



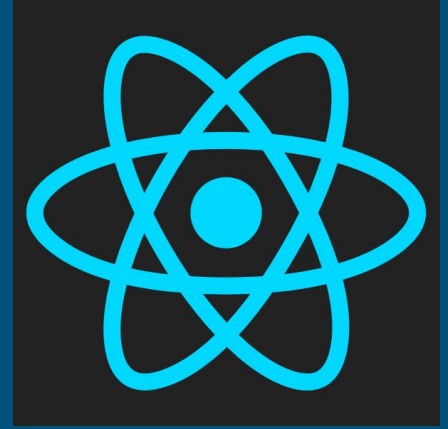
Treinamento Web Avançando



NodeJS



Framework em JS



NodeJS Estrutura

- Bloqueantes (Blocking-Thread)
 - Ruby
 - PHP
 - .NET
- Ryan Dahl, em 2009
 - + 14 colaboradores

Porque Node

- Plataforma escalável
- Baixo custo de memória e processador
- Programar diretamente com protocolos de Rede ou utilizar as Libs que acessam os recursos do sistema operacional (principalmente UNIX)
- JavaScript é uma linguagem de programação, graças ao JavaScript v8

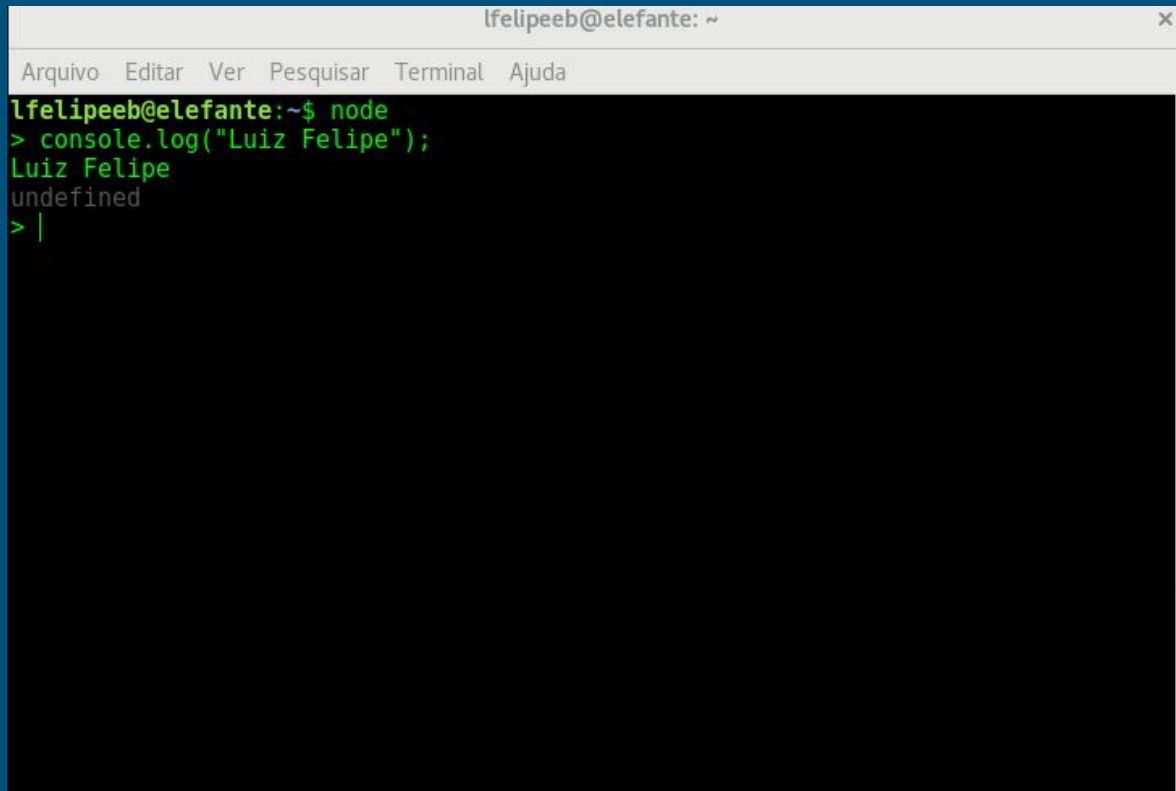
Tipo de programação

- Node é Single-Thread
- Mas possui possibilidade de Cluster
- Alternativa boa, fazer muito uso de programação assíncrona.
 - Execuções em background
 - Uso extremo de Callback
- Tudo de forma não bloqueante

Orientação a Eventos

- Node é orientado a Eventos como no HTML
- Não existe os eventos do HTML
- Node trabalha com Eventos do I/O do servidor
- Node é Client-Side
- Event-Loop é responsável por “ouvi” o servidor

Abra o terminal do NODE



```
lfelipeeb@elefante: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
lfelipeeb@elefante:~$ node  
> console.log("Luiz Felipe");  
Luiz Felipe  
undefined  
> |
```

NPM

- NPM já foi explicado
- Gerenciador Default do NodeJS
- Tudo no Node é Módulo
- “O termo módulo surgiu do conceito de que a arquitetura do Node.js é modular. E todo módulo é acompanhado de um arquivo descritor, conhecido pelo nome de package.json”


```
{  
  "name": "meu-primeiro-node-app",  
  "description": "Meu primeiro app em Node.js",  
  "author": "Caio R. Pereira <caio@email.com>",  
  "version": "1.2.3",  
  "private": true,  
  "dependencies": {  
    "modulo-1": "1.0.0",  
    "modulo-2": "~1.0.0",  
    "modulo-3": ">=1.0.0"  
  }  
}
```

package.json

Variáveis Globais

```
window.hoje = new Date();  
alert(window.hoje);
```

```
global.hoje = new Date();  
console.log(global.hoje);
```

```
human.js ×  
1 exports.hello = function (msg) {  
2   console.log(msg);  
3 };  
  
exports.hello()  
  
hello.js ×  
1 module.exports = function (msg) {  
2   console.log(msg);  
3 };
```

Primeira Aplicação

- Node é multi-protocolo
- Mais famoso é HTTP
 - HTTPS, FTP, SSH, DNS, TCP, UDP, WebSockets
- Na prática o Node serve para desenvolver aplicações middleware
- Não quer configurar tudo? Utilize um “módulo” já configurado
 - Express, Geddy, Sails, Socket.js

Como funciona. Você cria o HTTP, e invoca o `createServer` para levar o Callback ao EventLoop, cada vez que ele recebe uma requisição no serviço ele invoca a callback

Módulo URL

- href: Retorna a url completa: 'http://user:pass@host.com:8080/p/a/t/h?query=string#hash'
- protocol: Retorna o protocolo: 'http' • host: Retorna o domínio com a porta: 'host.com:8080'
- auth: Retorna dados de autenticação: 'user:pass'
- hostname: Retorna o domínio: 'host.com'
- port: Retorna a porta: '8080'
- pathname: Retorna os pathnames da url: '/p/a/t/h' • search: Retorna uma query string: '?query=string'
- path: Retorna a concatenação de pathname com query string: '/p/a/t/h?query=string'
- query: Retorna uma query string em JSON: {'query':'string'}
- hash: Retorna ancora da url: '#hash'

Módulo FS

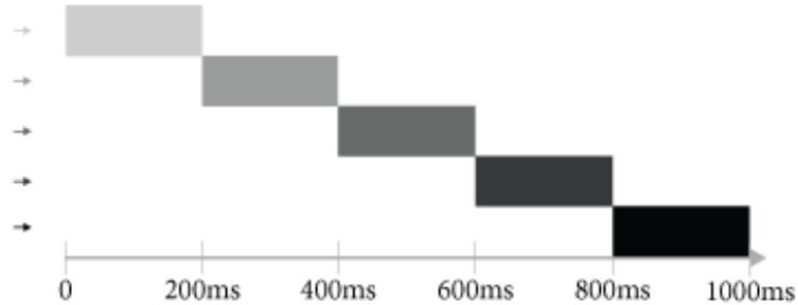
- Boa prática separa o código HTML do JavaScript
 - FS organiza diretórios e arquivos do Sistema
 - Final Sync() é para tratamento síncrono.
-
- <https://nodejs.org/api/fs.html>

Desafio implementar um Roteador de URL

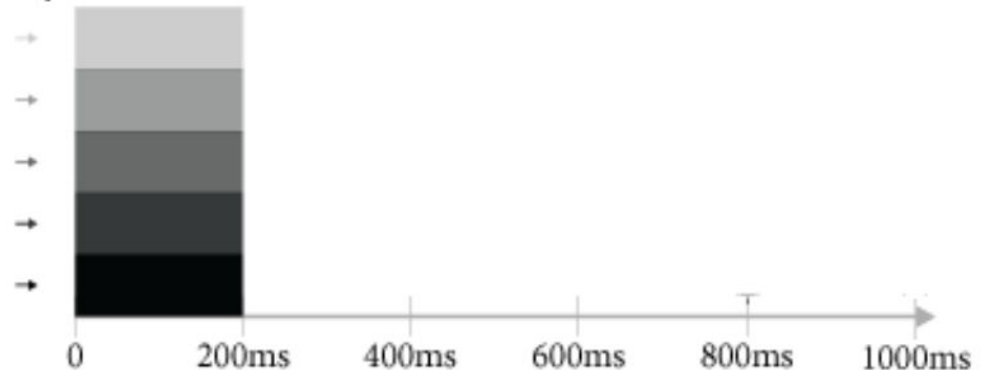
- O nome da Rota deve ser igual o do arquivo
- Se não achar o arquivo disparar a página ERRO
- Leitura do Arquivo deve ser assíncrona
- Dicas
 - `url.parse()` retorna o pathname digitado, pathname vazio == `/`
 - `response.end(html)` pode receber conteúdo HTML
 - `fs.exists(html)` para ver se o arquivo existe com mesmo nome do pathname

Porque Assíncrona?

Sync



Async



EventLoop

- libev
- libeio

Em C

- EventEmitter
- Trabalhe com EVENTOS



Callback Hell vs Heavens

```
var fs = require('fs');
fs.readdir(__dirname, function(err, contents) {
  if (err) { throw err; }
  contents.forEach(function(content) {
    var path = './' + content;
    fs.stat(path, function(err, stat) {
      if (err) { throw err; }
      if (stat.isFile()) {
        console.log('%s %d bytes', content, stat.size);
      }
    });
  });
});
```

```
var fs = require('fs');
var lerDiretorio = function() {
  fs.readdir(__dirname, function(err, diretorio) {
    if (err) return err;
    diretorio.forEach(function(arquivo) {
      ler(arquivo);
    });
  });
};
var ler = function(arquivo) {
  var path = './' + arquivo;
  fs.stat(path, function(err, stat) {
    if (err) return err;
    if (stat.isFile()) {
      console.log('%s %d bytes', arquivo, stat.size);
    }
  });
};
lerDiretorio();
```

Sexta-Feira



Express

Ele é um módulo para desenvolvimento de aplicações web de grande escala. Sua filosofia de trabalho foi inspirada pelo framework Sinatra da linguagem Ruby.

- MVR (Model-View-Routes);
- MVC (Model-View-Controller);
- Roteamento de urls via callbacks;
- Middleware; • Interface RESTFul;
- Suporte a File Uploads;
- Configuração baseado em variáveis de ambiente;
- Suporte a helpers dinâmicos;
- Integração com Template Engines;
- Integração com SQL e NoSQL;

Pretendo trabalhar com

- Node.js: Backend do projeto;
- MongoDB: Banco de dados NoSQL orientado a documentos;
- Redis: Banco de dados NoSQL para estruturas de chave-valor;
- Express: Framework para aplicações web;
- Socket.IO: Módulo para comunicação real-time;
- MongooseJS: ODM (Object Data Mapper) MongoDB para Node.js;
- Node Redis: Cliente Redis para Node.js;
- EJS: Template engine para implementação de html dinâmico;
- Mocha: Framework para testes automatizados;
- SuperTest: Módulo para emular requisições que será utilizado no teste de integração;
- Nginx: Servidor Web de alta performance para arquivos estáticos;