));
console.log(clc.red.bgWhite.underline("Underlined red text on white background."));
console.log(clc.red("Text in red"));

console.log(error("Error!"));
console.log(warn("Warning"));
console.log(notice("Notice"));
```

por ultimo iremos fazer uma tabela e deixar informações:

```
process.stdout.write(
  clc.columns([
    [clc.bold("First Name"), clc.bold("Last Name"), clc.bold("Age")],
    ["John", "Doe", 34],
    ["Martha", "Smith", 20],
    ["Jan", "Kowalski", 30]
  ])
);
```



# Unisenai

Ao término da programação em si, basta rodar no terminal integrado do vscode o comando : node index.js

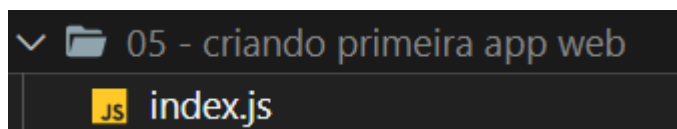
Efetutando esse comando, temos o seguinte resultado:

```
$ node index.js
Orange text on dark gray background
Mensagem verde;
Mensagem red;
Mensagem blue;
Mensagem yellow;
red blue red
red plain blue
Underlined red text on white background.
Text in red
Error!
Warning
Notice
First Name | Last Name | Age
John       | Doe       | 34
Martha     | Smith     | 20
Jan        | Kowalski  | 30
```

Todas as mensagens possíveis no console.log editáveis e configuráveis.

Por fim iremos ver como criar nosso primeiro servidor web com node.JS

Para isso vamos criar uma pasta:



Conforme imagem acima e um arquivo index.js.

Dentro iremos chamar o método http que já existe de forma nativa dentro do node.JS e criar sua configuração:

```
1 // Nesta aula veremos como proceder para criar nossa primeira aplicação web com Node.js. U
2 // Para isso, utilizaremos o pacote HTTP para criar um servidor e acessá-lo por meio do browser.
3 var http = require("http");
4
5 http.createServer(function(request, response){
6     response.write("Primeiros passos com Node.JS");
7     response.end();
8 }).listen(8081);
9
10 console.log("Servidor rodando em http://localhost:8081");
```

Os comandos response.write e response.end servem para pontuar uma escrita na página web carregada sem precisar criar um arquivo html algo do tipo.

realizando essa configuração no arquivo index.js basta rodar o comando dentro da pasta : node index.js

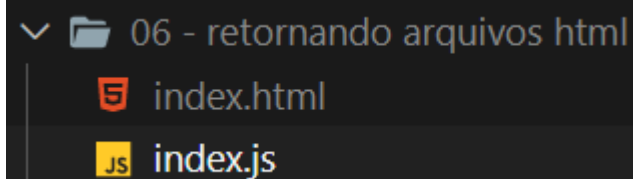
se tudo tiver corrido corretamente irá mostrar no console a seguinte msg:

```
$ node index.js
Servidor rodando em http://localhost:8081
```

e basta acessarmos o caminho indicado acima <http://localhost:8081> e vermos o servidor embutido.

## Primeiros passos com Node.JS

Agora vamos criar um projeto de servidor carregando um arquivo index.html fazemos um clone do projeto anterior para modificarmos algumas coisas:



teremos esta estrutura , por gentileza criem igual a imagem acima.

no arquivo index.js teremos a seguinte configuração:

```
var http = require("http");
var fs = require("fs");

http.createServer(function(request, response){
  fs.readFile("index.html", function(err, conteudo){
    if(err){
      console.log(err);
    }else {
      response.write(conteudo);
    }
    response.end();
  })
}).listen(8081);

console.log("Servidor rodando em http://localhost:8081");
```

Observem que agora chamamos o método fs que também é uma lib nativa do node.JS e já vem quando instalamos ele, ela será responsável como sabemos por ler documentos, arquivos, e etc.

Com o comando fs.readFile eu consigo realizar a leitura do arquivo que eu colocar entre as aspas da função e logo na sequência realizou uma tratativa de erro para controlar e ter certeza de que o conteúdo será exibido corretamente. e no final coloco para ele ser listado na porta 8081 e chamo um console.log para observar. Agora vamos criar o arquivo index.html, o mesmo terá a seguinte estrutura:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Olá Mundo NodeJS</title>
</head>
<body>

  <h1>Olá sou o arquivo html rodando com o NodeJS</h1>
</body>
</html>
```

Feito a criação do mesmo basta agora na pasta do projeto rodarmos o comando `node index.js` e vamos ver o que acontece?

```
$ node index.js
Servidor rodando em http://localhost:8081
```

Ao rodarmos aparece isso no console e ao abrirmos o link olhem só:



## Olá sou o arquivo html rodando com o NodeJS

para que possamos editar esse arquivo precisamos parar o comando `node index.js` no terminal use o atalho `ctrl + c`, isso será suficiente para parar o servidor e altere as informações no `index.html` para:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Olá Mundo NodeJS</title>
</head>
<body>

  <h1>Olá sou o arquivo html rodando com o NodeJS</h1>
  <p>Aprendendo NODEJS UNISENAI</p>
  <p>Com Professor Uchôa</p>
</body>
</html>
```

o mesmo ao rodar o comando novamente `node index.js` tem de ficar assim :

## Olá sou o arquivo html rodando com o NodeJS

Aprendendo NODEJS UNISENAI

Com Professor Uchôa

Ao término, enviar para o github de aulas de NodeJS que ja temos o caminho é este:

<https://github.com/uchoamaster/SENAI-NODE-JS>

### Procedimentos para postar no repositório:

**Passo 1:** realizar o fork do repositório;

**Passo 2:** certificar de que nao tem mais nada desatualizado no repositório com comando “ git pull”;

**Passo 3:** criar uma pasta com seu nome e postar seus projetos dentro dela cada um com a data correspondente, por exemplo: ead\_26\_08\_23 representando o ead desta semana.

**Passo 4:** realizar o commit e o push para o repositório;

**Passo 5:** Criar uma PR ( PULL REQUEST) pelo Github para que eu possa mergegear seus arquivos com a branche Principal e validar suas atividades.