



UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Um mundo estático

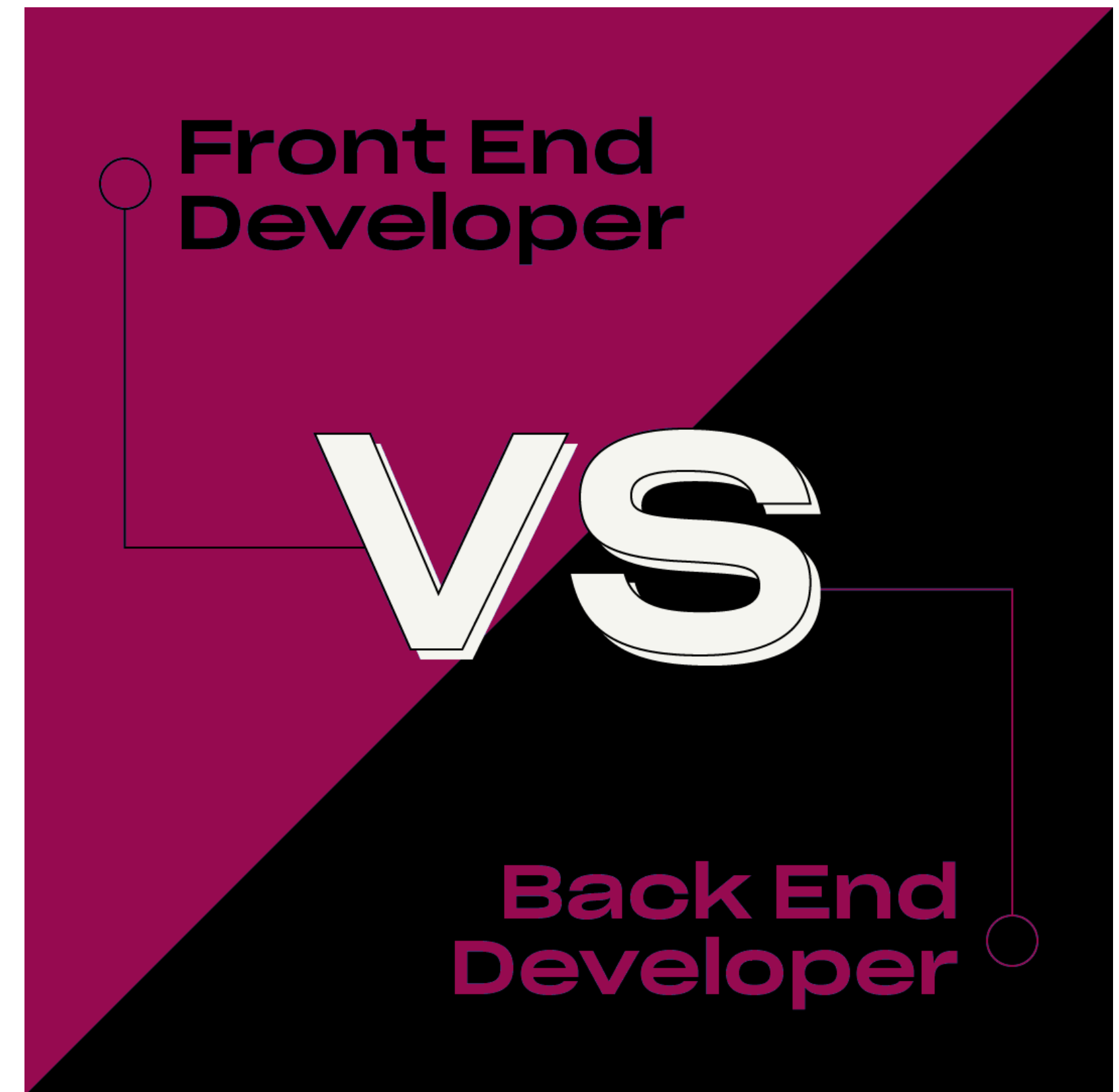
QXD0193 - Projetos de Interfaces Web

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Agenda

- Introdução
- De volta ao passado, a Web pré 1994
- Introdução ao Node
- Criando um projeto Node

Introdução



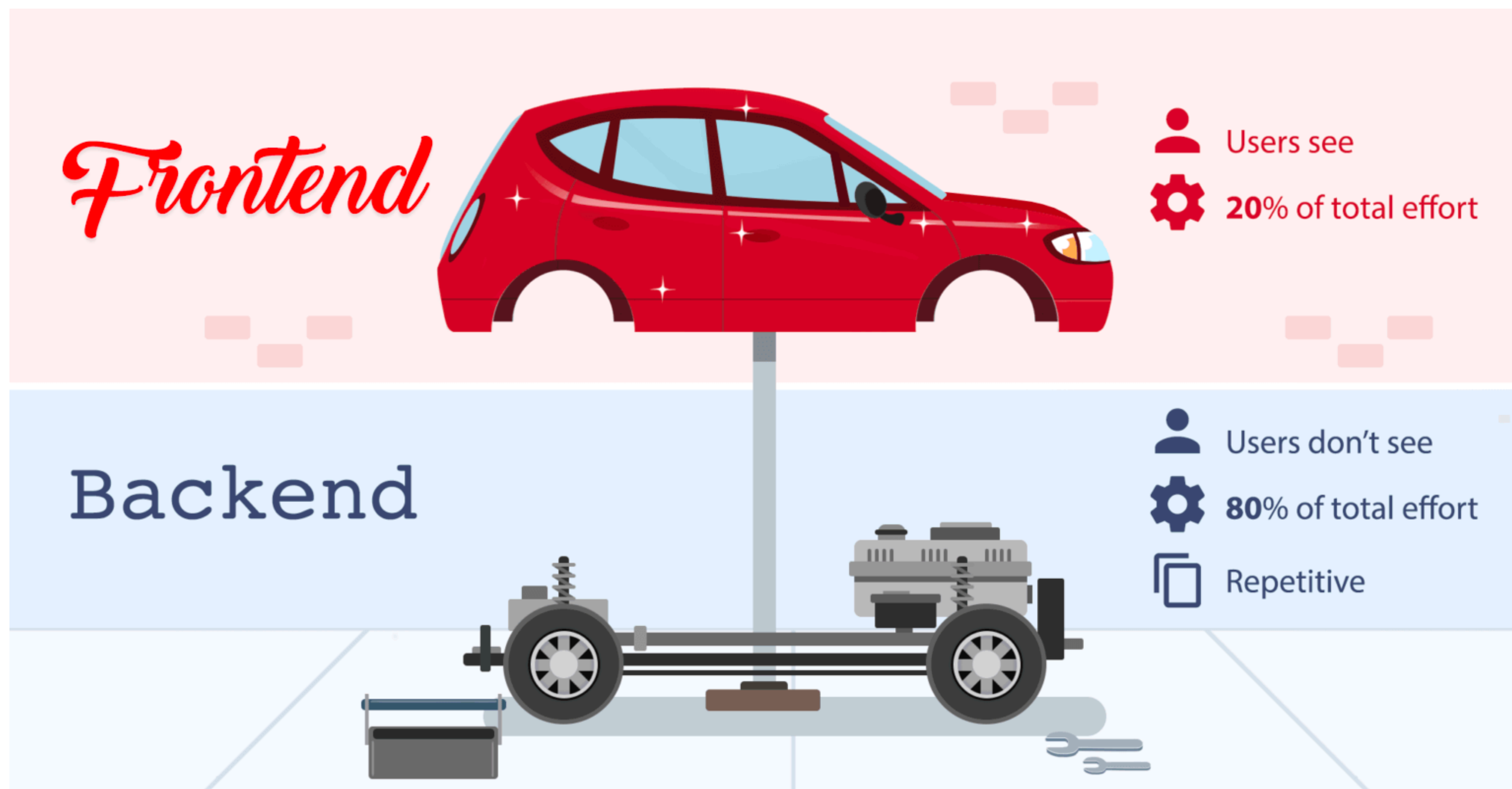
Introdução

Front End vs Back End

- Front End developer e Back End developer são posições que estão em alta
- No entanto, o que é um **Front End** ? O que é um **Back End**?
- O que faz um desenvolvedor de **Front End**?
- O que faz um desenvolvedor de **Back End**?

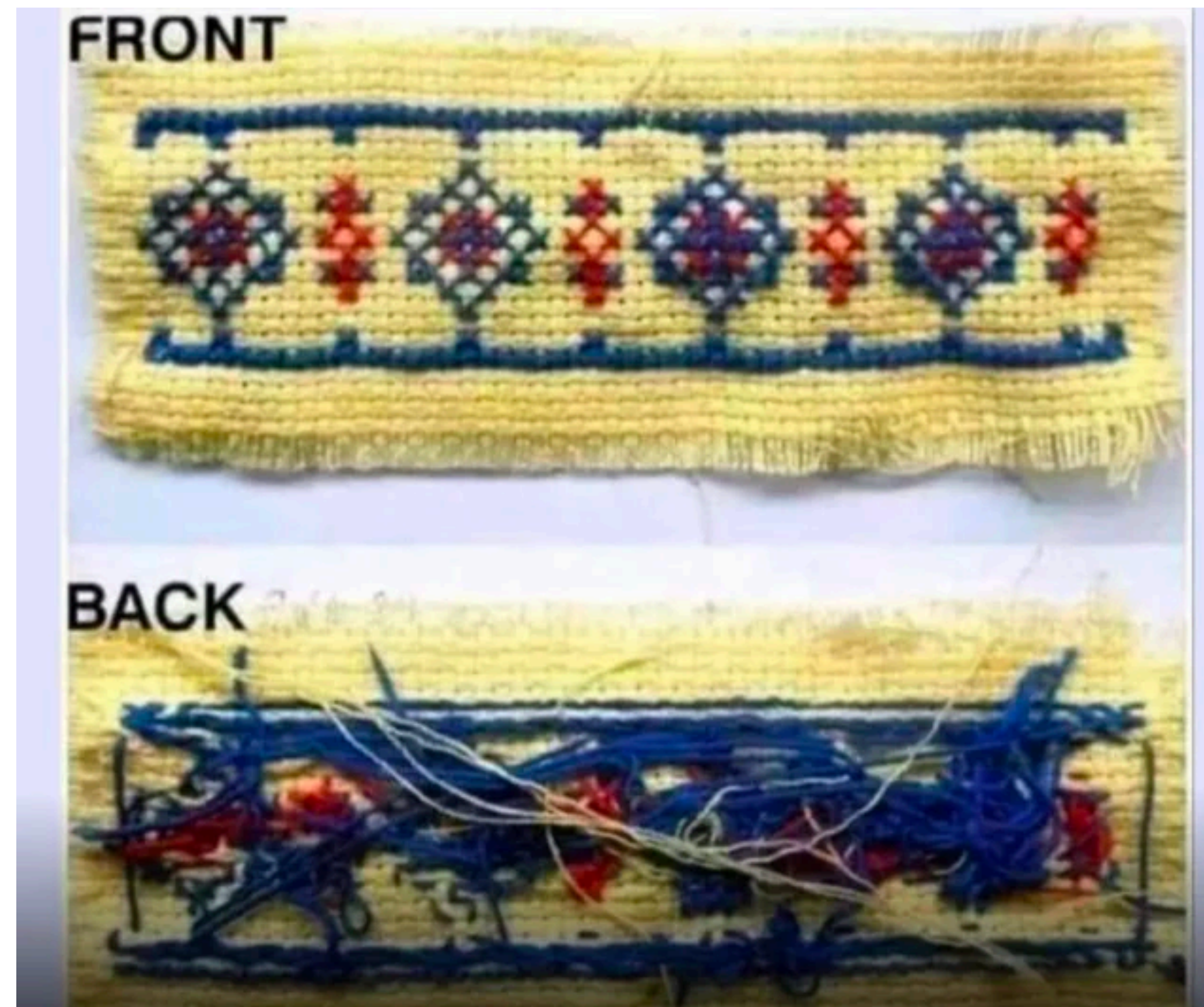
Introdução

Front End vs Back End



Introdução

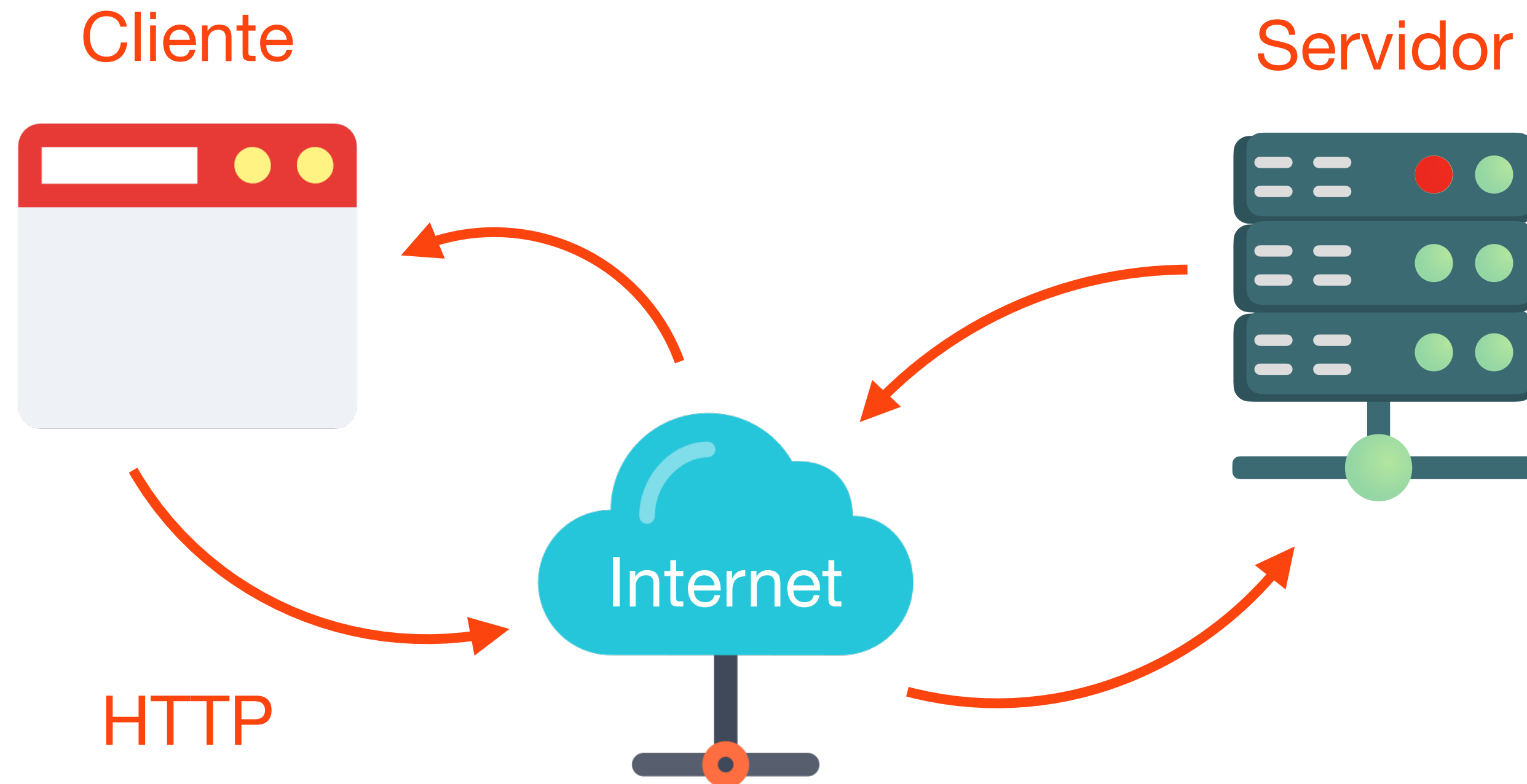
Front End vs Back End



Introdução

Front End vs Back End

- De maneira simplória:
 - O código do **Front End** é aquele **executado** no cliente ou no lado do cliente
 - O código do **Back End** é aquele **executado** no servidor ou no lado do servidor



Introdução

Front End vs Back End



De volta ao passado, a Web pré 1994

[Yahoo](#) - A Guide to WWW

[[What's New?](#) | [What's Cool?](#) | [What's Popular?](#) | [Stats](#) | [A Random Link](#)]

[Y](#) [Top](#) | [↑](#) [Up](#) | [🔍](#) [Search](#) | [✉](#) [Mail](#) | [+](#) [Add](#) | [??](#) [Help](#)

- [Art](#)(466) NEW
- [Business](#)(6426) NEW
- [Computers](#)(2609) NEW
- [Economy](#)(743) NEW
- [Education](#)(1487) NEW
- [Entertainment](#)(6199) NEW
- [Environment and Nature](#)(193) NEW
- [Events](#)(53) NEW
- [Government](#)(1031) NEW
- [Health](#)(367) NEW
- [Humanities](#)(163) NEW
- [Law](#)(163) NEW
- [News](#)(185)

De volta ao passado, a Web pré 1994

- O primeiro navegador da **World Wide Web** foi lançado em 1990
- Em outubro de 1991, **Tim Bernes-Lee** publica um documento descrevendo **18 tags do HTML**
- Em 1992 a primeira imagem da web foi publicada
- Em 1994, o **Mosaic**, primeiro navegador modo gráfico foi lançado pela Netscape



De volta ao passado, a Web pré 1994

- Nesta época não existia JavaScript e nem o CGI (*Common Gateway Interface*)
- Portanto a web era realmente estática e nenhum código era executado no cliente
- Neste cenário dois tipos de software se destacavam:
 - Navegadores
 - Servidores web
- Portanto podemos dizer que o desenvolvimento web nasceu no lado do servidor

De volta ao passado, a Web pré 1994

Servidor Web

- E se a gente voltasse no tempo e tivesse que construir um servidor web para atender as necessidades da época, o que precisaríamos fazer?
 - O que é um servidor web?
- Algumas dicas:
 - A comunicação entre cliente e servidor usa o protocolo HTTP
 - Por sua vez, HTTP é transportado via TCP

De volta ao passado, a Web pré 1994

Construindo nosso servidor web

- Primeiramente o servidor web deve ser apto a se comunicar com a rede (*socket*)
- O servidor precisa se capaz de se comunicar usando um formato específico determinado pelo protocolo (HTTP)
- O servidor deve ser capaz responder as requisições com arquivos estáticos localizando no seu sistema de arquivo

Um mundo estático

Construindo nosso servidor web

- Passo a passo:
 1. Criar um projeto NodeJs com suporte a Typescript
 2. Criar um HTTP Server usando o pacote HTTP
 3. Carregar um página HTML qualquer
 4. Analisar a URL e carregar o recurso pedido

Introdução ao Node



Introdução ao Node

Node

- É uma cross-platform runtime de código aberto que permite que desenvolvedores criem aplicações server-side em JS
- Executada “diretamente” no sistema operacional, fora do contexto do navegador
- Prover suporte a API mais tradicionais dos sistemas operacionais
- Ex: HTTP, FileSystem

Introdução ao Node

História

- Enquanto a web nasceu em 1990, JavaScript nasceu 1995, Node tem apenas 2009 anos
- Antes do sucesso do Node, a Netscape havia investido no LiveWire, o ambiente capaz de criar páginas web dinâmicas usando JavaScript no server-side, no entanto não obteve sucesso
- Aplicações *server-side* com JavaScript se popularizam a partir da introdução do Node.js
 - Fator decisivo: Timing
 - JavaScript passou ser utilizado em aplicações de maior porte graças a Web 2.0. Ex: Flickr, Gmail, etc.
 - Engine JavaScript melhoraram consideravelmente devido a competição entre navegadores
- Node usa a V8 ou Chrome V8, uma engine open-source JavaScript do Projeto Chromium que evoluiu bastante devido a essa competição

Introdução ao Node

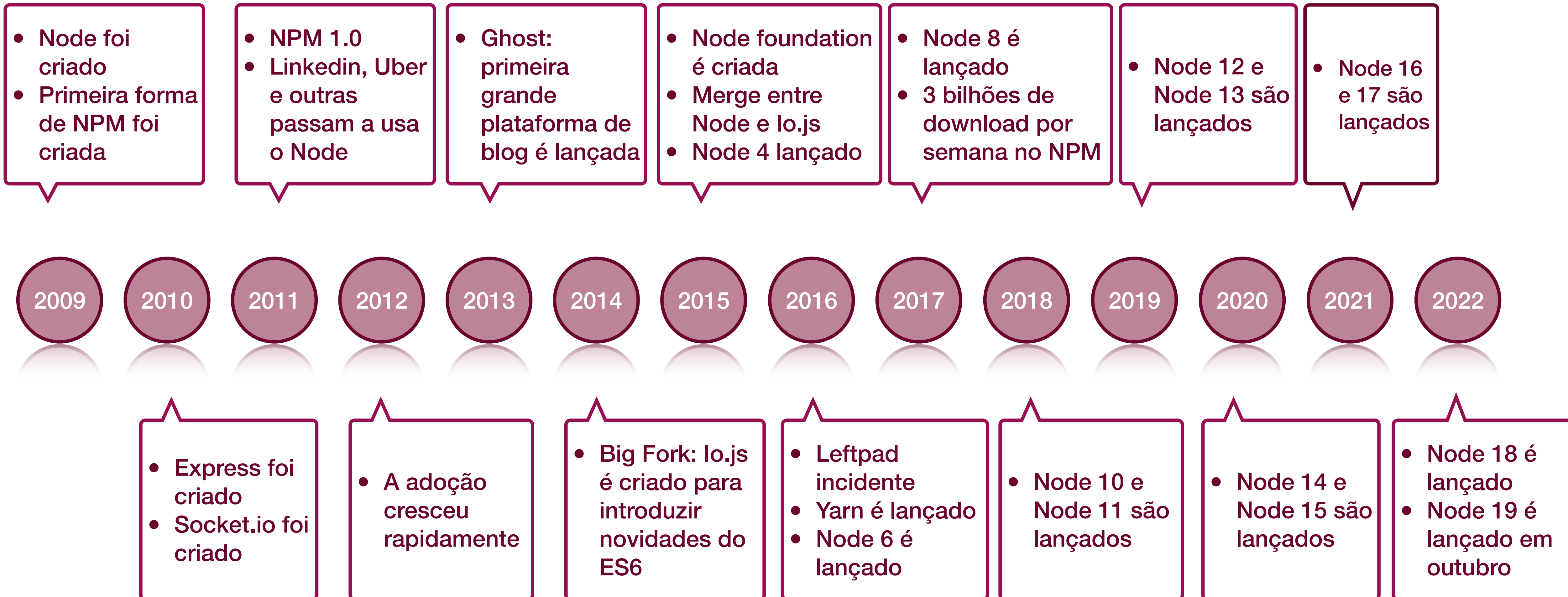
História - v8

- Engine de alto desempenho JavaScript e WebAssembly
- Escrita em C++
- Usada no Chrome e no Node entre outros projetos
- Compila e executa código JS, gerencia a alocação de memória e realiza a desalocação de objetos não necessário (*garbage collector*)



Introdução ao Node

História - Timeline



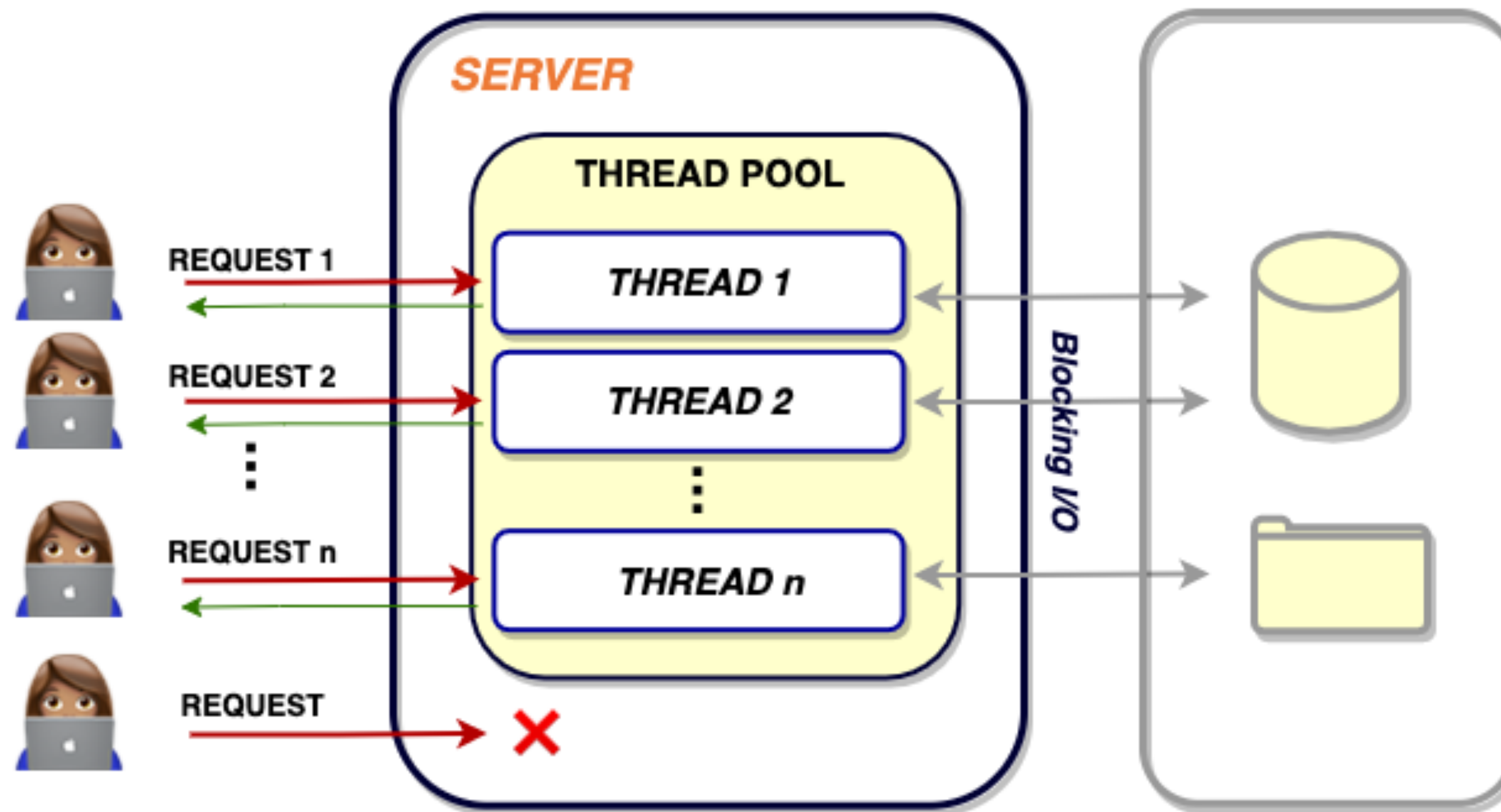
Introdução ao Node

Características

- Node.js app são executadas em um **único processo**
 - Não é necessária a criação de uma thread para cada requisição
- Fornece um conjunto de operações primitivas de I/O assíncronas
 - Evita que códigos de maneira geral sejam “**bloqueantes**”
- Escalável e mais simples de debugar, não há concorrência entre threads
- Novidades do **ECMAScript** podem ser usadas sem problemas já que o usuário possui o controle do ambiente de execução
 - No front-end dependemos dos navegadores

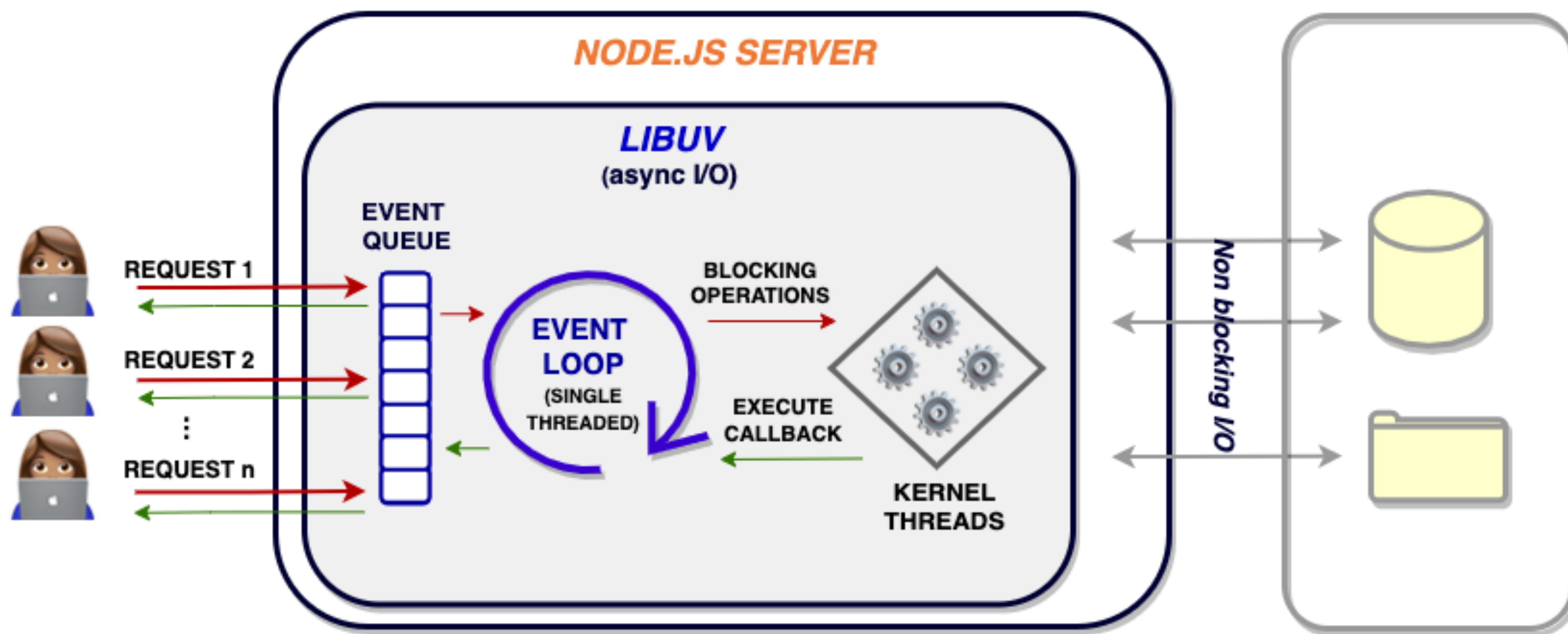
Introdução

Características



Introdução

Características



Introdução

Vantagens

- Excelente desempenho e escalável
- Escrito em JS, familiar para desenvolvedores Web
- Grande comunidade de usuários e desenvolvedores
- O gerenciador de pacote do Node, NPM, prover acesso a diversas bibliotecas reusáveis
 - Gerenciamento de dependências
- Portável, disponível para Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS, and NonStop OS

Criando um projeto Node



Criando um projeto Node

Hello World

- O seu primeiro programa em Node

```
console.log("Olá mundo");
```

```
node app.js
```

Criando um projeto Node

Módulos nativos

- [assert](#)
- [buffer](#)
- [child_process](#)
- [console](#)
- [cluster](#)
- [crypto](#)
- [dgram](#)
- [dns](#)
- [events](#)
- [fs](#)
- [http](#)
- [http2](#)
- [https](#)
- [net](#)
- [os](#)
- [path](#)
- [perf_hooks](#)
- [process](#)
- [querystring](#)
- [readline](#)
- [repl](#)
- [stream](#)
- [string_decoder](#)
- [timers](#)
- [tls](#)
- [tty](#)
- [url](#)
- [util](#)
- [v8](#)
- [vm](#)
- [wasi](#)
- [worker](#)
- [zlib](#)

Criando um projeto Node

Módulos nativos

Nome	Descrição
<u>console</u>	Prover um console para debug
<u>events</u>	Prover uma API para o gerenciamento de eventos
<u>fs</u>	Prover uma API para interagir com o sistema de arquivos
<u>http</u>	Prover uma implementação HTTP cliente/servidor
<u>os</u>	Prover propriedades e métodos utilitários relacionados ao sistema operacional
<u>path</u>	Prover utilitários para trabalhar com path e diretórios
<u>querystring</u>	Prover utilitários para “ <i>parsear</i> ” e formatar URL de string de consulta (querystring)

Criando um projeto Node

Módulos nativos

Nome	Descrição
<u>repl</u>	Prover uma implementação Read-Eval-Print-Loop (REPL) disponível como uma versão standalone, mas que também pode ser adicionada a outras aplicações
<u>timers</u>	Prover funções para agendar execuções de funções em um período futuro
<u>url</u>	Prover utilitários para resolução e “parseamento” de URL

Criando um projeto Node

NPM

- Node Package Manager - Gerenciador de pacotes do Node
- Inicialmente era uma maneira de fazer download e gerenciar as dependências
- Atualmente é também utilizado em projetos front-end
- Possui mais de 1.3 milhões de pacotes disponíveis
- Maior repositório de software do mundo



Criando um projeto Node

- Para iniciar um projeto **node**, é necessário criar um arquivo chamado **package.json**
 - Lista todas as dependências do projeto e suas versões
 - Torna o processo de build reproduzível e portanto mais fácil de compartilhar com outros desenvolvedores
 - Deve conter pelo menos o atributo **name** e **version**
- A maneira mais simples de criar esse arquivo é usando o comando:
 - **\$ npm init --yes**

Criando um projeto Node

```
{
  "name": "my package",
  "description": "",
  "version": "1.0.0",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/monatheoctocat/my_package.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/monatheoctocat/my_package/issues"
  },
  "homepage": "https://github.com/monatheoctocat/my_package"
}
```

Nome do diretório

Informação contida no README ou string vazia

Sempre 1.0.0

Script de test vazio

No caso de se um repositório git

Sempre vazio

Sempre vazio

Por padrão ISC

Caso hospedado no GitHub

Caso hospedado no GitHub

Criando um projeto Node

NPM e suas funções

- Instalar e atualizar dependências
 - `$ npm install` ou `$ npm install <package-name>`
 - `$ npm update` ou `$ npm update <package-name>`
- Versionamento
- Execução de tarefas
 - Ex: Executar em produção, testar ...

Criando um projeto Node

Pacotes populares

ex

Express



Socket.io

asynC

UNDERSCORE.JS

AXIOS



Fastify



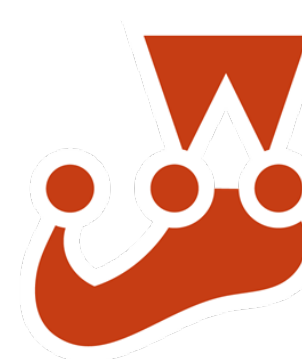
Rxjs



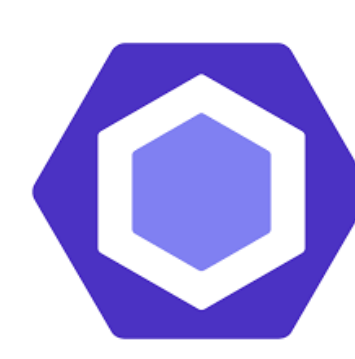
Lodash



Ramda



Jest



ESLint



Nodemon



Dotenv



Yargs



Passport



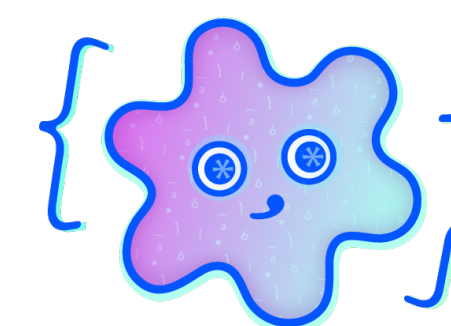
Nodemailer



Mongoose



date-fns

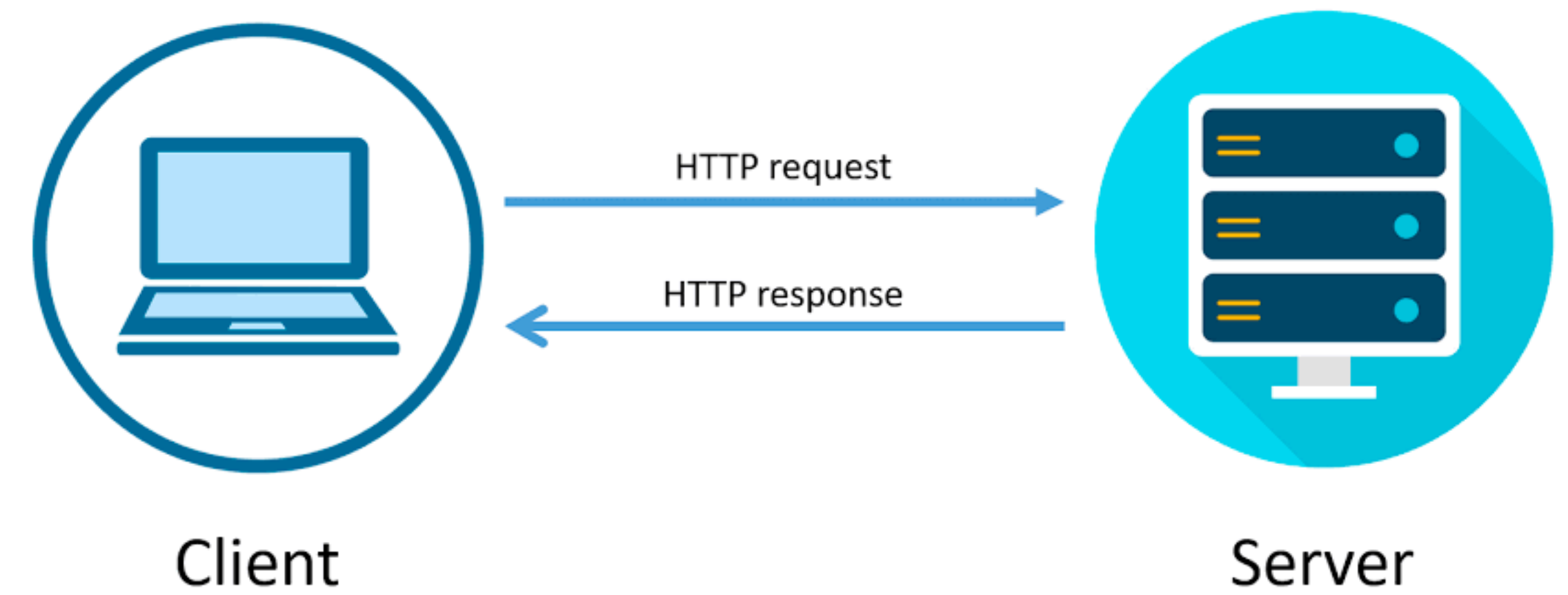


glob



[Lista completa contendo 40 pacotes](#)

Prática



Construindo nosso servidor web

- Criando um projeto Node com suporte a Typescript

```
# Criando um projeto usando o NPM
```

```
npm init -y
```

```
# Instalando os tipos do NodeJS do TypeScript
```

```
npm install -D typescript @types/node
```

```
# Gerando o arquivo de configuração do TypeScript
```

```
npx tsc --init
```

```
# Instalando o Nodemon e o ambiente de execução do  
TypeScript
```

```
npm install -D nodemon ts-node
```

Construindo nosso servidor web

- Configurando o package.json

```
# Adicionando o comando de execução
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "nodemon src/app.ts"
}
```

- Não esquece de criar o arquivo **app.ts** dentro da pasta **src**

Construindo nosso servidor web

- Criando um projeto Node com suporte a Typescript

```
import { IncomingMessage, ServerResponse, createServer } from 'http'

const port = 1010

const server = createServer(async (req: IncomingMessage, res: ServerResponse) => {
  console.log(`${req.method} - ${req.url}`)
  res.setHeader('Content-Type', 'text/html')
  res.writeHead(200)
  res.end('<html><head></head><body>Oi mundo</body></html>')
});

server.listen(port, () => {
  console.log(`Servidor pronto. Escutando requisições na porta ${port}`)
})
```

Criando um servidor HTTP

Iniciando o modo de escuta

Construindo nosso servidor web

- O resto é com vocês!

Referências

- [Web Design History Timeline](#)
- [Common Gateway Interface](#)
- [Build Your Own Web Server From Scratch In Node.JS](#)
- [Introduction to Node.js](#)
- [How To Create a Web Server in Node.js with the HTTP Module](#)
- [40 Useful NPM Packages for Node.js Apps in 2021](#)
- [The Node Core Modules](#)
- [Node.js Architecture and 12 Best Practices for Node.js Development](#)
- [How to set up TypeScript with Node.js and Express \(2023\)](#)

Por hoje é só