

DESENVOLVIMENTO WEB COM JAVASCRIPT

UMA ABORDAGEM 

Especialização em: DESENVOLVIMENTO WEB, CLOUD E
DISPOSITIVOS MÓVEIS

Professor: [Luiz Pedro Petroski](#)



FORMAÇÃO

- Graduação: Engenharia de Computação (UEPG)
- Pós-Graduação: MBA em Gestão de TI (Faculdade Sant'ana)
- Mestrado: Computação Aplicada (UEPG)

PROFISSIONAL

- Professor (Fateb)
- Professor Ensino técnico
- Gestor de TI no NUTE@D (UEPG)
- Fundador e CIO da PiSigma (Eventos científicos e Educação a distância)

EMENTA DA DISCIPLINA

- Introdução ao NodeJS
- Repositório Online NPM
- Módulos
- Servidores
- I/O Síncrono
- Express
- Mongoose

AVALIAÇÃO:

| | |
|-----------------------------------|------------|
| Participação nas aulas 11 e 12/05 | 1,0 ponto |
| Participação nas aulas 25 e 26/05 | 1,0 ponto |
| Atividade LearnNode | 3,0 pontos |
| Projeto Final | 5,0 pontos |

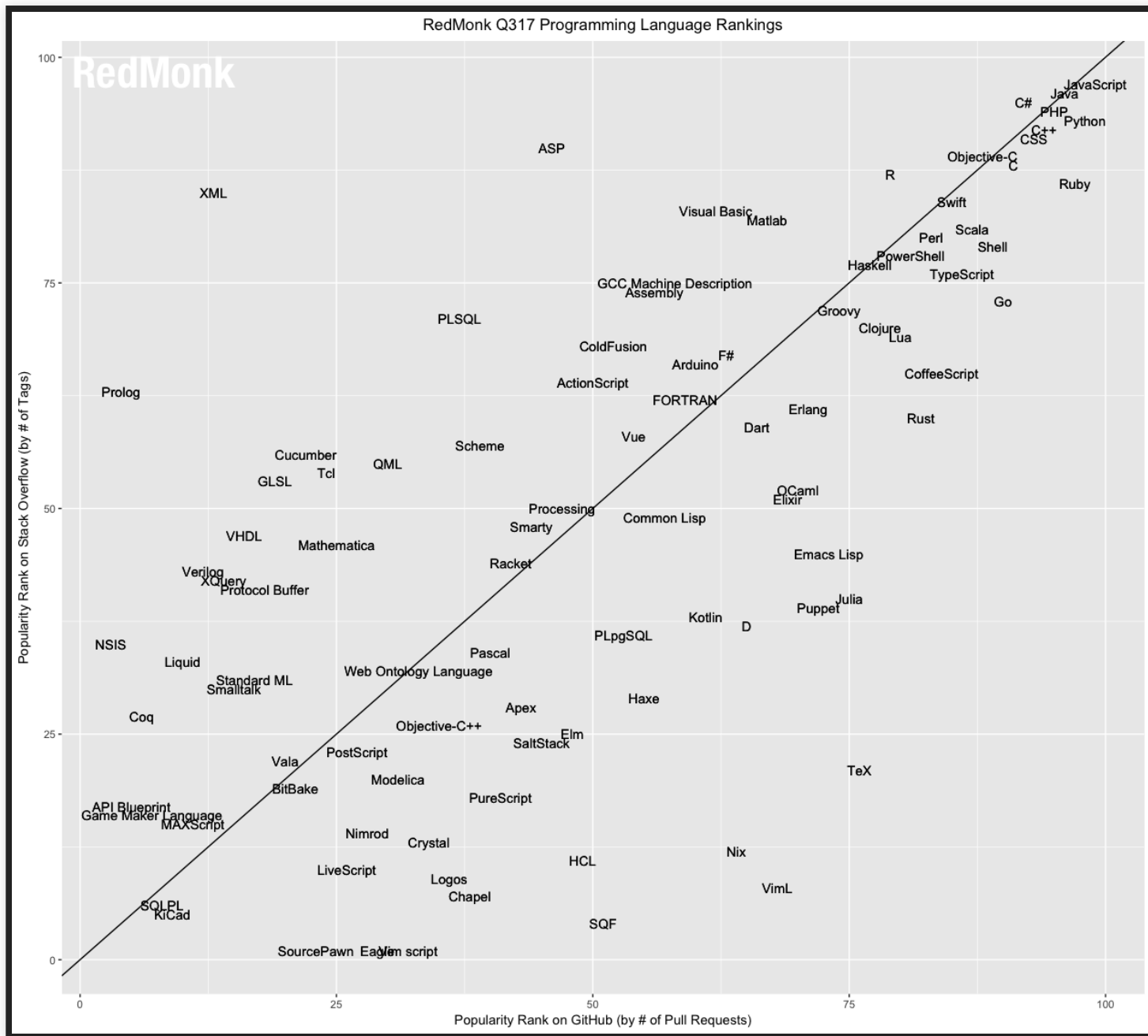
NODE JAVASCRIPT

O Node ampliou quase Universalmente o uso da linguagem JavaScript, e possibilitou explorar os seus recursos de forma mais efetiva

A LINGUAGEM MAIS POPULAR!!!



1. JavaScript
2. Java
3. Python
4. PHP
5. C#



★ QUAL A RAZÃO DESSA POPULARIDADE?

- Expansão da WEB
 - Redes Móveis
 - Tecnologia acessível
 - Poder de processamento e recursos no lado do cliente:
 - 📱 **HTML5:** Geolocalização, storage, notification, web socket...
- Qual a linguagem para manipular essas APIs?

BROWSERS



Opera



Google Chrome



Safari



Mozilla Firefox



Internet Explorer



Microsoft Edge

AMOR  / ÓDIO 

Alguém aqui odeia JavaScript?

Boa parte dos problemas tem relação com a DOM

Problemas de compatibilidade

Falta de padronização e briga dos grandes players do mercado

DOM

Ou Document Object Model, foi padronizado pela W3C em 1998 para representar documentos escritos em HTML, XHTML e XML.

Por meio dessa API, é possível acessar e manipular qualquer informação do documento

**VOCÊ COSTUMA LER O MANUAL
DE INSTRUÇÕES?**



EXEMPLO:

Como você usa o seu forno Micro-Ondas?



“A linguagem JavaScript é a única que as pessoas acham que não precisam aprender antes de começar a utilizar.”

(Douglas Crockford)

REVISÃO JAVASCRIPT

Pular Revisão Javascript

ORIGENS

Foi inspirada pela linguagem **HiperTalk**

Desenvolvida pela Apple para a plataforma **Hiper Card**

Linguagem Simples e amigável



Idéia de implementar um conceito semelhante no
browser

Contratou o Brendan Eich

Ele utilizou como base as linguagens Java, Scheme,
Self e algumas influencias de Perl

NASCIMENTO DO JAVASCRIPT

Surgiu no Netscape Navigator 2.0 beta

Gerra dos Browsers

Junção com a Sun Microsystems e foi criado o
JavaScript

Microsoft para não ficar para trás criou o JScript

PARA ENCURTAR A HISTÓRIA...

Para não perder para a gigante Microsoft a netscape busca padronizar (W3C e ISO)

E em 1997 Junto a ECMA Internacional foi registrado padronizado a linguagem

O nome correto é **ECMAScript**

CARACTERÍSTICAS

- Interpretada
- Orientação a objetos baseada em protótipos (não possui classes)
- Tipagem fraca
- Funções de primeira classe
- Atualmente está na versão 8

DECLARAÇÃO DE VARIÁVEIS

As variáveis devem ser declaradas pela palavra reservada **var** seguida do identificador

- Deve começar por letra, \$ ou _
- Após a primeira letra pode conter números
- Por convenção, começa com letra minúscula e usa camelCase
- case-sensitive

EXEMPLOS

```
var curso="DESENVOLVIMENTO WEB COM JAVASCRIPT";  
var notas=[80, 90, 70];  
var $scopo = {title: "JavaScript", version: "6"};  
var _nome = "Luiz";  
var codigoSecreto = 8732678632;  
var a=40, b=9, c=4;  
var teste;
```

NUMBER

- Não tem int, double, long, é todo number
- IEEE-754 (Standard for Floating-Point)
- binary64 ou Double precision
- Exceções!
- Math API

STRING

Sequencia de 0 ou mais caracteres

Strings são imutáveis

Podem ser declaradas com aspas simples ou dupas

string API

BOOLEAN

Pode assumir o valor true ou false

Cuidado com os valores truty e falsy

!!0 !!NaN !!" !!false !!null !!undefined

todos os demais são truty por padrão

DIFERENÇAS ENTRE UNDEFINED E NULL

O tipo undefined é retornado caso uma propriedade de um objeto seja consultada e não exista

O tipo null indica a ausência de valor em uma determinada propriedade já existente

OBJECT

Objeto é uma coleção dinâmica de **chaves e valores** de qualquer tipo de dado

É possível **adicionar** ou **remover** propriedades a qualquer momento

EXEMPLO

```
var pessoa = {};
```

EXEMPLO

```
var pessoa = {  
    nome: "João",  
    idade: 20,  
};
```

EXEMPLO

```
var pessoa = {  
    nome: "João",  
    idade: 20,  
    telefone: null,  
};
```

EXEMPLO

```
var pessoa = {  
    nome: "João",  
    idade: 20,  
    telefone: null,  
    endereco: {  
        logradouro: "Rua dos Bobos",  
        numero: 0,  
        bairro: "Centro"  
    }  
};
```


EXEMPLO

```
var pessoa = {};  
pessoa.nome="João";  
pessoa["nome"] = 'João';  
pessoa.endereco.bairro="Centro";  
pessoa["endereco"]["bairro"]="Centro";  
pessoa.endereco["logradouro"]="Rua dos bobos";  
pessoa.cor dos olhos = "Azul";  
pessoa["cor dos olhos"] = "Azul";
```

EXEMPLO

```
delete pessoa.idade;  
var pessoa = new Object();  
var pessoa = Object.create(...);
```

A LINGUAGEM JAVASCRIPT NAO TEM!

- Classe
- Construtor
- Método
- Módulo

MAS TEM FUNÇÃO!!!

Uma função é um objeto que contém um bloco de código executável.

O bloco é isolado!!

Função de primeira classe: Pode ser atribuída a uma variável, passada como parâmetro ou retornada por outra função.

FUNCTION DECLARATION

```
function soma(a, b){  
    return a+b;  
}  
soma(1, 3);
```

FUNCTION EXPRESSION

```
var soma = function(a, b){  
    return a+b;  
}  
soma(1, 3);
```

NAMED FUNCTION EXPRESSION

```
var soma = function soma(a, b){  
    return a+b;  
}  
soma(1, 3);
```

FORMAS DE INVOCAR UMA FUNÇÃO

- Invocando diretamente
- passado por parâmetro
- Retornando uma função
- Invocando função por meio de um objeto
- call e apply

INVOCANDO DIRETAMENTE

```
var soma = function soma(a, b){  
    return a+b;  
}  
soma(1, 3);
```

PASSANDO POR PARÂMETRO (LAMBDA)

```
var produto = {nome: 'Sapato', preco: 150};  
var formulaImpostoA=function (preco){return preco*0,1;};  
var calculaPreco = function(produto, formulaImposto){  
    return produto.preco+formulaImposto(produto.preco);  
}
```

RETORNO DE UMA FUNÇÃO

```
var helloworld = function () {  
    return function () {  
        return "Hello World";  
    }  
}  
console.log(helloworld);  
console.log(helloworld());  
console.log(helloworld())();
```

DENTRO DO OBJETO

```
var pessoa={  
    nome: "João",  
    idade: 20,  
    getIdade: function(){  
        return this.idade;  
    }  
}  
pessoa.getIdade();
```

DENTRO DO OBJETO

```
var getIdade = function(){  
    return this.idade;  
}  
var pessoa={  
    nome: "João",  
    idade: 20,  
    getIdade: getIdade  
}  
getIdade();  
pessoa.getIdade();
```

CALL E APPLY

Toda Função possui os métodos `call()` e `apply()`

Utilizados para indicar qual escopo a função deve ser executada

```
funcao.call(escopo, paramentro1, parametro2);  
funcao.apply(escopo, parametros);
```

CALL E APPLY

```
var getIdade = function(extra){  
    return this.idade + extra;  
}  
var pessoa={  
    nome: "João",  
    idade: 20,  
    getIdade: getIdade  
}  
getIdade();  
pessoa.getIdade();  
getIdade.call(pessoa, 3);  
getIdade.apply(pessoa, [3]);
```

FUNÇÕES CONSTRUTORAS VS FUNÇÕES FÁBRICA

Fábrica

```
var criarPessoa(nome, idade){  
    return {  
        nome: nome,  
        idade: idade  
    };  
};  
console.log(criarPessoa("Pedro", 20));
```


FUNÇÕES CONSTRUTORAS VS FUNÇÕES FÁBRICA

Construtoras

```
var Pessoa = function (nome, idade){  
    this.nome=nome;  
    this.idade=idade;  
};  
console.log(new Pessoa("Pedro", 20));  
var maria = {};  
Pessoa.call(maria, "Maria", 20);  
console.log(maria);
```

CLOSURES

```
var helloWorld = function (){  
    var message = "Hello world";  
    return function(){  
        return message;  
    };  
};  
var fnHello = helloWorld();  
console.log(fnHello());
```

CUIDADO COM O ESCOPO GLOBAL

```
var counter = 0;
var add = function(){
    return ++counter;
};
console.log(add());
console.log(add());
console.log(add());

var itens = [];
var add = function(item){
    itens.push(item);
    return itens;
};
console.log(add('A'));
console.log(add('B'));
```

DESAFIO!

Como evitar a situação anterior?

dica: pode utilizar objetos

SOLUÇÃO

```
var counter = {  
    value:0,  
    add: function(){  
        return ++this.value;  
    }  
};  
console.log(counter.add());  
console.log(counter.add());  
var itens = {  
    value:[],  
    add: function(item){  
        this.value.push(item);  
        return this.value;  
    }  
};
```

PROBLEMA: NAO TEM ENCAPSULAMENTO

```
var counter = {  
    value:0,  
    add: function(){  
        return ++this.value;  
    }  
};  
console.log(counter.add());  
console.log(counter.add());  
counter.value=undefined;  
console.log(counter.add());
```

LEMBRAM DA CLOUSURE?

FACTORY FUNCTION

```
var createCounter = function(){  
    value=0;  
    return {  
        add: function(){  
            return ++value;  
        }  
    };  
};  
var counter = createCounter();  
console.log(counter.add());  
console.log(counter.add());  
counter.value=undefined;  
console.log(counter.add());
```

ARRAY

São apenas objetos especiais, que oferece meios para acessar e manipular suas propriedades por meio de índices.

CRIANDO ARRAY

```
var carros = [];  
carros[0]='Gol';  
carros[1]='Palio';  
//ou  
var carros['Gol', 'Palio', 'Corsa'];  
//ou  
var carros = new Array();  
var carros = new Array('Gol', 'Palio', 'Corsa');  
var carros = new Array(10);
```

E O NODE

A linguagem ganhou ainda mais força com o Node

Possibilidade de interpretação do JavaScript no lado
do servidor

Express

ElectronJS

MAS ENTÃO O NODE NASCEU PARA EXECUTAR O JS NO SERVIDOR?

Não

Rayan Dahl

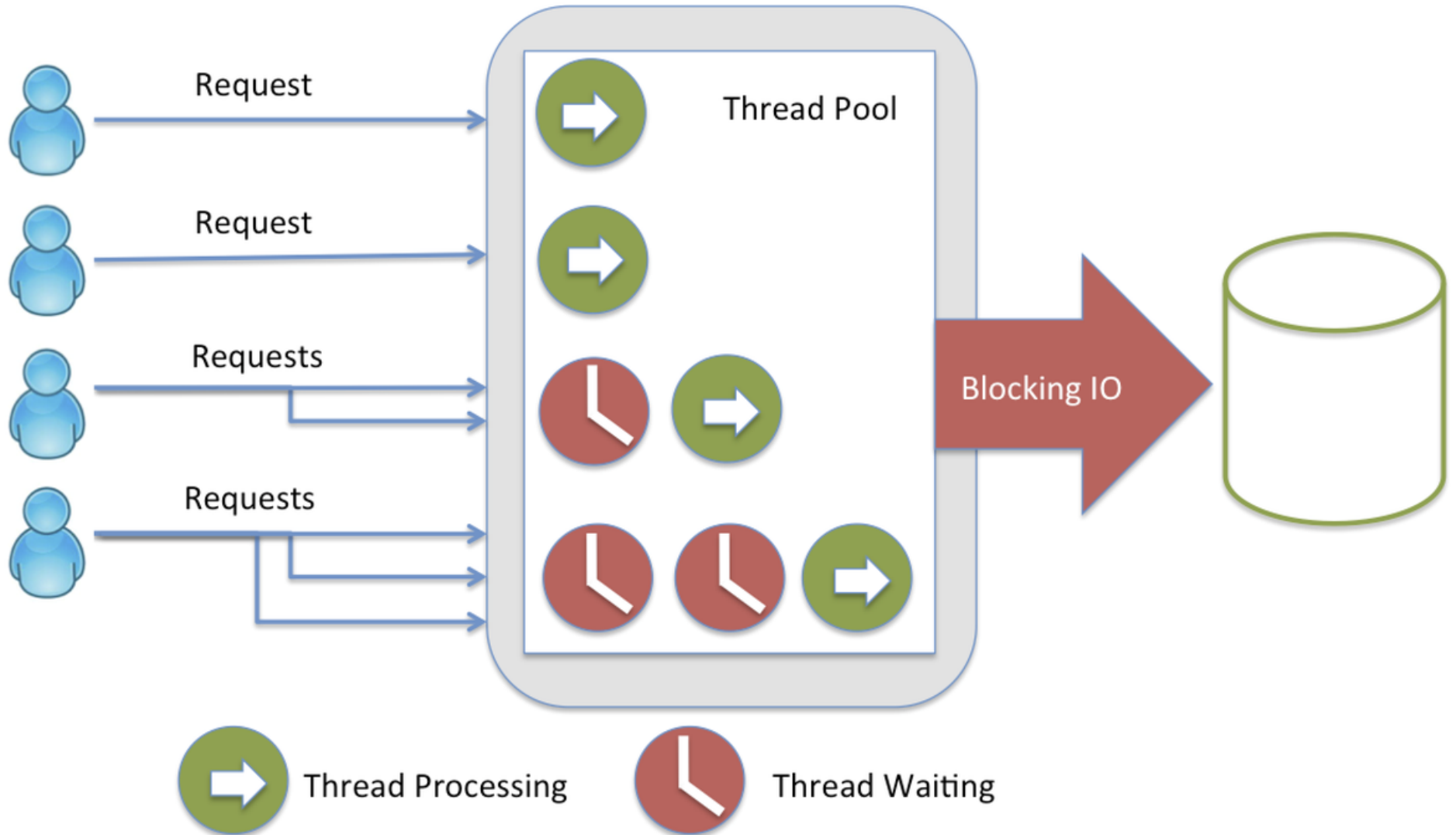
Matemático - Programação de sites (2006)



Flickr

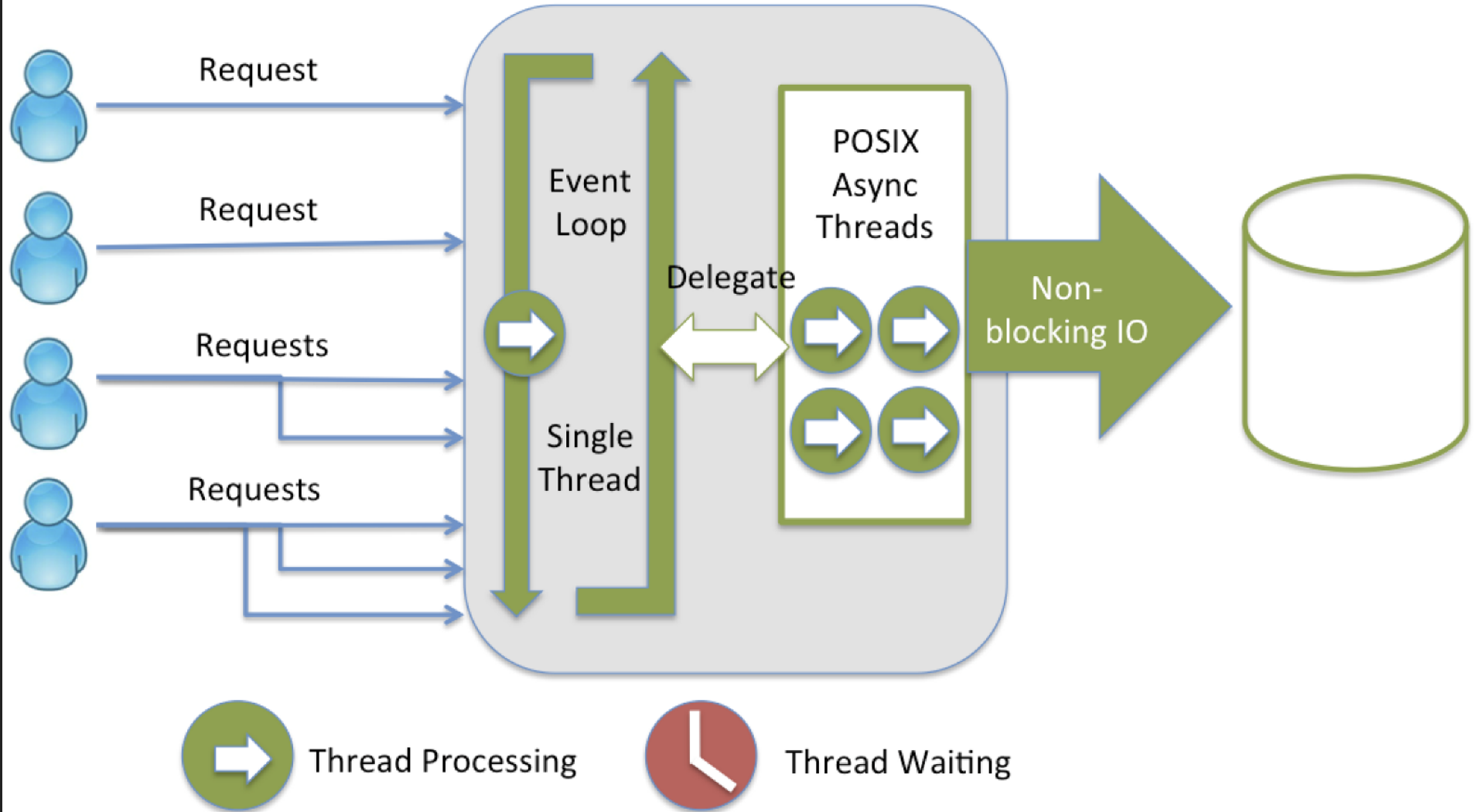
COMO TRABALHAR COM IO?

Multi Threaded Server

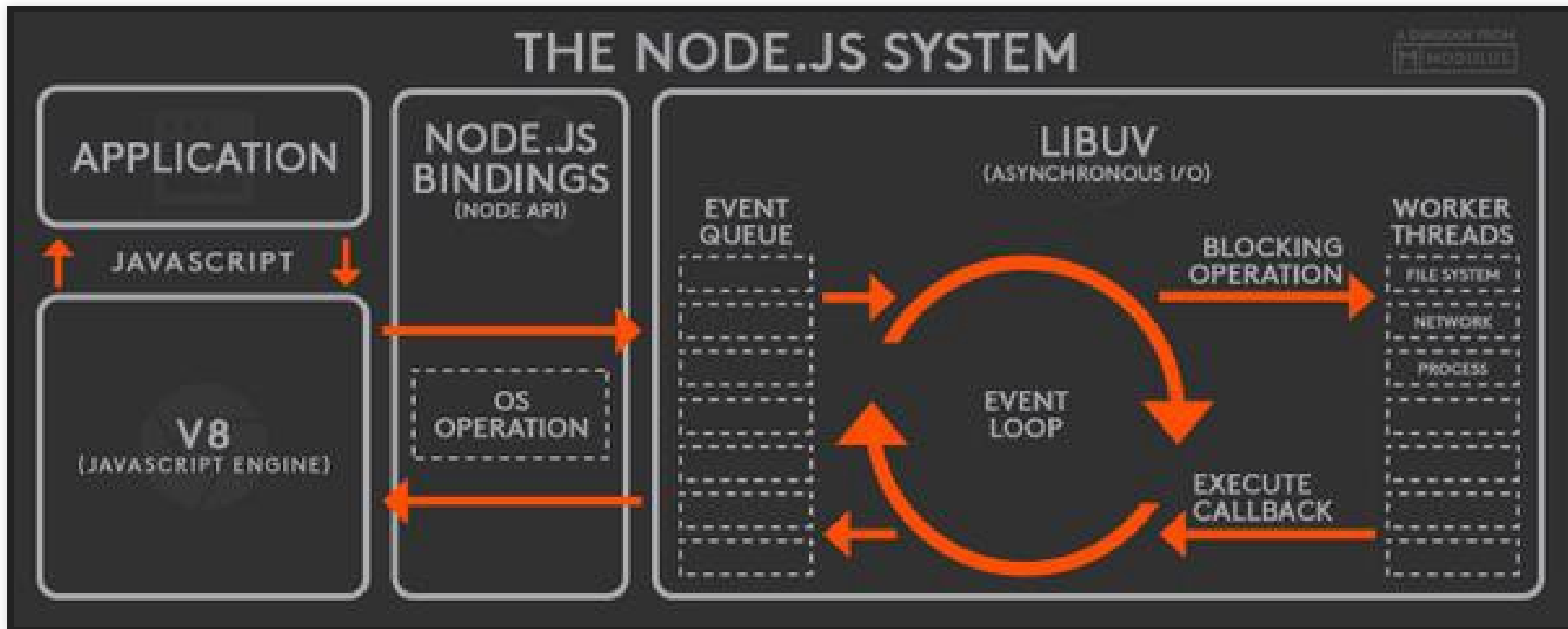


**COMO ISSO PODERIA SER
OTIMIZADO?**

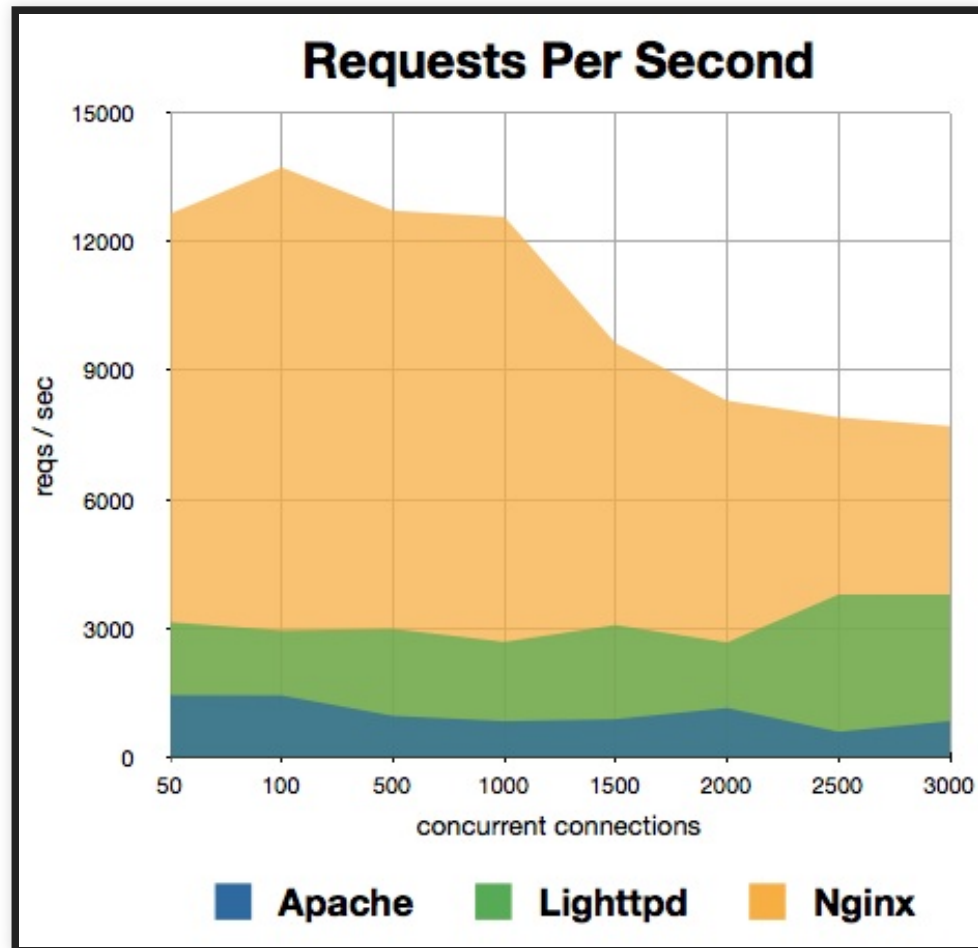
Node.js Server



ARQUITETURA DO NODE



WEB SERVER ORIENTADO A EVENTOS



VAMOS VER ISSO NA PRÁTICA

Vamos utilizar um algoritmo que force o uso de CPU

vamos notar o comportamento do Event Loop

primes-single-thread-server.js

SINGLE THREAD?

Dessa forma, como é possível escalar e atender vários clientes ao mesmo tempo?

Na verdade, existe um thread pool responsável pelo processamento de I/O

files.js

MAS E A ESCALABILIDADE NO EVENT LOOP?

Pode haver balanceador de carga

Mas podemos escalar via clusters (várias instancias!)

`primes-multi-thread.js`

BIBLIOTECA LIBUV



libuv

LIBUV

A libuv é uma biblioteca multi-plataforma responsável pela realização de **I/O assíncrono**, fornecendo a implementação do **event loop** e do thread pool, juntamente com o suporte a TCP e UDP socket, resolução de DNS, sistema de arquivos, processos entre outras.

MAS E PORQUE JAVASCRIPT?

As linguagens C, C++, Lua, haskel possuem muitas bibliotecas sincronas

JS tem a sintaxe amigável e tudo gira em torno de funções

Os navegadores já usavam JS e assíncrono (briga dos navegadores)



SERÁ QUE O NODE TEM PERFORMANCE?

Vamos fazer um teste bem simples, um comparativo
entre java e Javascript

Primes.java vs primes.js



O Node.js uma plataforma de código aberto para a execução de JavaScript no servidor. É composto pela junção do interpretador V8, do Google, com a biblioteca libuv e por um conjunto de módulos.

CORE MODULES

- http
- fs
- stream
- buffer
- net
- crypto
- ... <https://github.com/nodejs/node>

QUEM FINANCIOU

Joyent: Datacenter e cloud computing

Deu certo até certo ponto

A comunidade resolveu criar um fork

Surgiu o IO.js

O REENCONTRO!

Em 09/2015, com a saída da Joyent, io.js v3.3 se juntou com o Node.js v0.12, dando origem ao Node.js v4.0

QUAIS EMPRESAS USAM NODE?



CASES DESTAS GRANDES EMPRESAS

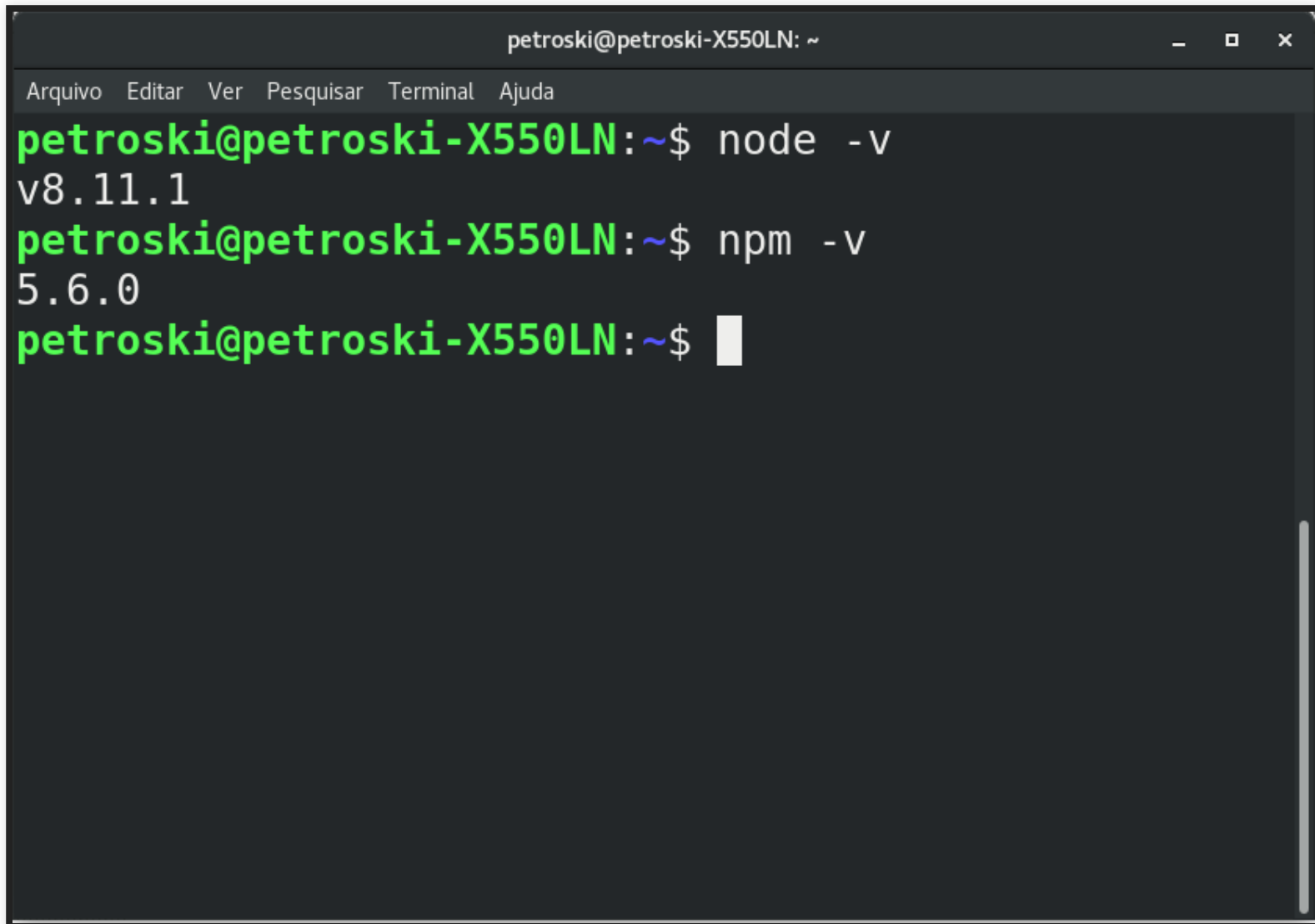
- [Linkedin - Entrevista com Kiran Prasada \(Lider de Desenvolvimento\)](#)
- [Node.js at NetFlix \(video - Inglês\)](#)
- [Node.js IBM \(video - Inglês\)](#)
- [Node.js at Uber \(video - Inglês\)](#)
- [Node.js at PayPal \(video - Inglês\)](#)

COMO INSTALAR

<https://nodejs.org/en/>

Seguir as instruções e instalar a ultima versão LTS

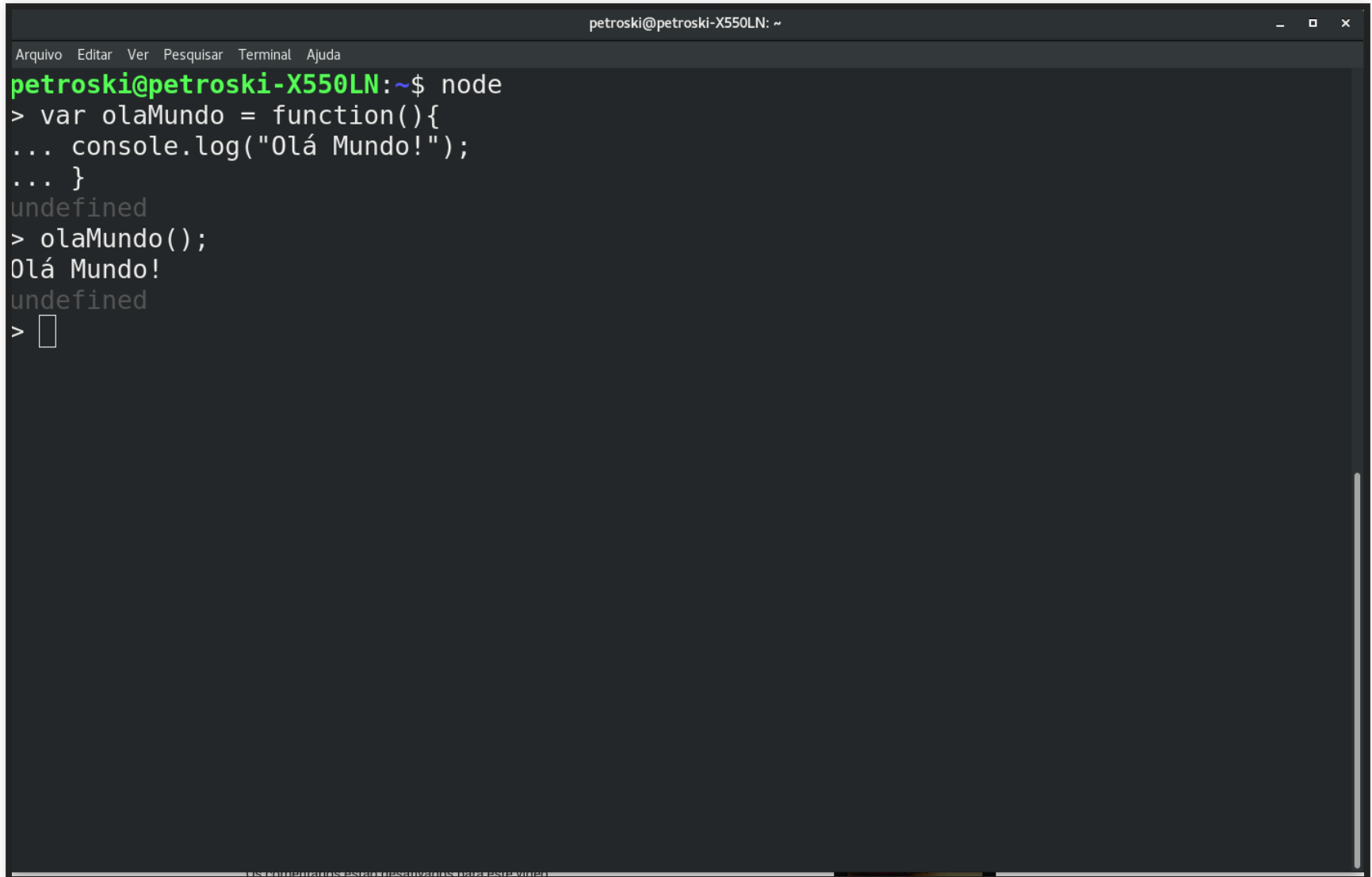
VERIFICAR A INSTALAÇÃO



A terminal window titled "petroski@petroski-X550LN: ~" with a menu bar containing "Arquivo", "Editar", "Ver", "Pesquisar", "Terminal", and "Ajuda". The terminal shows the following commands and outputs:

```
petroski@petroski-X550LN:~$ node -v
v8.11.1
petroski@petroski-X550LN:~$ npm -v
5.6.0
petroski@petroski-X550LN:~$
```

REPL (READ-EVAL-PRINT-LOOP)



```
petroski@petroski-X550LN: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
petroski@petroski-X550LN:~$ node  
> var olaMundo = function(){  
... console.log("Olá Mundo!");  
... }  
undefined  
> olaMundo();  
Olá Mundo!  
undefined  
> 
```

OU EXECUTA ARQUIVOS

hello.js

```
console.log("Olá mundo!");
```

Linha de comando

```
$ node hello.js
```

SUPOORTE AS VERSOES DO ECMASCRIPT

<http://node.green/>

NPM

Assim como o gems do Ruby ou o Maven do Java o Node possui o Node Package Manager

Simplifica o desenvolvimento e gestão de dependências

EXEMPLOS DE COMANDOS NPM

```
$ npm init //cria o arquivo package.json
$ npm install nome_do_modulo //Instala um módulo
$ npm install nome_do_modulo --save //Instala e salva no package.json
$ npm install nome_do_modulo -g //Instala global
$ npm remove nome_do_modulo //remove o pacote --save/g
$ npm update nome_do_modulo //atualiza o pacote
```

```
{
  "name": "app-node",
  "version": "1.2.3",
  "description": "Exemplo de package.json Node",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
  },
  "keywords": [
    "exemplo",
    "node",
    "teste"
  ],
  "author": "Luiz Pedro Petroski",
  "license": "MIT",
}
```

SISTEMAS DE MÓDULOS DO NODE

No Node.js, existe uma relação direta entre um
arquivo e um módulo

AS ESPECIFICAÇÕES DE MÓDULOS FORAM BASEADAS NO PADRÃO COMMONJS

<http://wiki.commonjs.org/wiki/Modules/1.1>



REQUIRE

A função require é responsável por retornar o que foi exportado de um outro módulo

POR EXEMPLO: TEMOS A SEGUINTE FUNÇÃO...

```
var max = 10000;  
var generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

INDEX.JS

```
var serialGenerator = require('./serialGenerator.js');  
console.log(serialGenerator);
```

SERÁ QUE FUNCIONA?

vamos testar?

Tudo que está dentro de um módulo é privado

O módulo precisa explicitar o que será exportado

CORRIGINDO...

```
var max = 10000;  
module.exports.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

CORRIGINDO...

```
var max = 10000;  
exports.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

CORRIGINDO...

```
var max = 10000;  
this.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```


INDEX.JS

```
var serialGenerator = require('./serialGenerator.js');  
console.log(serialGenerator());
```

QUAL A DIFERENÇA ENTRE MODULE.EXPORTS, EXPORTS E THIS?

```
console.log(module.exports === exports);  
console.log(module.exports === this);  
console.log(exports === this);
```

Todos apontam para a mesma referencia

CUIDADO!!

Quem retorna de fato é o module exports

COMO O MÓDULO É LOCALIZADO?

Primeiro, o algoritmo de busca tenta localizar um módulo core. Existem vários como: net, http, url, fs, zlib, crypto, events, stream, os, vm, util, entre outros.

DOCUMENTACAO SOBRE REQUIRE

Se o nome do módulo iniciar com '/', '../' ou './', o algoritmo de busca irá localizar o módulo por meio do caminho absoluto no sistema de arquivos.

Terminar o nome com '.js' é opcional, o algoritmo de busca irá tentar colocar o '.js' no final caso não encontre o módulo.

Se o módulo não começar com '/', '../' ou './' o algoritmo de busca da função require vai procurar dentro da pasta node_modules.

NPM

Ao utilizar o NPM, os módulos instalados são armazenados na pasta `node_modules`

OUTRAS FORMAS DE USAR O EXPORT

```
var max = 10000;  
var generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

POR MEIO DE UM OBJETO

```
var max = 10000;  
module.exports.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

POR MEIO DE UM OBJETO

```
var max = 10000;  
exports.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

POR MEIO DE UM OBJETO

```
var max = 10000;  
this.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```

POR MEIO DE UM OBJETO

```
var max = 10000;  
var generate = function() {  
    return Math.floor(Math.random() * max);  
};  
module.exports={  
    generate:generate  
};
```

POR MEIO DE UM OBJETO!

```
var max = 10000;  
var generate = function() {  
    return Math.floor(Math.random() * max);  
};  
exports={  
    generate:generate  
};
```

POR MEIO DE UM OBJETO!

```
var max = 10000;  
var generate = function() {  
    return Math.floor(Math.random() * max);  
};  
this={  
    generate:generate  
};
```


POR MEIO DE UM OBJETO

```
var createSerialGenerator = Function(){
  var max = 10000;
  var generate = function() {
    return Math.floor(Math.random() * max);
  };
  return {
    generate:generate
  };
}
module.exports = createSerialGenerator();
```

POR MEIO DE UM OBJETO

```
var SerialGenerator = Function(){  
    var max = 10000;  
    this.generate = function() {  
        return Math.floor(Math.random() * max);  
    };  
}  
module.exports = new SerialGenerator();
```

E SE NECESSITAR DOIS OBJETOS DO MESMO MÓDULO?

```
var    serialGeneratorA = require('./serialGenerator');  
var    serialGeneratorB = require('./serialGenerator');  
console.log(serialGeneratorA === serialGeneratorB);
```

O objeto será o mesmo: Cache

POR MEIO DE UM OBJETO

```
var createSerialGenerator = Function(){
  var max = 10000;
  var generate = function() {
    return Math.floor(Math.random() * max);
  };
  return{
    generate: generate
  }
}
module.exports = createSerialGenerator;
```

INDEX.JS

```
var createSerialGenerator = require('./serialGenerator.js');  
var      serialGeneratorA = createSerialGenerator();  
var serialGeneratorB = createSerialGenerator();  
console.log(serialGeneratorA===serialGeneratorB);
```

EXERCÍCIO DE SALA

- Calcular o valor do IPVA de um carro
- Utilizar dois arquivos de módulos, um para criar os objetos carros e outro para a fórmula do IPVA
- O carro deve conter: Marca, modelo, preço, ano de fabricação e se é utilitário ou não
- Carros com mais de 20 anos não pagam IPVA
- Carros utilitários pagam uma alíquota menor

VARIÁVEIS GLOBAIS

```
var max = 10000;  
var serialGenerator = require('./serialGenerator.js');  
console.log(serialGenerator());
```

SERÁ QUE O MÓDULO TEM ACESSO A VARIÁVEL GLOBAL?

```
module.exports.generate = function() {  
    return Math.floor(Math.random() * max);  
};
```


ESCOPO GLOBAL

```
console.log(global);
```

```
console.log(Object.keys(global));
```

Evite poluir o escopo global

MANEIRAS DE CRIAR VARIÁVEIS GLOBAIS

```
global.max = 10000;  
GLOBAL.max = 10000;  
root.max = 10000;
```

VARIÁVEIS GLOBAIS

```
global.max = 10000;  
var serialGenerator = require('./serialGenerator.js');  
console.log(serialGenerator());
```

AGORA SIM

```
module.exports.generate = function() {  
    return Math.floor(Math.random() * global.max);  
};
```

TAMBÉM É POSSÍVEL CRIAR UMA VARIÁVEL GLOBAL SEM A PALAVRA VAR

```
max = 10000;  
var serialGenerator = require('./serialGenerator.js');  
console.log(serialGenerator());
```

É POSSÍVEL CRIAR VARIÁVEIS ACESSÍVEIS EM VÁRIOS MÓDULOS, SEM SER GLOBAL?

Sim!! Com um módulo de configuração (config.js por exemplo)

CONFIG.JS

```
exports.max = 10000;
```

SERIALGENERATOR.JS

```
var config = require('./config');  
module.exports.generate = function() {  
    return Math.floor(Math.random() * config.max);  
};
```


VAMOS PRATICAR UM POUCO?

Instalar o pacote learnyounode

```
$ npm install -g learnyounode
```

UM FRAMEWORK RÁPIDO E MINIMALISTA PARA NODE

Instalar o pacote express-generator

```
sudo npm install express-generator -g
```

JÁ DEU PARA NOTAR O POTENCIAL DO NODE?

Consegue imaginar uma aplicação real em node?

Então mãos a obra!

PROJETO FINAL DA DISCIPLINA

Projeto de tema livre!

Grupos de 3 a 4 integrantes

Obrigatório uso do ExpressJS e Mongoose (próxima aula)

Não será necessário o projeto completo. Basta um crud funcionando completo

REFERÊNCIAS

- WILSON, Mike. Construindo Aplicações Node com MngoDB e Backbone. Rio de Janeiro: Editora Novatec, 2013
- PEREIRA, Caio Ribeiro. Aplicações Web real-time com NodeJS. Casa do Código, 2013, ISBN: 9788566250145
- FREEMAN, Eric, ROBSON, Elisabeth. Use a cabeça! Programação em HTML5. Editora AltaBooks, 2014, ISBN 9788576088455

Estes slides foram elaborados baseado no material de
Rodrigo Branas