



Front end



Back end



Banco de dados



Node.js é um interpretador de JavaScript assíncrono com código aberto orientado a eventos,, focado em migrar a programação do Javascript do cliente (*frontend*) para os servidores, criando aplicações de alta escalabilidade (como um servidor web), manipulando milhares de conexões/eventos simultâneas em tempo real.

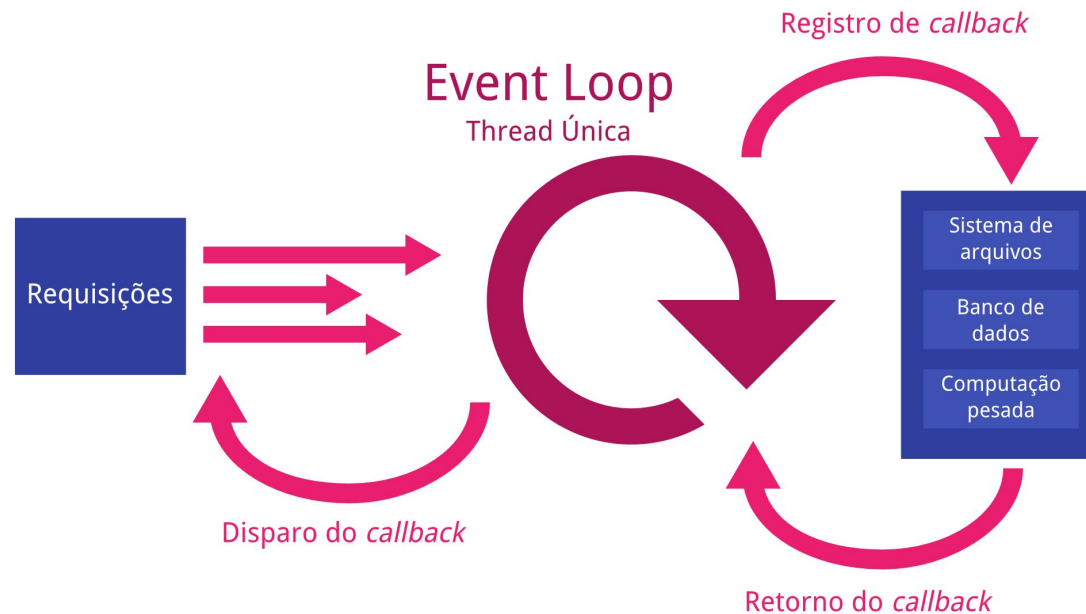


O Node.js é um ambiente JavaScript desenvolvido para ser executado em servidores e possui algumas características bem interessantes:

- Um motor de execução otimizado e rápido, construído sobre o V8 da Google
- Código executado de forma assíncrona por padrão — nada é bloqueante;
- Projeto baseado em iteração de eventos de forma parecida com os navegadores modernos;
- Acesso a rede como prioridade, você pode criar um servidor web com algumas poucas linhas de código;
- Uma potente biblioteca de streams;
- Um sistema gerenciador de pacotes amigável com alguns milhares de módulos open-source para escolha, o NPM.

Como o Node funciona

O Node trata conexões de forma diferente, uma única thread recebe todas as conexões, ou seja, não há concorrência de recursos. Essa única *thread*, chamada de **Event Loop** (que podemos traduzir para laço de eventos), controla todos os outros fluxos assíncronos. Assim, o Node elimina o gargalo de um máximo de requisições que os servidores convencionais sofrem





Quais as vantagens do Node.js?

Node.js usa Javascript

Node.js permite Javascript full-stack

Node.js é muito leve (pouco investimento em infraestrutura) e é multiplataforma

Quais as desvantagens do Node.js?

Node.js usa Javascript (tipagem fraca, POO fora do padrão)

Node.js é recente (2009)

Node.js é assíncrono (uso demasiado de call-backs, não recomendado para aplicações que exige muitos cálculos)

Modo console do Node (Executando Javascript)

No ambiente de linha de comandos digite

```
c:\> node
```

```
Welcome to Node.js v14.15.0.  
Type ".help" for more information.
```

```
>
```

Dentro do modo interativo execute comando JS

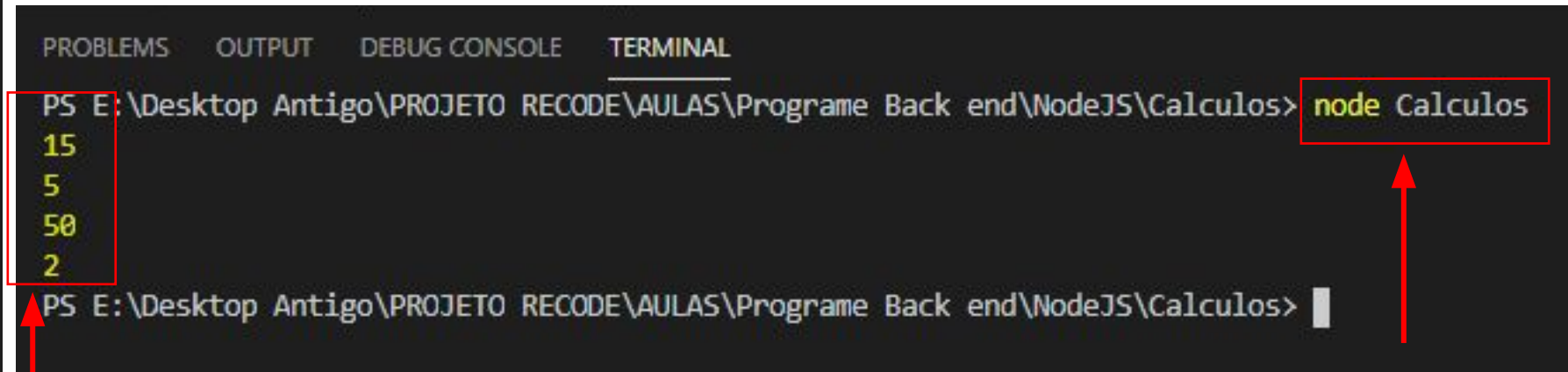
```
> console.log("Hello")
```

O comando será executado se tudo estiver correto.

Executando Javascript no Node

Calculos.js

```
function soma(x,y) {  
    return x + y  
}  
function subtracao(x,y) {  
    return x - y  
}  
function multiplicacao(x,y) {  
    return x * y  
}  
function divisao(x,y) {  
    return x / y  
}  
console.log(soma(10,5))  
console.log(subtracao(10,5))  
console.log(multiplicacao(10,5))  
console.log(divisao(10,5))
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
PS E:\Desktop Antigo\PROJETO RECODE\AULAS\Programe Back end\NodeJS\Calculos> node Calculos  
15  
5  
50  
2  
PS E:\Desktop Antigo\PROJETO RECODE\AULAS\Programe Back end\NodeJS\Calculos>
```

Saída no terminal

Executando um arquivo javascript no NODE



Prática - Mão na massa

Instalar, configurar e rodar os códigos demonstrados nos slides anteriores

Tempo: 40 minutos



Módulos: exports e require()

Módulos são cruciais para construção de aplicações em Node pois eles permitem incluir bibliotecas externas, como bibliotecas de acesso ao banco de dados, e ajudam a organizar seu código em partes separadas com responsabilidades limitada. Utilizar módulos em Node é simples, você usa a função **require()**, que recebe um argumento: o nome da biblioteca do core ou o caminho do arquivo do módulo que você quer carregar. O módulo **exports** torna a função passível de ser exportada.

Módulos: exports

Módulo soma.js

```
let soma = function(x, y) {  
  return x + y  
}  
  
module.exports = soma
```

Módulo multiplicação.js

```
let mult = function(x, y) {  
  return x * y  
}  
  
module.exports = mult
```

Módulo subtração.js

```
let subt = function(x, y) {  
  return x - y  
}  
  
module.exports = subt
```

Módulo divisão.js

```
let divisao = function(x, y) {  
  return x / y  
}  
  
module.exports = divisao
```

Módulo: require()

Módulos são cruciais para construção de aplicações em Node pois eles permitem incluir bibliotecas externas, como bibliotecas de acesso ao banco de dados, e ajudam a organizar seu código em partes separadas com responsabilidades limitada. Utilizar módulos em Node é simples, você usa a função **require()**, que recebe um argumento: o nome da biblioteca do core ou o caminho do arquivo do módulo que você quer carregar.

Estrutura do projeto anterior dividido em módulos

```
JS calculadora.js
JS divisao.js
JS multiplicacao.js
JS soma.js
JS subtracao.js
```

Arquivo calculadora.js importando os módulos

```
const subt = require('./subtracao')
const soma = require('./soma')
const mult = require('./multiplicacao')
const divisao = require('./divisao')
console.log(soma(40,2))
console.log(subt(40,2))
console.log(mult(40,2))
console.log(divisao(40,2))
```

```
node calculadora
```

Execute no terminal o módulo principal calculadora

Flávio Mota



Prática - Mão na massa - export e require()

Instalar, configurar e rodar os códigos demonstrados para montar a calculadora

Tempo: 40 minutos

Hello World – Create Server

server.js

```
var http=require('http');  
var server = http.createServer(function (req,res){  
    res.end("<html><body> Hello World okay! </body></html>");  
});  
server.listen(3000);  
console.log("Servidor Ativo!");
```

- solicitamos o módulo http e atribuímos à variável http
- do módulo http utilizamos o evento `createServer` para criar um servidor. Retorna uma instância do `http.server`
- A função `res.end()` é o que o servidor responderá ao ser solicitado.
- Usamos `server.listen(porta)`, para que o servidor seja criado na porta escolhida.
- Execute o arquivo `serve.js` com o servidor node

Criando nosso primeiro servidor em node

Arquivo servidor.js

```
var servidor = require('http');

servidor.createServer(function(req, res){
  res.end("<h1> Ola Mundo!!</h1>")
}).listen(8001);

console.log('Servidor subiu')
```

Módulo http sendo importado e atribuído a variável servidor

Método `createServer` cria um novo servidor com uma função call-back que recebe dois parâmetros 'req' e 'res', representando as requisições e respostas do servidor. O método `listen(8001)` representa a porta que o servidor está sendo escutado.

O método `end` do objeto `res` e a devolução visual no navegador

```
PS E:\Desktop Antigo\PROJETO RECODE\AULAS\Programe Back end\NodeJS\AppServidor> node servidor
Servidor subiu
```

Execute o arquivo no terminal.

Ola Mundo!!

<http://localhost:8001/> servidor rodando neste endereço local



Prática - Mão na massa - export e require()

Criar o servidor http usando o node.

OBS: criar servidor respondendo na porta 3000, 3001 e 3002 com respostas diferentes.

Exemplo: “Este servidor está na porta XXXX”, onde XXXX é o número da porta que está sendo usado

Tempo: 30 minutos

Criando rotas em node

Arquivo servidor2.js

```
var http = require('http');

var server = http.createServer(function(req, res){
  var categoria = req.url;

  if(categoria == '/front-end'){
    res.end("<html><body>Tecnologias Front-End: TypeScript, Angular, React..</body></html>");
  }else if(categoria == '/back-end'){
    res.end("<html><body>Tecnologias Back-End: NodeJS, Python, PHP, MySQL...</body></html>");
  }else if(categoria == '/infraestrutura'){
    res.end("<html><body>Azure Cloud, Linux, MySQL Server...</body></html>");
  }else{
    res.end("<html><body>Programador Full Stack</body></html>");
  }
});

server.listen(3000);
```

Criando rotas em node

Arquivo servidor3.js

```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (request, response) {
  var categoria = request.url;
  // A constante __dirname retorna o diretório raiz da aplicação.
  if (categoria == '/index') {
    fs.readFile(__dirname + '/index.html', function (err, html) {
      response.end(html);
    });
  } else if (categoria == '/teste') {
    fs.readFile(__dirname + '/teste.html', function (err, html) {
      response.end(html);
    });
  }
});
server.listen(3000, function () {
  console.log('Executando Site Pessoal');
});
```

Rota para index.html

← → ↻ ⓘ localhost:3000/index

Bem vindo ao meu site pessoal

Rota para teste.html

← → ↻ ⓘ localhost:3000/teste

Bem vindo a pagina de teste

Prática - Mão na massa - export e require()

Refazer os códigos anteriores treinando as rotas em Node

OBS: criar servidor respondendo na porta 3000, 3001 e 3002 com respostas diferentes.

Exemplo: “Este servidor está na porta XXXX”, onde XXXX é o número da porta que está sendo usado

Tempo: 40 minutos

Desafio 01 - Criando o servidor http e rotas no node

Crie 3 arquivos HTML: **artigos.html**, **contato.html** e **erro.html**

Coloque qualquer conteúdo para cada pagina html;

Ao digitar no browser o path: **/artigos** deve renderizar **artigos.html**

A regra anterior também se aplica para o arquivo **contato.html**;

Ao digitar qualquer path diferente de **/artigos** e **/contato** deve renderizar **erro.html**;

A rota principal **"/** deve renderizar **artigos.html**;

Tempo: 60 minutos

Ferramentas para aumentar nossa produtividade

- **npm:** já conhecido neste curso, vem instalado junto com o NodeJS e agora será usado como *gerenciador de dependências*.
- **Nodemon:** utilitário simples que reinicia automaticamente o NodeJS sempre que detectar uma alteração em qualquer arquivo do projeto.
- **express:** framework NodeJS para aplicações web. Implementa uma API e várias funções para controle de rotas, dos módulos etc.
- **EJS:** linguagem de modelagem para criação de páginas HTML utilizando JavaScript. Ajuda a separar as views dos scripts de controle.



Instalando e usando o nodemon

Com a pasta do projeto aberta digite no terminal o comando abaixo para uma instalação no seu computador

```
npm install nodemon -global // Instalação global
```

```
npm install nodemon -save // Instalação local
```

Para rodar sua aplicação digite **nodemon <nome do arquivo js>** tecle <enter>, agora sua aplicação está sendo monitorada e a cada mudança o nodemon reinicia o servidor automaticamente.

Instale o nodemon no seu projeto e faça o teste !!