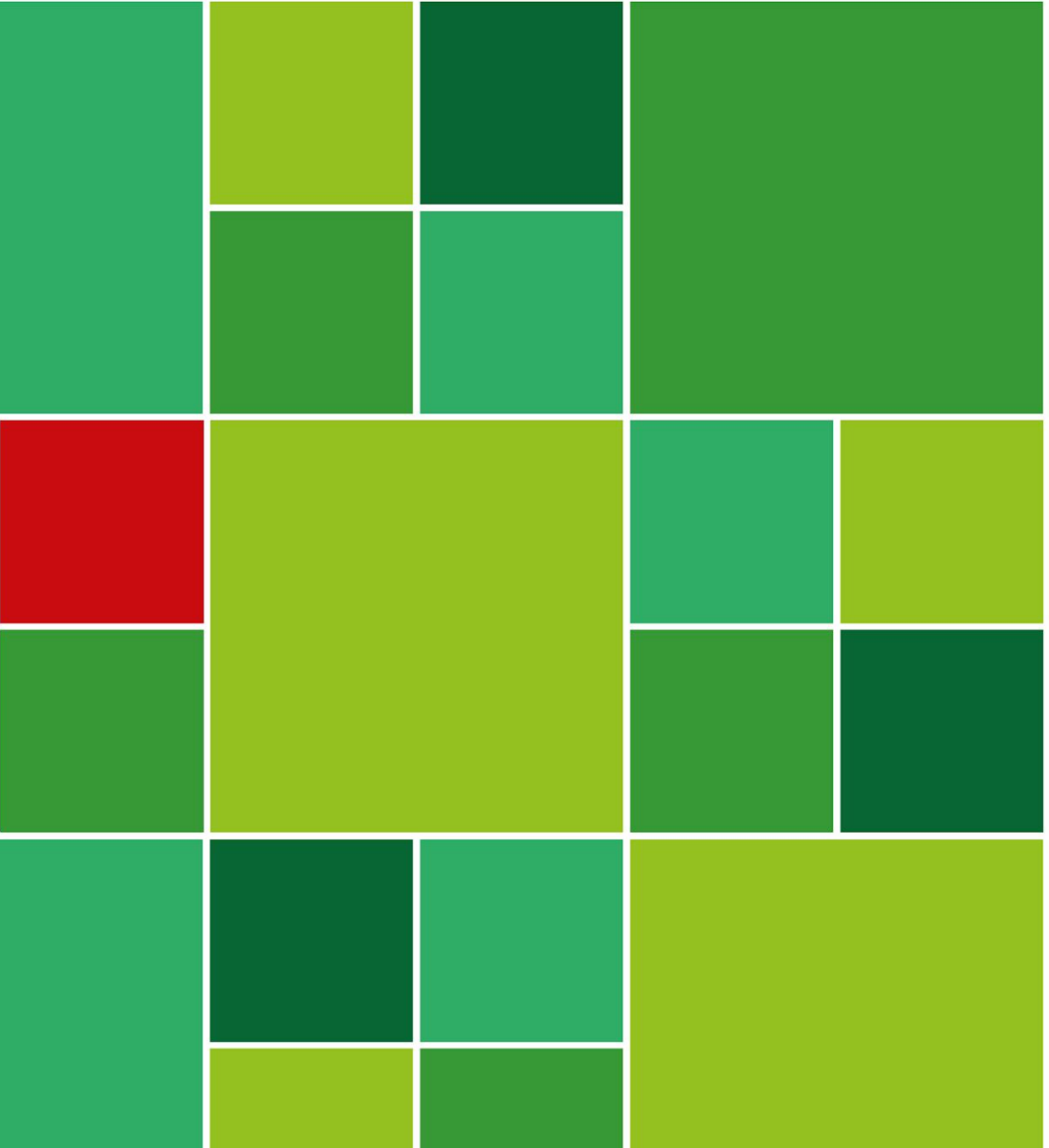


Programação WEB III

NodeJS

Prof. Dr. Danilo Barbosa

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Conteúdo

- Conceito
- npm
- Módulos do node
- CommonJS
- Exercicio
- Referências


node
JS

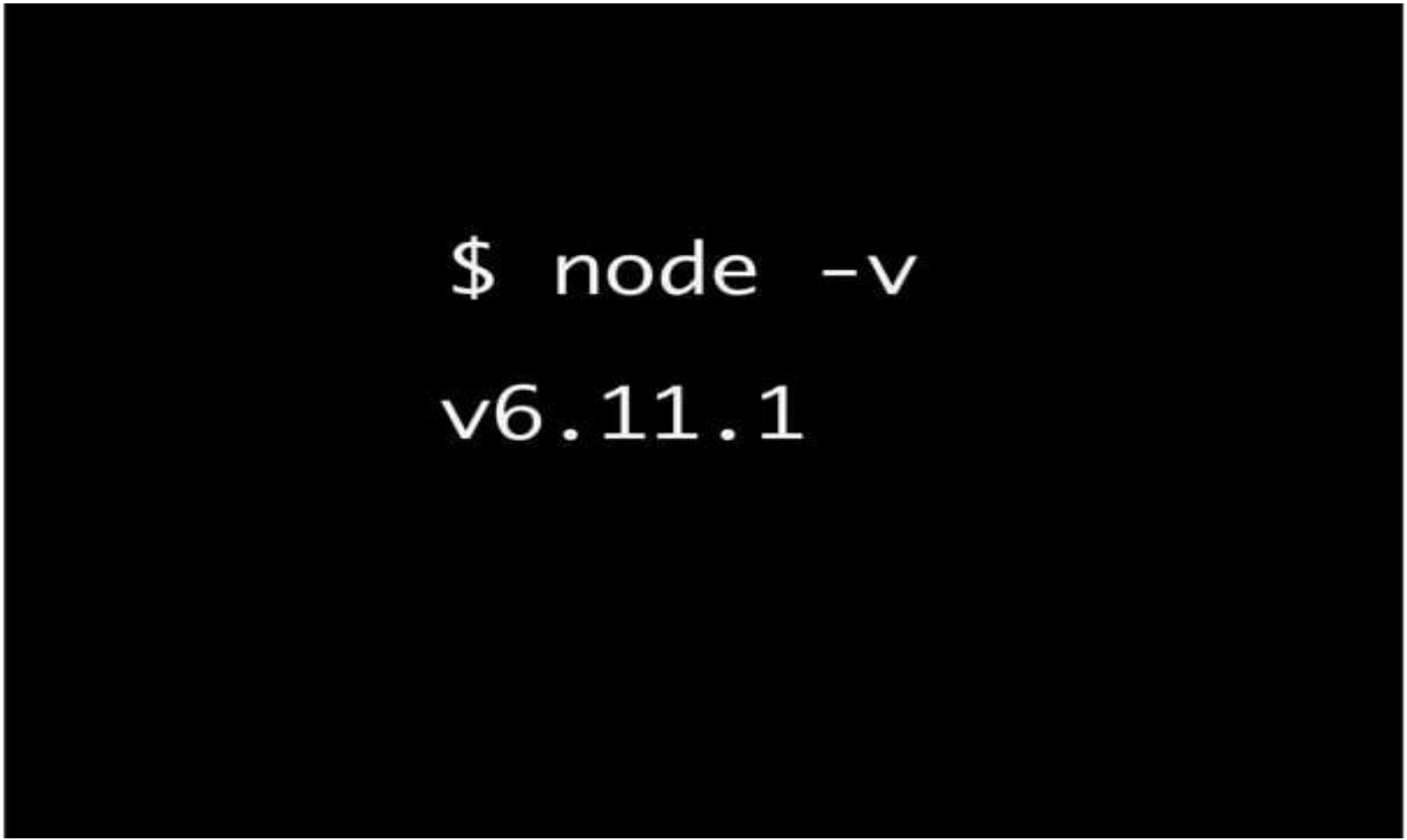
Node JS

"é uma plataforma para aplicações JavaScript criada por Ryan Dahl sob o ambiente de execução JavaScript do Chrome.

É possível utilizar bibliotecas desenvolvidas pela comunidade através de seu gerenciador de pacotes chamado *npm*"

Node JS

www.nodejs.org


$ node -v
v6.11.1

Configurando o ambiente

Adicionar uma variável de ambiente NODE_ENV no sistema operacional.

Linux ou OSX:

1. acessar com um editor de texto qualquer e em modo super user (**sudo**) o arquivo **.bash_profile** ou **.bashrc**
2. adicionar o seguinte comando: **export NODE_ENV='development**

No Windows:

1. Clique com **botão direito** no ícone **Meu Computador** e selecione a opção **Propriedades**
2. Clique em **Configurações avançadas do sistema**.
3. Na janela seguinte, acesse a aba **Avançado** e clique no botão **Variáveis de Ambiente**.
4. No campo **Variáveis do sistema** clique no botão **Novo**.
5. Em nome da variável digite **NODE_ENV** e em valor da variável digite: **development**.


$ node
> console.log('Hello World')
Hello World
undefined
>

npm

Node Package Manager

Comandos do npm

- Gerenciador de pacotes do node.js;
- Similar ao **Maven** do **Java** ou ao **Gems** do **Ruby;**
- Foi integrado ao instalador do Node.js a partir da versão **0.6.**

Comandos do npm

npm install nome_do_módulo: instala um módulo no projeto;

npm install -g nome_do_módulo: instala um módulo global;

npm install nome_do_módulo **--save**: instala o módulo no projeto, atualizando o package.json na lista de dependências;

npm list: lista todos os módulos do projeto;

npm list -g: lista todos os módulos globais;

Comandos do npm

npm remove nome_do_módulo: desinstala um módulo do projeto;

npm remove -g nome_do_módulo: desinstala um módulo global;

npm update nome_do_módulo: atualiza a versão do módulo;

npm update -g nome_do_módulo: atualiza a versão do módulo global;

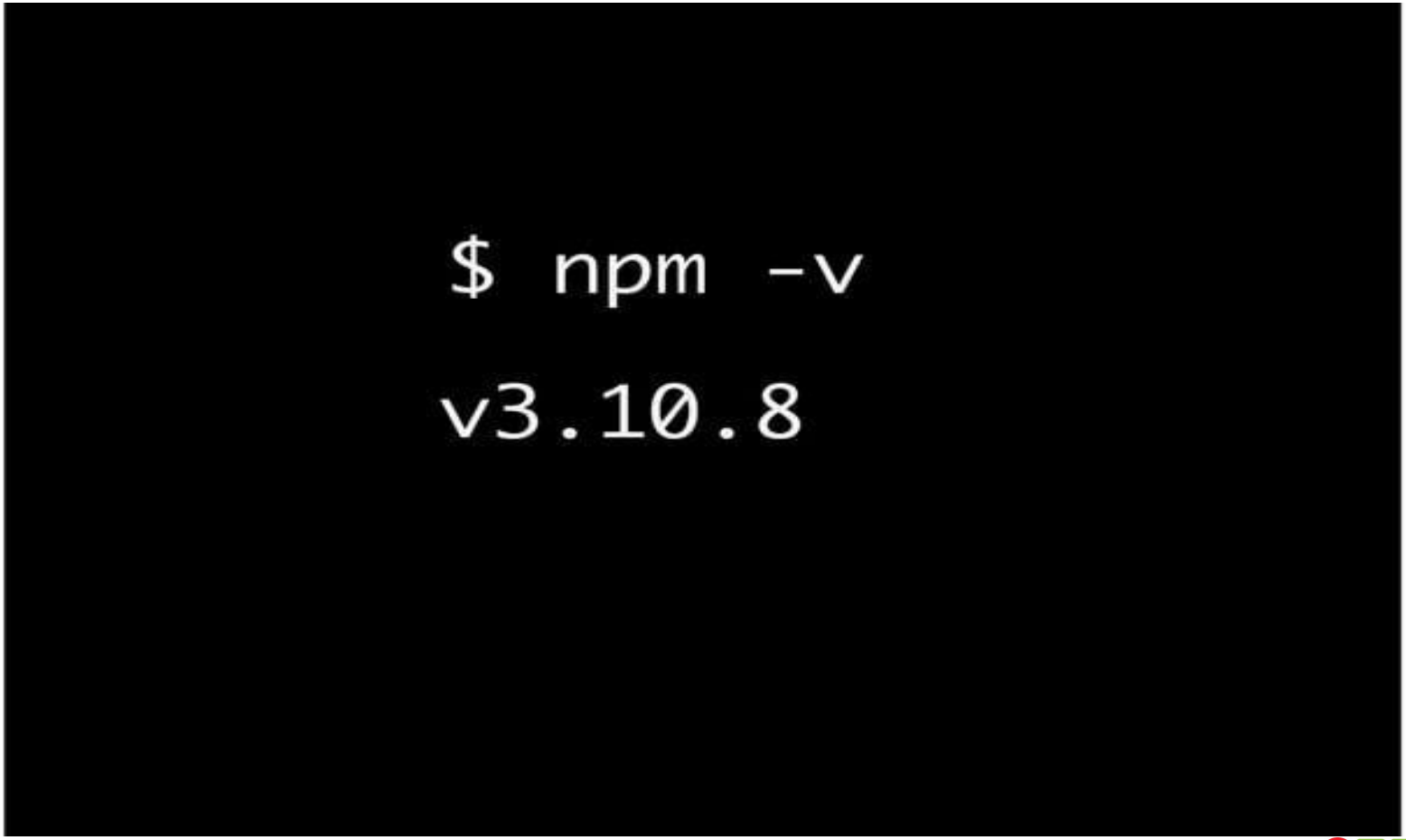
Comandos do npm

npm -v: exibe a versão atual do npm;

npm adduser nome_do_usuário: cria uma conta no npm, através do site https://npmjs.org.

npm whoami: exibe detalhes do seu per l público npm (é necessário criar uma conta antes);

npm publish: publica um módulo no site do npm (é necessário ter uma conta antes).


$ npm -v
v3.10.8

Criação de um projeto Node

Com o terminal aberto na pasta criada para a construção do projeto, iniciaremos a criação de um novo projeto Node com o comando de terminal npm init -y, sendo -y de yes.

npm init -y

Criação de um projeto Node

Feito isso, vamos abrir o Visual Studio Code na pasta do projeto, onde o arquivo JSON foi criado com as informações básicas.

O arquivo package.json é o arquivo principal de qualquer projeto em Node.js. É utilizado como arquivo manifesto, centralizando todas as informações necessárias para entender os componentes de um projeto

Criação de um projeto Node

package.json:

```
 {
  "name": "3266-express-mongo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Criação de um projeto Node

A única coisa que precisamos fazer neste momento no arquivo package.json é, em qualquer parte do objeto, adicionar uma linha com a propriedade type que será do tipo module.

```
{
  "name": "3266-express-mongo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",

// código omitido
```

Criação de um projeto Node

A propriedade type definida como module será usada para importar e exportar as partes, as funções, os módulos do nosso projeto, isto é, da nossa API, utilizando a sintaxe mais moderna do JavaScript.

Criação de um projeto Node

Package.json

```
{
 "name": "alura-curso-node",
 "version": "1.0.0",
 "description": "Módulo de primeiros passos no curso de Node.js",
 "main": "index.js",
 "scripts": {
   "start": "node index.js",
   "test": "jest"
 },
 "author": "Juliana",
 "license": "ISC",
 "dependencies": {
   "modulo-1": "^1.0.0",
   "modulo-2": "^1.0.3",
   "modulo-3": "^2.1.0"
 },
 "devDependencies": {
   "nodemon": "^2.0.15",
   "jest": "^27.2.1"
 }
}
```

Criação de um projeto Node

Criação do servidor

A primeira coisa que fazemos quando vamos criar uma API que precisa fornecer informações para outras partes do sistema é criar um servidor para justamente fornecer os dados, servindo como ponto de conexão.

o arquivo server.js um conjunto de rotas através de um objeto. Para isso, criaremos abaixo de PORT uma constante chamada rotas, que receberá uma abertura de chaves ({}).

Criação de um projeto Node

Arquivo server.js

```
import http from "http";

const PORT = 3000;

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end("Curso de Node.js");
});

server.listen(PORT, () => {
  console.log("servidor escutando!");
});
```

Criação de um projeto Node

Arquivo server.js

```
import http from "http";

const PORT = 3000;

const rotas = {
  "/": "Curso de Express API",
  "/livros": "Entrei na rota livros",
  "/autores": "Entrei na rota autores"
};

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end(rotas[req.url]);
});

server.listen(PORT, () => {
  console.log("servidor escutando!");
});
```

Criação de um projeto Node

Executando o arquivo

Agora precisamos apenas executar o arquivo e verificar se o nosso servidor está no ar e servindo arquivos. Para isso, vamos retornar ao terminal na pasta correta e pedir para o Node executar server.js com o comando:

```
node server.js
```

Criação de um projeto Node

Criando o arquivo .gitignore

Agora que já temos rotas para a nossa API, vamos criar um novo arquivo na raiz e chamá-lo de .gitignore. Dentro dele, vamos inserir a informação node_modules e salvar.

.gitignore:

node_modules

Criação de um projeto Node

Marcos está criando um novo projeto no Node.js e resolveu usar o módulo HTTP para auxiliá-lo na criação do servidor e suas rotas. Depois de usar o comando import http from “http”; para carregar o módulo HTTP, qual seria a forma adequada de criar o servidor?

Criação de um projeto Node

```
const server = http.listening((req, res) =>{ /* implementar o código */  });
```

```
const server = http.createServer((req, res) => { /* implementar o código */
```

```
const server = http.listen((req, res) =>{ /* implementar o código */  });
```

node modules

Modulos do Node

Todo projeto **Node.js** é chamado de **módulo**;

O termo **módulo**, **biblioteca** e **framework** possuem o mesmo significado (na prática);

O termo módulo surgiu do conceito de que a **arquitetura** do Node.js é **modular;**

Todo módulo é acompanhado de um arquivo descritor (**package.json**).

Package.json

```
{
   "name": "winter-is-coming-node-app",
   "description": "Meu primeiro app na Muralha",
   "author": "Jon Snow <jonsnow@norte.com>",
   "version": "1.2.3",
   "private": true,
   "dependencies": {
     "modulo-1": "1.0.0",
     "modulo-2": "~1.0.0",
     "modulo-3": ">=1.0.0"
   },
   "devDependencies": {
    "modulo-4": "*"
   }
}
```

CommonJS

CommonJS

- O Node.js utiliza nativamente o padrão ***CommonJS*** para organização e carregamento de módulos;
- Para criar um código Javascript que seja modular e carregável pelo **require**, utilizam-se as variáveis globais: **exports** ou **module.exports**.

hello.js

```
module.exports = function(msg) {
  console.log(msg);
};
```

Em **hello.js** carregamos uma única função modular;

human.js

```
exports.hello = function(msg) {
  console.log(msg);
};
```

Em **human.js** é carregado um objeto com funções modulares.

app.js

```
var hello = require('./hello');
var human = require('./human');

hello('Olá pessoal!');
human.hello('Olá galera!');
```

Exercícios

node modules

1. Crie uma pasta com o nome: node-exemplos
2. Crie um arquivo com o nome retangulo.js

```
var rect = {
  perimeter: function (x, y) {
            return (2*(x+y));
  },
  area: function (x, y) {
             return (x*y);
  }
};

function solveRect(l,b) {
    console.log("Solving for rectangle with l = " + l + " and b = " + b);

    if (l < 0 || b < 0) {
        console.log("Rectangle dimensions should be greater than zero:  l = "
                + l + ",  and b = " + b);
    }
    else {
  console.log("The area of a rectangle of dimensions length = "
                + l + " and breadth = " + b + " is " + rect.area(l,b));
  console.log("The perimeter of a rectangle of dimensions length = "
                + l + " and breadth = " + b + " is " + rect.perimeter(l,b));
    }
}

solveRect(2,4);
solveRect(3,5);
solveRect(-3,5);
```

Execute execute a aplicação com o comando: node retangulo.js

1. Crie um novo arquivo com o nome **retangulo-1.js**

```
exports.perimeter = function (x, y) {
        return (2*(x+y));
}

exports.area = function (x, y) {
        return (x*y);
}
```

2. Crie um novo arquivo com o nome **solucao-1.js**

```
var rect = require('./retangulo-1');

function solveRect(l,b) {
    console.log("Solving for rectangle with l = " + l + " and b = " + b);

    if (l < 0 || b < 0) {
        console.log("Rectangle dimensions should be greater than zero:  l = "
                + l + ",  and b = " + b);
    }
    else {
    console.log("The area of a rectangle of dimensions length = "
                + l + " and breadth = " + b + " is " + rect.area(l,b));
    console.log("The perimeter of a rectangle of dimensions length = "
                + l + " and breadth = " + b + " is " + rect.perimeter(l,b));
    }
}

solveRect(2,4);
solveRect(3,5);
solveRect(-3,5);
```

. Execute execute a aplicação com o comando: **node solucao-1**

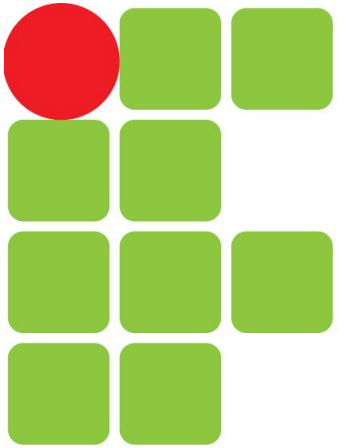
Referências

Node.js Ebook https://github.com/fraxken/ebook_nodejs?tab=readme-ov-file

Node-js - Casa do Codigo https://github.com/free-educa/books/blob/main/books/Aplicacoes%20web%20real%20time%20com%20Node-js%20-%20Casa%20do%20Codigo.pdf

Run JavaScript Everywhere https://nodejs.org/en

Introduction to Node.js (LFW111) https://training.linuxfoundation.org/training/introduction-to-nodejs-lfw111/



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO