

A large, thick green arrow pointing to the right, positioned behind the text "High performance. Delivered."

High performance. Delivered.

# Application Delivery Fundamentals: Node JS

## Module 1: Introdução ao Node JS

-

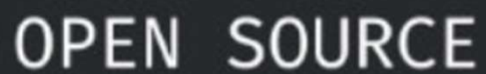




# O que é Node.JS


- **Node.js** é um interpretador de JavaScript de código aberto orientado a eventos, criado por Ryan Dahl em 2009, focado em migrar a programação do Javascript do cliente (**frontend**) para os servidores (**backend**), criando aplicações de alta escalabilidade (como um servidor web);
- O **Node.js** executa por linha de comando;
- Projetado para alta concorrência;
- Single Thread;
- Não bloqueante;

## 8207





# O que é Node.js

- O **Node.js** é orientado a eventos assíncronos. (*Event Driven*);
  - Executa melhor no Linux;
  - Open Source;
  - Criado sobre a **V8** javaScript Engine;
  - É 40% **javaScript** e 60% **C++**
- 



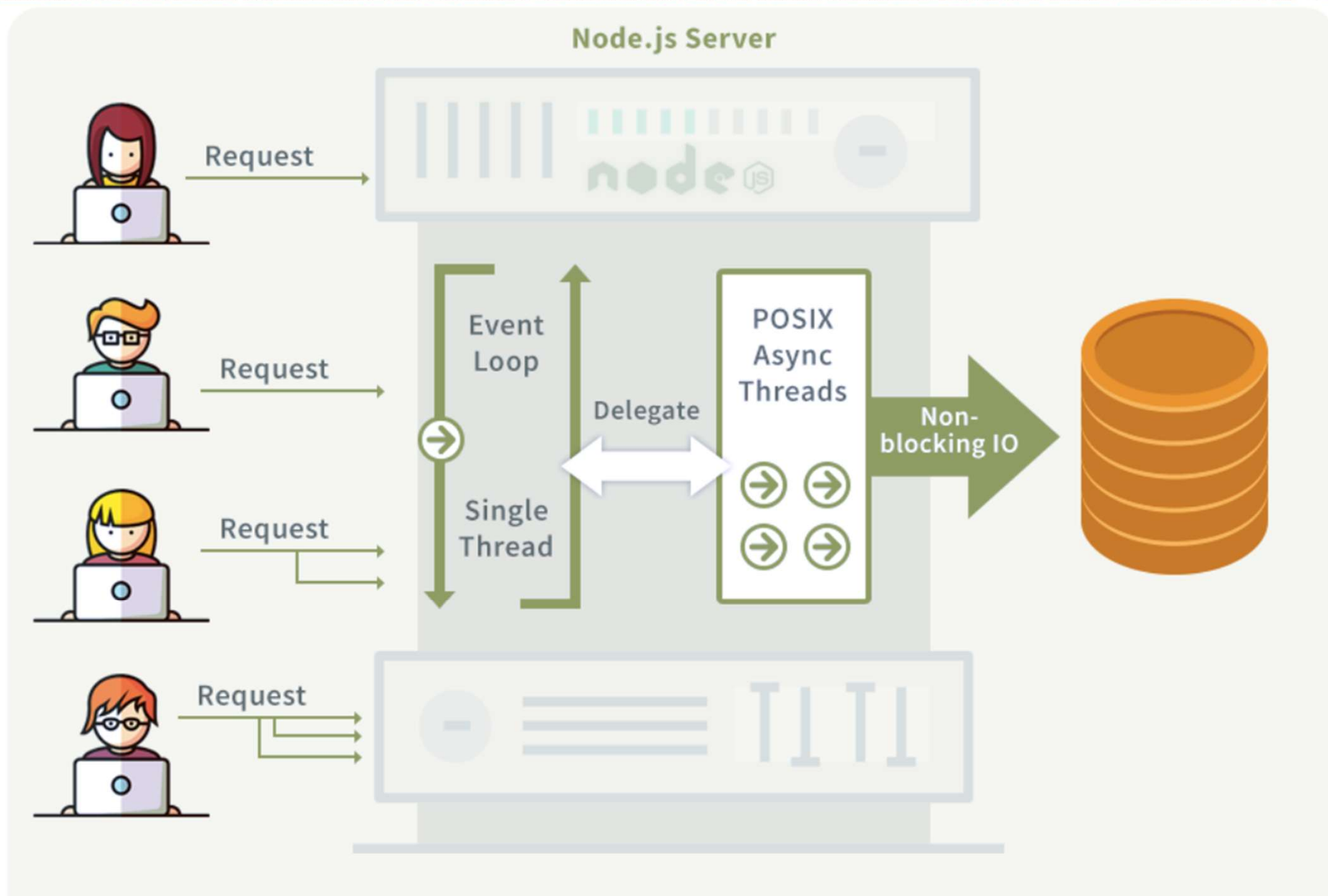
[illegible]

- [illegible]

# O que é o Node.js

2f4869b7e

38207



## 8207

- 
- THE NODE.JS SYSTEM**
- The diagram illustrates the Node.js system architecture, showing the flow of data and control between various components:
- APPLICATION**: The top-level user application.
  - V8 (JAVASCRIPT ENGINE)**: The JavaScript engine that executes the application code.
  - NODE.JS BINDINGS (NODE API)**: The layer that connects the V8 engine to the operating system and other system-level components.
  - LIBUV (ASYNCHRONOUS I/O)**: The library that handles asynchronous I/O operations, including the **EVENT QUEUE** and **EVENT LOOP**.
  - WORKER THREADS**: Threads that execute tasks in parallel, including **FILE SYSTEM**, **NETWORK**, and **PROCESS** operations.
- The flow of data and control is as follows:
- The **APPLICATION** sends **JAVASCRIPT** to the **V8** engine.
  - The **V8** engine sends data to the **NODE.JS BINDINGS**.
  - The **NODE.JS BINDINGS** send data to the **LIBUV** library.
  - The **LIBUV** library handles **ASYNCHRONOUS I/O** operations, including the **EVENT QUEUE** and **EVENT LOOP**.
  - The **LIBUV** library sends data to the **WORKER THREADS**.
  - The **WORKER THREADS** execute tasks and send data back to the **LIBUV** library.



# Node.JS



## Instalando o NODE.js



## Instalando o NODE.js;

- Instale o gerenciador de versão NVM;
- **nvm list available** – Lista as versões disponíveis na nuvem
- **nvm install 18.14.0** - Instala uma versão específica
- **nvm ls** - Lista as versões instaladas
- **nvm uninstall 18.14.0** – Desinstala uma versão;
- O programa **NPM** é instalado quando você instala o **Node.js**.
- Instala as dependências na pasta **node\_modules** local e atualiza o **package.json**. Por padrão, o **npm install** instalará todos os módulos listados como dependências no **package.json**



# Node.js

Command Prompt

```
C:\Devel\node>nvm list available
```

CURRENT	LTS	OLD STABLE	OLD UNSTABLE
17.8.0	16.14.2	0.12.18	0.11.16
17.7.2	16.14.1	0.12.17	0.11.15
17.7.1	16.14.0	0.12.16	0.11.14
17.7.0	16.13.2	0.12.15	0.11.13
17.6.0	16.13.1	0.12.14	0.11.12
17.5.0	16.13.0	0.12.13	0.11.11
17.4.0	14.19.1	0.12.12	0.11.10
17.3.1	14.19.0	0.12.11	0.11.9
17.3.0	14.18.3	0.12.10	0.11.8
17.2.0	14.18.2	0.12.9	0.11.7
17.1.0	14.18.1	0.12.8	0.11.6
17.0.1	14.18.0	0.12.7	0.11.5
17.0.0	14.17.6	0.12.6	0.11.4
16.12.0	14.17.5	0.12.5	0.11.3
16.11.1	14.17.4	0.12.4	0.11.2
16.11.0	14.17.3	0.12.3	0.11.1
16.10.0	14.17.2	0.12.2	0.11.0
16.9.1	14.17.1	0.12.1	0.9.12
16.9.0	14.17.0	0.12.0	0.9.11
16.8.0	14.16.1	0.10.48	0.9.10

This is a partial list. For a complete list, visit <https://nodejs.org/download/release>

```
C:\Devel\node>
```



# Modulo Node.JS

- Todo projeto **Node.js** é chamado de **modulo**;
- O termo modulo surgiu da arquitetura do **Node.js** que é modular;
- Todo modulo é acompanhado de um arquivo descritor;
- Deve estar presente na raiz do projeto; (**package.json**)
- Este arquivo descreve os metadados do projeto;

## 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": "*"
  }
}
```



## 58207

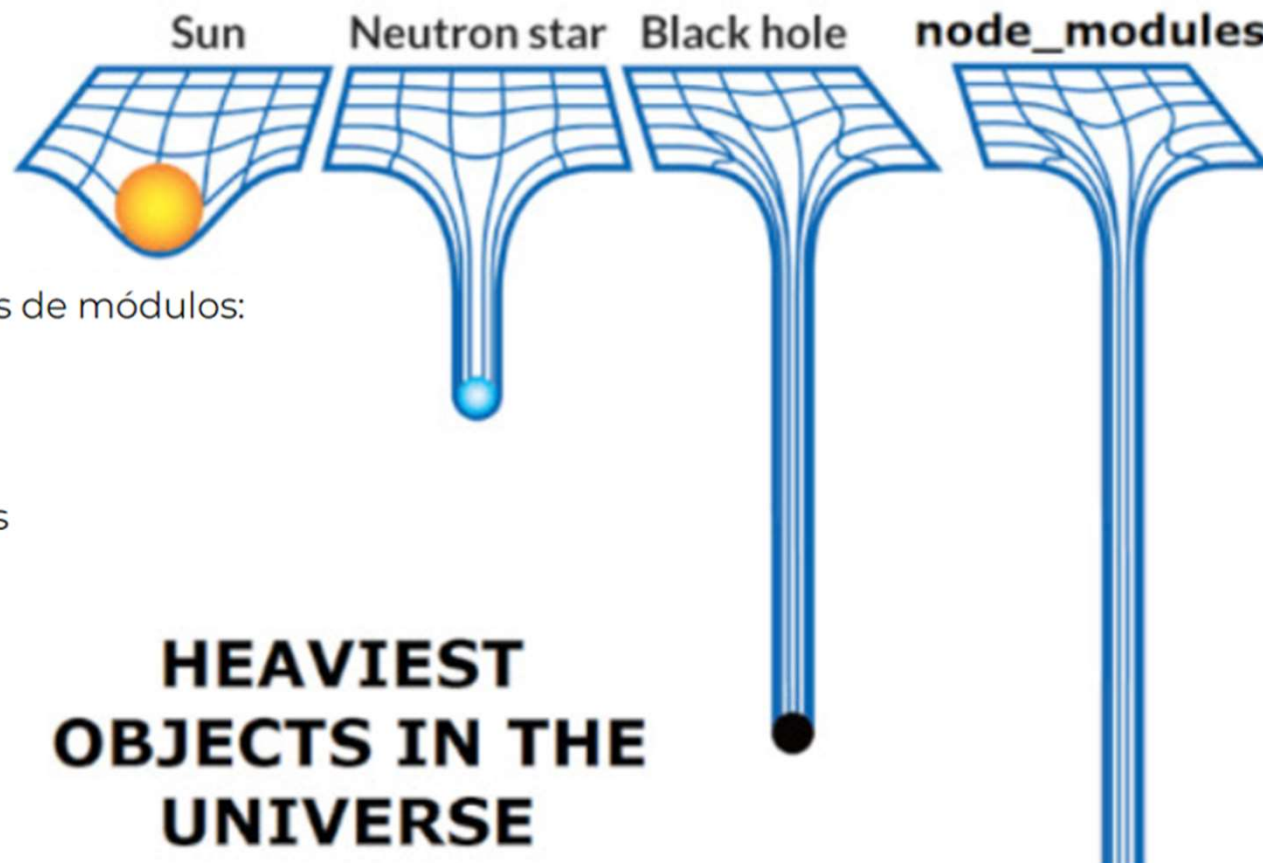


## 58207

### 3. Execute à aplicação: node retangulo.js

# MODULO

- Módulos são blocos de código reutilizáveis que podem ser exportados de um arquivo e importados para outro.
- Eles permitem que você organize seu código em partes menores e mais gerenciáveis, promovendo a reutilização e a manutenção.



O Node.js inclui três tipos de módulos:

- Módulos Core
- Módulos Locais
- Módulos de Terceiros

# Export

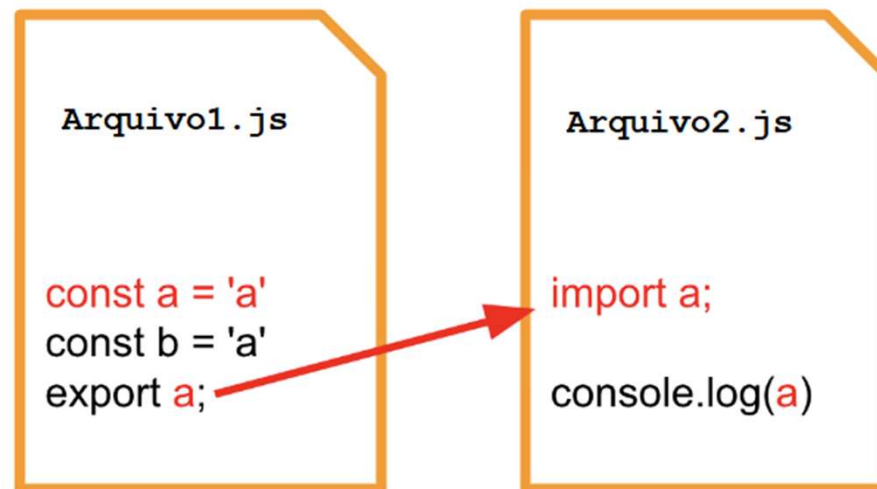
- Um módulo pode ser **exportado**;
- O que possibilita que ele seja importado por outros módulos;
- EXPORTANDO
  - `module.exports{<nome do módulo>}`
- IMPORTANDO
  - `var <varName> = require(<nome do módulo>)`





# Require

- Os módulos disponíveis no node podem ser importados para o arquivo .js utilizando a função “**require**”;
- A função **require**, ajuda na importação de módulos;





# Require

/projet

application.js

```
require('./module');  
require('module2');
```

module.js

```
/* JS code for the  
module */
```

/projet/node\_modules

module2.js

```
/* JS code for the  
module */
```



# Import

- Exemplo 1;

Message.js

```
module.exports = 'Hello world';
```

app.js

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

```
C:\> node app.js  
Hello World
```



# Import

- Exemplo 2;

data.js

```
module.exports = {  
  firstName: 'James',  
  lastName: 'Bond'  
}
```

- appData.js

```
var person = require('./data.js');  
console.log(person.firstName + ' ' + person.lastName);
```

- Node appData.js





# Import

- Exemplo 3;

Log.js

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

- appLog.js

```
var msg = require('./Log.js');  
  
msg('Hello World');
```

- Node appLog.js



# Import

- Exemplo 4;

Person.js

```
module.exports = function (firstName, lastName) {  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.fullName = function () {  
    return this.firstName + ' ' + this.lastName;  
  }  
}
```

- appPerson.js

```
var person = require('./Person.js');  
  
var person1 = new person('James', 'Bond');  
  
console.log(person1.fullName());
```

- Node appPerson.js

# Import

- Exemplo 5;
- Modulo1.js

```
console.log("Hello Accenture");  
module.exports = ()=> console.log("Node.js é o melhor back-end que existe !!");
```

- Modulo2.js

```
const mod = require('./modulo1.js');
mod();
mod();
mod();
```

- Node modulo2.js







## headline:

- ```
const readline = require('readline');
const readline = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

readline.question('Qual o seu nome? ', name => {
  console.log(`Ola ${name}!`);
  readline.close();
});
```

## prompt-sync:

- ```
const num = prompt('Informe um número: ');
console.log('Seu numero + 4 =');
console.log(Number(num) + 4);
```



# Input via teclado

## prompt-sync:

```
C:\Treinamentos\Curso\Node.js\Código\Exemplos 02>node teclado2.js
node:internal/modules/cjs/loader:1147
  throw err;
  ^

Error: Cannot find module 'prompt-sync'
Require stack:
- C:\Treinamentos\Curso\Node.js\Código\Exemplos 02\teclado2.js
  at Module._resolveFilename (node:internal/modules/cjs/loader:1144:15)
  at Module._load (node:internal/modules/cjs/loader:985:27)
  at Module.require (node:internal/modules/cjs/loader:1235:19)
  at require (node:internal/modules/helpers:176:18)
  at Object.<anonymous> (C:\Treinamentos\Curso\Node.js\Código\Exemplos 02\teclado2.js:1:16)
  at Module._compile (node:internal/modules/cjs/loader:1376:14)
  at Module._extensions..js (node:internal/modules/cjs/loader:1435:10)
  at Module.load (node:internal/modules/cjs/loader:1207:32)
  at Module._load (node:internal/modules/cjs/loader:1023:12)
  at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:135:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [
    'C:\\Treinamentos\\Curso\\Node.js\\Código\\Exemplos 02\\teclado2.js'
  ]
}

Node.js v20.10.0

C:\Treinamentos\Curso\Node.js\Código\Exemplos 02>
```



## prompt-sync:

- Npm install prompt-sync

```
C:\Treinamentos\Curso\Node.js\Código\Exemplos 02>npm install prompt-sync  
  
added 3 packages in 13s  
  
C:\Treinamentos\Curso\Node.js\Código\Exemplos 02>
```

# Atividade

- Alterar o **solucao-1.js**
- Obter via teclado os valores para cálculo do:
  - Perímetro do retângulo.
  - Área do retângulo.

