PRIMEIROS PASSOS COM





**SUMÁRIO**

[**O que é Node.js? 3**](#_yi4fckboutrl)

[**V8 JavaScript Engine**](#_4rbjsim77xea)[**3**](#_4rbjsim77xea)

[**Instalando o Node.JS**](#_6dg8alqh3t5n)[**4**](#_6dg8alqh3t5n)

[**Criando um servidor**](#_yab85j9fcxu2)[**5**](#_yab85j9fcxu2)

[**Entendendo o Request e o Response**](#_5nob6916fhnb)[**7**](#_5nob6916fhnb)

[**Melhorando nossa aplicação**](#_osifuhkso5my)[**9**](#_osifuhkso5my)

### 

### 

### O que é Node.js?

Antes de responder esta pergunta temos primeiro que responder o que Node não é. Ele NÃO é uma linguagem de programação muito menos um framework, como jQuery ou AngularJs.

**Node** é uma plataforma para desenvolvimento de aplicações web que roda do lado do servidor (server-side) que utiliza o **Chrome V8 JavaScript Engine**. Foi desenvolvido para construir aplicações rápidas e escaláveis. Sua arquitetura é totalmente non-blocking thread, ou seja, desenvolvido para não bloquear uma thread enquanto aguarda uma resposta, apresentando uma boa performance em consumo de memória e utilizando ao máximo o poder de processamento dos servidores de forma eficiente e eficaz.

Sendo assim fica mais fácil para o programador que é familiarizado com o JavaScript (Front-End), desenvolver a maior parte ou mesmo um aplicativo sozinho utilizando esta linguagem.

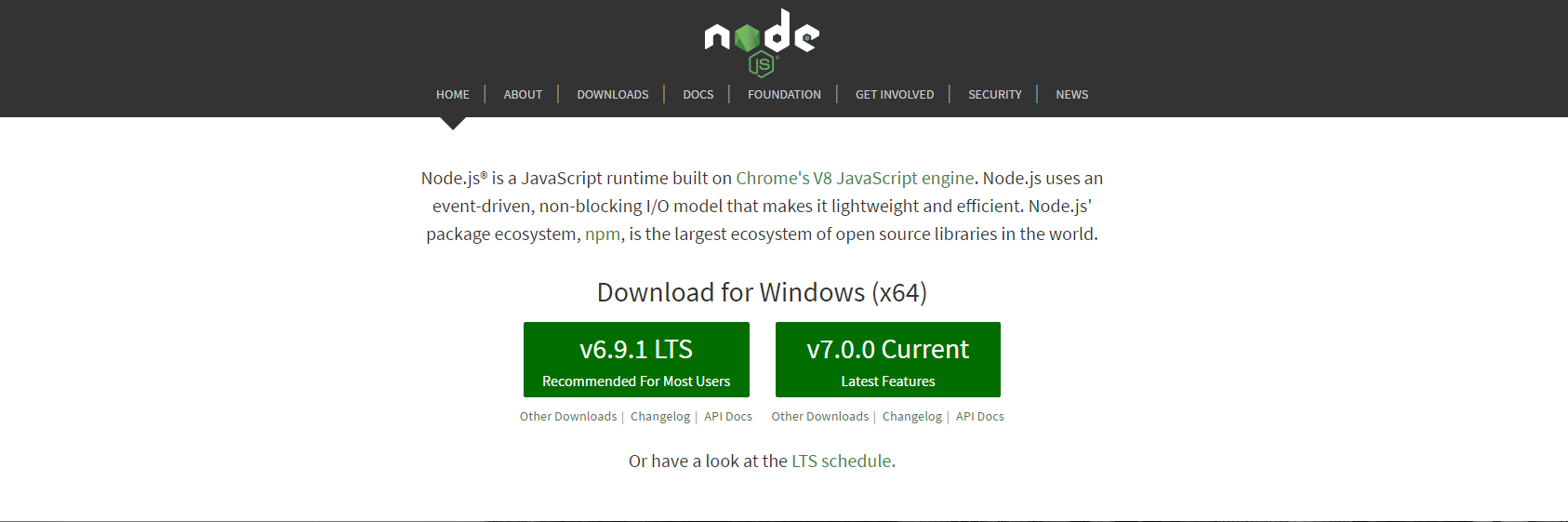
É amplamente usado com Mongodb, Express e AngularJS, formando assim o MEAN.

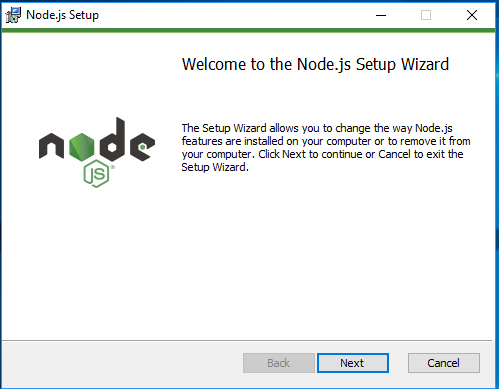
### V8 JavaScript Engine

É o interpretador de JavaScript open source implementado e mantido pelo Google, desenvolvido em C++ e utilizado pelo Chrome.

### Instalando o Node.JS

Para instalar o node é bem fácil e simples, basta fazer o download no site [nodejs.org](https://nodejs.org/en/) e instalar normalmente, não requer nenhuma configuração especial, basta next, next, next e finish.

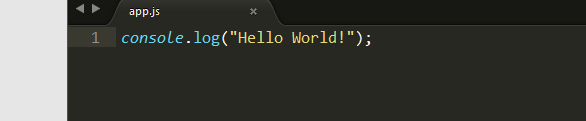




Com o instalador aberto basta clicar em Next, next, next e finish.

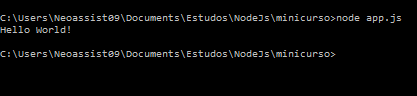
Após instalado vamos abrir um editor de texto, Nodepad ++, Sublime Text, etc e criarmos um arquivo chamado app.js.

Nele basta digitar:



Pelo prompt de comando ir até a pasta que salvou o arquivo e digitar

node app.js



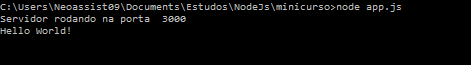
Vimos que assim como o javascript o console no node serve para escrevermos mensagens, só que estas mensagens aparecerão no prompt de comando, uma ótima forma de debugarmos.

### Criando um servidor

Agora que já vimos o básico como rodar um arquivo em node vamos criar um servidor web, para isso utilizaremos um módulo que já vem instalado com o node chamado http.



O resultado será:



A primeira linha criamos uma variável que recebe o valor do módulo http, estamos dizendo para o node que não poderemos trabalhar sem este módulo, então ele é requerido.

Na segunda linha criamos uma variável que receberá o valor da porta que utilizaremos para subir o servidor, que no nosso caso será a porta 3000, mas poderia ser 8080, 8008, 1000, 80, desde que nesta porta não esteja rodando nenhum outro serviço.

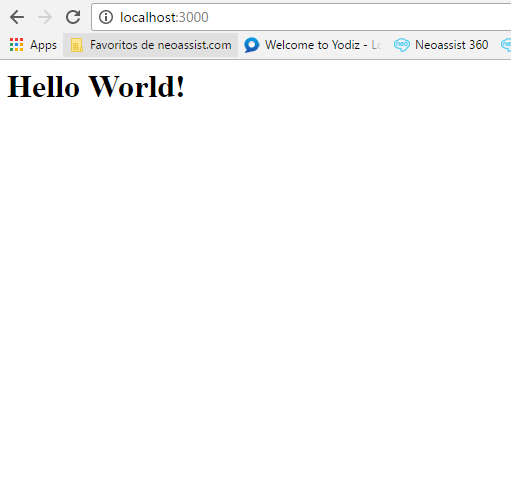
O módulo http tem uma função que se chama createServer, esta função serve para criar um servidor web, e recebe uma função callback que contém dois parâmetros, o primeiro é o request e o segundo o response.

Estes parâmetros são responsáveis pela comunicação entre cliente e servidor.

Agora que vimos que podemos criar um com o node vamos alterar um pouquinho nosso código.



Agora sim, temos algo visual na tela, para testar basta abrir o navegador e digitar: **localhost:3000**



### Entendendo o Request e o Response

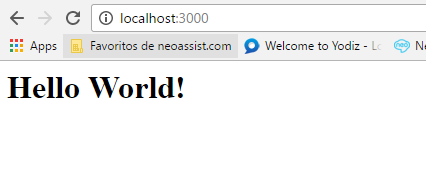
Como dito anteriormente quando criamos um servidor ele recebe uma função de callback que tem dois parâmetros, no exemplo acima utilizamos o response, para escrever na tela.

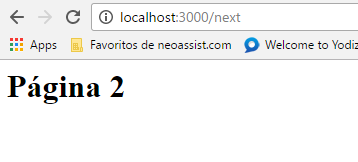
Agora vamos utilizar o **request** para pegar informações que o usuário passará.

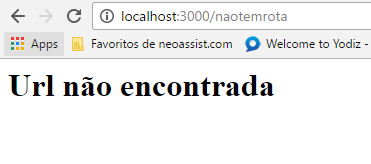
A primeira informação será a url digitada, essa informação será usada para desenvolvermos uma rota da nossa aplicação.



Agora vamos testar a rota.







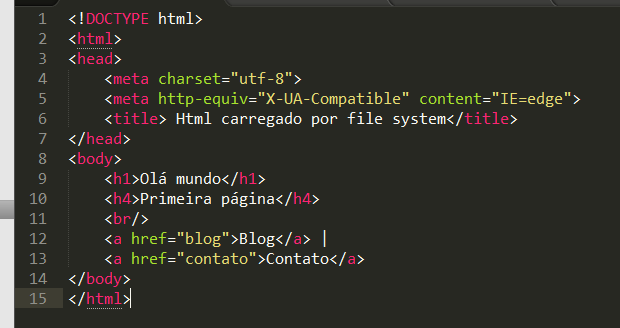
### Melhorando nossa aplicação

Quando formos trabalhar ficaria impossível escrever uma página toda dentro do nosso arquivo de rotas, imagine se tivéssemos várias páginas, cada uma fazendo uma coisa diferente, levando a caminhos diferentes. Para isso criaremos então uma nova aplicação.

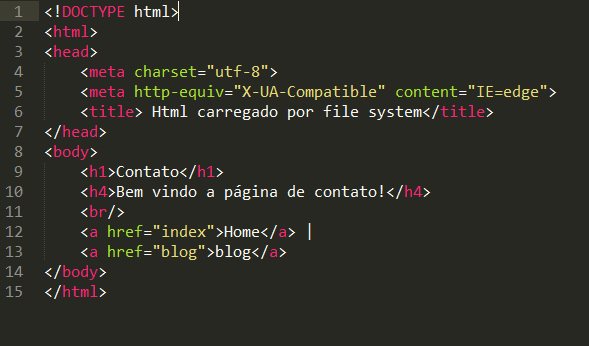
Vamos criar dentro da nossa aplicação uma pasta chamada parte2 e dentro dela duas pastas, uma chamada html e outra log.

Dentro da pasta html vamos criar os seguintes arquivos, index.htm, contato.html, blog.html e erro.html

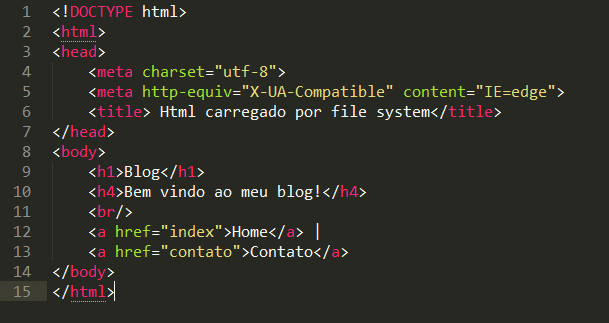
**Index.html**



**contato.html**



**blog.html**



**erro.html**



Com as páginas criadas temos que chamá-las, para isso usaremos 3 módulos nativos do node, que são o **HTTP**, **FS** e **URL**.

**HTTP** servirá para criar o servidor, o **FS** para ler os arquivos, tanto html, quanto o log da aplicação e o **URL**, para ler o caminho e fazer nossa rota.

Nosso arquivo ficará assim:



**Explicando o código:**

As três primeiras linhas estamos falando para o node que iremos utilizar.

Na linha 5 estamos atribuindo a variável path Log o caminho que gravaremos nosso arquivo de log.

Na linha 7 estamos criando uma função vai receber um parâmetro e retornar um caminho html.

Na linha 11 é a nossa função de rota, que receberá um caminho e vai testar se ele existe, retornando a página chamada ou uma página de erro.

Na linha 19 usamos o file system(**fs**) para testar se o arquivo existe, que será usada na função de rota.

Na linha 27 criamos nosso servidor, chamamos as páginas que foram requeridas e criamos o log da aplicação.

A primeira linha dessa função chama nossa função de rota e grava em uma variável para mostrar renderizar no browser.

A segunda linha é a mensagem que será gravada no log.

Na linha 31 usamos o file system para ler a página se ele encontrar a página vai usar o response para renderizar os dados, que no nosso caso é uma página html.

Na linha 35 o fs vai tentar escrever no arquivo de log a mensagem que criamos no início da função.

E por final a chamamos o servidor na porta 3000.