

Conteúdo

Módulo 1: Introdução ao Node.JS

- Introdução
- O que é Node.JS?
- A História do Node.JS
- Instalação do Node.JS
- Javascript Runtime e V8
- Entendendo o Motor V8
- Características do Node.JS
- Módulos
- Node Package Manager - NPM
- Certificações
- Utilização no Mercado
- Criando seu primeiro Programa em Node.JS
- Atividade extra - Calculadora Node.JS
- Conclusão

INTRODUÇÃO

Na atualidade, a criação de aplicações tem como foco arquiteturas que sejam escaláveis e na entrega de soluções em tempo real, além da atenção à componentização e segurança.

Além disso, soma-se a esse cenário a revolução iniciada pelos smartphones, com o uso cada vez mais intenso das mídias sociais e o aumento de soluções de IoT (Internet das Coisas). Nesse contexto, os paradigmas conhecidos no desenvolvimento de aplicações também têm passado por diversas mudanças que vão do Front-end ao Back-end, onde pensamos cada vez mais em uma solução como um todo, levando em consideração o consumo de dados e a disponibilidade de infraestrutura.

E é nessa conjuntura que nasce o Node.js, surgindo como uma solução poderosa e barata para a criação e a manutenção de ambientes de tecnologia com altas demandas. Então, vamos conhecer um pouco sobre essa ferramenta?

O Back-end é a parte do software responsável por lidar com os dados e com o processamento desses dados. Back-end e Front-end não estão no mesmo local.

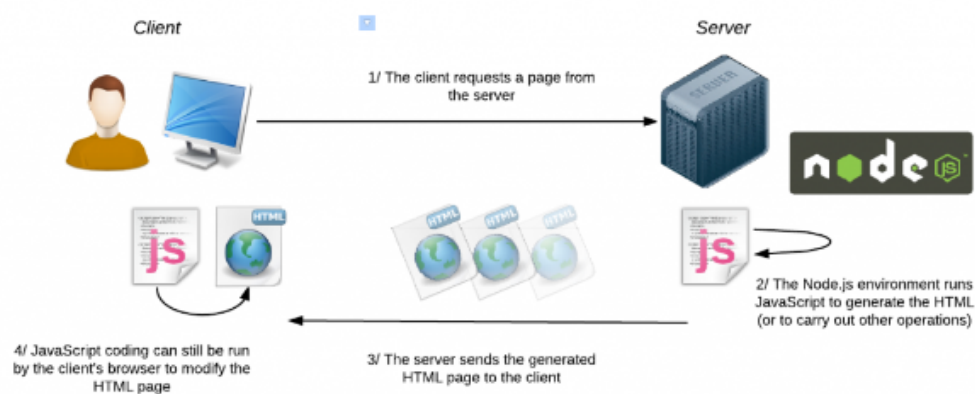
Quando falamos do Front-end estamos falando da parte da aplicação que fica com o usuário - seja em seu smartphone ou no seu computador.

O Back-end é armazenado e executado em um servidor, um computador central. É assim que ele consegue fornecer informações em tempo real.

O QUE É NODEJS?

O Node.js é uma plataforma server-side e com ele é possível criar aplicações JavaScript standalone, ou seja, que não dependem de um navegador para a execução, como estamos acostumados.

O JavaScript figura hoje como uma das linguagens mais utilizadas, e em grande parte isso se deve ao fato de ser uma linguagem base para dezenas de frameworks com alta popularidade e adesão na comunidade de desenvolvimento.



Por se tratar de uma linguagem popularmente conhecida para a construção de aplicações web mais interativas, o JavaScript possui grande foco no Front-end (client side), ou seja, é comumente utilizada para rodar no “lado cliente” da aplicação. Com a evolução das tecnologias web, tornou-se possível fazer o JavaScript rodar também no Back-end, e é nesse momento de consolidação de tecnologias e soluções que surge o Node.js.

Mas, afinal, o que é o Node.js? O Node.js é um ambiente de execução do código JavaScript do lado servidor (server side), que na prática se reflete na possibilidade de criar aplicações standalone (autossuficientes) em uma máquina servidora, sem a necessidade do navegador.

RELAÇÃO ENTRE NODEJS E BACK-END

Ser um programador Node.js significa ser um programador back-end, ou seja, você vai implementar o código da estrutura que forma a base de um site ou aplicativo.

Programar em Node.js pode te levar a duas carreiras, PROGRAMADOR BACK END NODE JS e PROGRAMADOR FULL-STACK(Profissional que domina o Nodejs e também alguma tecnologia frontend como Angular, ReactJS ou Vue.

A HISTÓRIA DO NODE.JS

O Node.js nasceu em 2009 como uma resposta às tentativas de rodar códigos JavaScript em modo server side, uma vez que a linguagem tinha como meta principal a manipulação do DOM (Document Object Model) e deixar as aplicações web mais interativas e dinâmicas.

Soma-se a isso o fato de que pessoas desenvolvedoras de JavaScript trabalham com uma linguagem simples, interpretada e que não necessita da instalação de ferramentas complexas de desenvolvimento.

Esses são alguns dos fatores motivadores do criador do projeto do Node.js, o engenheiro de software Ryan Dahl, responsável por esse ambiente de execução do código JavaScript fora do navegador, no lado servidor.

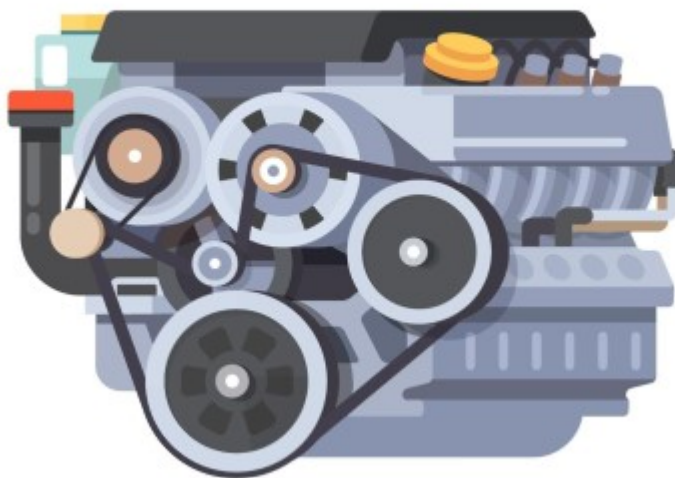
COMO INSTALAR O NODE.JS ?

O Node.js é uma ferramenta de código aberto e gratuita. Um belo atrativo, não?! Ela também é multiplataforma, o que nos permite entregar uma solução para rodar em ambientes Windows, Linux ou MacOS, e seu interpretador é baseado no V8 da Google (vamos conferir mais detalhes sobre ele a seguir).

Temos um excelente conteúdo mostrando em detalhes a instalação do Node.js em nossa **OT 02** então não se preocupem no momento, por hora .

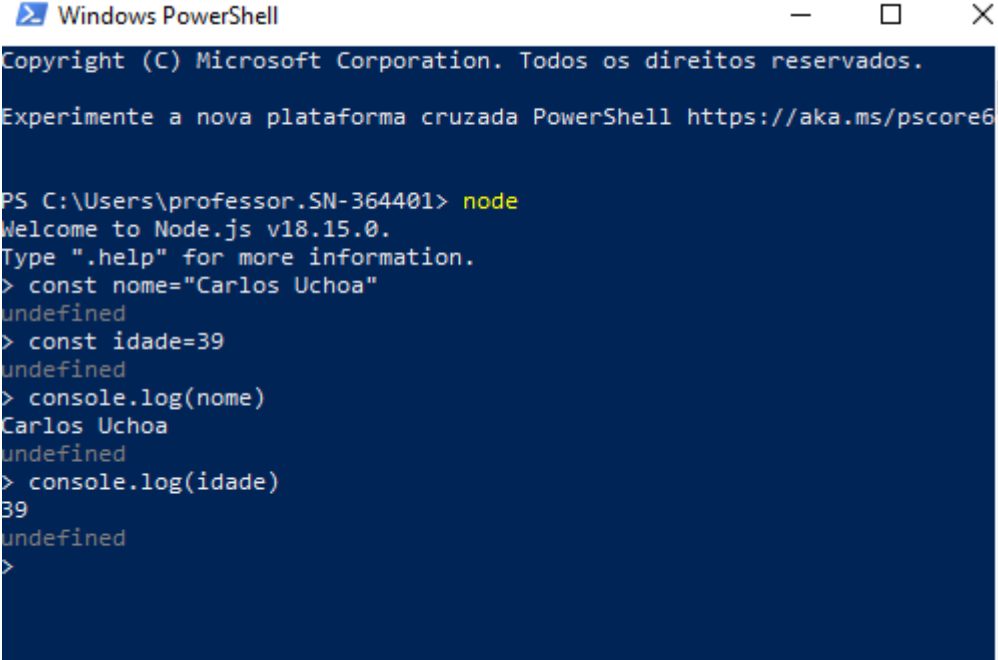
Recomendamos a leitura deste material para que você entenda como instalar e explorar as diferentes versões do Node.js.

JAVASCRIPT RUNTIME E MOTOR V8



O Node.js é a ferramenta que vai nos entregar a capacidade de interpretar código JavaScript, de maneira bem similar ao navegador. Quando executamos um comando escrito em JavaScript, o Node.js interpreta esse comando e faz a sua conversão para a linguagem de máquina a ser executada pelo computador. Por esse motivo, o Node.js também pode ser referido como um JavaScript Runtime, ou um programa de execução do JavaScript.

Observe a sequência de comandos JavaScript no terminal exibido abaixo:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.
Experimente a nova plataforma cruzada PowerShell https://aka.ms/powershell

PS C:\Users\professor.SN-364401> node
Welcome to Node.js v18.15.0.
Type ".help" for more information.
> const nome="Carlos Uchoa"
undefined
> const idade=39
undefined
> console.log(nome)
Carlos Uchoa
undefined
> console.log(idade)
39
undefined
>
```

Quando iniciamos o Node.js com o comando “**node**”, iniciamos um processo que engloba um interpretador JavaScript e um utilitário CLI (command-line interface), e é neste processo aberto no terminal que irá acontecer a interpretação e execução do JavaScript runtime. Para esse processo de interpretação, o Node.js faz uso do V8, mais precisamente conhecido como Chrome’s V8 JavaScript engine.

O V8 é um poderosíssimo interpretador JavaScript desenvolvido pela Google e utilizado pelo Chrome. Ele também é conhecido como a máquina virtual do JavaScript. Foi desenvolvido usando a linguagem C++, é de código aberto e nasceu com a intenção de acelerar a execução de aplicações desenvolvidas em JavaScript.

Por fornecer uma boa performance, várias plataformas têm adotado o Node.js como um solução viável e eficaz de tecnologia para Back-end, como o próprio Google, Netflix, entre outros.

ENTENDENDO O MOTOR

É interessante apontar que existem outros “motores” para JavaScript, como o **SpiderMonkey**, do Firefox, ou o **WebKit**, do Safari, mas o adotado pelo Node.js é o V8. De forma geral, o processo de funcionamento desses motores pode ser resumido nas seguintes etapas:

O motor acessa o código JavaScript, que veio da leitura da tag “**script**” de um arquivo HTML ou por meio do download de um arquivo JavaScript;

Executa o “**Parser**”, processo de análise e conversão;

Monta a **AST** (Árvore de Sintaxe Abstrata), um mapeamento que identifica o que cada parte do código está fazendo;

E, por fim, interpreta a AST. Nesta etapa, ao interpretar a AST, o motor V8 gera os bytecodes a serem executados pela máquina. Neste processo, ainda existe um estágio de otimização do código gerado.

Imagine a escrita da seguinte função em Javascript:

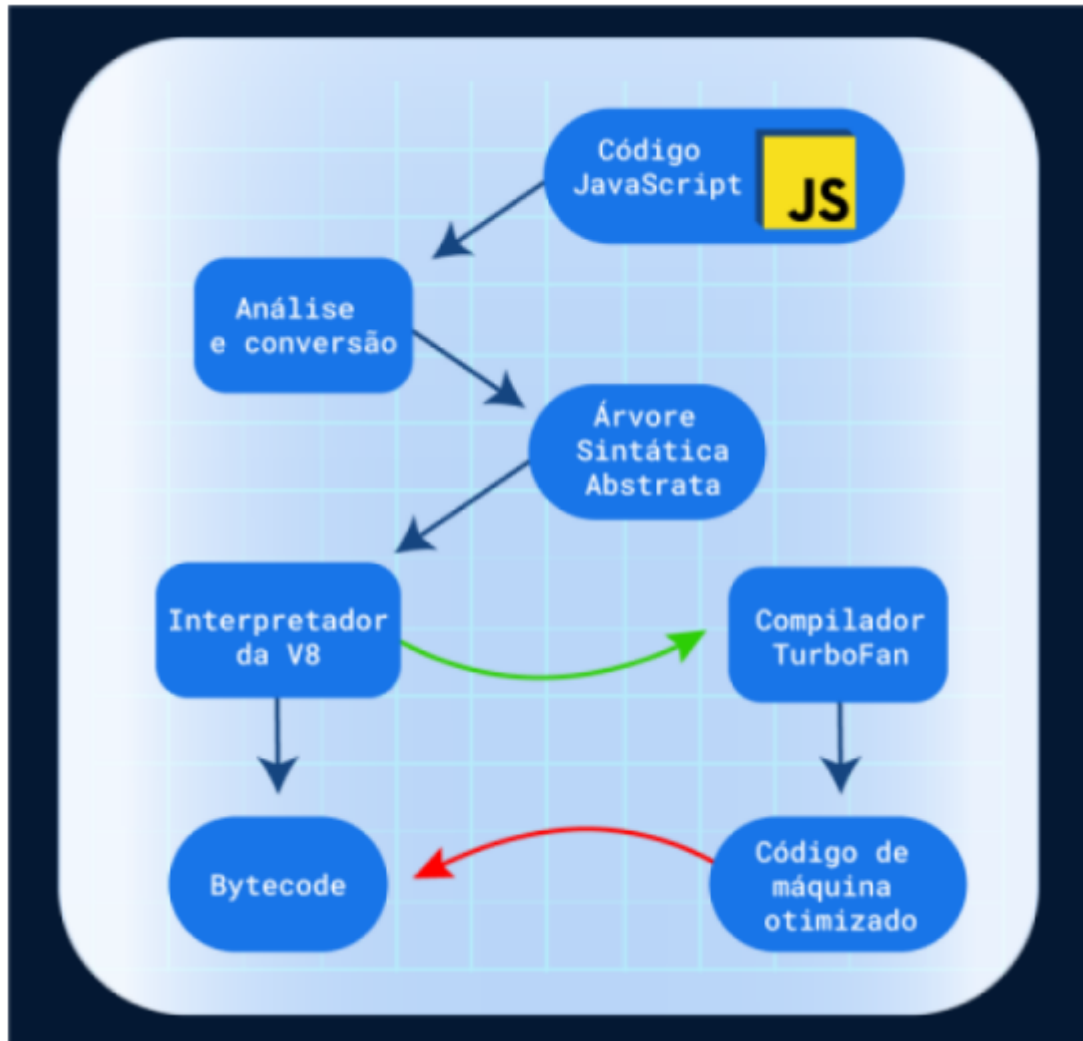


```
function soma(num1,num2) {  
  return num1 + num2;  
}
```

Nesse processo de análise e conversão teremos a seguinte AST:

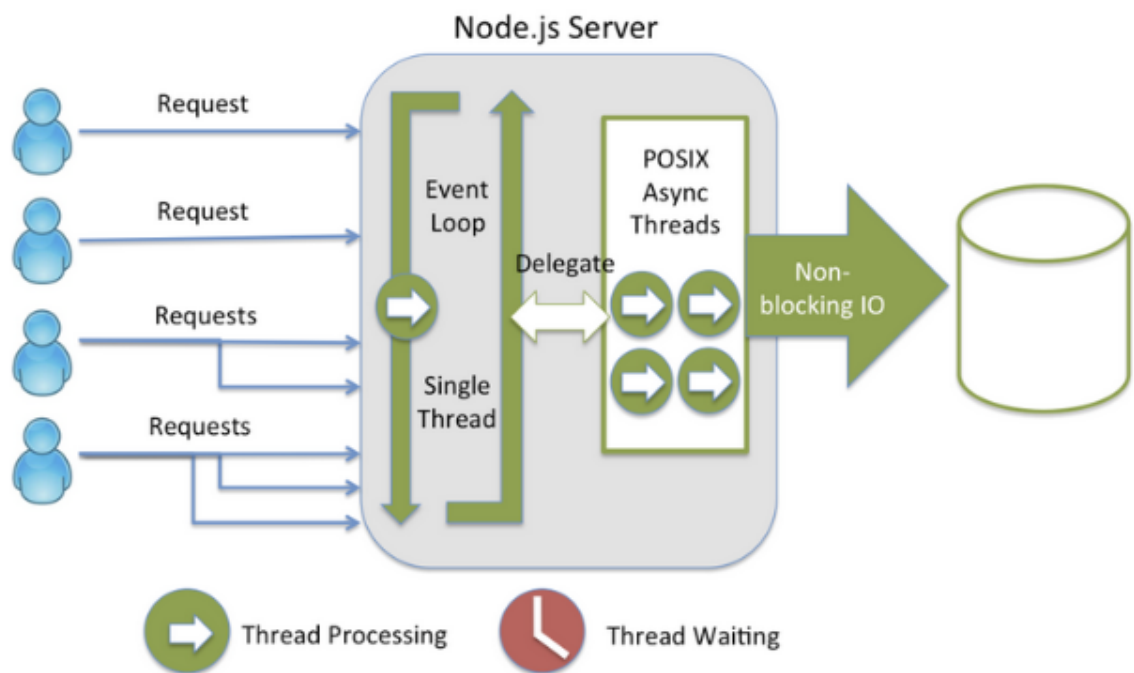
```
- Program {
  type: "Program"
  start: 0
  end: 50
  - body: [
    - FunctionDeclaration {
      type: "FunctionDeclaration"
      start: 0
      end: 50
      + id: Identifier {type, start, end, name}
      expression: false
      generator: false
      async: false
      + params: [2 elements]
      - body: BlockStatement {
        type: "BlockStatement"
        start: 25
        end: 50
        - body: [
          + ReturnStatement {type, start, end, argument}
        ]
      }
    }
  ]
  sourceType: "module"
}
```

Na imagem abaixo, temos uma abstração do processo de execução do código JavaScript pelo motor V8:



Com execução dessa sequência de processos apresentada no esquema acima, o nosso motor Javascript interpreta e compila o código para um bytecode otimizado para executar no computador.

ASSÍNCRONO, SINGLE THREAD E ORIENTADO A EVENTOS



Uma das características do Node.js é ser um ambiente de execução assíncrono. Com isso, ele trabalha de forma a não bloquear a aplicação no momento de sua execução, colocando os processos mais demorados para um segundo plano.

O Node.js se diferencia de outras plataformas consagradas de programação, como o Java, PHP e .NET, pelo fato de ser single thread, ou seja, o Node.js não inicia threads em paralelo como outras plataformas. Por se tratar de um sistema single thread, o Node.js não tem a necessidade do gerenciamento de múltiplas threads, otimizando, assim, o processo e o consumo de memória da aplicação.

A característica do Node.js que faz com ele não seja lento ou demore a processar a fila de requisição é ser não bloqueante; isso tem a ver com o sistema de callbacks do JavaScript e o loop de eventos. Para saber mais sobre esses recursos, confira o artigo Arquitetura do Node.js: entenda o loop de eventos.

CARACTERÍSTICAS DO NODE.JS

No decorrer do artigo já falamos de algumas características do Node.js, mas vale ressaltar que ele pode ser usado para:

Desenvolver soluções em API Rest;

Criação de chatbots;

Projetos de internet das coisas (IoT);e

Dar vida a soluções de web scraping, web servers e até mesmo aplicações desktop.

O Node.js possui uma versatilidade gigantesca, algumas outras características que não podemos deixar de comentar:

Escalabilidade: o Node.js foi pensando desde o início para entregar soluções para arquiteturas escaláveis, por isso sua adoção por grandes empresas de tecnologias;

Multiplataforma: podemos criar desde soluções web a aplicações desktop;

Open Source: como já citado, o Node.js tem seu código aberto, o que nos possibilita contribuir com o projeto e até mesmo realizar customizações;

Multi-paradigma: com o JavaScript podemos adotar várias formas de codificar usando paradigmas como a orientação a objetos, o funcional e o orientado a eventos.

Para se trabalhar com o Node.js é preciso conhecer as bases do JavaScript, que é uma linguagem com grande receptividade, principalmente para os iniciantes, pois possui uma curva de aprendizagem mais suave em comparação a outras linguagens.

MÓDULOS

O Node.js trabalha fortemente com o conceito de módulo, que visa organizar o código em partes pequenas e especializadas, além de aplicar o princípio de encapsulamento ao “esconder” o código e expor somente o necessário a outras partes da aplicação.

Originalmente o Node.js trabalhava com o sistema CJS (Common JS) de modularização, pois o JavaScript não tinha ferramentas específicas para isso. Posteriormente foi especificado o padrão “geral” de módulos do JavaScript, o EcmaScript Modules.

O Node.js vem adotando gradativamente o ECMAScript Modules desde a versão 13, o que traz muitas diferenças com relação à forma “original” do Node.js, o CJS. Para saber mais sobre esse tema, recomendamos a leitura do artigo [Um guia para importação e exportação de módulos com JavaScript](#).

USANDO O NPM



No processo de instalação do Node.js nos é entregue também o NPM (Node Package Manager), que vai nos permitir compartilhar os módulos que criamos com a comunidade além de possibilitar a reutilização de módulos criados por outras pessoas desenvolvedoras.

Para verificar a instalação e versão do NPM você pode executar o comando:

npm -v

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\professor.SN-364401> npm -v
9.6.7
PS C:\Users\professor.SN-364401>
```

Através do NPM podemos gerenciar todas dependências dos projetos JavaScript que estamos desenvolvendo, com ele temos acesso a uma gama enorme de bibliotecas e frameworks JS, e possibilita também a criarmos nossos próprios módulos e compartilhar com a comunidade.

CERTIFICAÇÕES

As certificações são sempre um ponto de muita discussão dentro das comunidades de tecnologia, mas é consenso que elas ajudam profissionais a adquirirem conhecimentos em diferentes áreas.

Além disso, certificações também são uma ótima oportunidade de conseguir mostrar para o mercado que você conhece os por menores de algumas tecnologias ou linguagens. Empresas como Microsoft, IBM, Google, Oracle e alguns institutos ou fundações promovem certificações a profissionais de tecnologia.

Mas existem certificações em Node.js? E a resposta é, sim! Através da OpenJS Foundation é possível conseguir certificações em Node.js, a fundação oferece duas opções de certificações para desenvolvedores. São elas:

OpenJS Node.js Services Developer (JSNSD);

OpenJS Node.js Application Developer (JSNAD).

1) OpenJS Node.js Services Developer (JSNSD)

Certifica a pessoa como competente na **criação de API Rest e Serviços com atenção especial à implementação de segurança.**



2) OpenJS Node.js Application Developer (JSNAD)

Esta certificação é para pessoas desenvolvedoras com capacidade de **criar aplicativos web** com Node.js.



SUA UTILIZAÇÃO NO MERCADO

Fica a pergunta: “E no mercado onde existe o Node.js rodando?”. Por se tratar de uma tecnologia flexível e escalável, hoje grandes empresas utilizam o Node.js como parte de sua stack. Entre elas, podemos citar:

- **GoDaddy**: que migrou e implementou diversos projetos de frontend e backend em Node.js, inclusive abandonando tecnologias .NET em prol do Node.js;
- **Netflix**: este é um caso bem emblemático por se tratar da maior plataforma de streaming da atualidade. Substituiu o Java pelo JavaScript para melhorar a performance de suas requisições;
- **NASA**: A engenharia da NASA desenvolveu uma API com Node.js para trabalhar a integração de bases de dados em um serviço de cloud;
- **Slack**: conhecida plataforma de comunicação pessoal ou profissional, roda suas aplicações e desenvolveu seu cliente de desktop com Node.js;
- **Uber**: a aplicação que conecta usuários e condutores foi construída com base no Node.js.

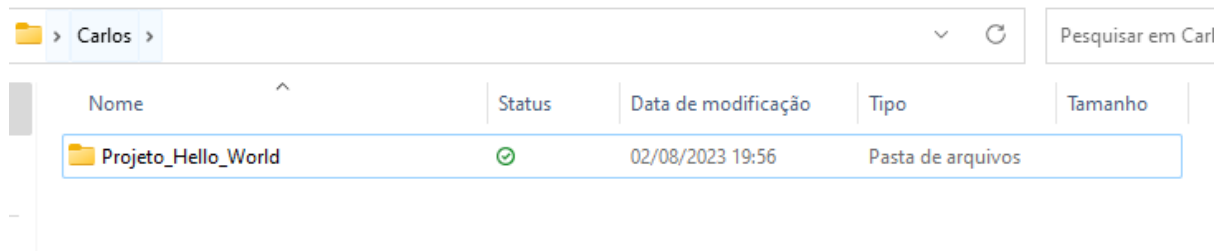
Podemos citar diversas startups e até gigantes da tecnologia como IBM que possuem soluções desenvolvidas com o Node.js. É uma tecnologia em grande expansão, que tem aberto inúmeras oportunidades de trabalho no Brasil e exterior.



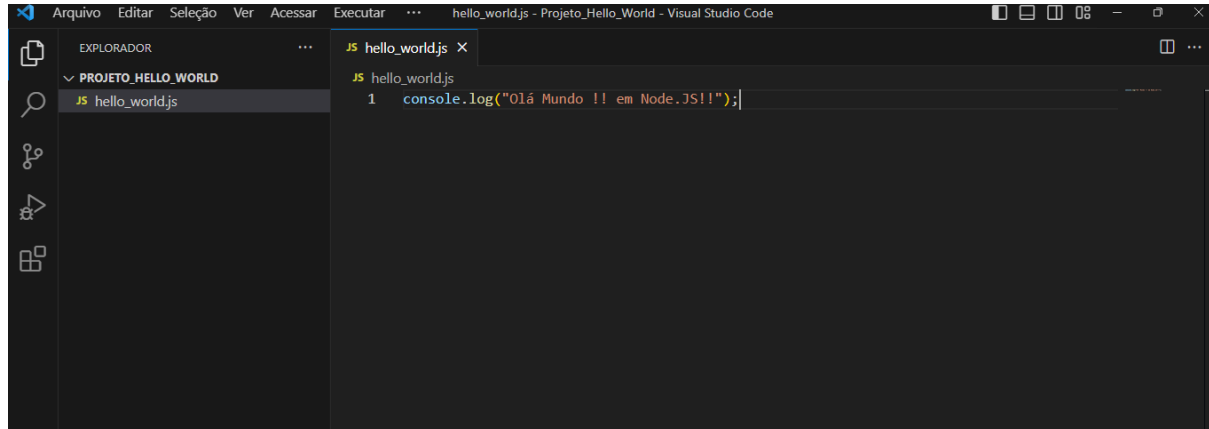
Node.js forma a base de muitos frameworks populares atualmente. Nesta aula tivemos uma visão geral desta tecnologia e vimos sua relação com o back-end.

Passo 01:

Crie um arquivo chamado `hello_world.js` na pasta do seu projeto, crie uma pasta com seu nome e dentro da pasta de seu nome crie uma pasta com o nome “Projeto_Hello_World”



Em seguida, abra esse arquivo em um editor de código, como , por exemplo, o VS Code. Insira o código `console.log("Olá Mundo !! em Node.JS!!");` nele e salve o arquivo.



Abra o seu terminal de comando e navegue até a pasta onde o arquivo está.

Agora, digite `node hello_world.js`.

Como podemos verificar na execução do comando o projeto foi executado com sucesso.

Unisenai

```
JS hello_world.js X
JS hello_world.js
1 console.log("Olá Mundo !! em Node.JS!!");

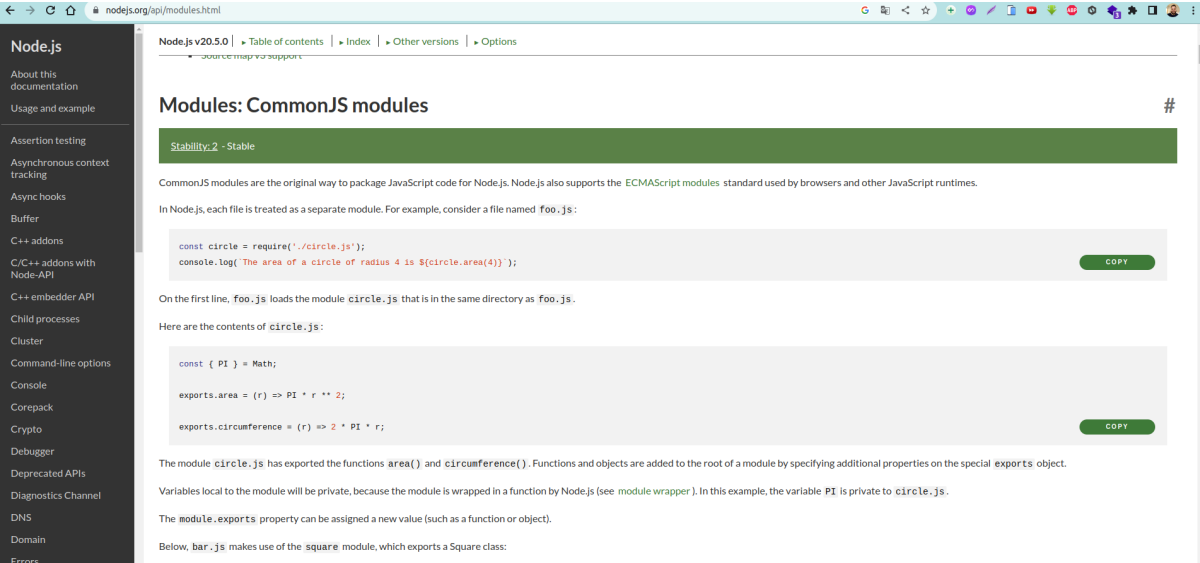
PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL

lenni@DESKTOP-VI20DSV MINGW64 ~/OneDrive/Área de Trabalho/Car...
$ node hello_world.js
Olá Mundo !! em Node.JS!!
```

Muito bem você agora aprendeu a criar seu primeiro projeto em Node.JS, agora iremos criar outro projeto um pouco mais robusto, mostrando algumas novidades que o Node.JS nos proporciona, onde iremos trabalhar com CommonJS Modules.

Para entender como funciona um pouco a sintaxe está na documentação do Node.JS:

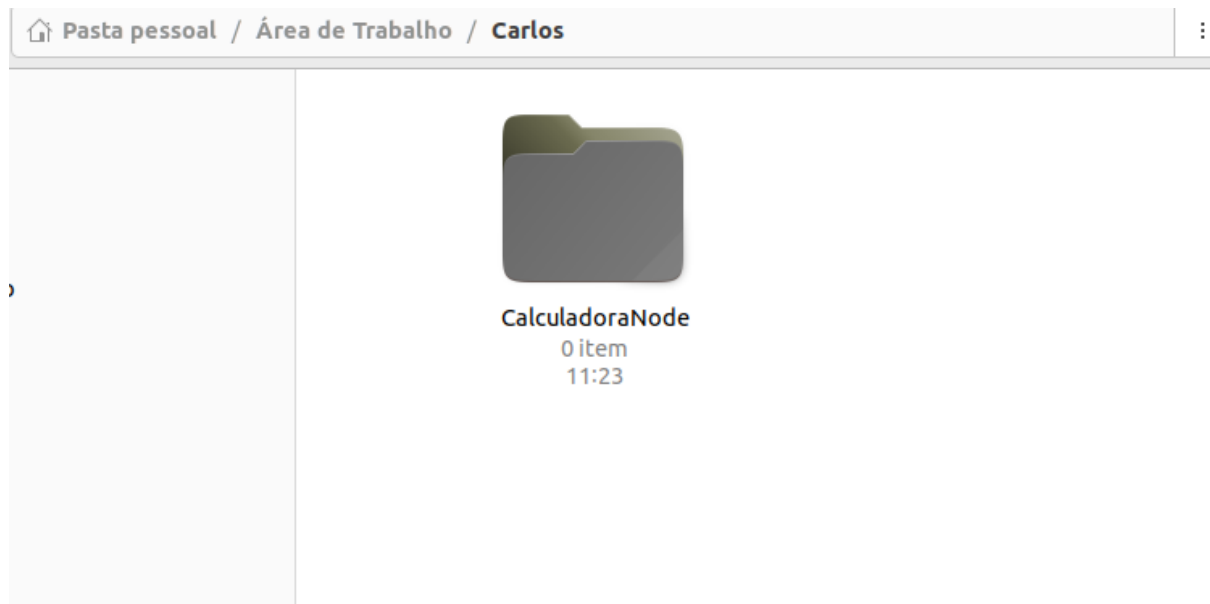
<https://nodejs.org/api/modules.html>



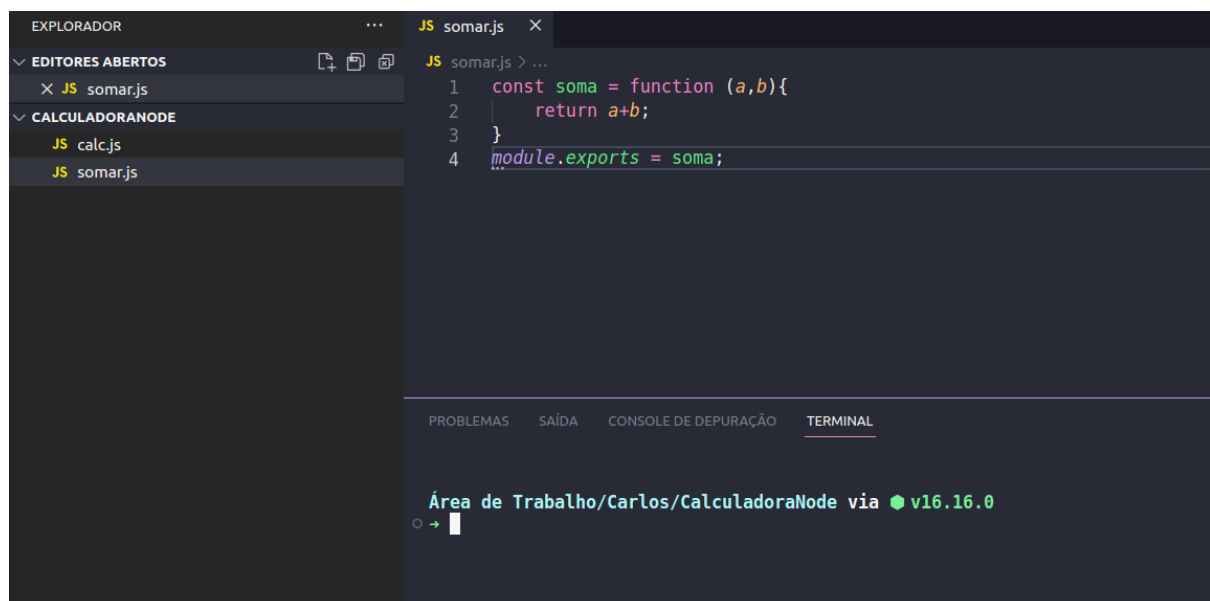
The screenshot shows the Node.js documentation page for CommonJS modules. The page title is "Modules: CommonJS modules". It includes a sidebar with navigation links for various Node.js topics. The main content area explains that CommonJS modules are the original way to package JavaScript code for Node.js and that Node.js also supports the ECMAScript modules standard. It provides an example of how to use the `require` function to load a module and how to export functions and objects using the `exports` object. The example code shows a module named `circle.js` that exports `area` and `circumference` functions, and a file `foo.js` that uses `require` to load `circle.js` and calls the `area` function. The page also mentions that variables local to the module are private and that the `module.exports` property can be assigned a new value.

UnISENAI

Então vamos lá, primeiro passo iremos criar dentro de nossa pasta com nosso nome, mais uma pasta do novo projeto chamado “**CalculadoraNode**”.



Após a criação da pasta abra ela com o **Vscode** e vamos dar início a criação de nossa calculadora com Node, **crie o primeiro arquivo chamado somar.js** e o seguinte código dentro :

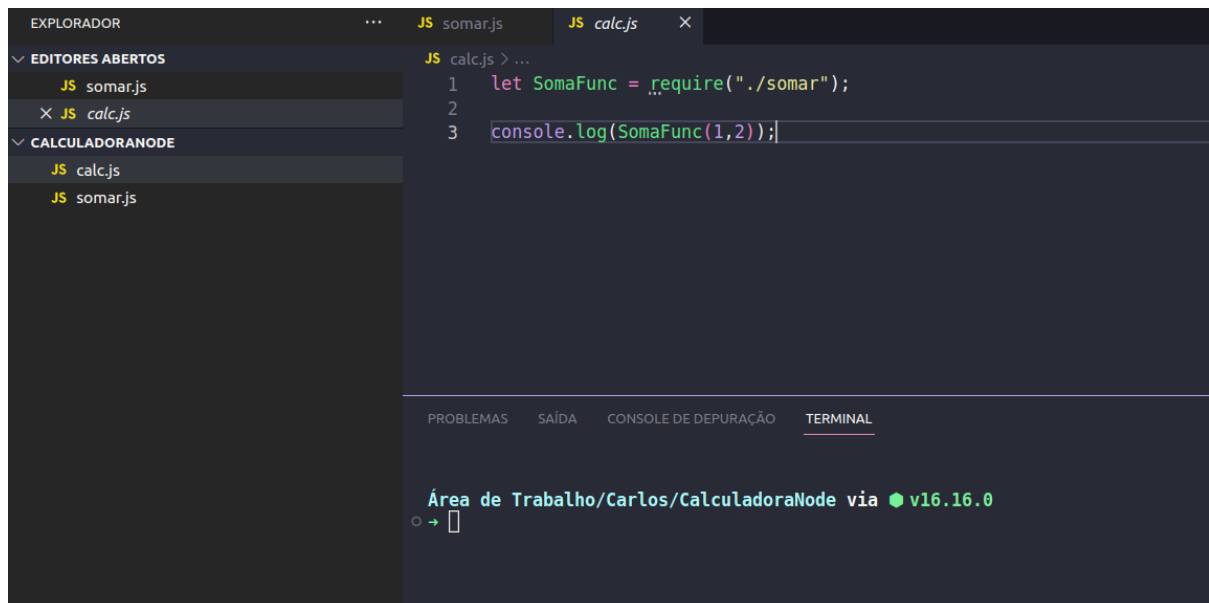


```
JS somar.js X
JS somar.js > ...
1  const soma = function (a,b){
2    return a+b;
3  }
4  module.exports = soma;
```

Notem que estou exportando a **function soma** que criamos para poder utilizar ela em outra página, desta forma conseguimos trabalhar com várias funções em outras telas com javascript no back-end.

Então iremos criar um outro arquivo chamado `calc.js`, onde será responsável por realizar todos os cálculos matemáticos, ou seja, iremos chamar todas as funções criadas.

No arquivo **calc.js**, iremos chamar a função soma através do comando **require** e iremos armazená-la em uma variável e rodar o comando `console.log`, passando 2 valores por parâmetros que irão realizar a soma já que criamos a lógica no arquivo **somar.js** onde realiza a soma de 2 números.



Para testarmos com Node.JS se está tudo funcionando basta rodarmos o comando :

node calc.js e verificarmos se está rodando o cálculo matemático de soma:

Unisenai

```
JS somar.js JS calc.js X
JS calc.js > ...
1 let SomaFunc = require("./somar");
2
3 console.log(SomaFunc(1,2));
```

```
Área de Trabalho/Carlos/CalculadoraNode via v16.16.0
● → node calc.js
3

Área de Trabalho/Carlos/CalculadoraNode via v16.16.0
○ →
```

Como podem verificar o valor deu 3, pois passamos na função **SomaFunc**, que chama a **function somar.js**, passando por parâmetro os valores 1 e 2 resultando no valor 3 e concluindo a soma.

Podemos dar uma melhoria nessa informação assim utilizando assim template strings e iterando variáveis e functions:

```
JS somar.js JS calc.js X
JS calc.js > ...
1 let SomaFunc = require("./somar");
2
3 console.log(`0 valor da soma é: ${SomaFunc(1,2)}`);
```

```
Área de Trabalho/Carlos/CalculadoraNode via v16.16.0
● → node calc.js
0 valor da soma é: 3
```

Agora vamos realizar os demais cálculos de matemática básica, sua atividade é criar os arquivos **divisao.js**, **subtracao.js**, **multiplicacao.js** e chamá-los no arquivo **cal.js** com seus respectivos **console.log** dando o resultado de cada operação, chegando em algo parecido a isso:

```
Área de Trabalho/Carlos/CalculadoraNode via v16.16.0
● → node calc.js
0 valor da soma é: 3
0 valor da divisão é: 5
0 valor da multiplicação é: 20
0 valor da subtração é: 8
```

Ao término desta OT, vamos subir todos os projetos em um novo repositório no Github criado para validar a atividade, irei colocar o link no Ava.

CONCLUSÃO

O JavaScript é uma linguagem muito popular e nos últimos anos vimos nascer vários frameworks que a utilizam. Com o Node.js é possível usar todo o poder do JavaScript também no backend, além de chatbots, IoT, desenvolvimento de aplicações desktop e até mesmo IA.

Para as pessoas desenvolvedoras, independente da linguagem, hoje, já é um diferencial ter conhecimentos em Node.js.