Módulo 16: TypeScript (Parte 03)

Boas-vindas!

Bem-vindo caro aspirante a dev!
Chegamos ao módulo de TypeScript, um superset da linguagem Javascript fortemente tipada e usualmente utilizada com classes. Neste módulo vamos aprender um pouco sobre os tipos de dados em TypeScript, enums, classes, tipos genéricos e interfaces. Também como criar seu primeiro servidor backend com TypeScript.

Ao final deste encontro você deverá:

- Entender o uso de Typescript no frontend
- Relacionar os conhecimentos de orientação a objetos ao DOM
- Entender os conceitos de Generics, Interfaces e Aliases
- Usar promises em Typescript

Referências básicas:

- **'CustomElements':**
  - https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements
  - https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM
- **'Generics':**
  - https://www.typescriptlang.org/docs/handbook/2/generics.html
- **'Tipos para Objetos':**
  - https://www.typescriptlang.org/docs/handbook/2/objects.html

Objetivos:

Neste encontro começamos compreendendo como utilizar o typescript em nosso frontend, fazemos um build do nosso código e um arquivo html e conectamos o javascript resultante da build. A partir deste ponto, veremos o que são **Custom Elements** (ref.1 e ref.2) e utilizaremos os conceitos de orientação a objetos sobre elementos chave para criar nosso elementos. Então aprendemos sobre **Generics** (ref.3), **Aliases e Interfaces** (ref.4) e aplicamos estes conceitos no uso de **promises**, especificando assim o tipo recebido em uma requisição **fetch**.

Exercícios:

1. Pra que serve o **Generics**?
2. Qual a diferença entre **Aliases** e **Interfaces**?
3. Você deve usar o último projeto em Typescript e enviar os arquivos com zip (exceto os módulos do node), seguindo estes passos:

- Crie um **Custom Element** chamado `EmailInput` herdando de `HTMLElement`, nele nós criamos um shadow e dentro deste shadow um input e adicionamos uma definição `customElements.define("email-input", EmailInput)`
- Crie um **RegexValidator** herdando de **StringValidator** e também checando se o regexp `/^(\w{1,}@\w{1,}\.(\w{3})(\.\w{2}){0,1})$/gim` existe no **data** recebido pelo construtor, caso não exista deve retornar `throw new Error("O formato está errado");`
- Adicione um evento "**onchange**" ao input dentro do Custom Element, instanciando o RegexValidator com o valor recebido pelo input.
- Compile seu código typescript para JS
- Crie um **index.html** que chame o EmailInput e o vincule ao seu index.html

Em síntese:

Nesta parte do módulo você conheceu um pouco sobre TypeScript, como conceitos básicos, os tipos nativos e iniciar um projeto.
Lembre-se que o foco deste estudo é sempre aprofundar o seu conhecimento em web e, portanto, não se limite às referências deste módulo e mergulhe no conhecimento sobre TypeScript assistindo tutoriais e manuais disponíveis na internet.

Na próxima aula continuaremos nossos estudos sobre o TypeScript. Bons estudos!