

EXERCÍCIO 01 - TYPESCRIPT

Aluno: Antônio Gabriel da Silva Moura

1) Verifique nas alternativas abaixo quais compilam ou não. Explique o motivo:

a) `let a = 10;`
`a = "2";`

R: Não irá compilar pois um tipo 'string' não pode ser atribuído para um tipo 'number'

b) `let b: any = 10;`
`b = 2;`

R: Irá compilar corretamente, pois o 'any' permite que o b receba qualquer outro tipo.

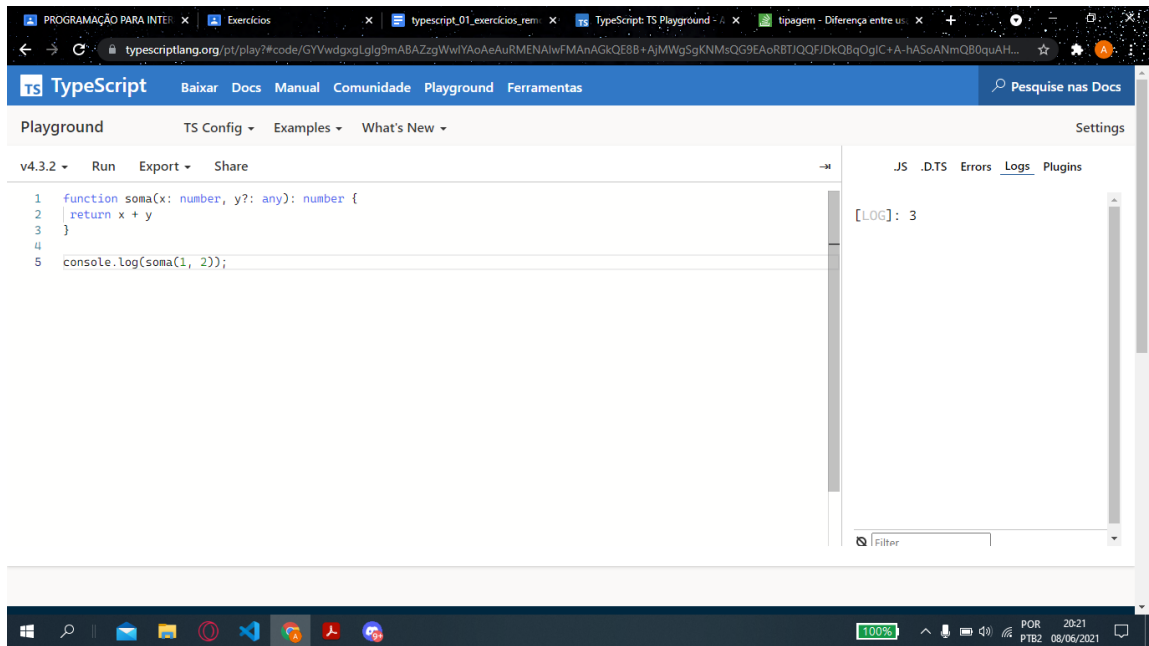
c) `let c: number = 10;`
`c = 2;`

R: Irá compilar corretamente, pois foi definido que o c seria do tipo 'number'.

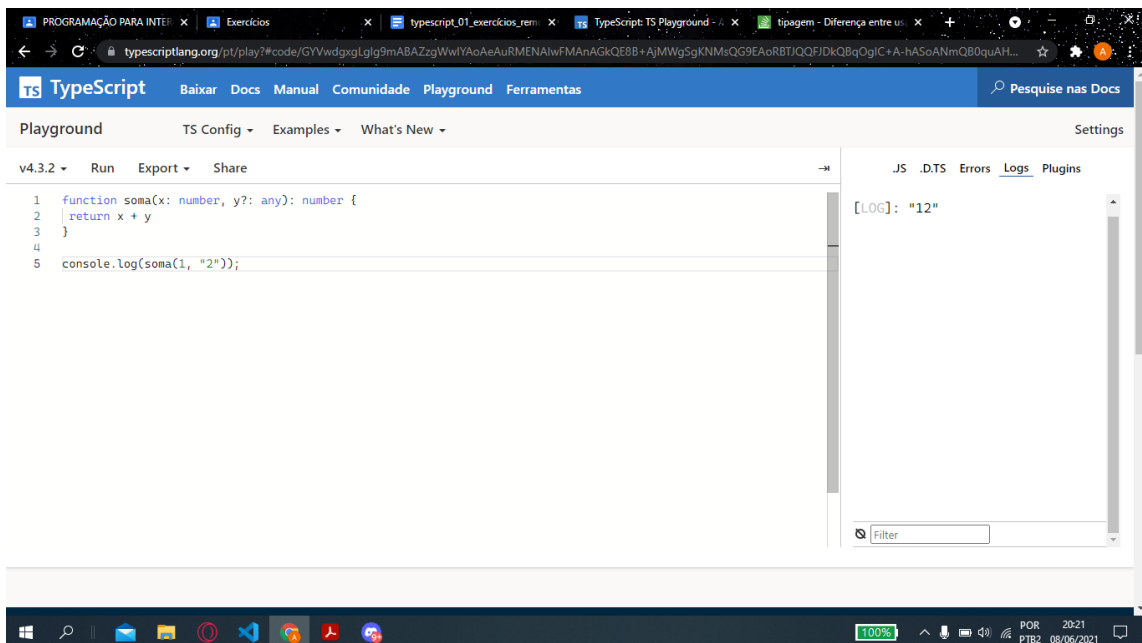
2) Dada a função soma abaixo, tente executar os scripts das alternativas e exiba os eventuais resultados:

```
function soma(x: number, y?: any): number {  
  return x + y  
}
```

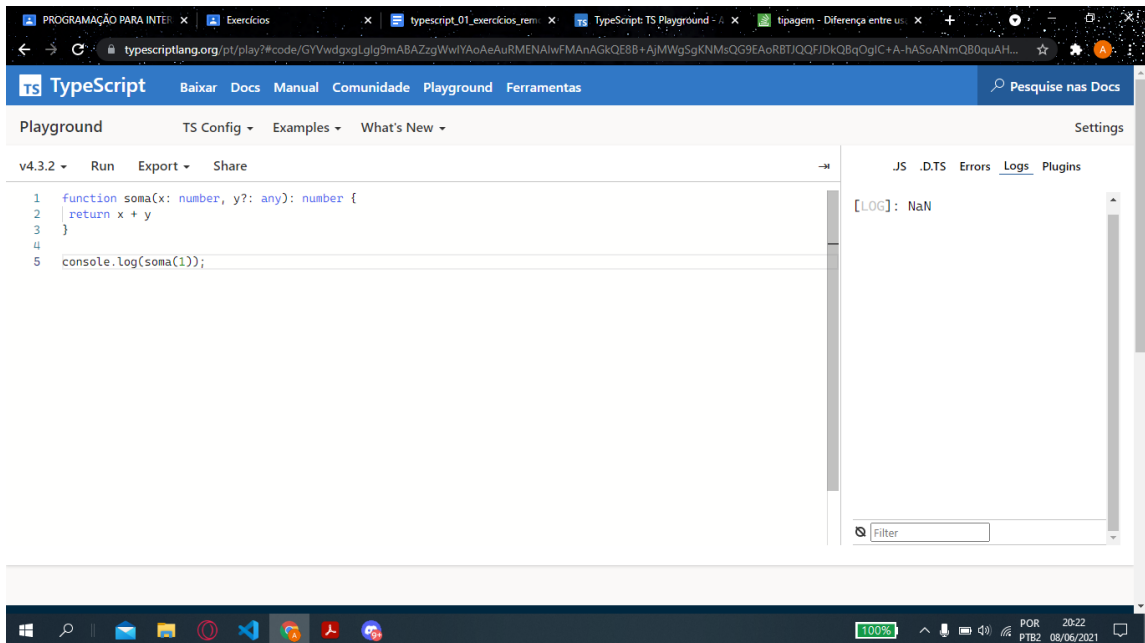
a) `console.log(soma(1, 2));`



b) `console.log(soma(1, "2"));`

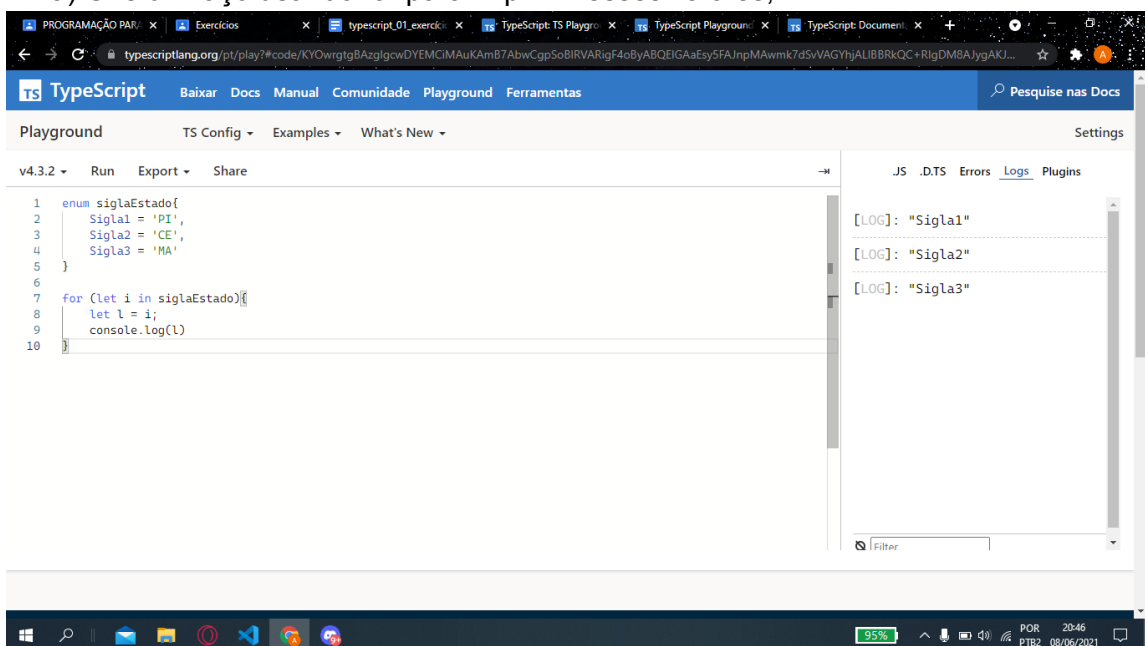


c) `console.log(soma(1));`

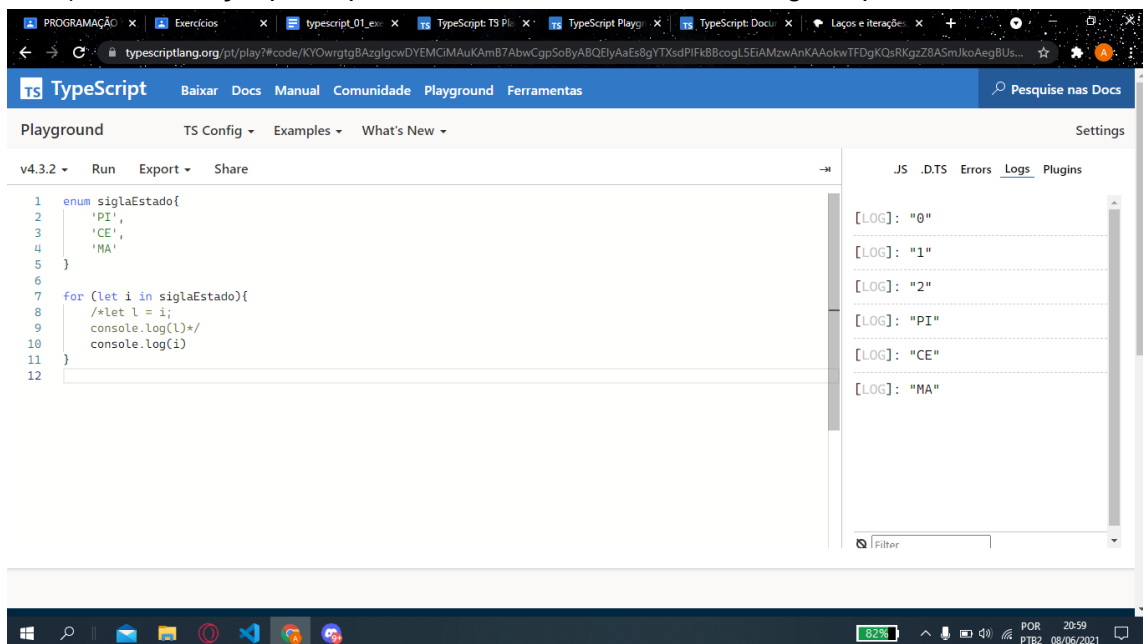


3) Crie uma enum com as siglas dos estados "PI", "CE", "MA" e implemente as duas alternativas abaixo:

a) Crie um laço usando for para imprimir esses valores;



b) Crie um laço que imprima os índices dessa enum e diga o que aconteceu.



The screenshot shows the TypeScript Playground interface. The code in the editor is as follows:

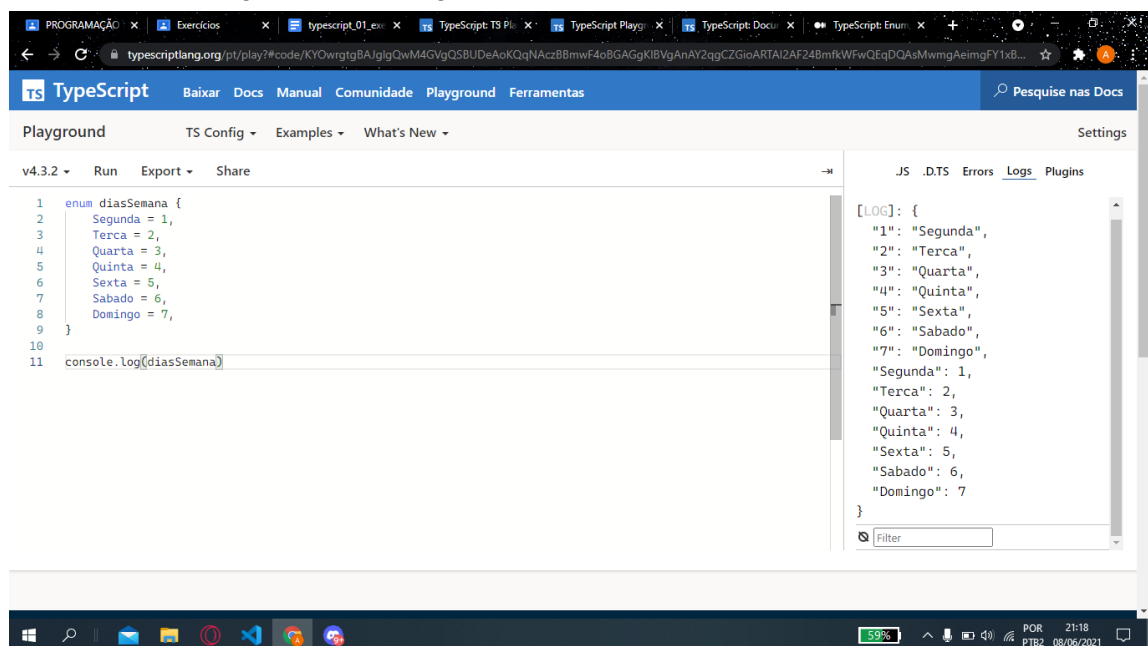
```
1 enum siglaEstado{
2   'PI',
3   'CE',
4   'MA'
5 }
6
7 for (let i in siglaEstado){
8   /*let l = i;
9   console.log(l)*/
10  console.log(i)
11 }
12
```

The right-hand pane shows the console output with the following log messages:

```
[LOG]: "0"
[LOG]: "1"
[LOG]: "2"
[LOG]: "PI"
[LOG]: "CE"
[LOG]: "MA"
```

4) Sobre enums, implemente o seguinte:

a) Crie uma enum chamada DiasSemana com os valores representando os dias da semana segunda a domingo;



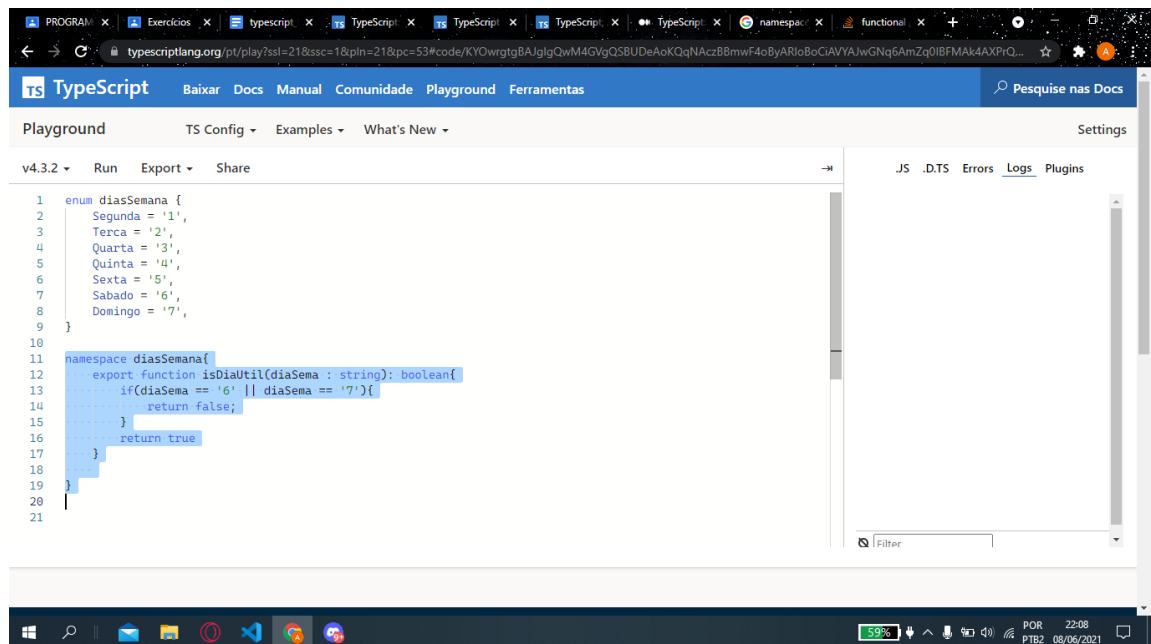
The screenshot shows the TypeScript Playground interface. The code in the editor is as follows:

```
1 enum diasSemana {
2   Segunda = 1,
3   Terca = 2,
4   Quarta = 3,
5   Quinta = 4,
6   Sexta = 5,
7   Sabado = 6,
8   Domingo = 7,
9 }
10
11 console.log(diasSemana)
```

The right-hand pane shows the console output with the following log message:

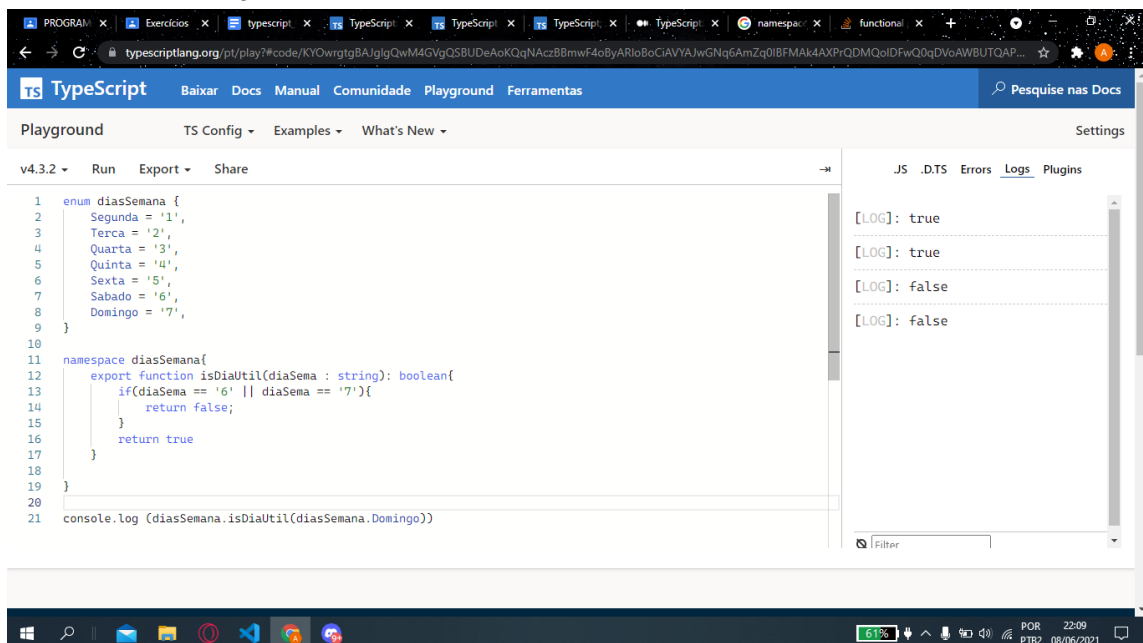
```
[LOG]: {
  "1": "Segunda",
  "2": "Terca",
  "3": "Quarta",
  "4": "Quinta",
  "5": "Sexta",
  "6": "Sabado",
  "7": "Domingo",
  "Segunda": 1,
  "Terca": 2,
  "Quarta": 3,
  "Quinta": 4,
  "Sexta": 5,
  "Sabado": 6,
  "Domingo": 7
}
```

- b) Crie um namespace com mesmo nome e dentro dele crie uma função chamada `isDiaUtil` receba um parâmetro do tipo `DiasSema` e retorna `false` se for um sábado ou domingo e retorna `true` caso contrário;



```
1 enum diasSemana {
2   Segunda = '1',
3   Terca = '2',
4   Quarta = '3',
5   Quinta = '4',
6   Sexta = '5',
7   Sabado = '6',
8   Domingo = '7',
9 }
10
11 namespace diasSemana {
12   export function isDiaUtil(diaSema : string): boolean {
13     if(diaSema == '6' || diaSema == '7'){
14       return false;
15     }
16     return true
17   }
18 }
19
20
21
```

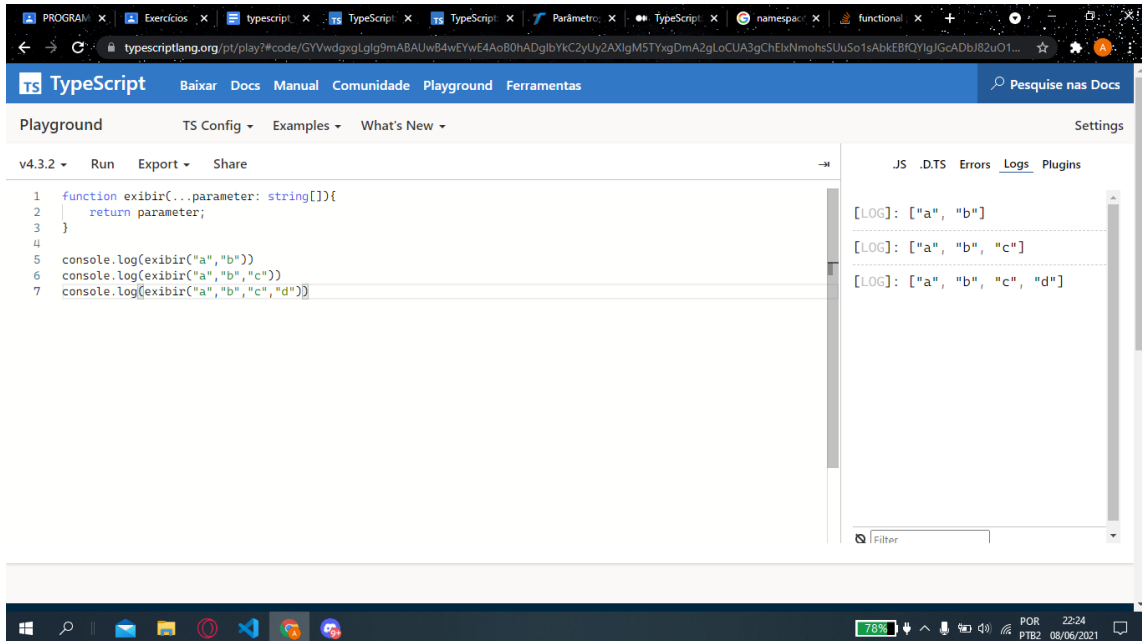
- c) Escreva também um script que declara uma variável do tipo da enum e que testa a função `DiasSemana.isDiaUtil()`.



```
1 enum diasSemana {
2   Segunda = '1',
3   Terca = '2',
4   Quarta = '3',
5   Quinta = '4',
6   Sexta = '5',
7   Sabado = '6',
8   Domingo = '7',
9 }
10
11 namespace diasSemana {
12   export function isDiaUtil(diaSema : string): boolean {
13     if(diaSema == '6' || diaSema == '7'){
14       return false;
15     }
16     return true
17   }
18 }
19
20 console.log (diasSemana.isDiaUtil(diasSemana.Domingo))
21
```

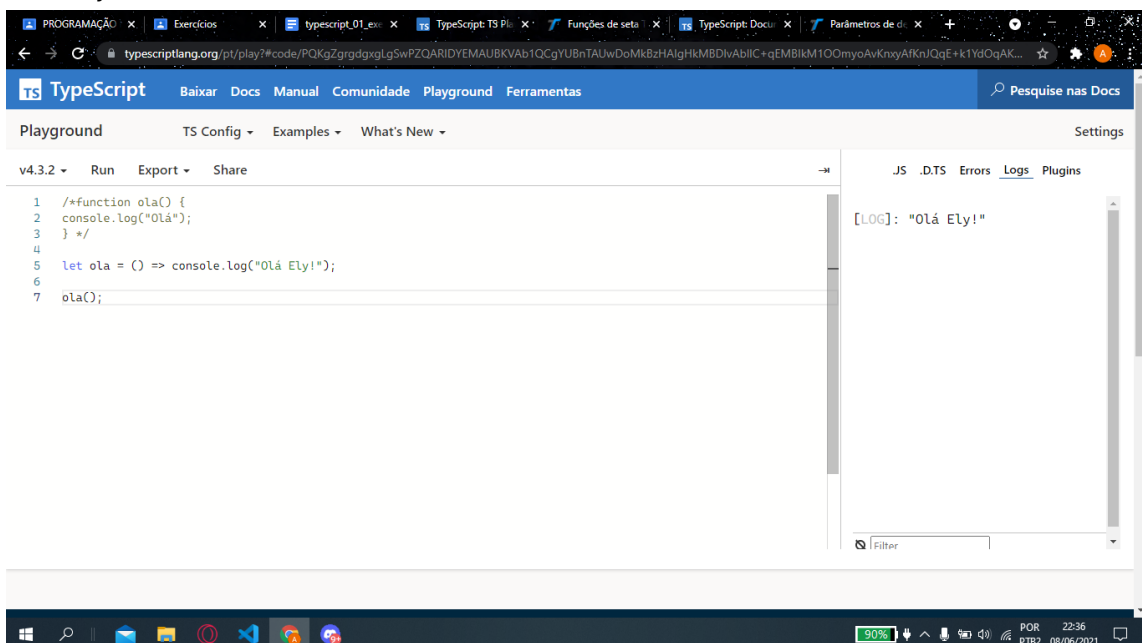
- 5) Crie uma função chamada `exibir` que receba como parâmetro um "rest parameter" representando strings. A função deve exibir no log cada um dos elementos do "rest parameter". Chame a função usando diferentes quantidades de parâmetros conforme abaixo:

```
exibir("a", "b");
exibir("a", "b", "c");
exibir("a", "b", "c", "d");
```



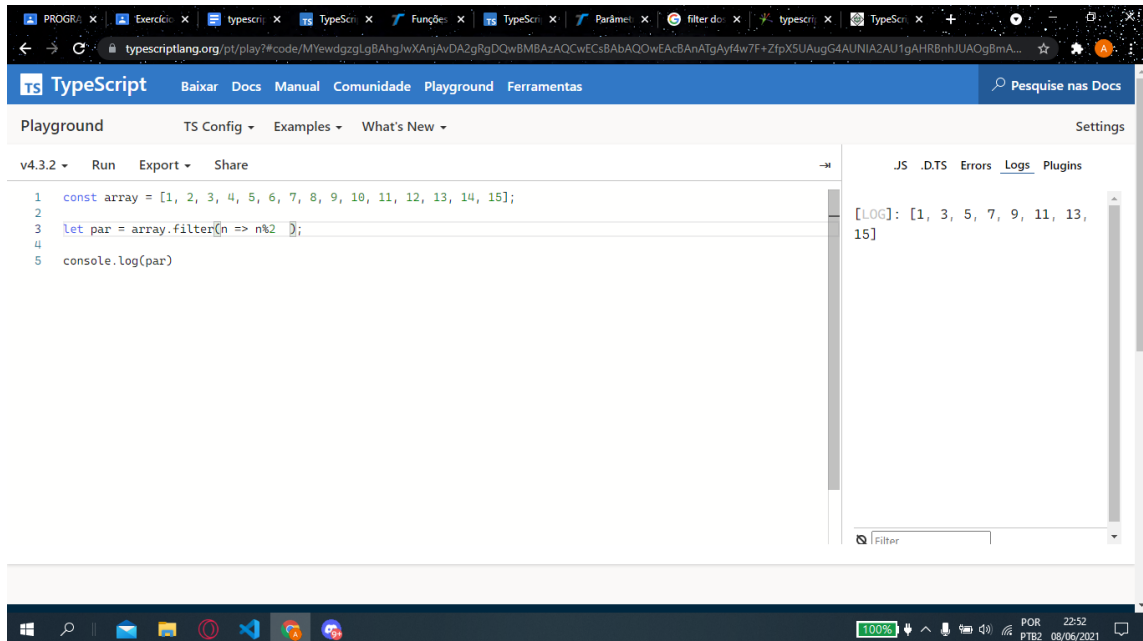
- 6) Converta em arrow function a seguinte função:

```
function ola() {
  console.log("Olá");
}
```



7) Dado método filter dos arrays, crie uma implementação usando arrow function que filtre todos os elementos pares do array abaixo:

```
const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
```



The screenshot shows the TypeScript Playground interface. The code editor contains the following code:

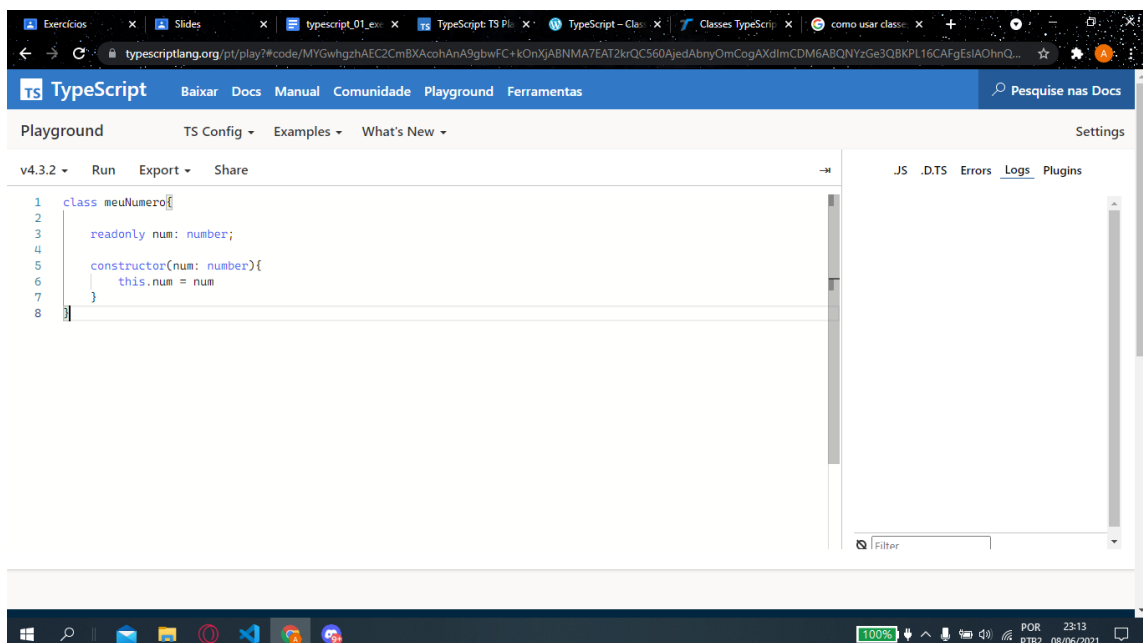
```
1 const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
2
3 let par = array.filter(n => n%2 !== 0);
4
5 console.log(par)
```

The right-hand pane shows the output in the console:

```
[LOG]: [1, 3, 5, 7, 9, 11, 13, 15]
```

8) Crie uma classe chamada MeuNumero tenha:

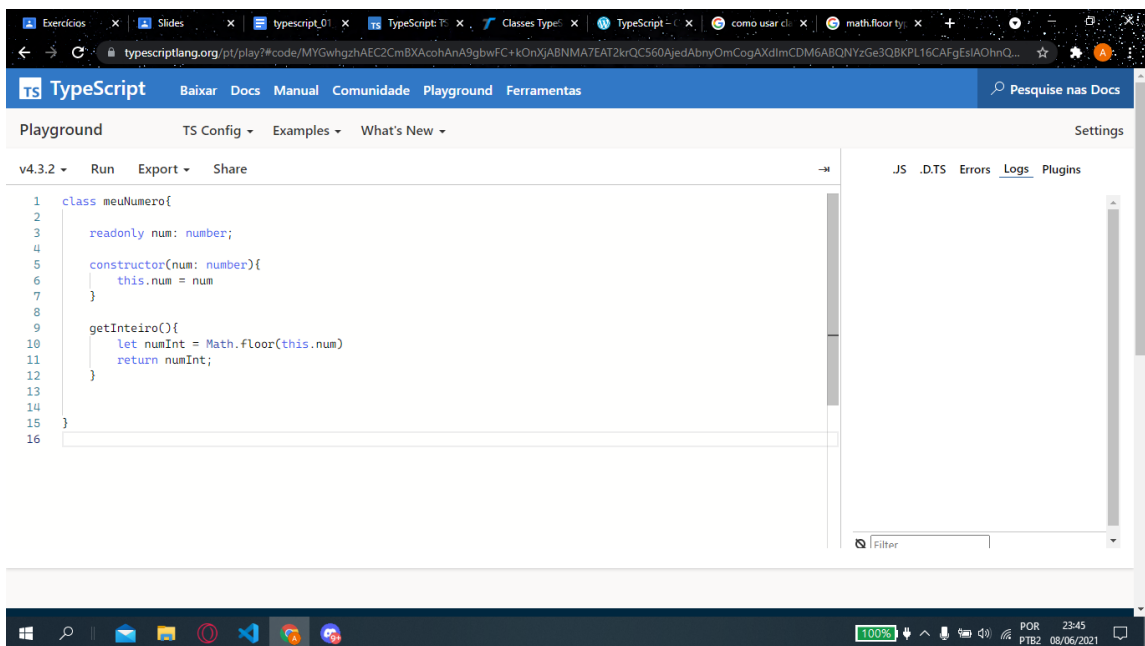
a) Um atributo chamado numero que seja somente leitura e inicializado no construtor;



The screenshot shows the TypeScript Playground interface. The code editor contains the following code:

```
1 class meuNumero{
2
3   readonly num: number;
4
5   constructor(num: number){
6     this.num = num
7   }
8 }
```

b) Um método chamado `getInteiro` que retorne a parte inteira do atributo `numero`;



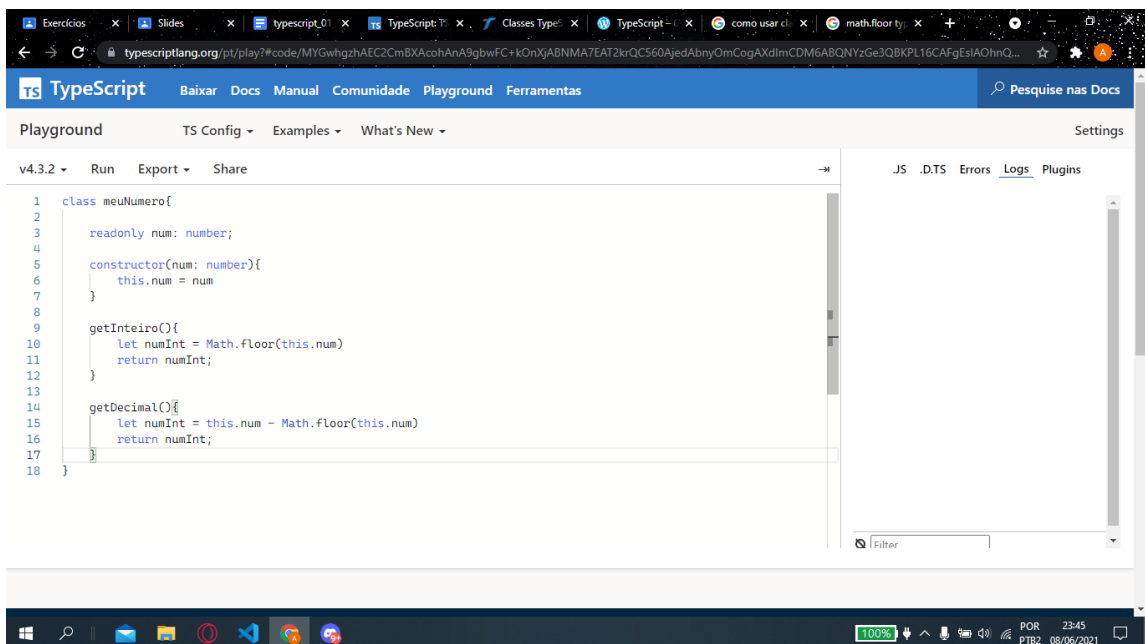
The screenshot shows the TypeScript Playground interface. The code editor contains the following TypeScript code:

```
1 class meuNumero{
2
3   readonly num: number;
4
5   constructor(num: number){
6     this.num = num
7   }
8
9   getInteiro(){
10    let numInt = Math.floor(this.num)
11    return numInt;
12  }
13
14
15
16 }
```

The right sidebar shows the 'Logs' tab, which is currently empty. The bottom status bar indicates the version is v4.3.2 and the file is named 'meuNumero.ts'.

c) Um método chamado `getDecimal` que retorne a parte decimal do atributo `numero`.

Dica: utilize a função `Math.floor(n)`

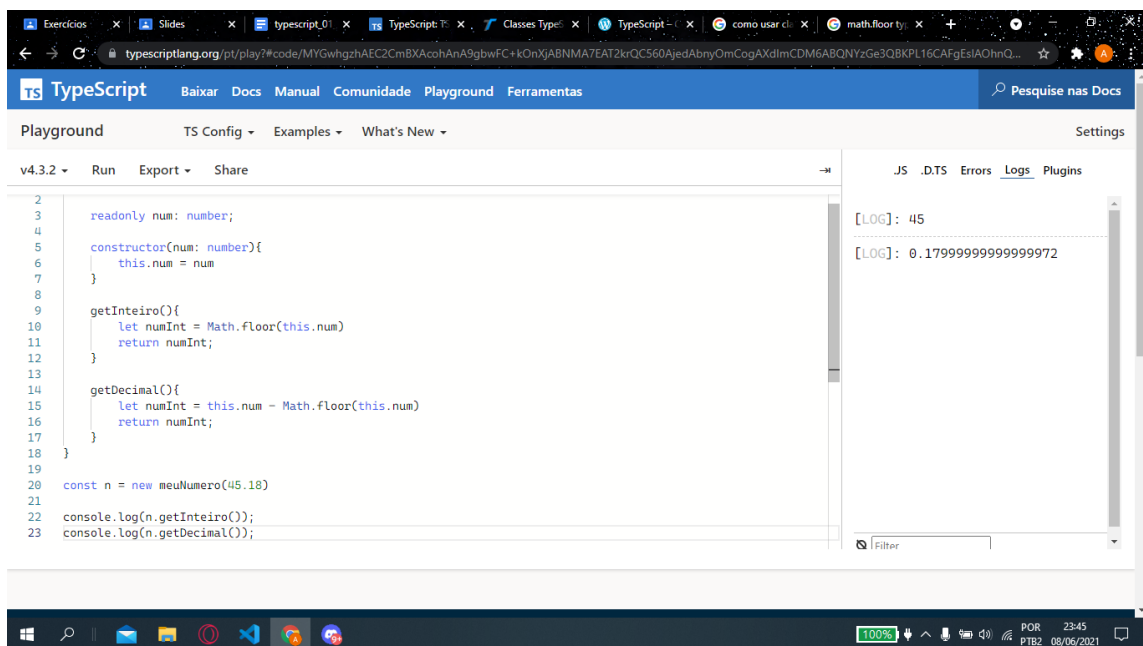


The screenshot shows the TypeScript Playground interface with the `getDecimal` method added to the `meuNumero` class. The code editor contains the following TypeScript code:

```
1 class meuNumero{
2
3   readonly num: number;
4
5   constructor(num: number){
6     this.num = num
7   }
8
9   getInteiro(){
10    let numInt = Math.floor(this.num)
11    return numInt;
12  }
13
14   getDecimal(){
15    let numInt = this.num - Math.floor(this.num)
16    return numInt;
17  }
18 }
```

The right sidebar shows the 'Logs' tab, which is currently empty. The bottom status bar indicates the version is v4.3.2 and the file is named 'meuNumero.ts'.

d) Instancie uma classe MeuNumero e teste os métodos da classe.



The screenshot shows the TypeScript Playground interface. The code editor on the left contains the following TypeScript code:

```
2  
3   readonly num: number;  
4  
5   constructor(num: number){  
6       this.num = num  
7   }  
8  
9   getInteiro(){  
10      let numInt = Math.floor(this.num)  
11      return numInt;  
12  }  
13  
14  getDecimal(){  
15      let numInt = this.num - Math.floor(this.num)  
16      return numInt;  
17  }  
18  }  
19  
20  const n = new meuNumero(45.18)  
21  
22  console.log(n.getInteiro());  
23  console.log(n.getDecimal());
```

The right-hand pane shows the execution results under the 'Logs' tab:

```
[LOG]: 45  
[LOG]: 0.17999999999999972
```

The bottom status bar indicates the file is named 'POR.PTB2' and the date is '08/06/2021'.

9) Crie uma classe chamada Transacao que tenha:

- Um atributo chamado valor e um outro chamado desconto, ambos somente leitura;
- Um método que calcule e retorne o valor do desconto aplicado ao valor original: $\text{valor} * (1 - \text{desconto}/100)$.
- Crie métodos de acesso get para ambos os atributos.
- Instancie uma classe Transacao e teste seus métodos