ES6 e TypeScript

-Introdução:

Neste documento serão abordados os conceitos, benefícios, criação e exemplos, tanto do Typescript quanto do ES6

-TypeScript:

Desenvolvido em 2012 pela Microsoft, o TypeScript pode ser considerado um superset, ou seja, ele não é uma linguagem de programação propriamente dita, mas ele tem funcionalidades que melhoram o JavaScript. Um bom exemplo disso é que é possível inserir código JavaScript dentro do type, pois o código será sempre transpilado para JS. Mas por que usar esse superset e quais são seus benefícios?

O TypeScript implementou conceitos e funcionalidades que não são somente com JavaScript puro, como por exemplo a tipagem estática a objetos. Pelo fato da linguagem não possuir uma tipagem bem desenvolvida, alguns bugs (como retorno de um valor undefined sendo passado de forma errada) são muito complicados de achar, pois são muitas e muitas variáveis. Abaixo pode-se ver alguns exemplos de tipagem e como são nomeados:

```
let ativo : boolean = true; // É assim que se nomeia uma variável do tipo booleana.

let quantidadeDeBananas: number = 8 ; // Quantidade de bananas é uma variável do tipo número, podendo ser tanto int quanto float.

let pessoa : string = "Constantino Neto"; // Constantino Neto é o valor da variável do tipo string neste exemplo.
```

Há outros tipos de tipagem, como Array , Enum e Any, mas outra coisa super interessante sobre o Typescript é que também é possível "tipar" funções, como por exemplo:

```
function anuncio() : void{
      alert("uStore é a maior empresa de nuvem do Brasil!")
}
```

Outra questão importante trazida pelo TypeScript, é que ela permite a utilização de programação orientada a objeto, dessa forma através dos quatro pilares (Herança, Encapsulamento, Polimorfismo e Abstração) podemos turbinar o nosso código. Um exemplo da eficiência do código estático do TypeScript pode ser visto mais adiante:

```
valor = 10.5;
let valor: number;
valor = 'frase qualquer'; // não é válido
```

A autenticação do tipo de objeto foi feita automaticamente!

Outros exemplos voltados para programação orientada a objeto podem ser vistos a seguir:

```
class ClasseExemplo {
      private _nome: string

      get nome(): string {
            return this._nome;
            }

      set nome(nome:string): void {
            this._nome = nome;
            }
}
```

Aqui podemos ver o princípio da abstração, pois o nome da classe está privado a só este arquivo, e o encapsulamento, pois pegamos ou “setamos” o nome através de funções que poderão ser chamadas em outros arquivos.

-ES6:

O ES5 ou ECMAScript 6(ou ECMAScript 2015, ano de lançamento), nada mais é do que uma padronização feita pelo ECMA (European Computers Manufacturers Association) do próprio JavaScript, ou seja, o ES6 é justamente o tão conhecido JavaScript (JavaScript hoje serve mais como um trademark). Mas o que o ES6 trouxe de novo?

Esta padronização trouxe novidades que fizeram a linguagem melhorar em vários quesitos, ela introduziu funcionalidades como por exemplo a arrow function (que nos permitiu criar funções sem a necessidade de nomeá-las), o const e o let, para nomeação (essa implementação faz com que a declaração por const ou let tenha um escopo de bloco, deixando assim o bloco mais organizado e tornando o var praticamente obsoleto) e várias outras mudanças nos parâmetros e funções. Seguem alguns exemplos das novidades:

```
tabela.addEventListener('click' ()=> {
      console.log("Um item da tabela foi clicado")
})


const express = require('express') // o valor dessa variável não se
altera

let contador = 0 // o valor da variável pode ser alterado
for(contador ; contador <= 9 ; contador++){
      console.log(contador)
}
```