



Fecomércio RS



Senac

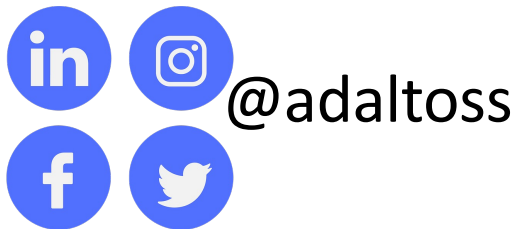
Desenvolvimento de Serviços e APIs

Aula07

Servidor Node.js

Prof. MSc. Adalto Selau Sparremerger

assparremerger@senacrs.com.br



Node.JS

- Criado pro Ryan Dahl
- Node.js é um ambiente de servidor de código aberto.
- Node.js permite executar JavaScript no servidor.
- Como um tempo de execução JavaScript assíncrono baseado em eventos, o Node.js foi projetado para construir aplicativos de rede escalonáveis.
- É um interpretador JavaScript desvinculado do navegador

Instalar Node.JS

- <https://www.youtube.com/watch?v=brSwmLQA0iA>
- <https://www.youtube.com/watch?v=gq9uGdZCKxl>

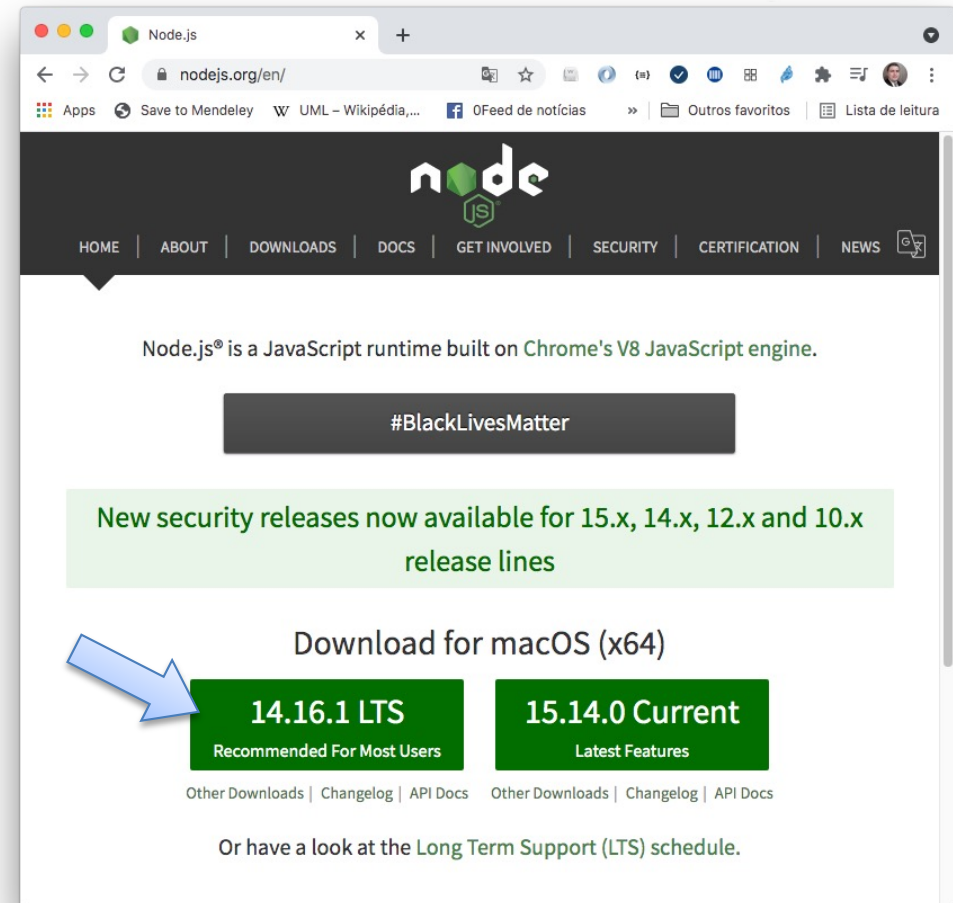


Fecomércio RS



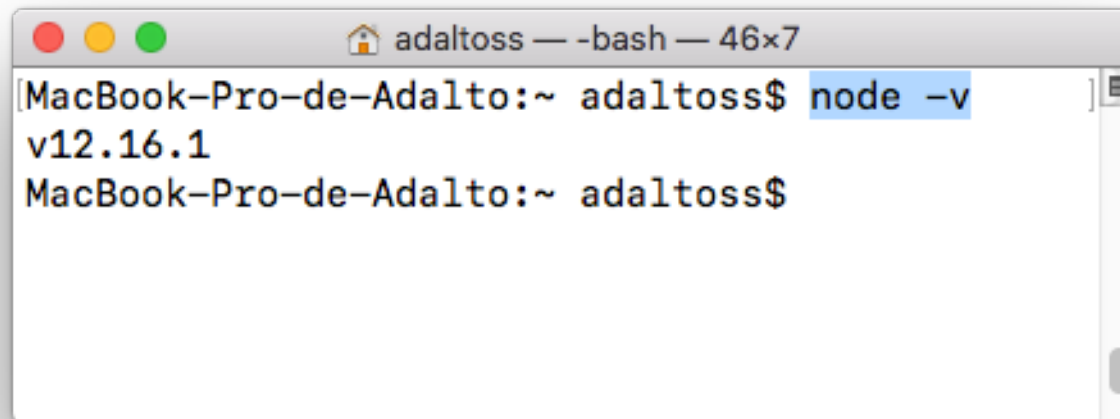
Instalação do Node.JS

- Acessar <https://nodejs.org/en/>
- Fazer o download da versão LTS para o seu sistema operacional e instalar como Administrador:



Instalação Node.JS

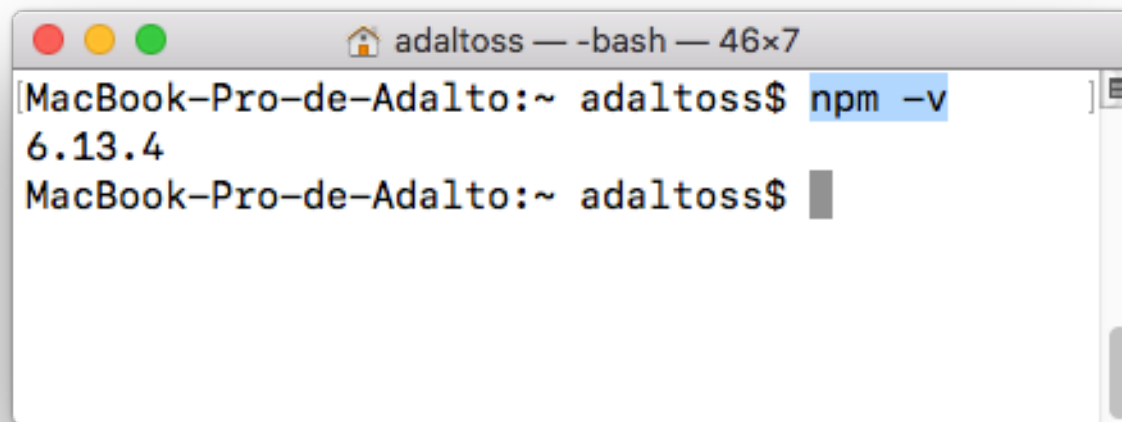
- Para verificar se a instalação foi concluída com sucesso abra o Prompt (Windows) ou o Terminal (Linux/Mac)
- Execute o comando ***node -v*** e verifique se é retornada a versão instalada do Node



```
adaltoss — -bash — 46x7
[MacBook-Pro-de-Adalto:~ adaltoss$ node -v
v12.16.1
MacBook-Pro-de-Adalto:~ adaltoss$
```

NPM – Node Package Manager

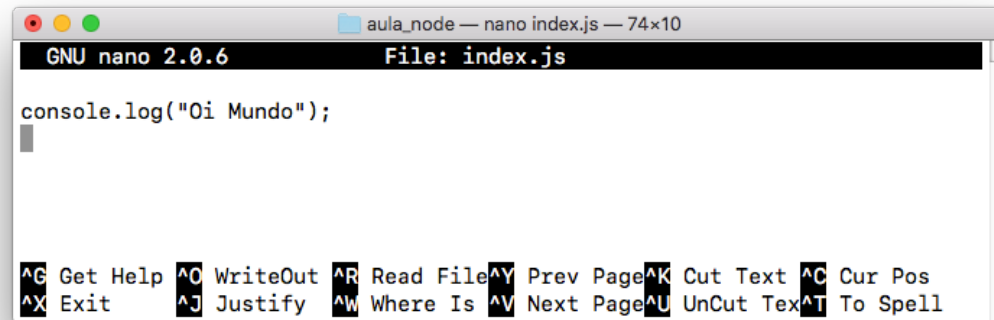
- É o Gerenciador de Pacotes do Node que é instalado juntamente com o Node, no momento de sua instalação.
- Execute o comando ***npm -v*** e verifique se é retornada a versão instalada no npm



```
adaltoss — -bash — 46x7
MacBook-Pro-de-Adalto:~ adaltoss$ npm -v
6.13.4
MacBook-Pro-de-Adalto:~ adaltoss$
```

Hello World

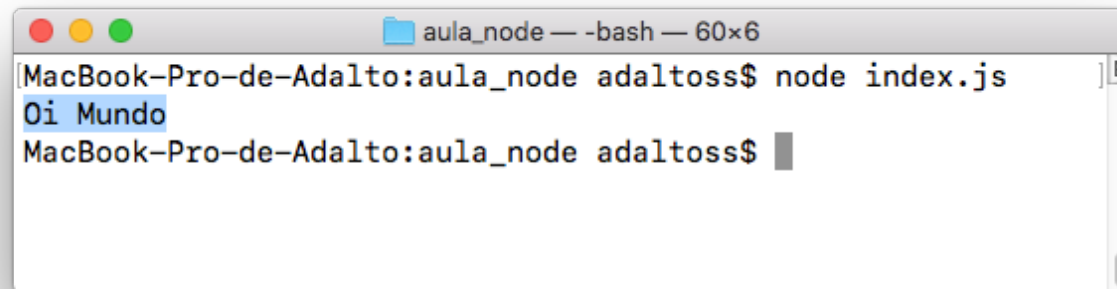
- Em uma pasta de sua preferência, crie um arquivo ***index.js*** e escreva dentro o comando ***console.log("Oi Mundo");***



```
GNU nano 2.0.6 File: index.js
console.log("Oi Mundo");
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

- Execute no terminal o comando ***node index.js***



```
MacBook-Pro-de-Adalto:aula_node adaltoss$ node index.js
Oi Mundo
MacBook-Pro-de-Adalto:aula_node adaltoss$
```

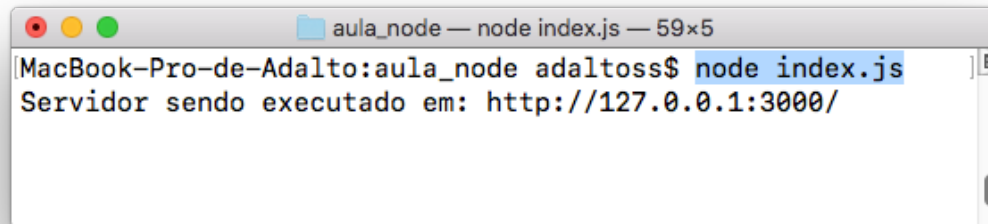
Criar um servidor Node

- Substitua o conteúdo do arquivo *index.js* pelo seguinte:

```
1  const http = require('http');
2
3  const hostname = '127.0.0.1';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    res.statusCode = 200;
8    res.setHeader('Content-Type', 'text/plain');
9    res.end('Oi Mundo');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Servidor sendo executado em: http://${hostname}:${port}/`);
14 });
```

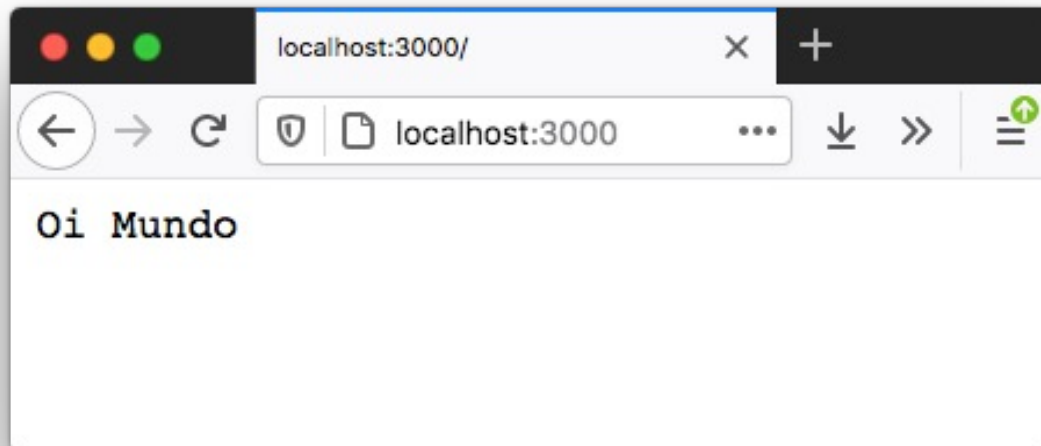

Criar um servidor Node

- Execute novamente no terminal, o comando ***node index.js***

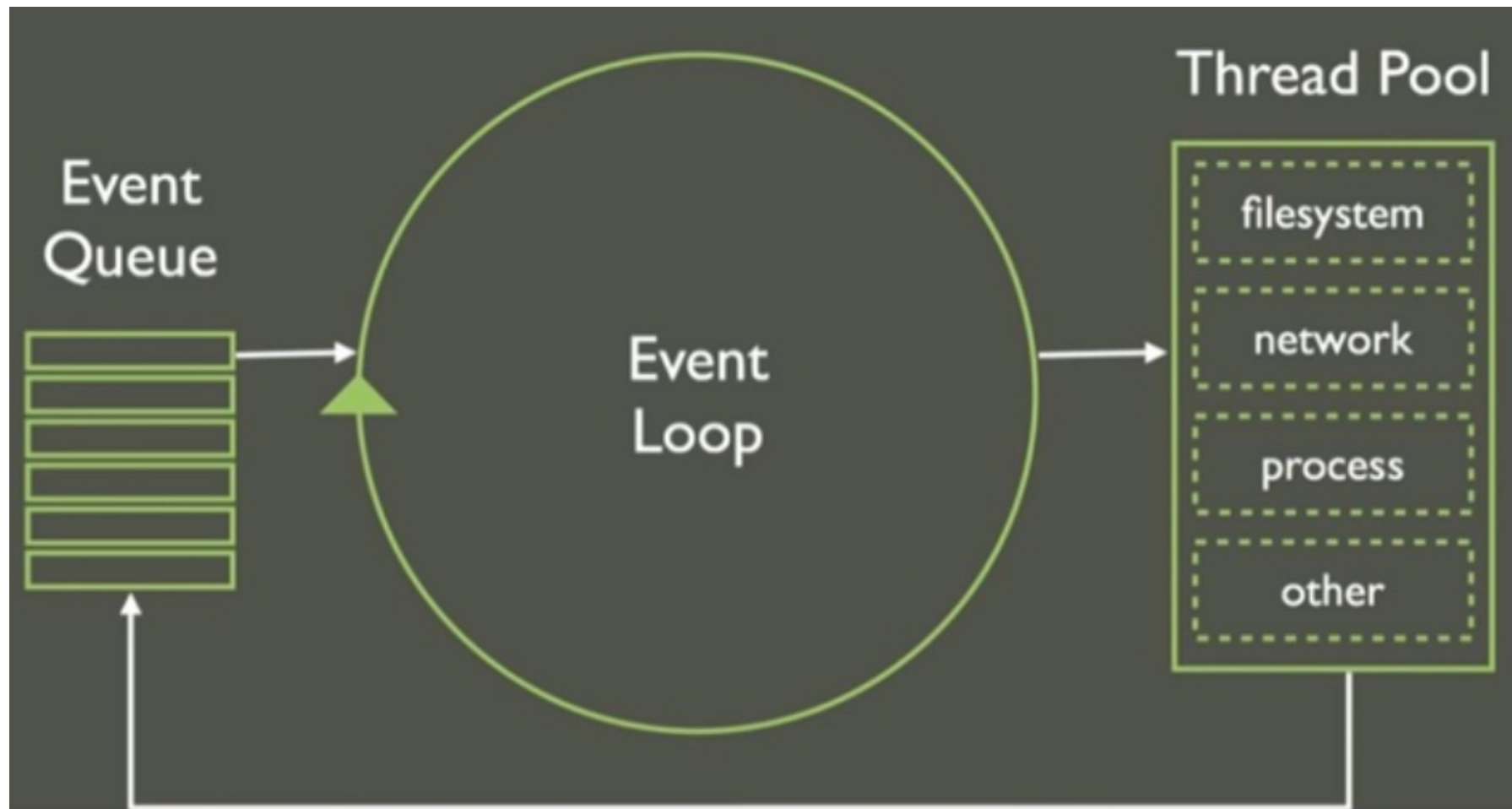


```
aula_node — node index.js — 59x5
MacBook-Pro-de-Adalto:aula_node adaltoss$ node index.js
Servidor sendo executado em: http://127.0.0.1:3000/
```

- Abra o navegador digite o endereço: ***localhost:3000***



Node



Fonte: <http://www.dotnetcurry.com/nodejs/1143/nodejs-tutorial-series-beginner-experienced-developer>



Fecomércio RS



Node

- No exemplo de servidor que acabamos de construir, muitas conexões podem ser tratadas simultaneamente. A cada conexão, uma função de callback é disparada, mas se não houver trabalho a ser realizado, o Node.js ficará inativo.
- Isso contrasta com o modelo de simultaneidade mais comum de hoje, no qual threads de sistema operacional são utilizadas.
- A rede baseada em thread é relativamente ineficiente e muito difícil de usar. Além disso, os usuários do Node.js estão livres da preocupação de travar o processo, já que não há travas.
- Quase nenhuma função no Node.js realiza I/O diretamente, então o processo nunca bloqueia. Como nada bloqueia, sistemas escaláveis são muito fáceis de desenvolver em Node.js.

Node

- O Node.js é semelhante em design e influenciado por sistemas como Ruby's Event Machine e Python's Twisted.
- Node.js leva o modelo de evento um pouco mais longe. Ele apresenta um loop de eventos como uma construção de tempo de execução em vez de uma biblioteca.
- Em outros sistemas, há sempre uma chamada de bloqueio para iniciar o loop de eventos. Normalmente, o comportamento é definido por meio de retornos de chamada no início de um script e, no final, um servidor é iniciado por meio de uma chamada de bloqueio como **EventMachine::run()**.
- No Node.js, não existe essa chamada start-the-event-loop. O Node.js simplesmente entra no loop de eventos após executar o script de entrada. O Node.js sai do loop de eventos quando não há mais callbacks para executar. Esse comportamento é como o JavaScript do navegador - o loop de eventos é escondido do usuário.



Node

- HTTP é um cidadão de primeira classe em Node.js, projetado com streaming e baixa latência em mente. Isso torna o Node.js adequado para a base de uma biblioteca ou estrutura da web.
- O Node.js sendo projetado sem threads não significa que você não pode tirar proveito de vários núcleos em seu ambiente.
- Os processos filhos podem ser gerados usando a API **child_process.fork()** e são projetados para serem fáceis de comunicar.
- Construído sobre a mesma interface está o módulo de cluster, que permite compartilhar sockets entre processos para habilitar o balanceamento de carga sobre seus núcleos.



Acesso a banco com Node

- No Terminal, vamos instalar o module MYSQL:
 - ***npm install mysql***
- Crie um arquivo dê a ele um nome, por exemplo: ***buscaProdutos.js***

```
1  var mysql = require('mysql');
2
3  var conn = mysql.createConnection({
4    host: "localhost",
5    user: "root",
6    password: "",
7    database: "api_rest"
8  });
9
10 conn.connect(function(erro) {
11   if (!erro) {
12     conn.query("SELECT * FROM tbl_produtos", function(err, result, fields) {
13       if (!err) {
14         console.log(result);
15       } else {
16         console.log(err);
17       }
18     });
19   }
20 });
```

Referências

- <https://nodejs.org/en/>
- <https://www.youtube.com/watch?v=brSwmLQA0iA>
- <https://www.youtube.com/watch?v=gq9uGdZCKxI>



Fecomércio RS





Fecomércio RS



Senac