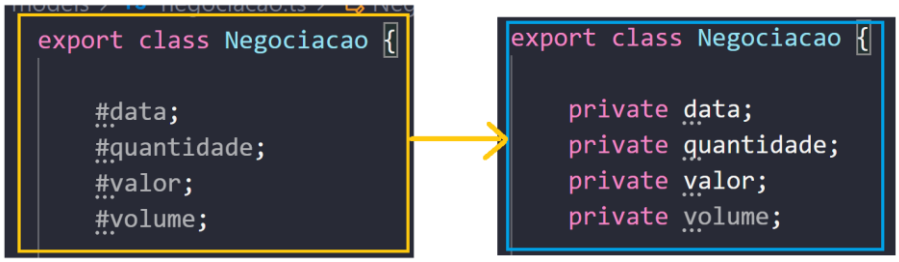
TypeScript - Encapsulamento

Agora que eu estou utilizando **typeScript** que me dá poderes de orientação a objetos, eu posso utilizar **modificadores de acesso**, então invés de eu **escrever em sintaxe de javascript** para **deixar meus atributos privados** eu posso usar a palavra reservada ***"private"*** que vai ter o **mesmo comportamento** *(impedir que agentes externos modifiquem o valor da minha classe):*



Em typeScript os Getters não podem ter o mesmo nome do atributo, então existe algumas abordagens para resolver isso, você pode usar métodos com verbo getter(estilo Java) ou mudar o nome *

```
getData(){
    return this.data;
}
```

TypeScript – Tipagem

TypeScript assume por default *(caso você não explicite)* o tipo **any** nos atributos das classes, mas isso não é tipar, um any é qualquer coisa. Então é legal que o **compilador esteja configurado pra não aceitar o any de maneira implícita**, somente explicita:

```
"compilerOptions": {
    "outDir": "dist/js",
    "target": "ES6",
    "noEmitOnError": true,
    "noImplicitAny": true
},
"include": ["app/**/*"]
```

Agora essa configuração **vai ajudar muito a lidar com atributos tipados**, o que nos leva a outro recurso do **TypeScript**, a tipagem estática, que nada mais é do que **definir tipos**:

```
export class Negociacao {

    private data : Date;
    private quantidade : number;
    private valor : number;
    private volume : number;

    constructor(data : Date, quantidade : number, valor : number){
        this.data = data;
        this.quantidade = quantidade;
        this.valor = valor;
    }
```

```
export class NegociacaoController{

    private inputData : HTMLInputElement;
    private inputQuantidade : HTMLInputElement;
    private inputValor : HTMLInputElement;
```

A **definição do tipo** não se aplica somente a atributos, também é **possível fazer o mesmo em retornos de funções:**

```
criaNegociacao() : Negociacao{
```