

# MEAN Stack

Bruno Gama Catão



# MEAN - Tecnologias

- MongoDB
- Express
- AngularJS
- Node.js





# Ideia

- Apresentar de forma rápida os principais conceitos;
- Mostrar passo-a-passo como construir uma aplicação utilizando essas tecnologias.

# Ambiente de testes

- Você vai precisar:
  - Editor de texto:
    - *Sublime Text*
  - Browser:
    - *Chrome*
  - Conexão com a Internet
  - Um shell decente:
    - ex: *zsh ou bash*



# Introdução a JavaScript

Esta imersão é baseada no ótimo  
livro “JavaScript: The Good Parts”  
de Douglas Crockford



# História e por quê ...

- É a linguagem do *browser*
- A linguagem mais popular do mundo
- Feita em 1995 para o Netscape 2.0 sob o nome de Mocha depois LiveScript e por fim JavaScript (ECMA Script)
- A Microsoft adotou JavaScript no IE 3.0 em 1996



Brendan Eic

# O porquê da má fama

- Sempre foi “a outra”
- Nome terrível !
- Control + C e Control + V
- APIs diferentes entre browsers
- Algumas características bem ruins (*bad parts*)



# The bad parts

- Variáveis globais
- Escopo
- Inserção de ponto-e-vírgula
- Palavras reservadas
- parseInt
- +
- Valores falsos
- ===



... vamos nos focar nas partes boas !

# The good parts

- Funções como valores
- Objetos dinâmicos
- Tipos fracos
- JSON



```
var x = 'Este é um string';
var n = 32; // Um número

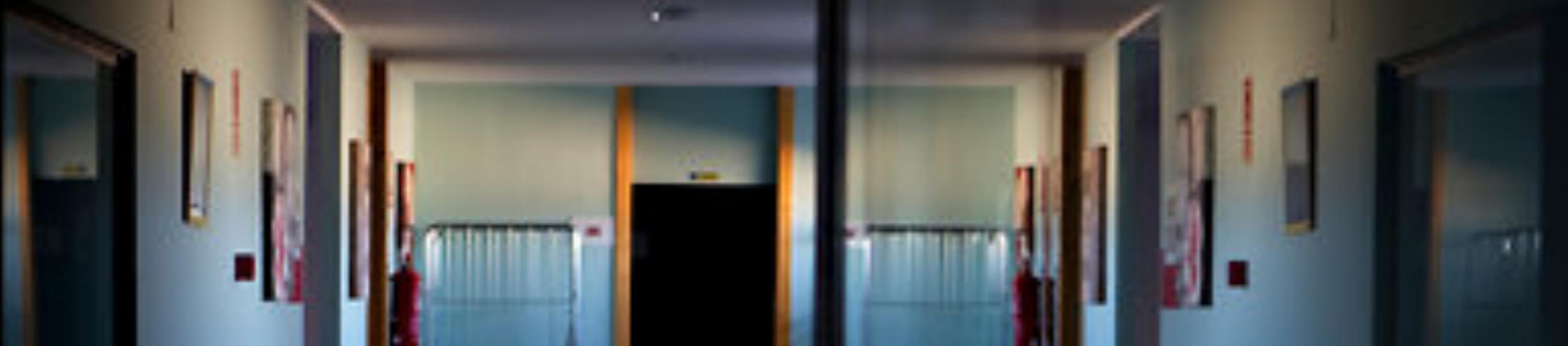
// Funções
var soma = function(a, b) { return a + b; };
var subtrai = function(a, b) { return a - b; };

var operacao = soma;

var resultado = operacao(3, 4);
```

```
var objeto = {
    nome: 'Fulano de Tal',
    dnasc: new Date('1990-04-23'),
    endereco: {
        rua: 'Floriano Peixoto',
        numero: 100,
        bairro: 'Centro',
        cidade: 'Campina Grande',
        uf: 'PB'
    },
    ehAniversario: function() {
        var hoje = new Date();
        return (hoje.getMonth() == this.dnasc.getMonth() &&
            ||| ||| hoje.getDate() == this.dnasc.getDate());
    }
};

console.log(objeto.nome);
console.log('Aniversário ? ' + objeto.ehAniversario());
```

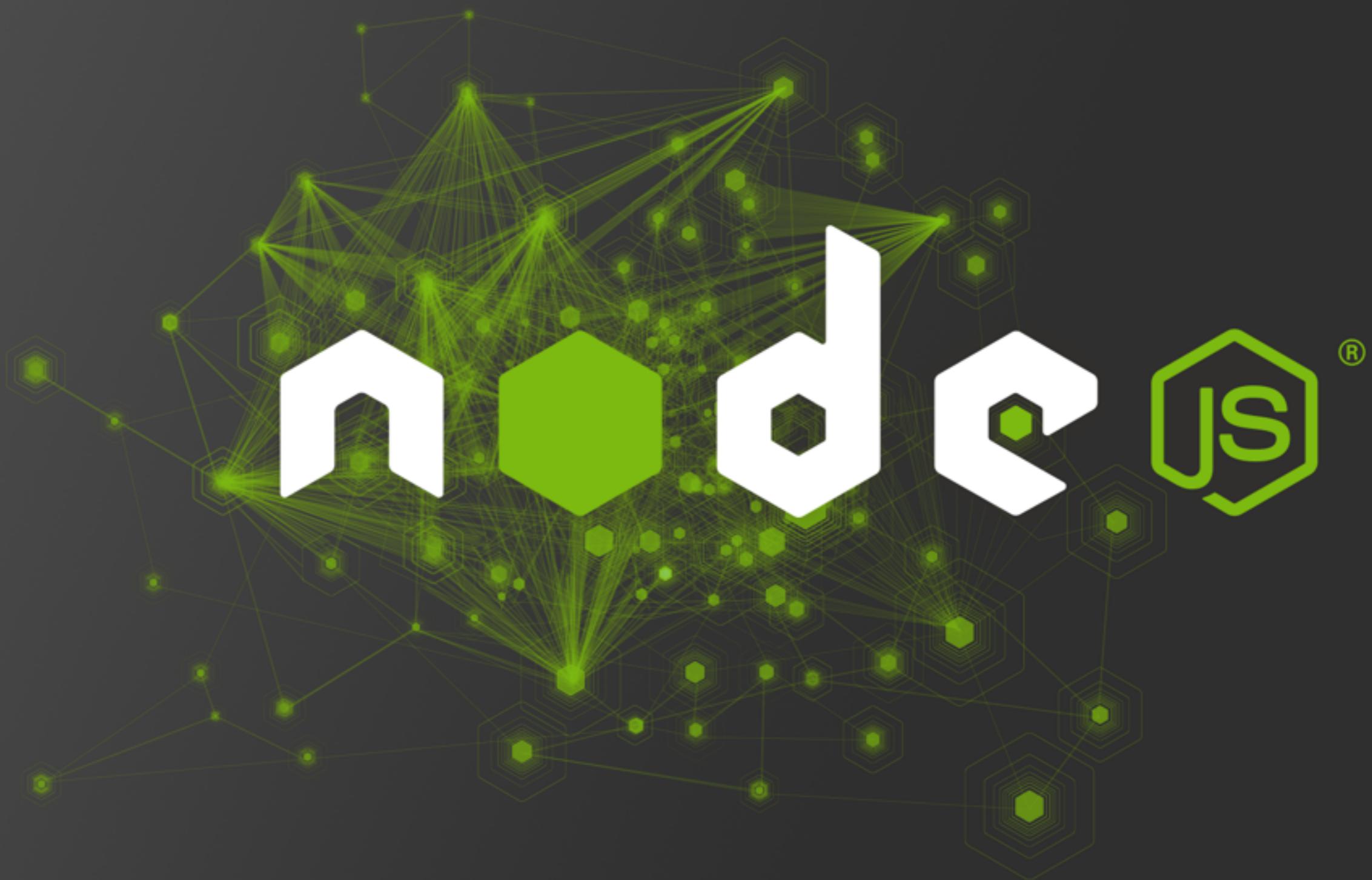


... e mais importante



não dá para fugir





# NodeJS

- JavaScript no Servidor !
- Um único Thread
- Escala
- Velocidade



# NodeJS - História

- Criado por Ryan Dahl em 2009
- Baseado no interpretador V8 do Google
- Código aberto, porém é mantido pela empresa Joyent



<http://nodejs.org>

<http://www.joyent.com>

# Você vai precisar ...

- Baixar o Git (e instalar)
- Baixar o NodeJS (e instalar)

# GIT ?!



- Sistema de controle de versões com excelente suporte para projetos colaborativos
- Feito por Linus Torvalds
- Base da maior rede social / repositório de código online - GitHub (<https://github.com/>)
- A maioria dos projetos open source atuais está hospedada em algum repositório git

# O que significa GIT ?

- Segundo o readme no código fonte do GIT:

GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

- **random three-letter combination that is pronounceable**, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- **stupid**. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "**global information tracker**": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "**goddamn idiotic truckload of sh\*t**": when it breaks

# Baixe o GIT

<https://git-scm.com/downloads>

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It handles all your branching and merging needs, and its distributed nature makes it ideal for both small teams and large organizations.

Git is maintained by the Linux Foundation, which handles legal and financial needs for the project. Conservancy is currently raising funds to continue their mission. Consider [becoming a supporter!](#)



--local-branching-on-the-cheap

Search entire site...

About

Documentation

Blog

**Downloads**

GUI Clients

Logos

Community

## Downloads



Mac OS X



Windows



Linux



Solaris

Older releases are available and the [Git source repository](#) is on GitHub.

Latest source Release

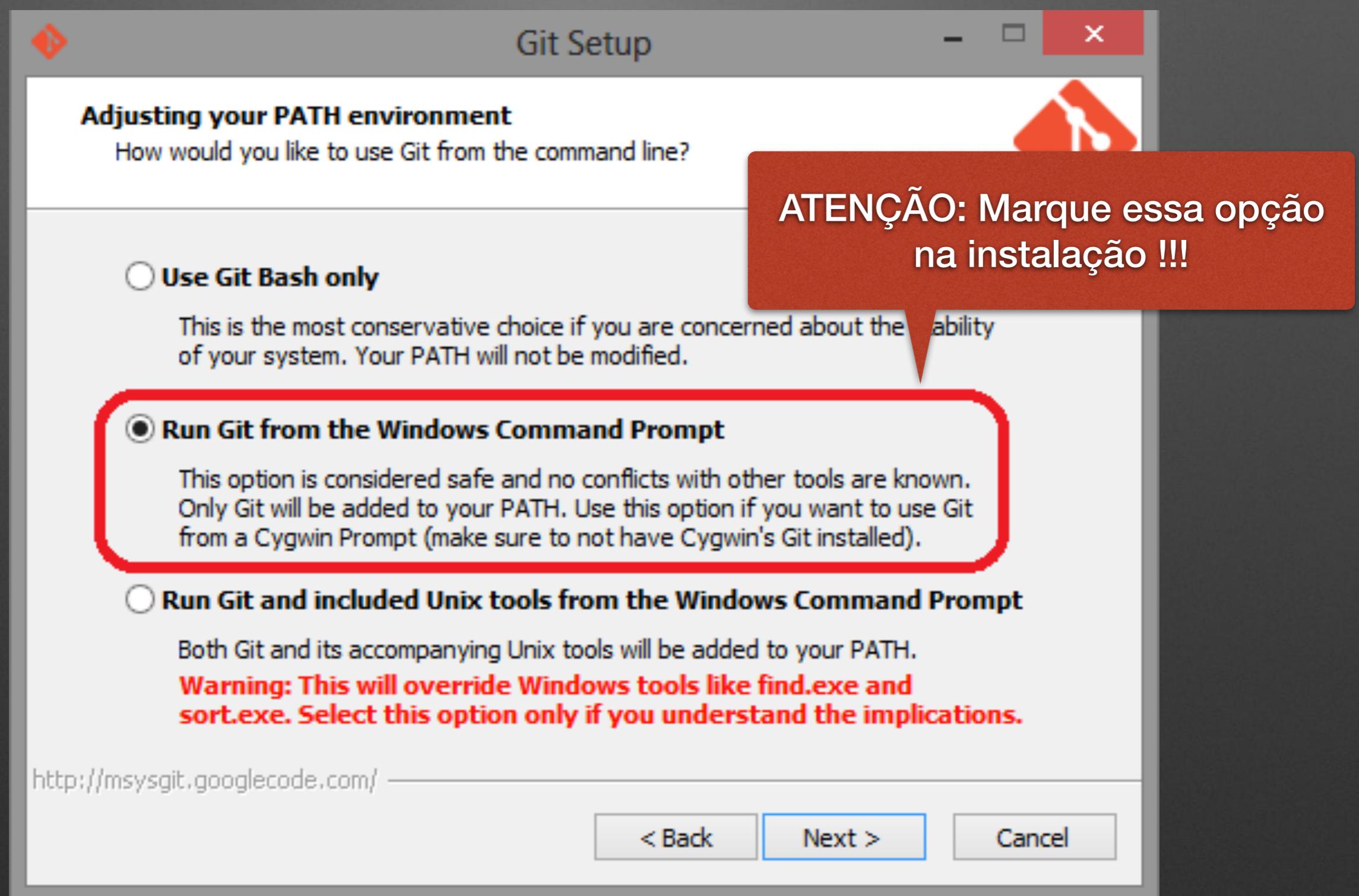
**2.7.0**

[Release Notes \(2016-01-04\)](#)

[Downloads for Mac](#)



# Instale o GIT



# Baixe o NodeJS

A screenshot of a web browser displaying the Node.js download page at <https://nodejs.org/en/download/>. The page features the Node.js logo at the top center. Below the logo is a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, FOUNDATION, GET INVOLVED, SECURITY, and NEWS. The DOWNLOADS link is currently highlighted. A large downward-pointing arrow is positioned below the navigation bar. The main content area is titled "Downloads" and contains a sub-section titled "Windows Installer".

## Downloads

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

The "Downloads" section includes two green boxes on the left: one labeled "LTS" (Mature and Dependable) and another labeled "Stable" (Latest Features). To the right are four download options: "Windows Installer" (MSI file), "Macintosh Installer" (PKG file), and "Source Code" (tar.gz file). Each option has its corresponding icon above the text.

LTS Mature and Dependable	Stable Latest Features
Windows Installer node-v4.2.6-x86.msi	Macintosh Installer node-v4.2.6.pkg
Source Code node-v4.2.6.tar.gz	

[Windows Installer \(.msi\)](#)

32-bit

64-bit

[Windows Binary \(.exe\)](#)

32-bit

64-bit

[Mac OS X Installer \(.pkg\)](#)

64-bit

# Gerenciamento de Pacotes

- NPM - "npm is not an acronym";
- Gerenciador de pacotes JavaScript;
- Vem de brinde no Node.js;
- Dá para pesquisar pacotes em:
  - <https://www.npmjs.com/>



# Projeto com suporte a NPM

- Crie um diretório para seu projeto;
- Nesse diretório, execute: npm init;
- Responda um monte de perguntas ;)

```
~ >>> mkdir meu_projeto
~ >>> cd meu_projeto
~/meu_projeto >>> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
```

# Projeto com suporte a NPM

- Será criado um arquivo “package.json”

```
{  
  "name": "meu_projeto",  
  "version": "1.0.0",  
  "description": "Meu Projeto suporte a NPM",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "Bruno Gama Catao",  
  "license": "ISC"  
}
```

# Instalando um pacote

- npm install colors --save

```
{  
  "name": "meu_projeto",  
  "version": "1.0.0",  
  "description": "Meu Projeto suporte a NPM",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "Bruno Gama Catao",  
  "license": "ISC",  
  "dependencies": {  
    "colors": "^1.1.2"  
  }  
}
```

# Teste de utilização do pacote

- Crie um arquivo index.js

```
var colors = require('colors');

console.log('Hello World'.green);
console.log('JSDay'.yellow);
console.log('FACISA'.rainbow);
```

- Executando:

```
~/meu_projeto >>> node index.js
Hello World
JSDay
FACISA
```



# MongoDB

- HuMONGOus (gigantesco)
- Banco NoSQL orientado a documentos
- Desenvolvido em 2007 pela 10gen

# MongoDB - Conceitos

- **Documento:**
  - Conjunto de dados (semelhante a um objeto)
  - Os dados não precisam seguir a um esquema
  - É possível criar documentos com dados simples, compostos (outros documentos) e coleções (arrays)
- **Coleções:** conjunto de documentos
- **Banco de Dados:** conjunto de coleções

# Próximos Passos

- Baixe o MongoDB
  - <https://www.mongodb.org/>
- Instale o MongoDB
  - Instalação simples (basta escolher o local e pronto)
  - Crie a pasta padrão do banco de dados:
    - C:\data\db
  - Execute o servidor (mongod)

# Observação

- Caso queira usar outra pasta para o banco de dados:
- mongod --dbpath outra\_pasta
- Ex:

```
~/meu_projeto >>> mkdir db
~/meu_projeto >>> mongod --dbpath db
```

# mongo shell

- Execute o aplicativo “mongo.exe” / "mongo"
- No shell vamos digitar alguns comandos

# Criar / Usar um Banco de Dados

- `use nome_do_banco_de_dados`
- Ex:
  - `use blog`

# Inserir um documento em uma coleção

- db.nome\_da\_colecao.insert(objeto)
- Ex:

```
db.posts.insert({autor: 'Fulano', texto: 'Teste', criado_em: new Date()})
```

```
> use blog
switched to db blog
> db.posts.insert({autor: 'Bruno', texto: 'Teste', criado_em: new Date()})
WriteResult({ "nInserted" : 1 })
```

# Consultando todos os documentos de uma coleção

- db.nome\_da\_colecao.find( )
- Ex:

- db.posts.find( )

id criado automaticamente

```
{ "_id" : ObjectId("56ac0bcd18608059723e7bb9"), "autor" : "Fulano", "texto" : "123  
Testando", "criado_em" : ISODate("2016-01-30T01:03:09.072Z") }
```

# Consultando documentos por valor

- `db.nome_da_colecao.find(objeto)`
- Ex:
  - `db.posts.find({autor: “Fulano”})`

# Atualizando um documento

- `db.nome_da_colecao.update(obj_busca, obj_atualizacao)`
- Ex:
  - `db.posts.update({autor: “Fulano”}, {texto: “CENSURADO”})`

# Operadores de Atualização

- Realizam operações sobre os valores de um documento
- Alguns operadores:
  - **\$inc** - Incrementa o atributo pelo valor informado
  - **\$mul** - Multiplica o atributo pelo valor informado
  - **\$set** - Atribui o valor informado ao atributo
  - **\$min** - Atualiza o valor apenas se for menor que o informado
  - **\$max** - Atualiza o valor apenas se for maior que o informado

# Operadores de Atualização

- Exemplo:
  - use estoque
  - db.produtos.insert({cod: 123, nome: 'Durex', qtd: 10})
  - db.produtos.update({cod: 123}, {\$inc: {qtd: -1}})
  - db.produtos.find( )

```
{ "_id" : ObjectId("56ac0f7018608059723e7bba"), "cod" :  
123, "nome" : "Durex", "qtd" : 9 }
```

# Excluindo um documento

- `db.nome_da_colecao.remove(objeto)`
- Ex:
  - `db.produtos.remove({cod: 123})`

# Fechando o shell

- Digite "exit"

# MongooseJS

- Mapeia objetos JavaScript para documentos MongoDB
- Instalação via npm
- No seu projeto, abra a janela do terminal e digite:
  - `npm install mongoose --save`

Não feche a janela  
do mongod !

**Abra o arquivo index.js**

# Use o módulo mongoose

- `var mongoose = require('mongoose');`

# Abra uma conexão com o banco de dados

- `mongoose.connect('mongodb://localhost/estoque');`

endereço do host

nome do banco de dados

# Obtenha a conexão

- var db = mongoose.connection;

# Defina o esquema de um produto

```
var ProdutoSchema = new mongoose.Schema( {  
    cod: Number,  
    nome: String,  
    qtd: Number  
} );
```

# Crie um modelo a partir do esquema

```
var Produto = mongoose.model('Produto', ProdutoSchema);
```

# Criando um produto quando a conexão estiver aberta

```
db.once('open', function () {  
  
  var coca = new Produto({cod: 321, nome: "Coca Cola", qtd: 22});  
  
  coca.save(function(err) {  
  
    if (err) throw err;  
  
    console.log("Produto criado com sucesso!");  
  
  } );  
  
} );
```

```
var colors = require('colors');
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/estoque');

var db = mongoose.connection;

var ProdutoSchema = new mongoose.Schema({
  cod: Number,
  nome: String,
  qtd: Number
});

var Produto = mongoose.model('Produto', ProdutoSchema);

db.once('open', function(){
  var coca = new Produto({cod: 321, nome: 'Coca Cola', qtd: 10});
  coca.save(function(error) {
    if (error) {
      console.log('Erro: '.red, error);
    } else {
      console.log('Produto salvo com sucesso'.green);
    }
    db.close();
  });
});
```

# Express

Framework web rápido, flexível e  
minimalista para [Node.js](#)

```
$ npm install express --save
```



# Hello World

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World, para variar ;)');
});

app.listen(3000, function () {
  console.log('Servidor escutando na porta 3000!');
});
```

Execute: node index.js

# Hello World



# Roteamento

- **app.METHOD(PATH, HANDLER)**
- app - Instância do express
- METHOD - método HTTP (get, put, post, delete);
- PATH - URL relativa
- HANDLER - função que será executada

# Roteamento - Exemplos

```
app.get('/', function (req, res) {  
  res.send('Hello World!');  
});
```

```
app.put('/user', function (req, res) {  
  res.send('Got a PUT request at /user');  
});
```

```
app.delete('/user', function (req, res) {  
  res.send('Got a DELETE request at /user');  
});
```

# Servindo Arquivos Estáticos

```
app.use(express.static('public'));
```

ou

```
app.use('/static', express.static('public'));
```

# Gerador do Express

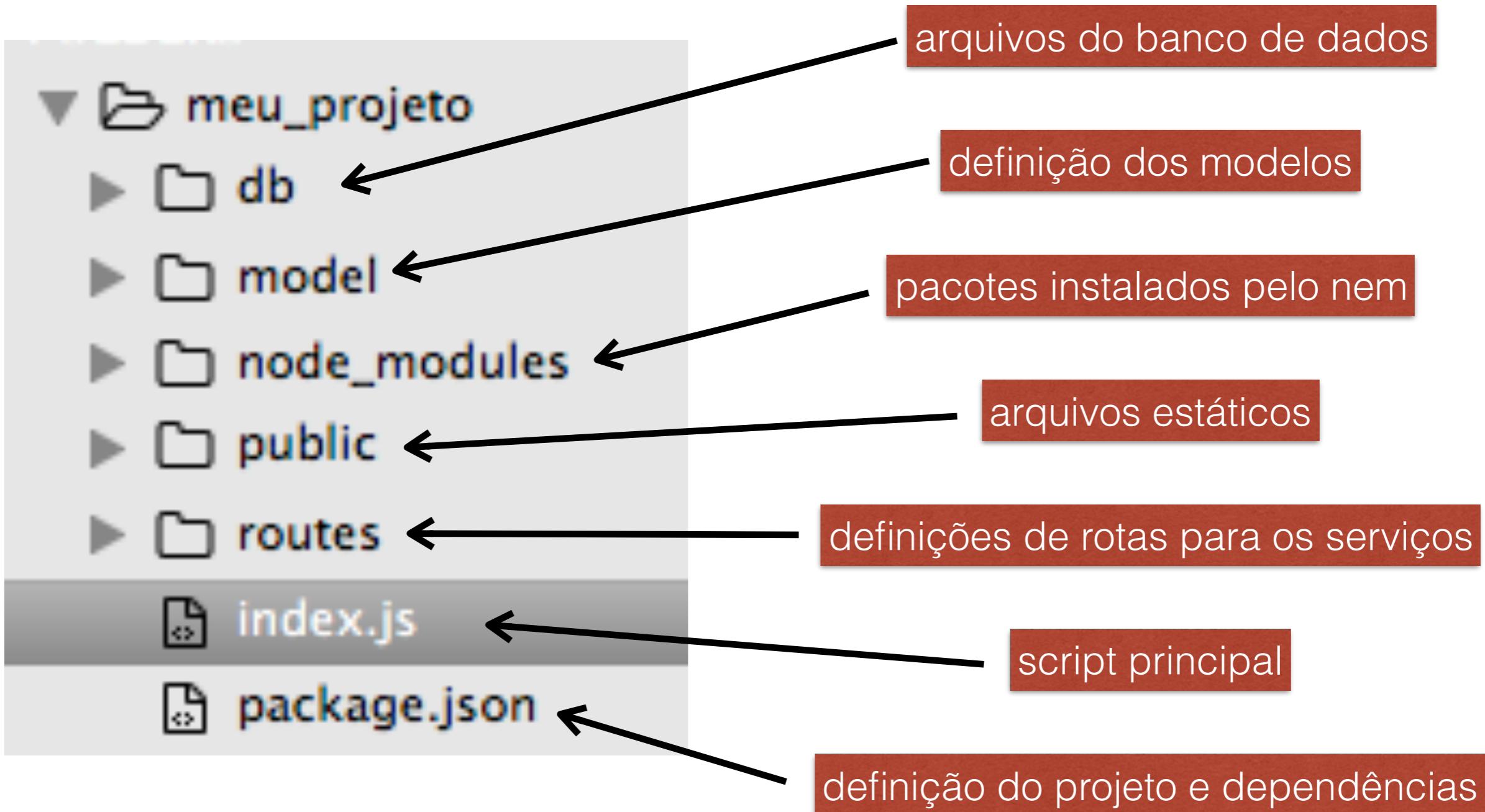
- O Express possui um gerador de aplicativos tradicionais:
  - templates de páginas processadas por engines;
- Instalação:
  - npm install express-generator -g
- Uso:
  - express meu\_app

**Não usaremos  
esse gerador**

# Vamos criar uma API

- Web Service RESTFull;
- Utilizando JSON para comunicação;
- Próximos passos:
  - Organizar o projeto em módulos;
  - Instalar pacotes adicionais;
  - Criar módulos restantes.

# Estrutura de Diretórios



# Pacote body-parser

- Converte parâmetros da requisição HTTP em objetos JSON;
- Instalação:
  - `npm install body-parser --save`

# model/produto.js

```
var mongoose = require('mongoose');

var ProdutoSchema = new mongoose.Schema({
  cod: Number,
  nome: String,
  qtd: Number
});

var Produto = mongoose.model('Produto', ProdutoSchema);

module.exports = Produto; valor que será exportado pelo módulo
```

# Definindo o serviço routes/produtos.js

- Importações:

```
var express = require('express');
var router = express.Router();
var Produto = require('../model/produto');
```

módulos locais são referenciados pelo caminho relativo

# Definindo o serviço routes/produtos.js

```
// LISTA TODOS OS TODOS OS PRODUTOS
router.get('/', function(req, res, next) {
  Produto.find(function(err, produtos) {
    if (err) next(err);
    res.send(produtos);
  });
});
```

# Definindo o serviço routes/produtos.js

```
// ADICIONA UM PRODUTO
router.post('/', function(req, res, next) {
  var produto = new Produto(req.body);
  produto.save(function(err, produto) {
    if (err) next(err);
    res.send(produto);
  });
});
```

# Definindo o serviço routes/produtos.js

```
// OBTEM UM PRODUTO
router.get('/:id', function(req, res, next) {
  Produto.findById(req.params.id, function(err, produto) {
    if (err) next(err);
    res.send(produto);
  });
});
```

# Definindo o serviço routes/produtos.js

```
// ALTERA UM PRODUTO
router.put('/:id', function(req, res, next) {
  Produto.findByIdAndUpdate(
    req.params.id,           // id
    {$set: req.body},        // novos valores
    {new: true},             // cria caso não exista
    function(err, produto) {
      if (err) next(err);
      res.send(produto);
    });
});
```

# Definindo o serviço routes/produtos.js

```
// REMOVE UM PRODUTO
router.delete('/:id', function(req, res, next) {
  Produto.findByIdAndRemove(req.params.id,
    function(err, produto) {
      if (err) next(err);
      res.send(produto);
    });
});
```

# Definindo o serviço routes/produtos.js

```
module.exports = router;
```

# Resumo do serviço

```
var express = require('express');
var router = express.Router();
var Produto = require('../model/produto');

router.get('/', function(req, res, next) {
  Produto.find(...);
});

router.post('/', function(req, res, next) {
  new Produto(req.body).save(...);
});

router.get('/:id', function(req, res, next) {
  Produto.findById(...);
});

router.put('/:id', function(req, res, next) {
  Produto.findByIdAndUpdate(...);
});

router.delete('/:id', function(req, res, next) {
  Produto.findByIdAndRemove(...);
});

module.exports = router;
```

# index.js

```
var express      = require('express');
var mongoose    = require('mongoose');
var bodyParser = require('body-parser');

var app = express();

app.use(bodyParser.json());
app.use(express.static('public'));
app.use('/produtos', require('./routes/produtos'));

mongoose.connect('mongodb://localhost/estoque');

var db = mongoose.connection;
db.once('open', function() {
  app.listen(3000, function() {
    console.log('Servidor escutando na porta 3000');
  });
});
```



**ANGULARJS**

by **Google**

by **Google**

# Bower



- Gerenciador de pacotes para a web (cliente);
- Instalação:
  - `npm install -g bower`
- Configurando o bower no projeto
  - `bower init`

# Bower



- Instalando componentes:
  - bower install angular --save
  - bower install angular-resource --save

# Bower



- Adicionando o diretório bower\_components às rotas estáticas do Express:
  - **app.use(express.static('bower\_components'));**

agora, todos os arquivos importados pelo bower estarão disponíveis no browser

# public/index.html

- **Dica:**
  - No Sublime Text:
    - Crie um arquivo;
    - Salve-o com a extensão html
    - Digite html e aperte TAB

# public/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>

</body>
</html>
```

tudo criado “automagicamente” !

# Importando scripts

os scripts angular e angular-resource foram importados pelo Bower

```
<script src="/angular/angular.js"></script>
<script src="/angular-resource/angular-resource.js"></script>
<script src="/js/app.js"></script>
```

ainda vamos criar o script /js/app.js

# Template do conteúdo

```
<h1>Lista de Produtos</h1>
<table>
  <thead>
    <tr>
      <td>Código</td><td>Nome</td><td>Quantidade</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>001</td><td>Produto</td><td>5</td>
    </tr>
  </tbody>
</table>
```

tabela simples, mostrando um produto fictício

# public/index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Lista de Produtos</title>
  <script src="/angular/angular.js"></script>
  <script src="/angular-resource/angular-resource.js"></script>
  <script src="/js/app.js"></script>
</head>
<body>
  <h1>Lista de Produtos</h1>
  <table>
    <thead>
      <tr>
        <td>Código</td><td>Nome</td><td>Quantidade</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>001</td><td>Produto</td><td>5</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

# Testando nosso template

- Execute o servidor:
  - node index.js
- Abra no browser:
  - <http://localhost:3000>

# Resultado



# Testando o Angular

- Na tag html adicione um atributo:
  - **ng-app**
- Em algum ponto do corpo da página adicione uma expressão:
  - **$\{{2 + 2}\}$**

# Testando o Angular

```
<!DOCTYPE html>
<html ng-app>    atributo ng-app
<head>
    <title>Lista de Produtos</title>
    <script src="/angular/angular.js"></script>
    <script src="/angular-resource/angular-resource.js"></script>
    <script src="/js/app.js"></script>
</head>
<body>
    <h1>Lista de Produtos {{2 + 2}}</h1>
```

expressão que será interpretada pelo Angular

# Resultado



# public/js/app.js

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

# public/js/app.js

criação do módulo 'loja'

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

# public/js/app.js

importando o módulo ngResource

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

# public/js/app.js

```
var loja = angular.module('loja', ['ngResource']);
```

criação de uma fábrica 'Produto' utilizando o componente \$resource

```
loja.factory('Produto', function($resource) {  
  return $resource('/produtos/:id');
```

}); configuração do componente \$resource com a URL do serviço RESTful

```
loja.controller('lojaCtrl', function ($scope, Produto) {  
  $scope.produtos = Produto.query();  
});
```

# public/js/app.js

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

criação do controlador principal da aplicação 'lojaCtrl'

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

# public/js/app.js

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

o controlador depende dos componentes \$scope e Produto

# public/js/app.js

```
var loja = angular.module('loja', ['ngResource']);

loja.factory('Produto', function($resource) {
  return $resource('/produtos/:id');
});

loja.controller('lojaCtrl', function ($scope, Produto) {
  $scope.produtos = Produto.query();
});
```

no início, o controlador irá carregar os produtos e guardar na variável produtos

a variável produtos está definida no componente \$scope, ou seja, acessível na página web

# Modificações no index.html

- Mudar o valor do atributo ng-app:
  - **ng-app=“loja”**
- Adicionar um atributo ng-controller no body:
  - **ng-controller=“lojaCtrl”**
- Para cada produto do escopo, exibir seu conteúdo:
  - **<tr ng-repeat=“p in produtos”>**
    - **<td>{{p.nome}} ...**

```
<!DOCTYPE html>
<html ng-app="loja">
<head>
  <title>Lista de Produtos</title>
  <script src="/angular/angular.js"></script>
  <script src="/angular-resource/angular-resource.js"></script>
  <script src="/js/app.js"></script>
</head>
<body ng-controller="lojaCtrl">
  <h1>Lista de Produtos</h1>
  <table>
    <thead>
      <tr>
        <td>Código</td><td>Nome</td><td>Quantidade</td>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="p in produtos">
        <td>{{p.cod}}</td><td>{{p.nome}}</td><td>{{p.qtd}}</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

# Resultado



# Resultado Final

The screenshot shows a web browser window titled "Lista de Produtos" with the URL "localhost:3000/#". The page contains a form for adding a new product and a table displaying existing products.

**Form (Top Section):**

- Produto:** (Section title)
- Código:
- Nome:
- Qtd:
- 

---

**Lista de Produtos:** (Section title)

Código	Nome	Quantidade	Editar
321	Coca Cola	10	<a href="#">Editar</a>
321	Coca Cola	10	<a href="#">Editar</a>
123	Fanta	22	<a href="#">Editar</a>

# Resultado Final

- Disponível em:
  - [https://github.com/brunogamacatao/mean\\_jsday](https://github.com/brunogamacatao/mean_jsday)



Dúvidas ?



MOVIECLIPS.COM

Muito Obrigado