Módulo 16: TypeScript (Parte 04)

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:

- Aplicar o uso de promises em Typescript
- Entender o tipo Partial<T>, Promise<T> e Array<T>
- Entender Typescript para backend

Referências básicas:

- **Fetch:** https://www.newline.co/@bespoyasov/how-to-use-fetch-with-typescript--a81ac257
- **Partial:** https://www.typescriptlang.org/docs/handbook/utility-types.html#partialtype
- **Promise:** https://www.educba.com/typescript-promise-type/
- **Array:** https://www.typescriptlang.org/docs/handbook/2/everyday-types.html#arrays
- **Errors:** https://fettblog.eu/typescript-typing-catch-clauses/

Objetivos:

Neste encontro começamos uso de **promises** em uma requisição **fetch**, então aprendemos sobre **Partial<T>**(ref.3), **Promises<T>** e **Array<T>** (ref.4) e especificando assim o tipo recebido.

Exercícios:

1. Você deve usar o último projeto em Typescript e enviar os arquivos com zip (exceto os módulos do node), seguindo estes passos:

   Os regex necessários serão:

   - email:

     ```
     /^(\w{1,}@\w{1,}\.(\w{3})(\.\w{2}){0,1})$/gim
     ```

   - password:

     ```
     /^\w{1,}$/gim
     ```

   - name:

     ```
     /^([a-z]{1,})([ ]{1}[a-z]{1,}){0,}$/gim
     ```

   → Modifique o **RegexValidator** para uma classe abstrata e adicione uma propriedade "get" com o nome "regex" que retorne um RegExp vazio: **new RegExp('')**;

   → Crie um **EmailValidator**, **PasswordValidator** e **NameValidator** herdando de **RegexValidator**, estes devem sobrescrever a propriedade "regex" para retornar as respectivas expressões regulares;

   → Compile seu código typescript para JS;

Exercícios Continuação:

2. Você deve usar o projeto anterior e enviar os arquivos com zip (exceto os módulos do node), seguindo estes passos:

   → Crie as classes EmailInput, NameInput e PasswordInput herdando de HTMLElements;

   → Valide as respostas no evento “onchange” de cada input e faça um try/catch e em caso de captura de erro limpe o input.value;

   → Crie as seguintes Interfaces com os atributos especificados:

      i. **APIResponse<T>**: data: **T**, errors: **Array<strings>**;

      ii. **UserData**: id, email e name;

      iii. **LoginData**: id

   → Crie um botão que quando clicado chame a função que chama cada requisição (**não execute a função caso algum dos inputs necessários esteja vazio**): Cadastrar, Logar e Atualizar;

   → Para cada botão faça uma função que execute as respectivas requisições para um servidor localhost:8000 com os seguintes dados:

      i. POST “/accounts/” => body: email, name e password;
         - response: **Response**<**UserData**>

      ii. POST “/accounts/login” => body: email e password;
         - response: **Response**<**LoginData**>

      iii. PATCH “/accounts/” => body: email, name e password;
         - response: **Response**<**UserData**>

   → Compile seu código typescript para JS;

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!