Colorindo a TV

Imagem de Alexander Antropov por Pixabay

Módulo 3 - Aula 1 - Hard

RESILIA | Senac

Node.js

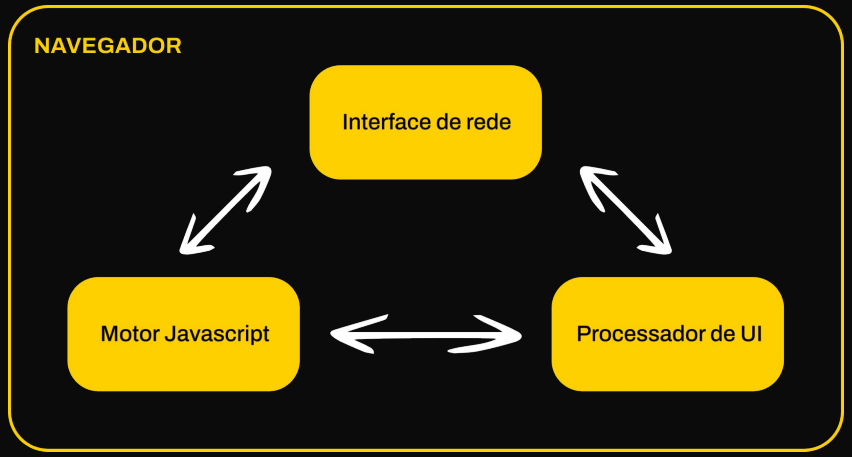
É um runtime de javascript, baseado no motor V8 , que nos permite rodar códigos Javascript fora do navegador.

Suas grandes vantagens são:

- Leveza
- Bibliotecas e comunidade
- Permitir a execução em multi-plataforma

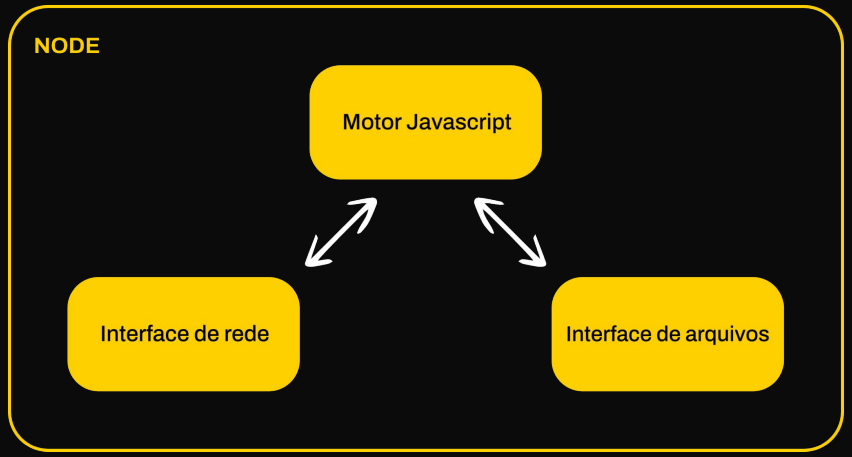
Node.js

Comparando com o navegador, seria como tirar o interpretador de JS do chrome o permitindo funcionar por si só.



Node.js

Com isso, o node passa a ter sua própria interface para se comunicar com a rede e com o sistema de arquivos.



Vamos ver o node.js na prática?

Olá node

Implemente um “hello node” no terminal usando o node.js:

- Crie um arquivo hello-node.js
- Utilize seus conhecimentos em JavaScript para que o programa exiba a mensagem “olá mundo!” no console.
- Pesquisar como executar um arquivo .js com o node.
- Execute o programa utilizando um terminal/powershell com node.

Promises

Promises são um mecanismo para executar código de forma assíncrona. Assim como no mundo real, as promessas podem demorar algum tempo para serem cumpridas ou descumpridas (resolvidas ou rejeitadas).

Promises - Estrutura e declaração

A estrutura de declaração de uma Promise é a abaixo:

```
const promessa = new Promise((resolve, reject) => {
  // Lógica de resolver ou rejeitar a promessa
});
```

Promises - Tratamento

Para cumprir ou descumprir (resolver e rejeitar) uma *promise* utilizamos as funções resolve e reject passadas para ela.
Ex:

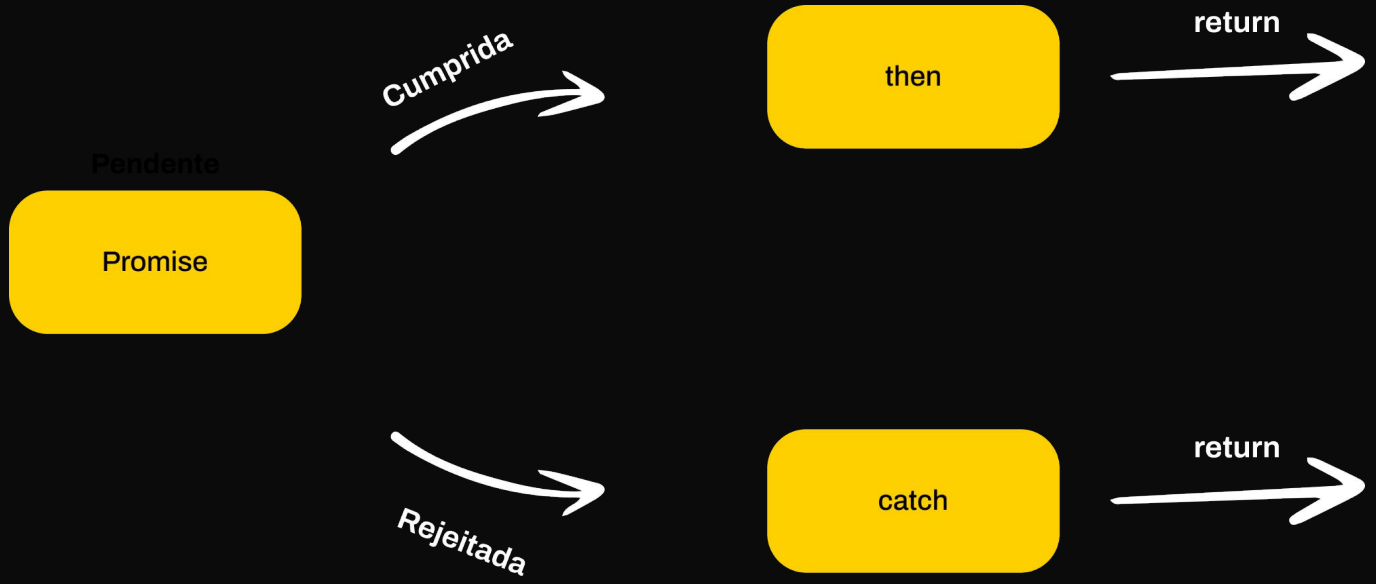
```
const promessa = new Promise((resolve, reject) => {
  if (condiçãoSatisfeita) {
    resolve("Valor a ser retornado se cumprida");
  } else {
    reject("Valor a retornado se descumprida");
  }
});
```

Promises - Tratamento

Para tratar de forma assíncrona o resultado de uma Promise, utilizamos then (quando ela é cumprida) e catch (quando descumprida).
Ex:

```
const promessa = new Promise((resolve, reject) => {
  if (condiçãoSatisfeita) {
    resolve("Valor a ser retornado se cumprida");
  } else {
    reject("Valor a retornado se descumprida");
  }
})
  .then((retorno) => console.log(`Promessa resolvida com: ${retorno}`))
  .catch((erro) => console.log(`Promessa rejeitada com: ${erro}`));
```

Promises - graficamente



A mãe de Carlos o prometeu que iria comprar um **carro** para ele, porém o Carlos precisa da nossa ajuda para simular os **possíveis resultados desta promessa,**

vamos lá?

LIVE CODING

Atividade: A espera de um sinal

A estação espacial internacional precisa manter-se em contato constante com a terra. No entanto, ocorre um delay de 2.5 segundos nessa comunicação. Além disso, existe a possibilidade dessa comunicação ser perdida. Utilize node para simular essa situação, utilizando:

- Uma promessa que:
  - Executa uma Chamada a função fornecida (comunicacaoPerdida)
  - Caso a comunicação tenha sido perdida, rejeita a promessa com: “Comunicação perdida”
  - Caso a comunicação tenha sido enviada, resolve a promessa com: “Ok, todos vivos na estação”
  - Trata o caso de sucesso (then) exibindo: `Sucesso:  ${msgSucesso}`
  - Trata o caso de falha (catch) exibindo: `Falha: ${msgFalha}`

O PROBLEMA DAS CORES

Cores no terminal

Cores são usadas por sistemas/programas para sinalizar uma importância.

Ex:

- Verde: Sucesso
- Amarelo: Aviso/Alerta
- Vermelho: Erro grave/fatal

Código de cores

Podemos utilizar diferentes códigos de cores em strings para modificar a aparência da saída no terminal.

Ex:

Cor da fonte:

- FgBlack = "\x1b[30m"
- FgRed = "\x1b[31m"
- FgGreen = "\x1b[32m"
- FgYellow = "\x1b[33m"
- FgBlue = "\x1b[34m"

Bibliotecas/pacotes

São códigos criados com uma finalidade específica, envelopados e distribuídos para que outras pessoas possam utilizá-los.
Ex:

- JQuery
- Chalk

Bibliotecas/pacotes - Node

Em Node existem diversas bibliotecas disponíveis. Algumas são nativas e outras são disponibilizadas pela comunidade.

Onde encontramos os pacotes?

NPM

NPM significa Node Package Manager. Ele é responsável por gerenciar os pacotes disponibilizados pela comunidade que queremos utilizar nos nossos projetos.

NPM - Intro

Para instalar um pacote (como o chalk) utilizamos o comando abaixo:

```
npm install chalk@4
```

Chalk

O Chalk é uma biblioteca que abstrai os códigos das cores e torna mais fácil a utilização de strings coloridas.

Chalk - Importando

Para importar uma biblioteca utilizaremos a palavra require e uma variável que receberá o conteúdo (funções e classes) fornecido por essa biblioteca como no exemplo abaixo:

```
const chalk = require("chalk");
```

Chalk - Olá mundo colorido

O Chalk possui uma vasta gama de funcionalidades e funções. Nós queremos utilizar funções básicas para exibir a seguinte saída no console:

"Olá mundo colorido"

Atividade em pares