

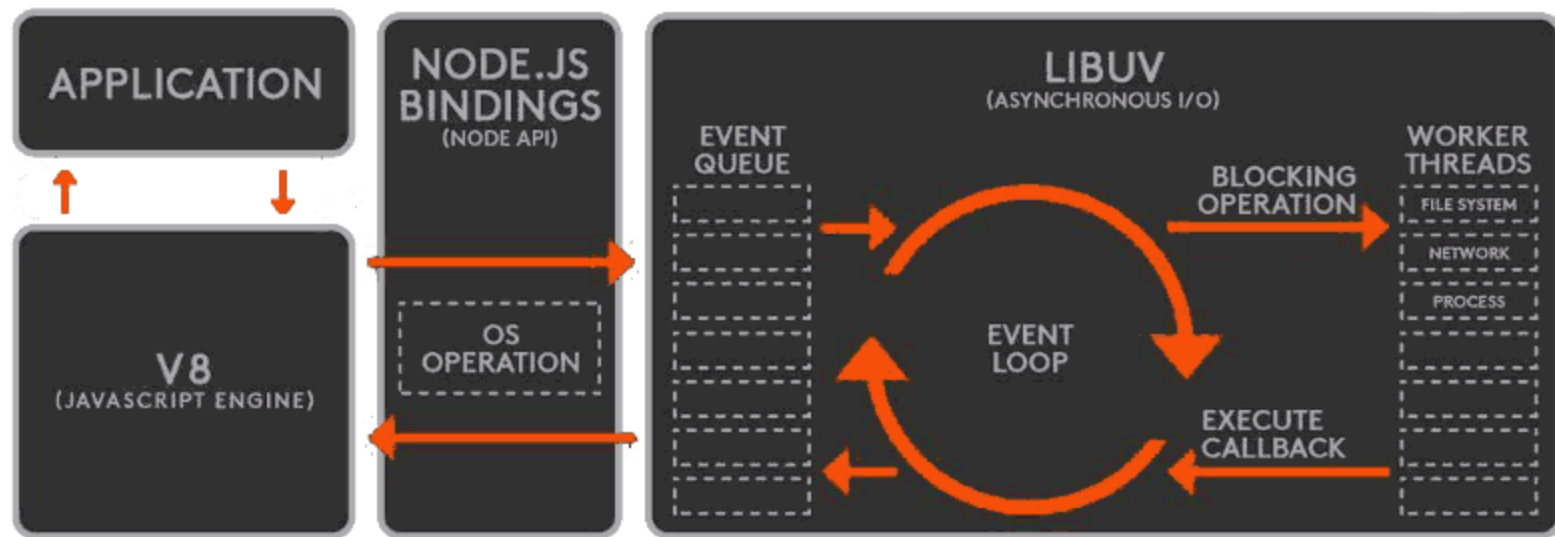
Aula 01

Node.js é um **interpretador** de JavaScript que funciona do lado do servidor criado em cima do [V8](#) que é o motor de JavaScript da Google e que roda no seu Chrome, além disso ele conta com outras bibliotecas que o auxiliam no gerenciamento dos processador, como por exemplo a **Libuv** que falaremos mais adiante.

O Node.js age como uma ponte entre uma API acessável via JavaScript e funções em C++ do V8, foi criado por Ryan Dahl em 2009.

Conta-se que Ryan se inspirou depois de ver barra de progresso de upload de arquivos no [Flickr](#), percebeu que o navegador não sabia o quanto do arquivo foi carregado e tinha que consultar o servidor web.

THE NODE.JS SYSTEM



O Node.js pode ser considerado uma plataforma de execução de aplicações em JavaScript no lado do servidor.

Platform

node standard library

node bindings
(http, socket, file system)

v8

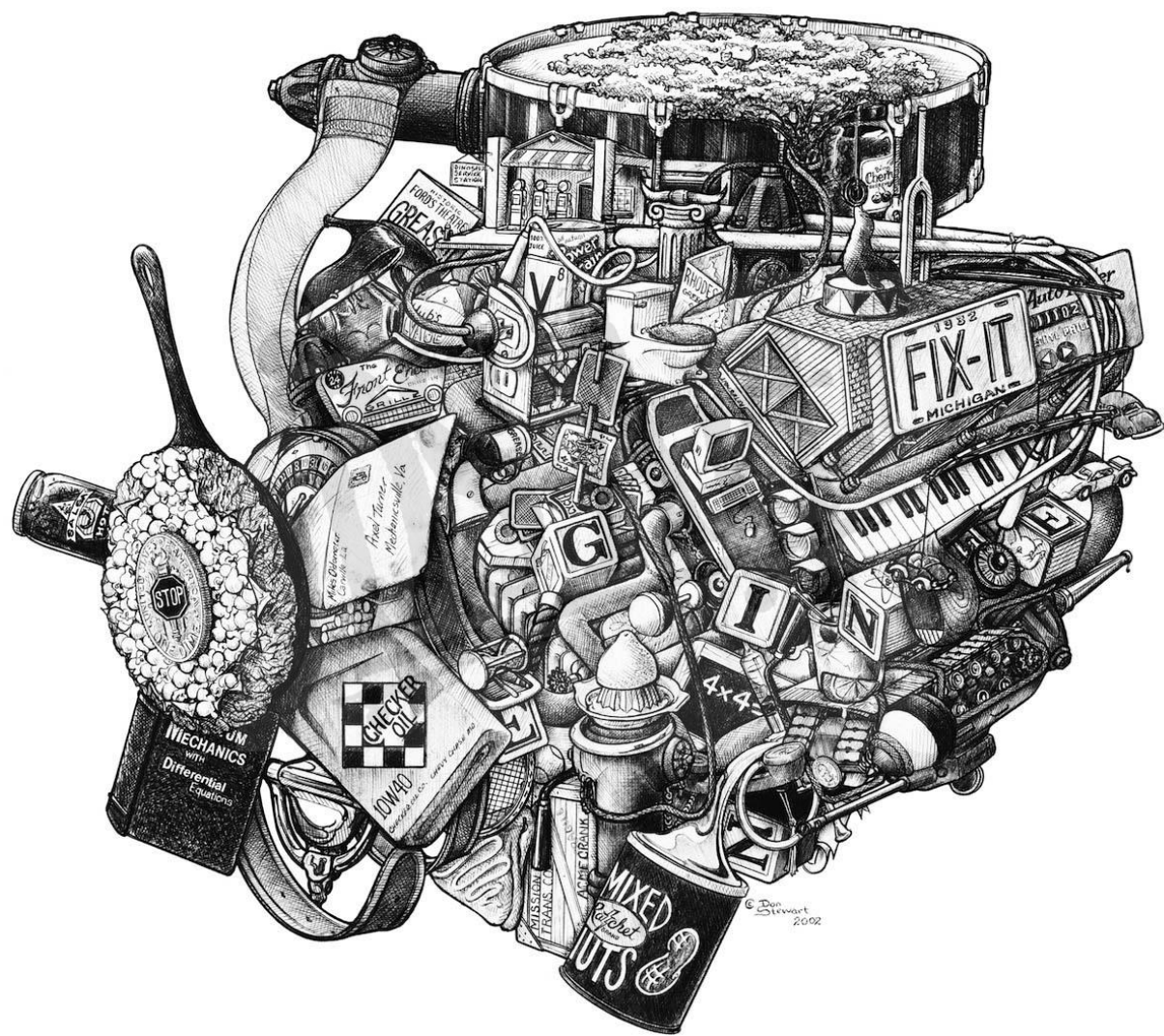
thread pool
(libeio)

event loop
(libev)

crypto
(OpenSSL)

DNS
(c-ares)

V8



Então o que é o tal do **V8** que é a base fundamental do Node.js?

Ele é nada menos que o **interpretador de JavaScript**, tipo uma máquina virtual, desenvolvido pelo Google e usado no Chrome. Feito em C++ e open-source.

O trabalho dele é basicamente *compilar* o código de JavaScript para o código nativo de máquina para depois executá-lo.

Ele levou a velocidade dos códigos compilados para o JavaScript.

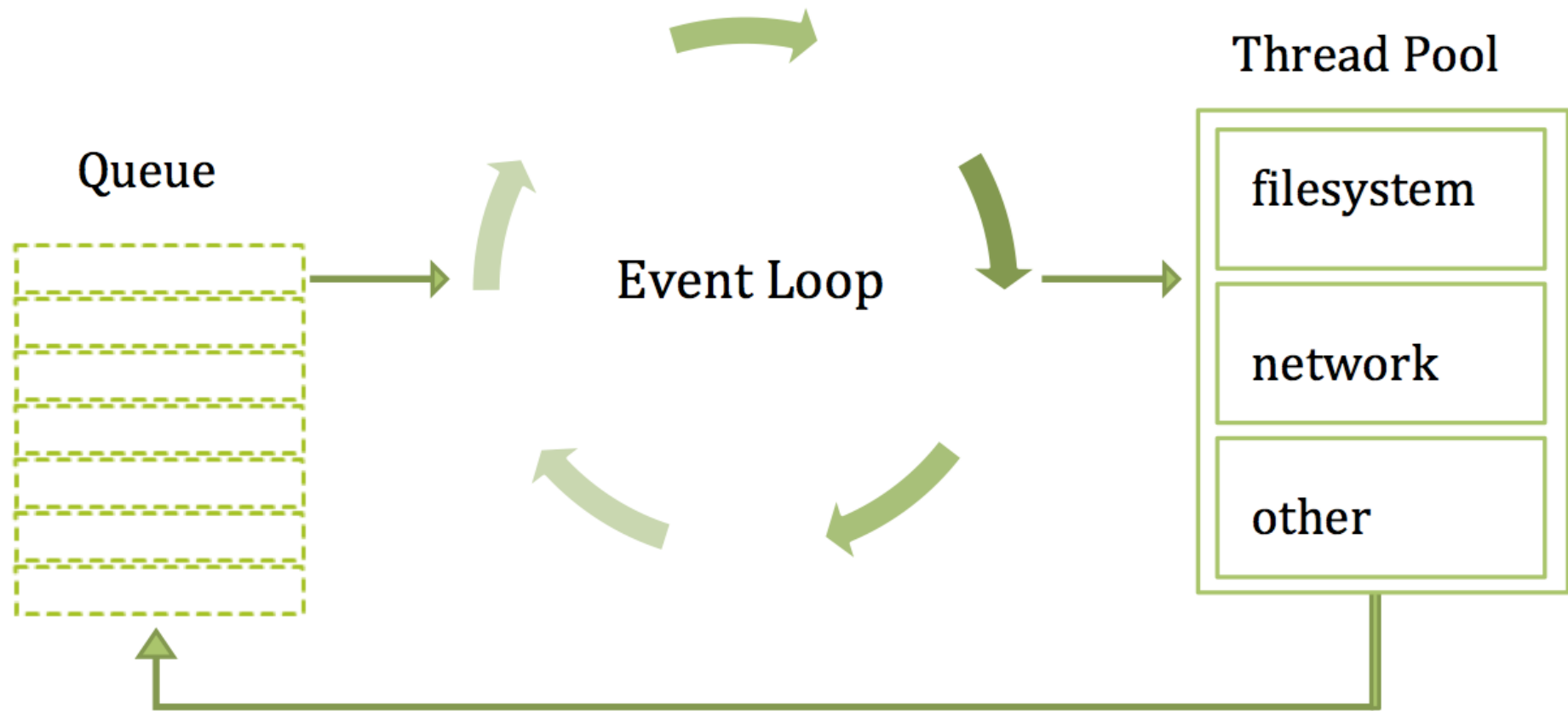
Single Thread

O Node.js trabalha *apenas* com uma thread, podendo ser criadas outras, com isso economizando muita memória, diferentemente da forma que o Apache trabalha e você percebe claramente a diferença de utilização de memória, já que com apenas uma **thread** você não precisa criar um processo novo para cada usuário conectado, acarretando também em uma economia de CPU.

Mas como ele **consegue gerenciar com apenas uma thread?**

Event Loop

O Event Loop nada mais é que uma fila **infinita** que recebe todos os eventos emitidos pelo Node.js, isso inclui as requisições que recebemos no servidor HTTP.



Quando o evento chega para ser executado no *Event Loop*, caso ele seja assíncrono, ele será enviado para onde deve ser executado, por exemplo: filesystem, network, process, etc.

Como o processo é **assíncrono** ele irá executar e só após sua finalização que ele dispara o *trigger* para seu *callback*, esse voltando para a fila que irá ser executada pelo ***Event Loop***.

Logo o processo não fica parado aguardando
sua finalização
no *Event Loop*

O mesmo acontece com nossos sistemas, quando você **envia uma requisição assíncrona** você não tem a certeza quando ela irá retornar, por isso usamos *Promises*.

I/O Async

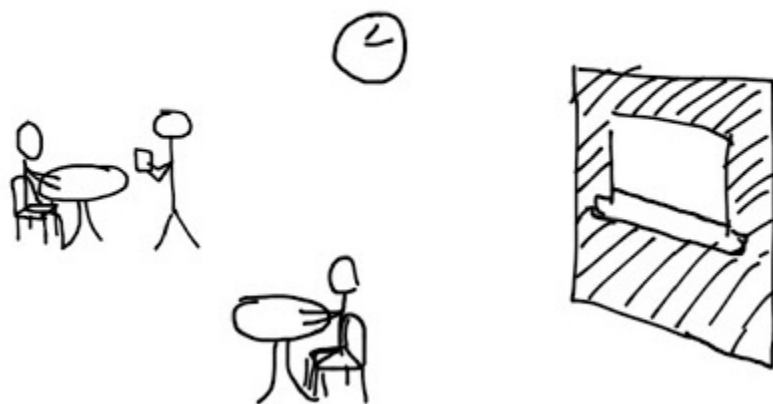
Qualquer função do Node.js, por padrão, é **assíncrona** por isso sempre precisamos de uma função que executará após o final desse processamento, essa que executa posteriormente é chamada de *callback*, falaremos muito mais sobre isso futuramente.

Mas então o que quer dizer que o
I/O é assíncrono?

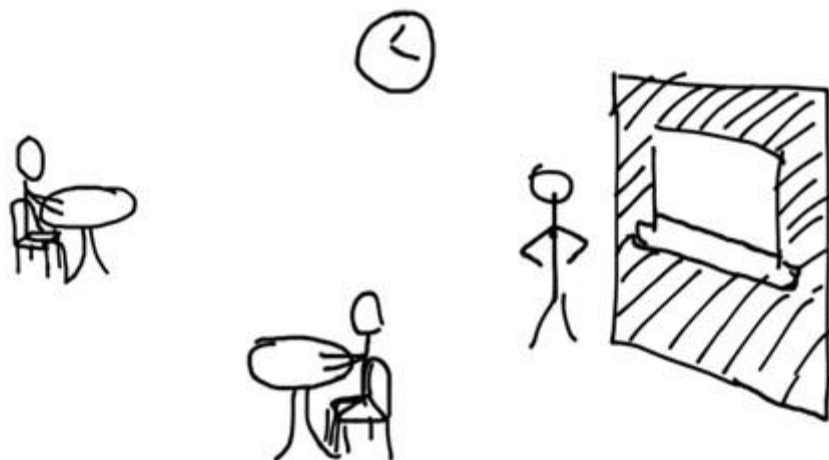
Basicamente diz que **qualquer leitura ou escrita de dados não espera seu processo finalizar para continuar** o *script*, nesse caso os processos ocorrem “*paralelamente*” à execução.

Para termos uma ideia melhor de como é o funcionamento assíncrono, vamos pensar um restaurante sendo **síncrono**.

○ restaurante síncrono

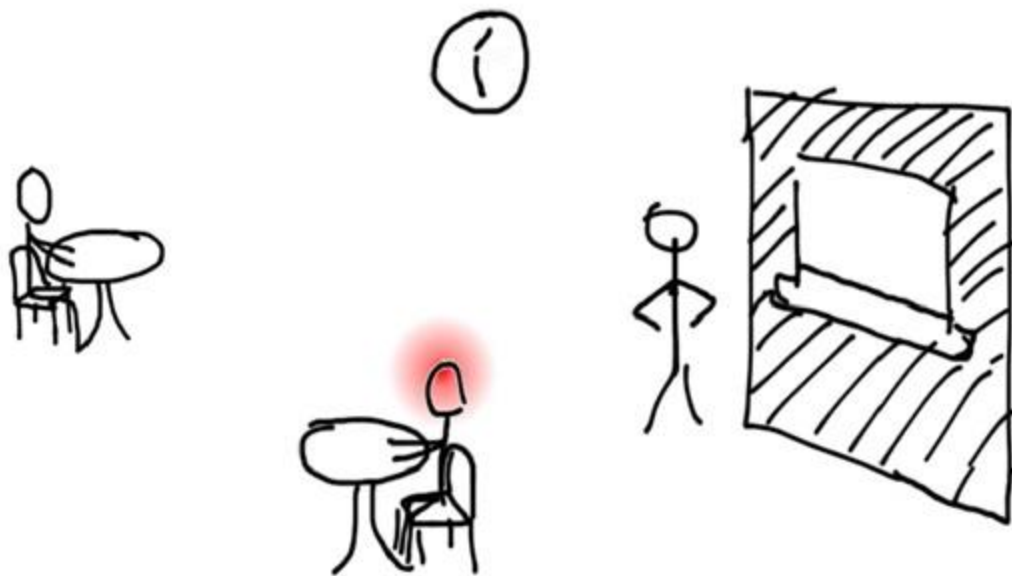


○ restaurante síncrono



No restaurante **síncrono** quando uma mesa é atendida ela precisa receber seu pedido antes que o garçom possa atender outra mesa!!!

○ restaurante síncrono

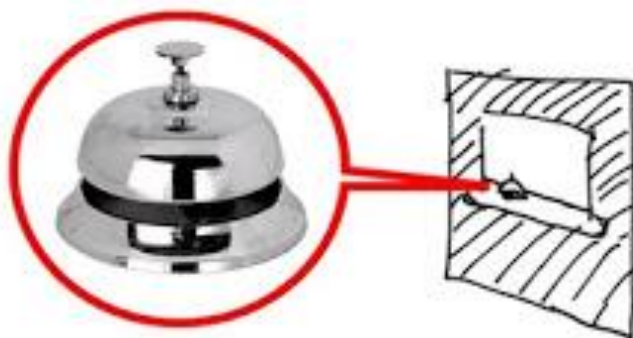


Agora no restaurante **assíncrono** o mesmo garçom pode atender vários pedidos e enviá-los para a cozinha.

Será a cozinha a responsável por responder cada pedido na ordem que para eles forem mais importantes ou mais rápidos. Nesse caso a ordem da resposta dos pedidos pode ser diferente da ordem pedida para a cozinha.

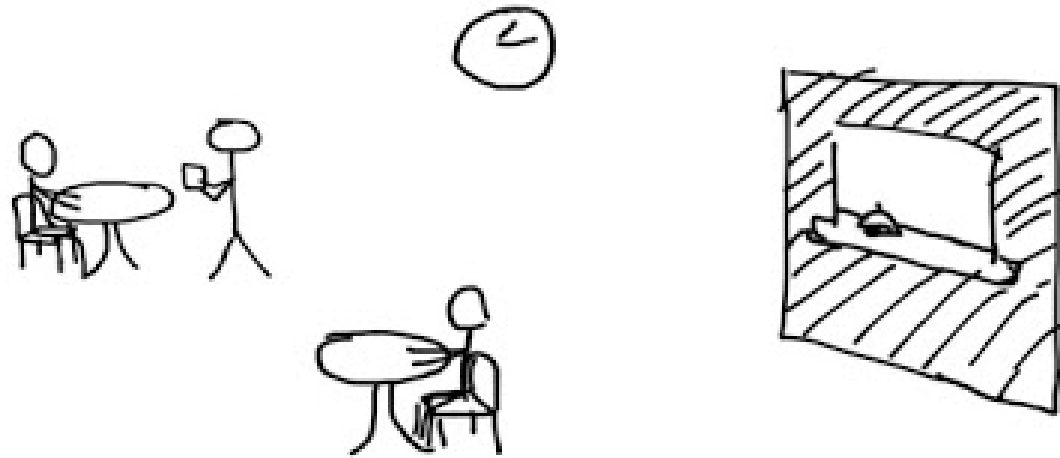
Quando um pedido é finalizado no **Restaurante**
Assíncrono uma campanha/evento é emitido.

○ restaurante assíncrono



Agora no **Restaurante Assíncrono** o garçom pode atender todas as mesas que existirem apenas enviando seus pedidos para serem executados na **cozinha***.

○ restaurante assíncrono



○ restaurante assíncrono



O mesmo acontece com nossos sistemas, quando você **envia uma requisição assíncrona** você não tem a certeza quando ela irá retornar, por isso usamos *Promises*, mas isso é um assunto posterior.

Node também é orientado a eventos

Eventos simplificam a programação
assíncrona

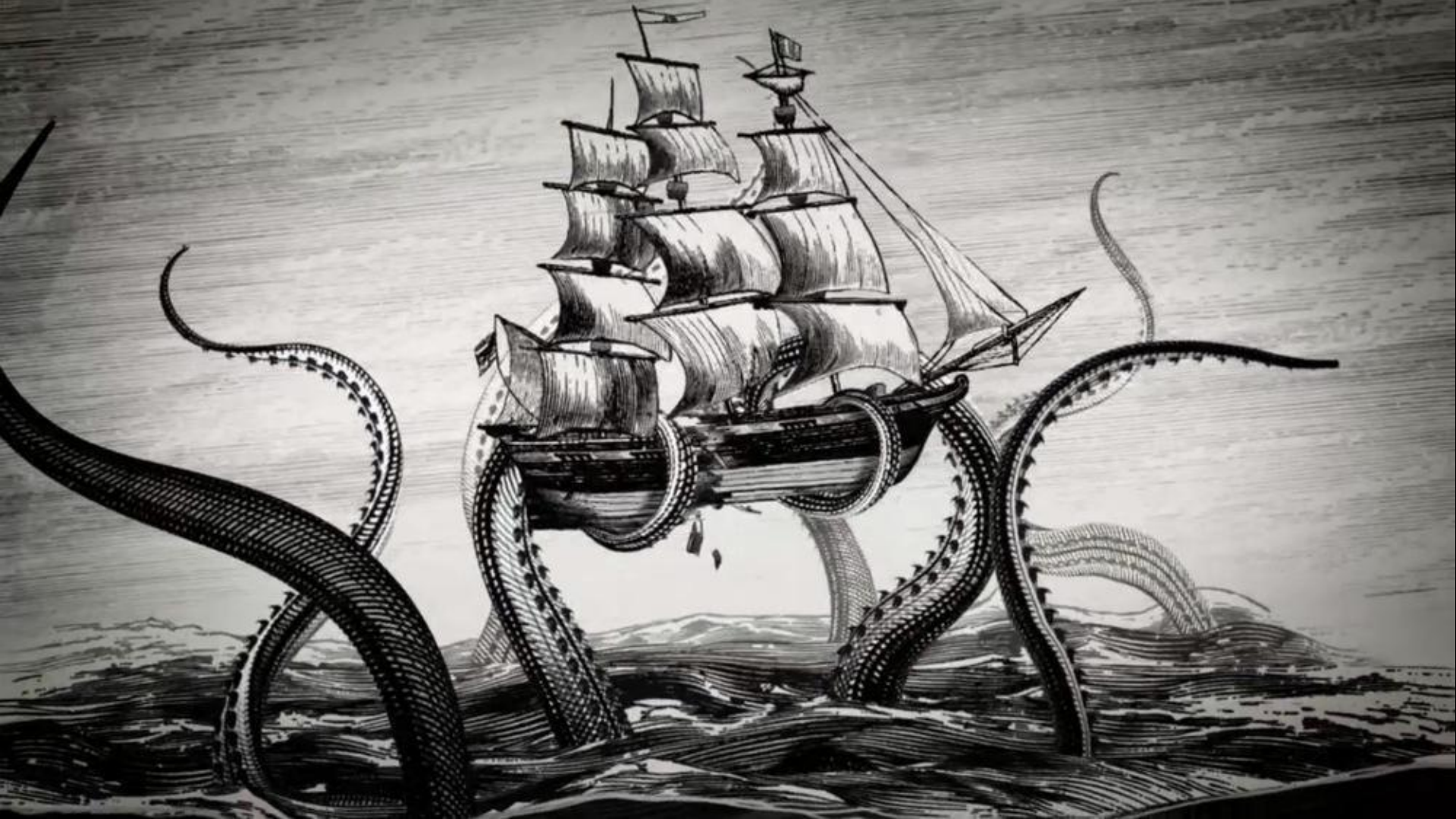
Eventos podem ser emitidos quando
determinada tarefa está pronta

Um listener é o responsável por ‘pegar’ esse evento quando ele ocorrer.

Use Case - Paypal

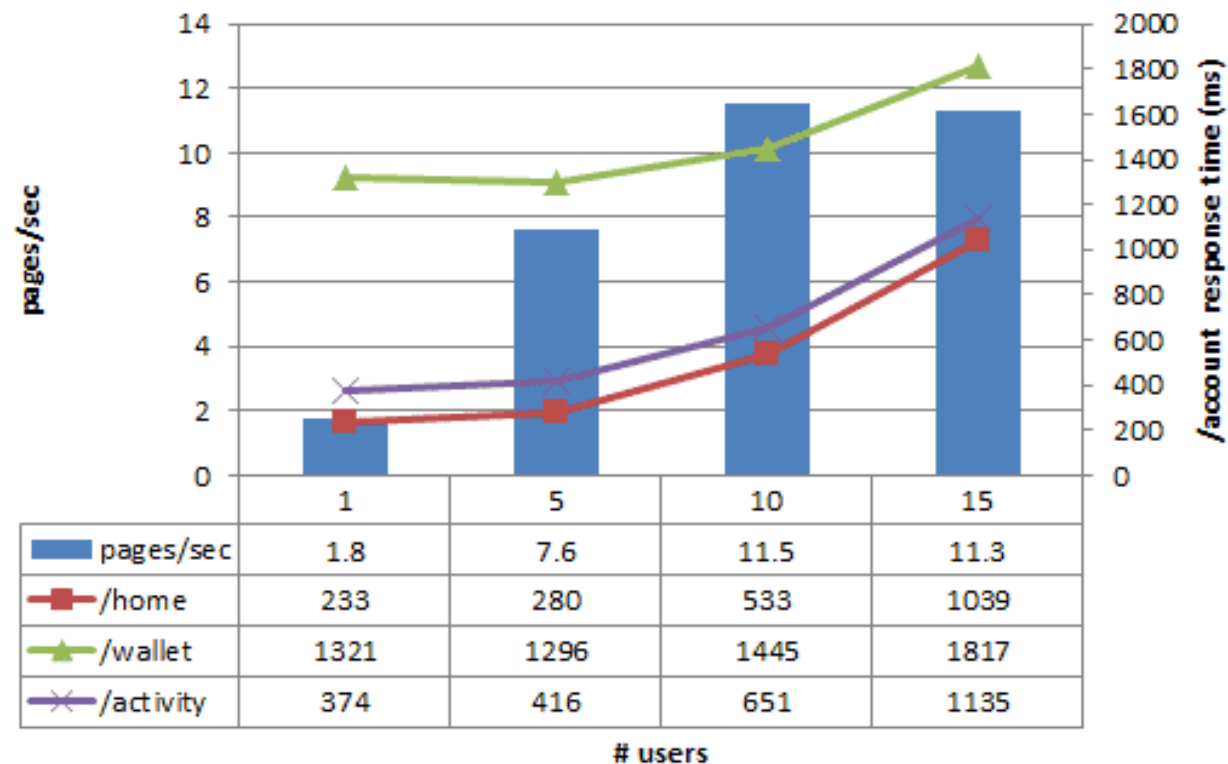
PayPalTM

<https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/>

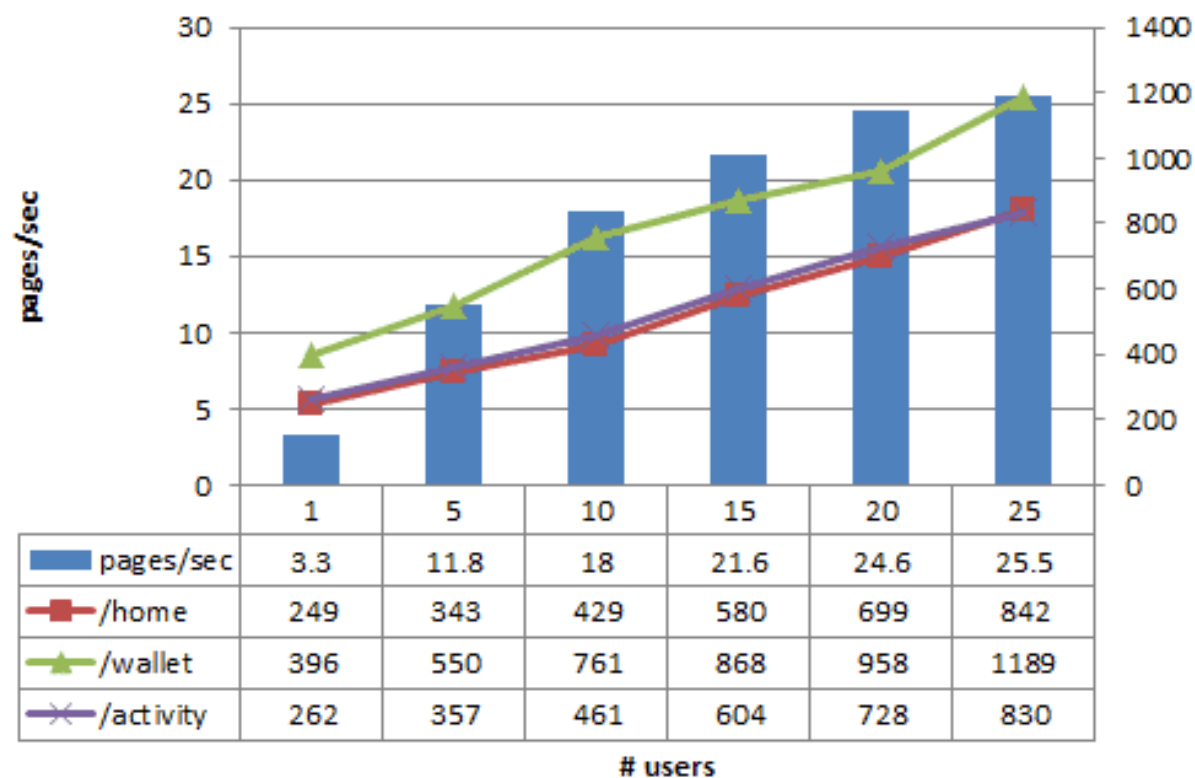


- Built almost **twice as fast** with **fewer people**
- Written in **33% fewer lines of code**
- Constructed with **40% fewer files**

Java application



Node.js application



- **Double the requests per second vs. the Java application.** This is even more interesting because our initial performance results were using a single core for the node.js application compared to five cores in Java. We expect to increase this divide further.
- **35% decrease in the average response time** for the same page. This resulted in the pages being served **200ms faster**— something users will definitely notice.

Instalação

Você pode baixar os instaladores em
nodejs.org/en/download/

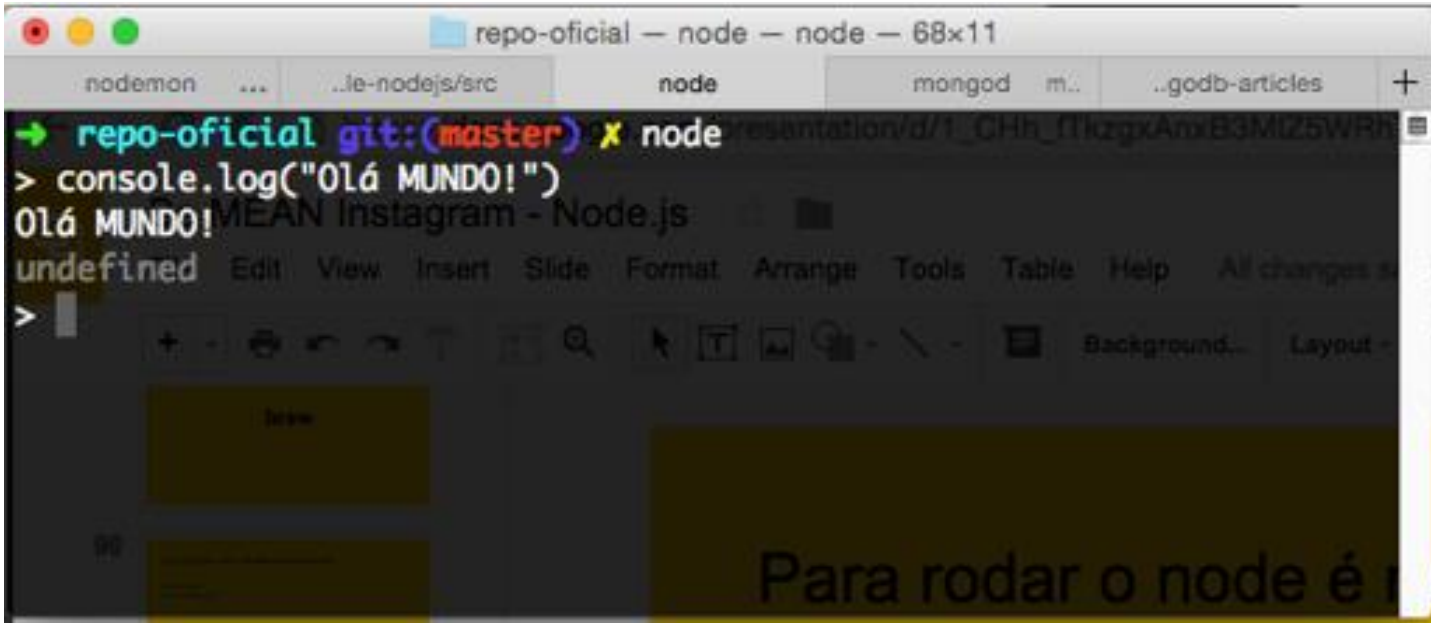
Teste a versão do Node.js como seguinte comando:

```
node -v
```

```
v6.2.0
```

Rodando o Node

Para rodar o node é muito simples, basta executar o comando **node**.



The screenshot shows a terminal window titled "repo-oficial — node — node — 68x11". The terminal has several tabs open: "nodemon", "...le-nodejs/src", "node", "mongod", "m..", and "..godb-articles". The "node" tab is active. The terminal output shows the command `node` being executed, followed by `> console.log("Olá MUNDO!")`, which results in the output `Olá MUNDO!`. Below this, the word `undefined` is visible. The terminal also shows a menu bar with options like "Edit", "View", "Insert", "Slide", "Format", "Arrange", "Tools", "Table", "Help", and "All changes". At the bottom of the terminal, there is a large yellow banner with the text "Para rodar o node é".

```
repo-oficial git:(master) x node
> console.log("Olá MUNDO!")
Olá MUNDO!
undefined
>
```