



Introdução ao Node.js



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >
Coding School



Temas

1

Node.js

2

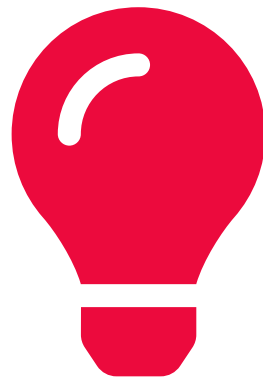
NPM



1 | Node.js



É um **ambiente** de
execução, ou seja, nos
permite executar **Javascript**
fora de um **navegador**.





Arquitetura NODE JS

Todos os navegadores possuem um **mecanismo Javascript** para ler e renderizar código JS. Isso torna a linguagem dependente de um **navegador** para ser executado.

Os próprios navegadores usam motores diferentes. Por conta desta variedade, é que às vezes o mesmo código JS pode se comportar de forma diferente, dependendo do navegador em que ele está sendo executado.



Motor
CHAKRA



Motor
SpiderMonkey



Motor
v8



É um interpretador Javascript

Node.js é um software de código aberto e multiplataforma, que executa códigos JavaScript no backend/servidor e frontend/interface, construído sob o motor (engine) **v8** do Google Chrome. Isso o torna um ambiente de execução Javascript e faz com que a linguagem não dependa mais do navegador para ser executada.

Desta forma, podemos programar tanto o Front-end como o Back-end na mesma linguagem: **JavaScript**. Além disso, é uma ferramenta mantida pela fundação Node.js em parceria com a Linux Foundation.





Instalando o Node JS

A primeira coisa a fazer é baixar o Node.js no seu site oficial: nodejs.org/pt-br
Junto com o Node.js, vamos instalar o gerenciador de pacotes **NPM**, que veremos com profundidade mais tarde.

Para verificar se ele foi instalado corretamente, abra um terminal e execute o comando `node -v` OU `node --version` .

Tenha em mente que, ao instalar o Node, não estamos instalando um software, mas sim um ambiente de execução.





Testando o Node JS

- Para testar o NodeJS, crie uma pasta na Área de Trabalho chamada **Node**.
- Abra o editor de código Visual Studio Code. Vá até o menu **File > Open Folder** e selecione a pasta que acabamos de criar chamada Node.
- Crie nela, um arquivo chamado test.js e escreva o seguinte script dentro dele:

```
console.log('Testando o Node!');
```

- Salve o arquivo e em seguida, abra o terminal. Para fazer isso, vá até **Terminal > New terminal**.
- No terminal, digite o seguinte comando e pressione enter: `node test.js`
- Se tudo correr bem, veremos a seguinte mensagem no terminal:

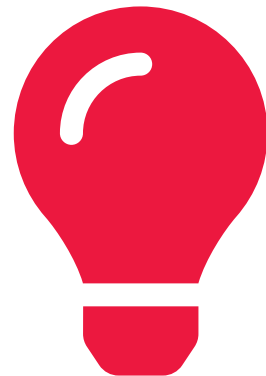
Testando o Node!

2 | NPM



“ O **NPM** é o **gerenciador**
de **pacotes** Node, que nos
permite baixar e **instalar**
bibliotecas para incorporar
aos nossos **projetos**.

”





Introdução ao NPM

Quando instalamos o Node em nossos computadores, várias bibliotecas são instaladas para uso **global**. Uma delas é o **NPM**: Node Package Manager.

Através dele, instalamos as bibliotecas que consideramos necessárias para o funcionamento ou desenvolvimento de nossas aplicações.

Podemos instalá-las **localmente**, para uso em um projeto específico, ou **globalmente**, para utilizarmos em qualquer lugar do nosso computador.



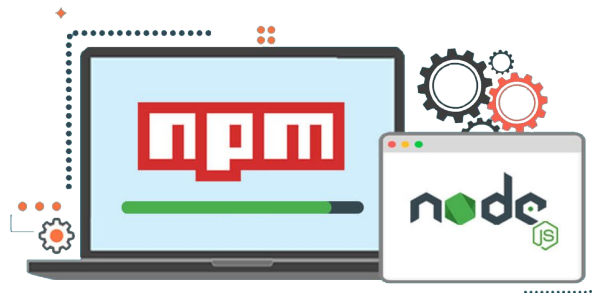


O que são bibliotecas?

São **blocos** de **código** que nos permitem abordar soluções específicas dentro da aplicação que estamos desenvolvendo.

Em desenvolvimento web, há situações que se repetem em vários projetos. Algumas delas são: gerenciar o upload de arquivos, validar um formulário ou restringir o acesso a um usuário que não está registrado.

As bibliotecas vêm para **facilitar** os problemas que sabemos que vamos encontrar ao desenvolver a nossa aplicação.





Usando o NPM

Quando o Node é instalado, é gerado o comando *npm* para uso no terminal.

Para confirmar que a instalação está correta, podemos digitar um dos comandos abaixo no Terminal, que diz qual é a versão do NPM que está instalada em nosso Sistema Operacional.

```
>_ npm -v
```

```
>_ npm --version
```



Usando o NPM

A primeira coisa a fazer para usar o npm é **inicializar** nosso projeto Node, executando o comando:

```
>_ npm init
```

Este comando irá criar um arquivo **package.json**, dentro do qual todas as configurações do projeto serão salvas.

No momento, a propriedade que mais nos interessa neste arquivo é a *main*. Ela refere-se ao **entry point**, ou seja, ao ponto de entrada da nossa **aplicação**, onde colocaremos o nome do nosso arquivo principal, que, por convenção, costumamos chamar *app.js*



package.json

```
{  
  "name": "app",  
  "version": "1.0.0",  
  "description": "",  
  "main": "app.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

Caminho para **entry point** (*ponto de entrada*)



Instalando bibliotecas

Para instalar uma biblioteca, usamos o seguinte comando:

```
>_ npm install PACKAGE --save
```

Onde iremos substituir a palavra *PACKAGE* pelo nome da biblioteca que queremos instalar.

A opção *--save* faz com que a biblioteca fique registrada no **package.json**, na propriedade *dependencies*.





package.json

```
"main": "app.js",  
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1"  
},  
"author": "",  
"license": "ISC",
```

```
  "dependencies" {  
    "moment": "^2.24.0"  
  }  
}
```

Referência à(s)
biblioteca(s) que
instalamos em
nosso projeto.



Dentro da pasta **node_modules** serão criadas as pastas das bibliotecas que instalamos.

Cada pasta conterá os **arquivos necessários** para poder trabalhar com essa biblioteca dentro do projeto.





Atualizando bibliotecas

Quando queremos que um pacote (biblioteca) seja atualizado, precisamos executar o seguinte comando:

```
>_ npm update
```

Desta forma, ele irá atualizar os pacotes listados como dependências em seu arquivo **package.json**, baixando as versões mais atuais para a pasta **node_modules** e removendo as versões anteriores.





Removendo bibliotecas

De vez em quando, podemos perceber que alguma biblioteca instalada não é mais necessária. Para remover, usamos o comando:

```
>_ npm uninstall PACKAGE
```

Este comando vai procurar essa dependência dentro do **package.json**, remover do arquivo e em seguida, apagar os arquivos da biblioteca que estão na pasta **node_modules**.

Perceba que, diferente do comando **update**, aqui é obrigatório avisar qual o nome do pacote a ser removido. Sem esse nome, você receberá um erro na tela.



DigitalHouse>
Coding School