

WebDev - Módulo 3

Roteiro de aula

Aula 1 - Hard: Colorindo a TV



Tópicos da aula:

- Introdução a pacotes/bibliotecas node
- Bibliotecas + NPM
- Introdução à Promises



Objetivos da aula:

- Compreender os princípios de node.js
- Compreender a introdução de Promises
- Compreender bibliotecas/pacotes
- Conhecer algumas bibliotecas existentes
- Entender a importância das cores no output do console
- Empregar a ideia das bibliotecas como uma possível resolução do problema das cores.



Atividades da Aula

→ Atividade 1: 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.

→ Atividade 2 : 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}`

→ Conteúdo 3 : Um pacote de cores - Chalk

- Link repositório do [npm](#)
- repositório do pacote [chalk](#) do node



Momento Aprendizagem em Pares

- Esse momento é dedicado para vocês desenvolverem suas demandas e entregas para o curso.
- Utilize esse tempo da maneira que preferir, mas atente-se às aulas que você deve realizar as entregas.
 - ◆ **Dica:** Nas Propostas dos projetos, vocês encontram uma sugestão de organização para a realização das atividades.
 - ◆ **Lembre-se:** O momento de Aprendizagem em Pares é justamente para fazer trocas e aprender em comunidade, aproveite seus colegas e se desenvolvam juntos!
- **Entregas:**
 - ◆ Projeto individual: aula 5 - HARD
 - ◆ Projeto em grupo: aula 10 - HARD
 - ◆ Apresentação do projeto: aula 10 - HARD



Revisão da aula

- Vamos fazer uma introdução ao **Node.js**, com atividades mão na massa para colorir uma tv. A palavra "node" pode ser traduzida do inglês como "nó" e essa alusão é pelo fato de o node.js ser um **eficiente construtor de back-end para nossas aplicações, com importantes bibliotecas que facilitam a vida dos programadores, na integração entre o cliente o servidor.** Muita dessa performance do node.js se dá pelo motor V8 do Google que é um runtime de javascript que nos permite rodar códigos Javascript fora do navegador, tendo como principais vantagens a sua leveza, as vastas bibliotecas e comunidades sobre a linguagem, e a possibilidade de execução em multi-plataforma.

O Node.js traz uma nova dinâmica ao ter sua própria interface para se comunicar com a rede e com o sistema de arquivos. A sua arquitetura é **assíncrona e orientada**

a eventos. Como funciona o clássico “**hello node**” no **Node.js**? Usando o terminal com o node.js, os passos são os seguintes:

- crie um arquivo hello-node.js;
- utilize os conhecimentos aprendidos para exibir a mensagem olá mundo no console;
- execute o programa utilizando um terminal/powershell com node.

→ Promises no Node.js


Outro ponto importante que compõem as principais funcionalidades do node são as **promises**. A palavra, do inglês, quer dizer promessa, que é um **mecanismo para executar código de forma assíncrona**. Assim como no mundo real, as promessas podem demorar algum tempo para serem cumpridas ou descumpridas e, portanto, podem ter três estados distintos:

Pendente: Quando a promessa for criada, o resultado é indefinido.

Cumprida: Quando a promessa é finalizada com sucesso, o resultado é um valor.

Rejeitada: Quando a promessa é finalizada com erro, o resultado é um objeto de erro.

A estrutura de declaração de uma Promise está representada abaixo:



```
1 const promessa = new Promise((resolve, reject) => {  
2   // Lógica de resolver ou registrar a promessa  
3 });
```

Para cumprir ou descumprir (resolver e rejeitar) um promise utilizamos as funções resolve e reject passadas para ela como no exemplo abaixo:

```
1  const promessa = new Promise((resolve, reject) => {
2    if (condiçãoSatisfeita) {
3      resolve("Valor a ser retornado se cumprida");
4    } else {
5      reject("Valor a retornado se descumprida");
6    }
7  });
```

- Durante a aula trouxemos ainda um **problema para ser resolvido**. O contexto foi o seguinte: 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.
- Para resolver esse problema, vamos desenvolver uma chamada a função fornecida (comunicacaoPerdida) e caso a comunicação tenha sido perdida, rejeitar a promessa com — “Comunicação perdida” . Caso a comunicação tenha sido enviada, resolver a promessa com: “Ok, todos vivos na estação”. Tratar o caso de sucesso (then) exibindo: `Sucesso: \${msgSucesso}`. Trata o caso de falha (catch) exibindo: `Falha: \${msgFalha}`.
- Para tratar de forma assíncrona o resultado de uma Promise, utilizamos **then**(quando ela é cumprida) e **catch**(quando descumprida), como representado abaixo:

```
1  const promessa = new Promise((resolve, reject) => {
2    if (condiçãoSatisfeita) {
3      resolve("Valor a ser retornado se cumprida");
4    } else {
5      reject("Valor a retornado se descumprida");
6    }
7  })
8  .then((retorno) => console.log(`Promessa resolvida com: ${retorno}`))
9  .catch((erro) => console.log(`Promessa rejeitada com: ${erro}`));
```

- Um outro tema abordado na aula, foi o processo de como funcionam as **bibliotecas do Node.js**. Mas peraí, o que entendemos por **biblioteca**, em programação?
- **São códigos criados com uma finalidade específica, envelopados e distribuídos para que outras pessoas possam utilizá-los**. Exemplo: JQuery, Express e o Chalk que vamos utilizar nessa aula. Outro fator crucial no desenvolvimento em Node.js é o **npm(Node Package Manager)** que vem junto com a instalação do node.js e é bastante útil.
- Mas **como usar essas bibliotecas?** Vamos **fazer um start** e precisamos, primeiramente, importar a biblioteca. Para isso, utilizamos a palavra `require` e uma variável que receberá o conteúdo (funções e classes) fornecido por essa biblioteca como no exemplo abaixo:




- Em seguida, é só **utilizar os recursos de que a biblioteca dispõe**. No nosso caso, na aula, vamos utilizar a biblioteca e vamos criar nossa primeira operação com bibliotecas, colorindo o nosso "hello node" usando a biblioteca importada.



Para ajudar



Links interessantes:

- Para aprender mais sobre Promise - JavaScript
<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise>
- GitHub - chalk/chalk:  Terminal string styling done right
<<https://github.com/chalk/chalk#readme>>
- Chalk Biblioteca NPM - Pacotes em Node.js
<<https://codezup.com/chalk-library-npm-package-in-node-js/>>
- NPM Pacote Manager & Node Modules Directory | Node.js
<<https://codezup.com/npm-package-manager-node-modules-directory-node-js/>>