TypeScript

Basicamente falando o TypeScript é um ***superSet(super conjunto)*** do javascript, então **tudo que você faz no** javascript **você pode fazer em** TypeScript, mas não o contrário.

O TypeScript traz fortemente os conceitos de orientação a objetos, então agora as variáveis podem ter tipos explícitos *(tipagem estática)* por exemplo.

Um objeto criado só com javaScript

Digamos que por exemplo eu precise **criar um objeto** que represente uma **negociação** no **javaScript**, eu poderia fazer isso da seguinte maneira:

```
export class Negociacao {

    #data;
    #quantidade;
    #valor;
    #volume;

    constructor(data, quantidade, valor){
        this.#data = data;
        this.#quantidade = quantidade;
        this.#valor = valor;
    }
```

```
get data(){
    return this.#data;
}

get quantidade(){
    return this.#quantidade
}

get valor(){
    return this.#valor
}

get volume(){
    return  this.#quantidade * this.valor;
}
```

Esse **objeto** tem **algumas regras**, depois que ele for *construído(instanciado)* **seus valores não podem ser mudados**, ele **obrigatoriamente precisa ter uma data, um valor e uma quantidade**, o volume **é calculado a partir do valor e quantidade.**

Para aplicar essas regras eu vou ter **atributos (privados)**, um **construtor** e **getters para leitura desses atributos**, além de um **getter específico que tem uma regra de negócio.**

Se eu colocar isso na prática eu posso observar se os comportamentos esperados estão refletindo no código *(ele está **sendo criado**, o **cálculo de volume está correto**, e não é possível alterar o valor de uma propriedade que já foi atribuída, ótimo):*

```
const negociacao = new Negociacao(new Date(), 10, 600)
console.log(negociacao)
console.log(negociacao.volume)
negociacao.valor = 600
```

```
app.js:4
Negociacao {#data: Wed Jun 01 2022 16:03:49 GMT-0300
▶ (Horário Padrão de Brasília), #quantidade: 10, #valo
r: 600, #volume: undefined}
6000                                          app.js:5
▶ Uncaught TypeError: Cannot set property     app.js:6
valor of #<Negociacao> which has only a getter
    at app.js:6:18
```

Legal, mas e se eu começasse a **errar algumas coisas na hora do código**, sou humano afinal de contas:

```
const negociacao = new Negociacao()
console.log(negociacao)
negociacao.quantidad = 2;
console.log(negociacao.quantidad)
```

```
app.js:4
Negociacao {#data: undefined, #quantidade: undefined, #valor: undefined, #volume: undefined}
app.js:4
Negociacao {#data: undefined, #quantidade: undefined, #valor: undefined, #volume: undefined}
  quantidad: 2
  #data: undefined
  #quantidade: undefined
  #valor: undefined
  #volume: undefined
  data: (…)
  quantidade: (…)
  valor: (…)
  volume: (…)
  [[Prototype]]: Object
2
app.js:6
```

Pera aí, eu consegui criar um objeto de negociação sem ter os valores necessários para cria-lo ***(quebrei uma regra do meu modelo)***, eu **também errei o nome de um atributo**, mas **JS** é dinâmico então **ele simplesmente criou uma nova propriedade.**

Mas meu código compilou de boa ué, e é por isso que é um problema. **Todos os meus possíveis erros passaram da minha fase de compilação**, muito em fator de que o **JS** é uma linguagem dinâmica e sem tipagem.

Compilador do TypeScript

Quando instalamos o **TypeScript** em um projeto, temos que configurar seu compilador *(o compilador do **TypeScritp** só funciona com node, pois ele precisa ser compilado em ECMASCRIPT para o navegador rodar)*. ***Em frameworks de front end, esse processo de compilação automática, configuração e afins já vem um default útil pra maioria dos casos****

O arquivo principal de configuração do **TypeScript** é o ***"tsconfig.json".*** Esse arquivo pode conter informações de compilação **como diretório alvo dos arquivos compilados, a versão do JavaScript que será usada na compilação, restringir a compilação a funcionar somente se não houver erros em arquivos ts a localização dos arquivos que devem ser compilados e etc...:**

```
app.ts    tsconfig.json
tsconfig.json > ...
1  {
2      "compilerOptions": {
3          "outDir": "dist/js",
4          "target": "ES6",
5          "noEmitOnError": true
6      },
7      "include": ["app/**/*"]
8  }
```