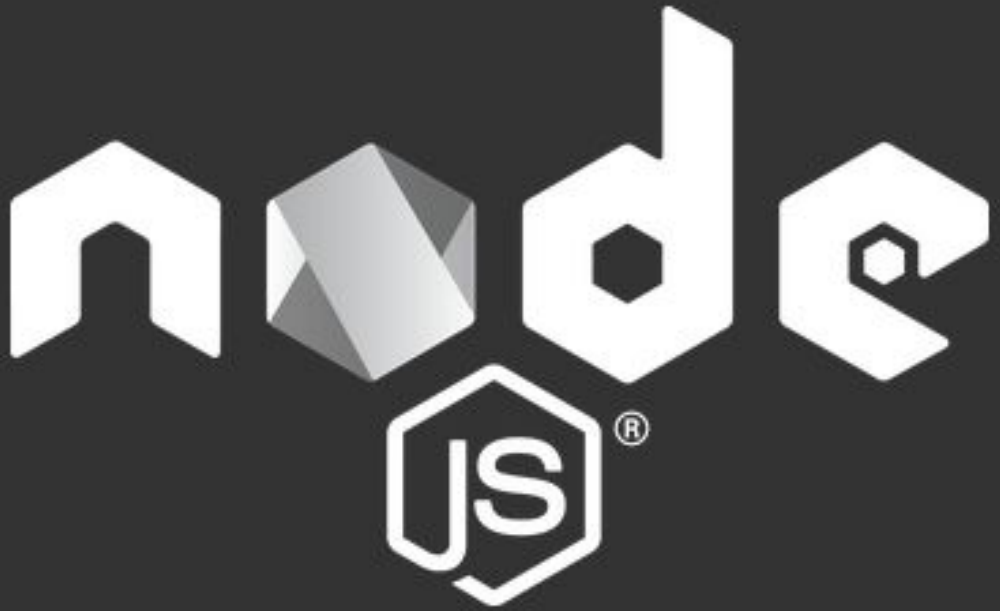

node
JS®

Node.js é uma plataforma para desenvolvimento de aplicações *server-side* baseadas em rede utilizando **JavaScript** e o **V8 JavaScript Engine**

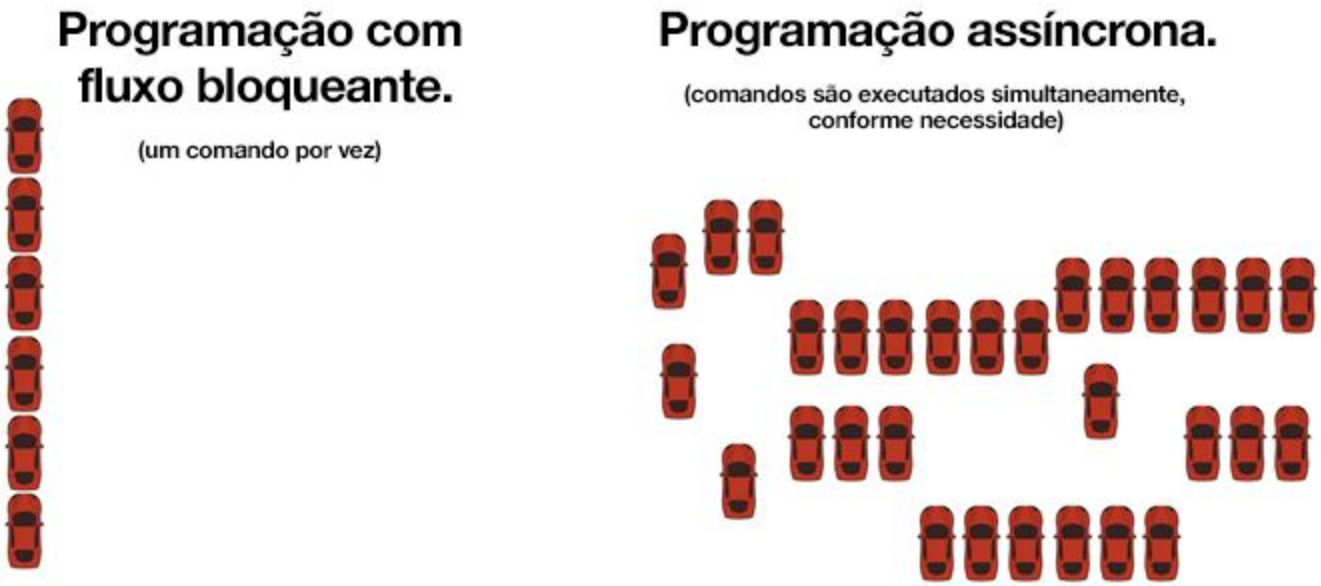
Podemos criar uma variedade de aplicações *Web* utilizando apenas código em**JavaScript**.

V8 JavaScript Engine

V8 é o nome do interpretador JavaScript, também chamado de máquina virtual Javascript *(ou engine)*, desenvolvido pela Google e utilizado em seu navegador Google Chrome. A proposta do V8 é acelerar o desempenho de uma aplicação compilando o código Javascript para o formato nativo de máquina antes de executá-lo, permitindo que rode a velocidade de um código binário compilado[1].

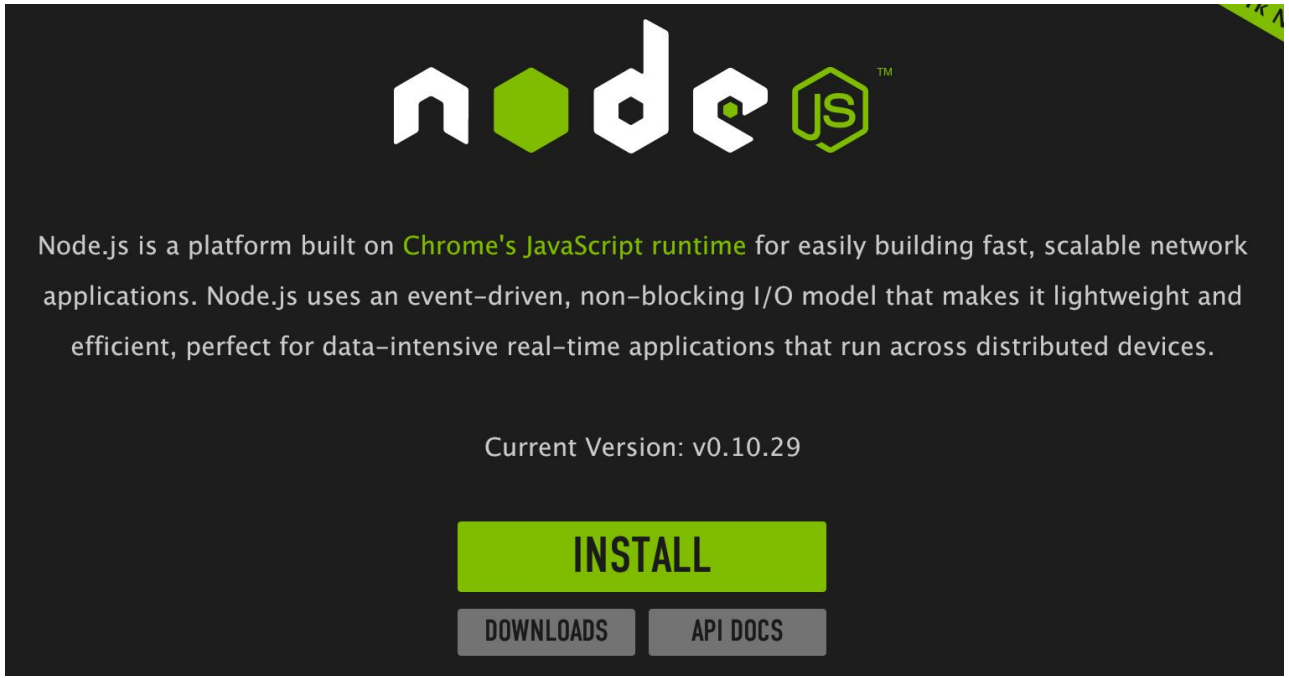
- **Similar Apache** e **IIS, porem muito mais simples**
- **Node.js** é extremamente simples, por isso pode não ser a solução ideal para todos os casos
- *Single threaded*
- **Node.js** utilizar uma abordagem não obstrutiva, essa diferença vai ser imperceptível na maioria dos casos.

- *Single threaded*



Instalando o Node.js

A instalação do **Node.js** é extremamente simples e completamente multi-plataforma, tudo que você precisa fazer é visitar a pagina Nodejs(https://nodejs.org/en/) , clicar em “INSTALL” e seguir as instruções.



Após a instalação, basta executar o seguinte comando no seu terminal para verificar se foi instalado corretamente:

```
$ node -v
> v0.10.26
```

```
> a = 10
10
> b = 20
20
> a + b
30
>
```

deve retornar a versão do node que foi instalada, como por exemplo *v0.10.26*.

Simple web server Nodejs

Crie um arquivo chamado “simpleWebserver.js”;
Vamos iniciar o servidor:

```
// Requisitamos o módulo http do Node
var http = require('http');

// Executamos o método para criar o servidor
var server = http.createServer(function(request, response) {
  response.writeHead(200, {'Content-Type': 'text/html'});
    response.write('<h1>Página inicial</h1>');
  response.end();

});

// Iniciamos o servidor criado
server.listen(3000, function() {
  console.log('Servidor rodando!');
});
```

Simple web server Nodejs, verificando a url

```
// Verificamos a URL
 if (request.url === '/') {
   response.write('<h1>Página inicial</h1>');
 } else if (request.url === '/sobre'){
   response.write('<h1>Sobre</h1>');
 } else {
   response.write('<h1>Página não encontrada
:(</h1>');
 }
```

Simple web server Nodejs, Parametros URL

```
// Verificamos a URL

var url = require('url');
var result = url.parse(request.url, true);

    for(var key in result.query){
   response.write("<h2>"+key+" :
"+result.query[key]+"</h2>");
  }
```

Simple web server Nodejs, Modulo FS

```
 // Verificamos a URL
var fs = require('fs');
  fs.readFile(__dirname + '/index.html', function(err,
html){
   response.writeHeader(200, {'Content-Type':
'text/html'});
   response.write(html);
   response.end();
   });
```

Não obstrutivo

Todos os recursos presentes no **Node.js** e também a maioria das bibliotecas feitas para ele adotaram um padrão não obstrutivo de escrever código, isso quer dizer que em **Node.js** você geralmente vai conseguir estruturar seu código de uma maneira que operações que não dependem de nada que está sendo executado possam ser executadas de forma independente.

```
var frase;

    carregaFrase = function (callback) {
      setTimeout(function() {
        //Simula leitura da frase no banco de dados.
        frase = "Minha frase obstrutiva";
        callback();
      }, 3000)
    }

    imprimeFrase = function () {
      console.log(frase);
    }

    carregaFrase(imprimeFrase);
```

Gerenciando Modulos

```
var circulo = require('./circulo.js');
response.write( 'Um circulo de raio 4
tem area de '
        + circulo.area(4));
```

```
// circulo.js
var PI = Math.PI;

exports.area = function (r) {
  return PI * r * r;
};

exports.circunferencia =
function (r) {
  return 2 * PI * r;
};
```

Pense no seguinte: aqui estou, criando meu aplicativo, onde um usuário irá cadastar login e senha. Esta senha precisará ser criptografada e, por suposto, todos os dados serão armazenados em um banco. Preciso realmente desenvolver tudo isto? Ou será que existe alguma biblioteca de código livre que possa fazer isto por mim? Reinventar a roda não é algo interessante para quem é desenvolvedor, eu acredito. Então, o mais sensato, prático e seguro é procurar alguma biblioteca que faça isso para você.

FAKER

```
// NPM INSTALL FAKER

var faker = require('faker');

   var randomName = faker.name.findName();
  var randomImage = faker.internet.avatar();

  response.write('Your random name: ' + randomName
+ "<img src='"+randomImage + "'/>");
  response.end();
```

HTTP-SERVER

```
npm install http-server
npm install jitsu -g

cd myapp/
jitsu install http-server
```

Uglifycss

```
$ npm install uglifycss
```

```
$ uglifycss -max-line-len n [filename] [...] > output
```

Controlando as Versões dos Módulos

Como posso saber qual versão do módulo que instalei irá rodar? Ela pode mudar, caso façam algum update? Tem como controlar isso?” É claro! Abra seu arquivo package.json e procure por este trecho:

dependencies { “angular” : “^1.5.5”; }

^ = isto quer dizer que toda vez que o módulo for instalado, qualquer atualização entre 1.5 a 1.9 será instalada.

~ = Esta opção irá instalar somente as versões em que houve correções de bugs, ou seja, atualização de patches

dependencies { “angular” : “1.5.5”; }