Aula Node

# O que é ?

O Node.js pode ser definido como um ambiente de execução Javascript ***server-side***.

**Node.js** é um [interpretador](https://pt.wikipedia.org/wiki/Interpretador_de_JavaScript) de [JavaScript](https://pt.wikipedia.org/wiki/JavaScript) [assíncrono](https://pt.wikipedia.org/wiki/XMLHttpRequest) com [código aberto](https://pt.wikipedia.org/wiki/C%C3%B3digo_aberto) [orientado a eventos](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_orientada_a_eventos), criado por [Ryan Dahl](https://pt.wikipedia.org/w/index.php?title=Ryan_Dahl&action=edit&redlink=1) em 2009, focado em migrar a programação do Javascript [do cliente](https://pt.wikipedia.org/wiki/Lado_cliente) (*frontend*) para os [servidores](https://pt.wikipedia.org/wiki/Server-side), criando aplicações de [alta escalabilidade](https://pt.wikipedia.org/wiki/Escalabilidade) (como um servidor [web](https://pt.wikipedia.org/wiki/Web)), manipulando milhares de conexões/eventos simultâneas em tempo real numa única máquina física.

O Node.js (ambiente de execução Javascript no servidor) foi implementado baseado no interpretador [V8](https://pt.wikipedia.org/wiki/V8_(JavaScript)) *JavaScript Engine* (interpretador de JavaScript em [C++](https://pt.wikipedia.org/wiki/C%2B%2B) com código aberto do Google, utilizado no Chrome), com desenvolvimento mantido pela fundação Node.js em parceria com a [Linux Foundation](https://pt.wikipedia.org/wiki/Linux_Foundation).

Apesar de recente, o Node.js já é utilizado por grandes empresas no mercado de tecnologia, como **Netflix**, **Uber** e **LinkedIn**.

# Características

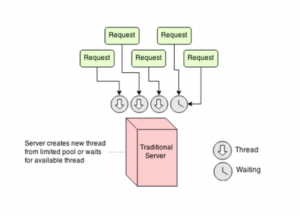
A principal característica e diferença de outras tecnologias (como [PHP](https://pt.wikipedia.org/wiki/PHP), [Java](https://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o)), [C#](https://pt.wikipedia.org/wiki/C_(linguagem_de_programa%C3%A7%C3%A3o))) é a execução das requisições/eventos em [*single-thread*](https://en.wikipedia.org/wiki/single_thread)*,* onde apenas uma [*thread*](https://pt.wikipedia.org/wiki/Thread_(computa%C3%A7%C3%A3o)) (chamada de *Event Loop*) é responsável por executar o código Javascript, sem a necessidade de criar nova *thread* que utilizaria mais recursos computacionais (por exemplo memória RAM) e sem o uso da fila de espera.

Pense na thread como uma área que foi reservada em seu processador para tratar/processar alguma informação.

# Mas o que isso significa na prática?

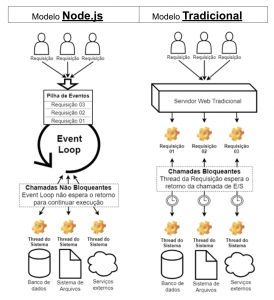
Em um servidor web utilizando linguagens tradicionais, para cada requisição recebida é criada uma nova ***thread*** para tratá-la. A cada requisição, serão demandados recursos computacionais (memória RAM, por exemplo) para a criação dessa nova ***thread***. Uma vez que esses recursos são limitados, as ***threads*** não serão criadas infinitamente, e quando esse limite for atingido, as novas requisições terão que esperar a liberação desses recursos alocados para serem tratadas.

A figura ao lado representa esse cenário em um servidor tradicional.

No modelo Node.js, apenas uma ***thread*** é responsável por tratar as requisições. Essa thread é chamada de ***Event Loop***, e leva esse nome pois cada requisição é tratada como um evento. O ***Event Loop*** fica em execução esperando novos eventos para tratar, e para cada requisição, um novo evento é criado.

Apesar de ser ***single-threaded***, é possível tratar requisições concorrentes em um servidor Node.js. Enquanto o servidor tradicional utiliza o sistema ***multi-thread*** para tratar requisições concorrentes, o Node.js consegue o mesmo efeito através de chamadas de E/S (entrada e saída) **não-bloqueantes**. Isso significa que as operações de entrada e saída (ex: acesso a banco de dados e leitura de arquivos do sistema) são assíncronas e não bloqueiam a ***thread***. Diferentemente dos servidores tradicionais, a ***thread*** não fica esperando que essas operações sejam concluídas para continuar sua execução.

A figura abaixo representa a diferença de funcionamento de um servidor web tradicional e um Node.JS:



# Vantagens de Uso

* Flexibilidade: O Node.js possui o gerenciador de pacotes reusáveis NPM (*Node Package Manager*), que também é o maior repositório de softwares do mundo, dando ao interpretador um potencial a ser utilizada em qualquer situação. O pacote mais conhecido é o Express.js, um framework completo para desenvolvimento de aplicações Web.
* Leveza: Um ambiente Node.js não exige muitos recursos computacionais tradicionais. Se utilizado em conjunto com a ferramentas [Docker](https://pt.wikipedia.org/wiki/Docker_(software)), o ganho na velocidade de *deploy* e replicação de máquinas pode ser muito significativo e, em ambientes escaláveis, de micro-serviços e *serverless*, representa menos custo e mais eficiência. Um exemplo de uso é uma aplicação de conversa/bate papo (webchat).
* Suporte: Conta com suporte das principais empresas de serviços de [armazenamento na nuvem](https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_em_nuvem), como a AWS, Google Cloud e, Microsoft Azure, com suporte nativo ao Node.JS.

### Produtividade:

* + O gerenciador NPM fornece pacotes reusáveis disponível gratuitamente para uso em grupo.
  + Mesma linguagem no **frontend** e **backend**, que pode representar ganhos de reuso de código e criação de equipes multidisciplinares, reaproveitando recursos.
  + A possibilidade de deploys e iterações mais rápidas, e resolução de problemas "*on the fly",* também permitem a criação de soluções próprias e inovadoras, como a Uber faz para resolver alguns problemas.

# Casos de uso mais comuns

## Aplicações em Tempo Real

Um exemplo comum é uma aplicação de conversa (chat). Tal aplicação exige muito pouco processamento e basicamente consiste em transferir as mensagens de um lado para outro.

## Ambientes Escaláveis

O Node.js é bastante indicado para ambientes escaláveis (com grande número de conexões concorrentes), já que tem potencial para suportar um número maior de conexões simultâneas do que servidores tradicionais.

## Camada de Entrada do Servidor

O Node.js faz pouco processamento de dados e apenas passa a requisição para frente, se comunicando com serviços de ***backend***.

### Mocks e Protótipos

Por utilizar uma linguagem bastante conhecida no mundo Web, o Node.js possibilita criar ***mocks*** e protótipos de APIs e serviços de ***backend*** com grande rapidez, podendo assim simular a comunicação com um serviço externo, por exemplo.

## API com NoSQL por trás

As base de dados NoSQL são baseadas em JSON (*JavaScript Object Notation*), portanto, sua comunicação com Node.js é bastante intuitiva. Com isso, não é necessário converter modelos de dados, por exemplo, pois os mesmos objetos JavaScript armazenados na base de dados podem ser enviados para o front-end sem a necessidade de nenhum tipo de tratamento ou conversão.

Fontes:

<https://pt.wikipedia.org/wiki/Node.js>

<https://www.opus-software.com.br/node-js/>