Usando funções



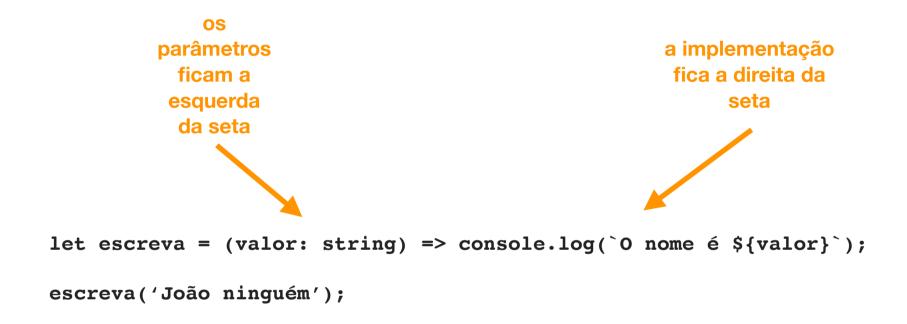
```
parâmetro
                                                        retorno
                               com tipo
                                                       com tipo
let temMaisTitulos = function(titulos : number) : boolean{
  return titulos < 35;
let numero = 8;
console.log(`Ter ${numero} é suficiente
                                               para passar
{temMaisTitulos(8) ? 'SIM': 'NÃO'}`);
            expressão
                                          template
             ternária
                                           string
```

O que acontece caso seja passado uma string?

Usando funções



Arrow function



O que acontece caso seja passado um número

Funções



Escreva uma função que faça uso de parâmetros com valor padrão





Define atributos e comportamentos



Sintaxe

```
class Laptop {
  tela: number;
  constructor (tela: number) {
    this.number = number;
  ligarMonitor() {
    console.log('O monitor do laptop foi ligado!');
let computador = new Laptop(14);
computador.ligarMonitor();
```



Herança

```
class Lenovo extends Laptop {
  constructor () {
    super(21);
  aumentarBrilho(valor: number) {
    console.log(`Brilho subiu ${valor} pontos`);
let computador = new Lenovo();
computador.ligarMonitor();
computador.aumentarBrilho(3);
```



Interfaces

```
class Lenovo extends Laptop {
  constructor () {
    super(21);
  aumentarBrilho(valor: number) {
    console.log(`Brilho subiu ${valor} pontos`);
let computador = new Lenovo();
computador.ligarMonitor();
computador.aumentarBrilho(3);
```



Interfaces

Uma interfaces define um contrato que toda classe que a implemente é obrigada a seguir, a cumprir.

```
interface Gamer {
 memoriaVideo: number;
class Lenovo extends Laptop implements Gamer {
 memoriaVideo: number = 512;
  constructor () {
    super(21);
  aumentarBrilho(valor: number) {
    console.log(`Brilho subiu ${valor} pontos`);
```