

# TypeScript

## Sys4soft - João Ribeiro

[https://www.youtube.com/watch?v=DF9jY0ITt3Q&list=PLXik\\_5Br-zO9SEz-3tuy1UlcU6X0GZo4i](https://www.youtube.com/watch?v=DF9jY0ITt3Q&list=PLXik_5Br-zO9SEz-3tuy1UlcU6X0GZo4i)

Resumo do curso feito por Roberto Pinheiro

## Aula 02 - Instalação e fluxo de utilização do TypeScript

JavaScript



TypeScript



- Instale o Node JS.
- Posteriormente instale o TypeScript

**npm install -g typescript**

- entre com o comando:

**tsc**

```
C:\sites\vuejs-sys4soft
> tsc
Version 3.7.4
Syntax: tsc [options] [file...]

Examples: tsc hello.ts
          tsc --outFile file.js file.ts
          tsc @args.txt
          tsc --build tsconfig.json

Options:
  -h, --help                Print this message.
  -w, --watch               Watch input files.
  --pretty                 Stylize errors and messages using color and context (experimental)
  --all                    Show all compiler options.
  -v, --version             Print the compiler's version.
  --init                   Initializes a TypeScript project and creates a tsconfig.json file.
  -p FILE OR DIRECTORY, --project FILE OR DIRECTORY
                           Compile the project given the path to its configuration file, or t
                           o a folder with a 'tsconfig.json'.
  -b, --build              Build one or more projects and their dependencies, if out of date
  -t VERSION, --target VERSION
                           Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015
                           ', 'ES2016', 'ES2017', 'ES2018', 'ES2019' or 'ESNEXT'.
  -m KIND, --module KIND   Specify module code generation: 'none', 'commonjs', 'amd', 'system
                           ', 'umd', 'es2015', or 'ESNext'.
  --lib                    Specify library files to be included in the compilation.
                           'es5' 'es6' 'es2015' 'es7' 'es2016' 'es2017' 'es2018' 'es2019' '
                           es2020' 'esnext' 'dom' 'dom.iterable' 'webworker' 'webworker.importscripts' 'scripthost' 'es2015.core' 'es2015.collect
                           ion' 'es2015.generator' 'es2015.iterable' 'es2015.promise' 'es2015.proxy' 'es2015.reflect' 'es2015.symbol' 'es2015.symbo
                           l.wellknown' 'es2016.array.include' 'es2017.object' 'es2017.sharedmemory' 'es2017.string' 'es2017.intl' 'es2017.type
                           darrays' 'es2018.asyncgenerator' 'es2018.asynciterable' 'es2018.intl' 'es2018.promise' 'es2018.regexp' 'es2019.array'
                           'es2019.object' 'es2019.string' 'es2019.symbol' 'es2020.string' 'es2020.symbol.wellknown' 'esnext.array' 'esnext.symbo
                           l' 'esnext.asynciterable' 'esnext.intl' 'esnext.bigint'
  --allowJs                Allow javascript files to be compiled.
```

# Aula 04 - Typescript Basic Types - Parte 1

## aula-04\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Typescript Basic Types - 1</title>
</head>
<body>
  <p id="info"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-04\code.ts

```
function executar(){

  // string
  let valor1: string = "João";

  // boolean
  let valor2: boolean = true;

  // number
  let valor3: number = 23;
  let valor4: number = 16.5;
  let valor5: number = 0xA;
  let valor6: number = 0b1100100;
  let valor7: number = 0o144;

  // concatenando
  let primeiro_nome: string = "João";
  let nome_completo = `${primeiro_nome} Ribeiro`;

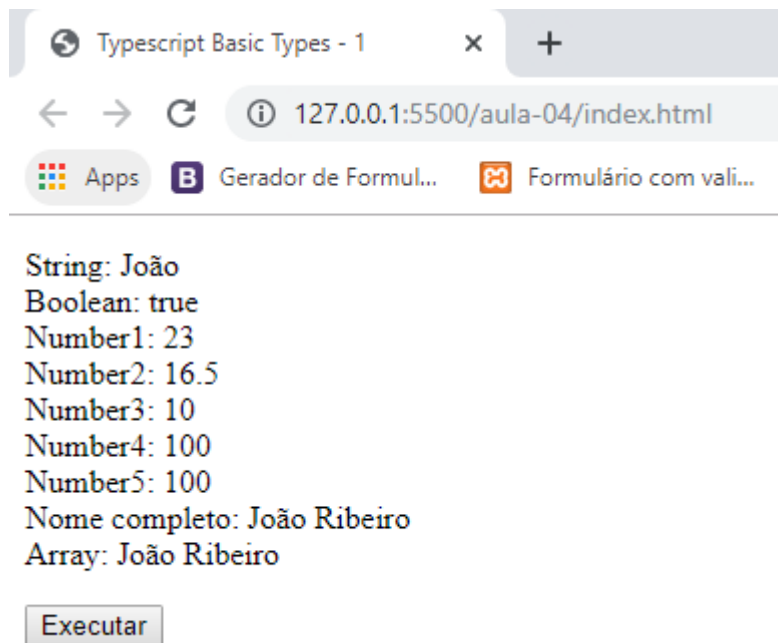
  // arrayx
  let nomes: string[] = ['João', 'Ana', 'Carlos'];
  let sobrenomes: Array<string> = ['Silva', 'Ribeiro', 'Oliveira'];

  let valor = "String: " + valor1 + "<br>Boolean: " + valor2 + "<br>Number1: " + valor3
  + "<br>Number2: " + valor4 + "<br>Number3: " + valor5 + "<br>Number4: " + valor6
  + "<br>Number5: " + valor7 + "<br>Nome completo: " + nome_completo
  + "<br>Array: " + `${nomes[0]} ${sobrenomes[1]}`;
  document.getElementById("info").innerHTML += valor;
}
```

tsc code

## aula-04\code.js

```
function executar() {  
  // string  
  var valor1 = "João";  
  // boolean  
  var valor2 = true;  
  // number  
  var valor3 = 23;  
  var valor4 = 16.5;  
  var valor5 = 0xA;  
  var valor6 = 100;  
  var valor7 = 100;  
  // concatenando  
  var primeiro_nome = "João";  
  var nome_completo = primeiro_nome + " Ribeiro";  
  // arrayx  
  var nomes = ['João', 'Ana', 'Carlos'];  
  var sobrenomes = ['Silva', 'Ribeiro', 'Oliveira'];  
  var valor = "String: " + valor1 + "<br>Boolean: " + valor2 + "<br>Number1: " + valor3  
    + "<br>Number2: " + valor4 + "<br>Number3: " + valor5 + "<br>Number4: " + valor6  
    + "<br>Number5: " + valor7 + "<br>Nome completo: " + nome_completo  
    + "<br>Array: " + (nomes[0] + " " + sobrenomes[1]);  
  document.getElementById("info").innerHTML += valor;  
}
```



## Aula 05 - Typescript Basic Types - Parte 2

### aula-05\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Typescript Basic Types - 2</title>
</head>
<body>

  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <p id="info5"></p>
  <p id="info6"></p>
  <p id="info7"></p>
  <p id="info8"></p>

  <button type="button" onclick="executar()">Executar</button>

</body>
</html>
```

### aula-05\code.ts

```
function executar(){

  // tuple
  let dados: [string, number] = ['João', 43];

  // enum
  enum rgb {red=10, breen, blue};
  let cor: rgb = rgb.blue;
  let cor_nome: string = rgb[10];

  enum pontos {derrota=0, empate=1, vitoria=3};
  let resultado: pontos = pontos.vitoria;

  enum genero {masculino, feminino};
  let eu: genero = genero.masculino;

  // any
  let valor: any = 'João';
  valor = 43;

  let valor2: any[] = ['João', 43, false];
  valor2[0] = 2017;
```

```

let nome = "João";
let sobrenome = "Ribeiro";

document.getElementById("info1").innerHTML = `${dados[0]} tem ${dados[1]} anos de idade.`;
document.getElementById("info2").innerHTML = cor_nome;
document.getElementById("info3").innerHTML = cor.toString();
document.getElementById("info4").innerHTML = resultado.toString();
document.getElementById("info5").innerHTML = eu.toString();
document.getElementById("info6").innerHTML = valor;
document.getElementById("info6").innerHTML = valor;
document.getElementById("info7").innerHTML = valor2[0].toString();
document.getElementById("info8").innerHTML = nome + " " + sobrenome;
}

```

tsc code

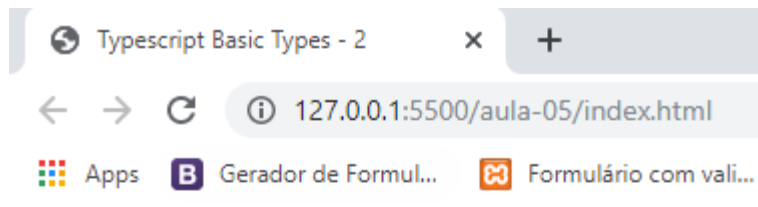
## aula-05\code.js

```

function executar() {
  // tuple
  var dados = ['João', 43];
  // enum
  var rgb;
  (function (rgb) {
    rgb[rgb["red"] = 10] = "red";
    rgb[rgb["green"] = 11] = "green";
    rgb[rgb["blue"] = 12] = "blue";
  })(rgb || (rgb = {}));
  ;
  var cor = rgb.blue;
  var cor_nome = rgb[10];
  var pontos;
  (function (pontos) {
    pontos[pontos["derrota"] = 0] = "derrota";
    pontos[pontos["empate"] = 1] = "empate";
    pontos[pontos["vitoria"] = 3] = "vitoria";
  })(pontos || (pontos = {}));
  ;
  var resultado = pontos.vitoria;
  var genero;
  (function (genero) {
    genero[genero["masculino"] = 0] = "masculino";
    genero[genero["feminino"] = 1] = "feminino";
  })(genero || (genero = {}));
  ;
  var eu = genero.masculino;
  // any
  var valor = 'João';
  valor = 43;
  var valor2 = ['João', 43, false];
  valor2[0] = 2017;
  var nome = "João";
  var sobrenome = "Ribeiro";
  document.getElementById("info1").innerHTML = dados[0] + " tem " + dados[1] + " anos de idade.";
}

```

```
document.getElementById("info2").innerHTML = cor_nome;  
document.getElementById("info3").innerHTML = cor.toString();  
document.getElementById("info4").innerHTML = resultado.toString();  
document.getElementById("info5").innerHTML = eu.toString();  
document.getElementById("info6").innerHTML = valor;  
document.getElementById("info6").innerHTML = valor;  
document.getElementById("info7").innerHTML = valor2[0].toString();  
document.getElementById("info8").innerHTML = nome + " " + sobrenome;  
}
```



João tem 43 anos de idade.

red

12

3

0

43

2017

João Ribeiro

Executar

# Aula 06 - TypeScript destructuring

## aula-06\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>TypeScript destructuring</title>
</head>
<body>
  <p id="info"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-06\code.ts

```
function executar(): void{

  // destructuring
  let valores = ['João', 'Ribeiro'];
  let [nome, sobrenome] = valores;
  document.getElementById("info").innerHTML = teste(['João', 'Ribeiro']);
}

function teste([nome, sobrenome]: [string, string]): string{
  return `${nome} ${sobrenome}`;
}
```

tsc code

## aula-06\code.js

```
function executar() {
  // destructuring
  var valores = ['João', 'Ribeiro'];
  var nome = valores[0], sobrenome = valores[1];
  document.getElementById("info").innerHTML = teste(['João', 'Ribeiro']);
}

function teste(_a) {
  var nome = _a[0], sobrenome = _a[1];
  return nome + " " + sobrenome;
}
```

João Ribeiro

Executar



# Aula 07 - TypeScript spread

## aula-07\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>TypeScript spread</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-07\code.ts

```
function executar(): void{

  // spread
  let nomes_1 = ['João', 'Carlos'];
  let nomes_2 = ['Ana', 'Cristina'];
  let nomes_totais = ['Ronaldo', ...nomes_1, 'Antonio', ...nomes_2];
  document.getElementById("info1").innerHTML = JSON.stringify(nomes_totais);

  let pessoa = {
    nome: "João",
    sobrenome: "Ribeiro",
    idade: 43
  };

  let nova_pessoa = {
    ...pessoa,
    nome: "Joaquim"
  };

  document.getElementById("info2").innerHTML = nova_pessoa.nome;
}
```

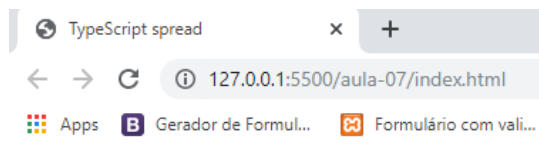
tsc code

## aula-07\code.js

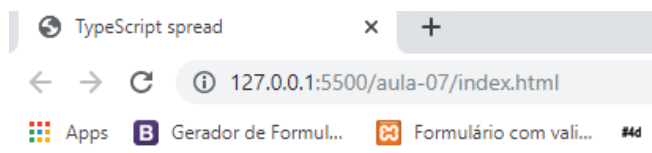
```
var __assign = (this && this.__assign) || function () {
  __assign = Object.assign || function(t) {
    for (var s, i = 1, n = arguments.length; i < n; i++) {
      s = arguments[i];
      for (var p in s) if (Object.prototype.hasOwnProperty.call(s, p))
        t[p] = s[p];
    }
    return t;
  };
  return __assign.apply(this, arguments);
};

var __spreadArrays = (this && this.__spreadArrays) || function () {
  for (var s = 0, i = 0, il = arguments.length; i < il; i++) s += arguments[i].length;
  for (var r = Array(s), k = 0, i = 0; i < il; i++)
    for (var a = arguments[i], j = 0, jl = a.length; j < jl; j++, k++)
      r[k] = a[j];
  return r;
};

function executar() {
  // spread
  var nomes_1 = ['João', 'Carlos'];
  var nomes_2 = ['Ana', 'Cristina'];
  var nomes_totais = __spreadArrays(['Ronaldo'], nomes_1, ['Antonio'], nomes_2);
  document.getElementById("info1").innerHTML = JSON.stringify(nomes_totais);
  var pessoa = {
    nome: "João",
    sobrenome: "Ribeiro",
    idade: 43
  };
  var nova_pessoa = __assign(__assign({}, pessoa), { nome: "Joaquim" });
  document.getElementById("info2").innerHTML = nova_pessoa.nome;
}
```



Executar



["Ronaldo","João","Carlos","Antonio","Ana","Cristina"]

Joaquim

Executar

## Aula 08 - Interfaces

### aula-08\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Interfaces</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

### aula-08\code.ts

```
function executar(): void{

  // interfaces
  interface intPessoa {
    nome: string;
    sobrenome?: string;
  }

  interface intCoordenadas {
    x: number;
    readonly y: number;
  }

  function pessoa(dados: intPessoa){
    let final = "";
    if(dados.nome){
      final = dados.nome;
    }
    if(dados.sobrenome){
      final += " " + dados.sobrenome;
    }
    return final;
  }

  let d = {nome:'João'};
  let nome_completo = pessoa(d);

  let posicao: intCoordenadas = {x: 100, y: 200};
```

```

posicao.x = 500;

document.getElementById('info1').innerHTML = nome_completo;
document.getElementById('info2').innerHTML = "Posição x = " + posicao.x + ' - Posição y = ' +
posicao.y;

}

```

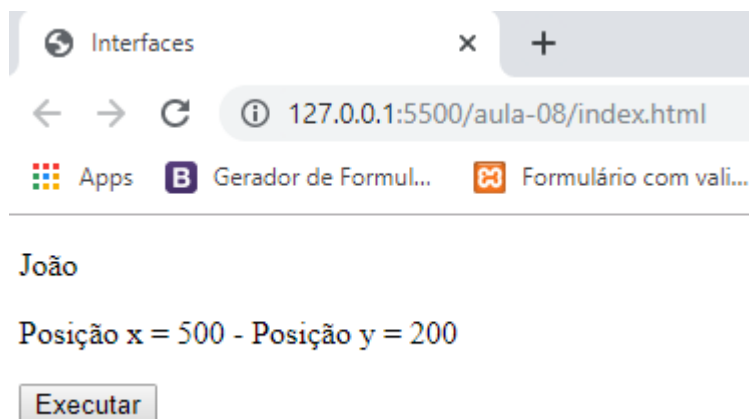
tsc code

## aula-08\code.js

```

function executar() {
  function pessoa(dados) {
    var final = "";
    if (dados.nome) {
      final = dados.nome;
    }
    if (dados.sobrenome) {
      final += " " + dados.sobrenome;
    }
    return final;
  }
  var d = { nome: 'João' };
  var nome_completo = pessoa(d);
  var posicao = { x: 100, y: 200 };
  posicao.x = 500;
  document.getElementById('info1').innerHTML = nome_completo;
  document.getElementById('info2').innerHTML = "Posição x = " + posicao.x + ' - Posição y = ' +
posicao.y;
}

```



## Aula 09 - Classes

### aula-09\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Interfaces</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

### aula-09\code.ts

```
function executar(): void{

  // interfaces
  interface intPessoa {
    nome: string;
    sobrenome?: string;
  }

  interface intCoordenadas {
    x: number;
    readonly y: number;
  }

  function pessoa(dados: intPessoa){
    let final = "";
    if(dados.nome){
      final = dados.nome;
    }
    if(dados.sobrenome){
      final += " " + dados.sobrenome;
    }
    return final;
  }

  let d = {nome:'João'};
  let nome_completo = pessoa(d);

  let posicao: intCoordenadas = {x: 100, y: 200};
```

```

posicao.x = 500;

document.getElementById('info1').innerHTML = nome_completo;
document.getElementById('info2').innerHTML = "Posição x = " + posicao.x + ' - Posição y = ' +
posicao.y;

}

```

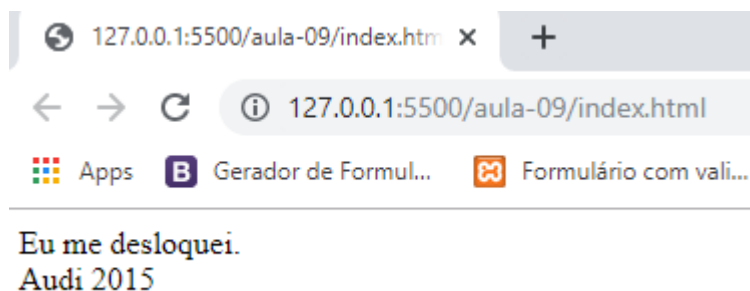
**tsc -w code.ts**

## **aula-09\code.js**

```

function executar() {
  // classe
  var veiculo = /** @class */ (function () {
    function veiculo(m, a) {
      this.marca = m;
      this.ano = a;
    }
    veiculo.prototype.mover = function () {
      return "Eu me desloquei.";
    };
    return veiculo;
  }());
  var automovel = new veiculo('Audi', 2015);
  // automovel.ano = 2010;
  // automovel.marca = "BMW";
  document.write(automovel.mover());
  document.write("<br>");
  document.write(automovel.marca + " " + automovel.ano);
}

```



# Aula 10 - Classes - Hereditariedade

## aula-10\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Classes - Hereditariedade</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-10\code.ts

```
function executar(){

  // classe - hereditariedade
  class animal{
    designacao: string;

    constructor(designacao: string){
      this.designacao = designacao;
    }

    mover(distancia: number): string{
      return `deslocou-se por ${distancia} metros`;
    }
  }

  class cavalo extends animal{

    mover(distancia: number): string{
      return `${this.designacao} a cavalgar...` + super.mover(distancia);
    }
  }

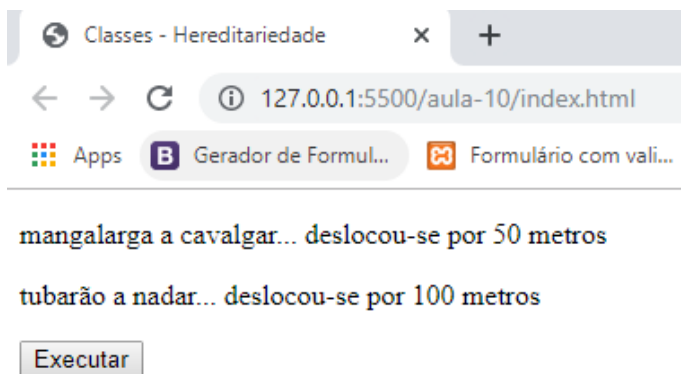
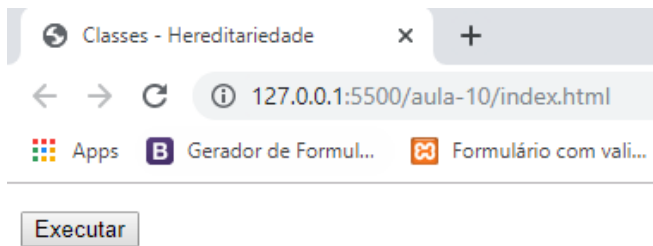
  class peixe extends animal{

    mover(distancia: number): string{
      return `${this.designacao} a nadar...` + super.mover(distancia);
    }
  }

  let c = new cavalo('mangalarga');
  let p = new peixe('tubarão');
```

```
document.getElementById("info1").innerHTML = c.mover(50);  
document.getElementById("info2").innerHTML = p.mover(100);  
}
```

tsc -w code.ts





# Aula 11 - Classes - Modificadores

## aula-11\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Classes - Modificadores</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-11\code.ts

```
function executar(){

  // classe - modifiers
  class humano{
    private nome: string;
    protected genero: string;

    constructor(n: string){
      this.nome = n;
    }

    setNome(n:string){
      this.nome = n;
    }

    getNome(){
      return this.nome;
    }

    private falar(texto: string){
      return `Eu, ` + this.nome + `, disse ` + texto;
    }

    public gritar(){
      return this.falar('OLÁ');
    }
  }
}
```

```

class homem extends humano{

    setGenero(g:string){
        this.genero = g;
    }

    getGenero(){
        return this.genero;
    }

}

let pessoa = new humano('José');
pessoa.setNome('João');
let pn = pessoa.getNome();

let h = new homem('João');
h.setGenero('masculino');

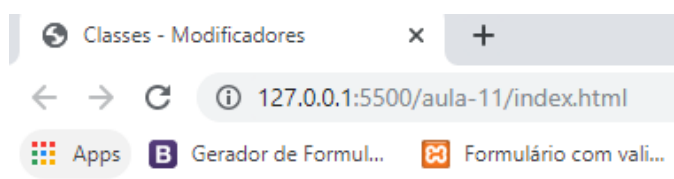
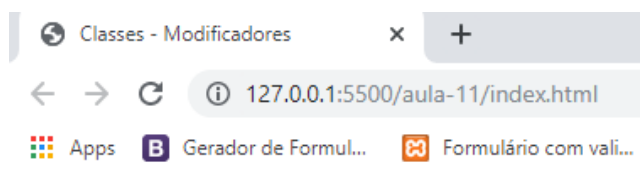
let g = h.getGenero();

document.getElementById("info1").innerHTML = pessoa.gritar();
document.getElementById("info2").innerHTML = "Genero: " + g;

}

```

**tsc -w code.ts**



Eu, João, disse OLÁ

Genero: masculino

Executar

## Aula 12 - Readonly e accessors

### aula-12\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Classes - Readonly e accessors</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

### aula-12\code.ts

```
function executar(){

  // classe - readonly e accessors

  class homem {

    readonly nome: string;
    readonly idade: number;

    constructor(n: string, i: number){
      this.nome = n;
      this.idade = i;
    }

  }

  class mulher {
    private _nome: string;
    private _sobrenome: string;

    // setter

    set Nome(n: string){
      if(n !== 'x'){
        this._nome = n;
      } else {
        this._nome = "Nome indefinido";
      }
    }

    // getter
```

```

    get Nome(){
        return this._nome;
    }

}

let h = new homem('João', 43);
// h.nome = 'João';

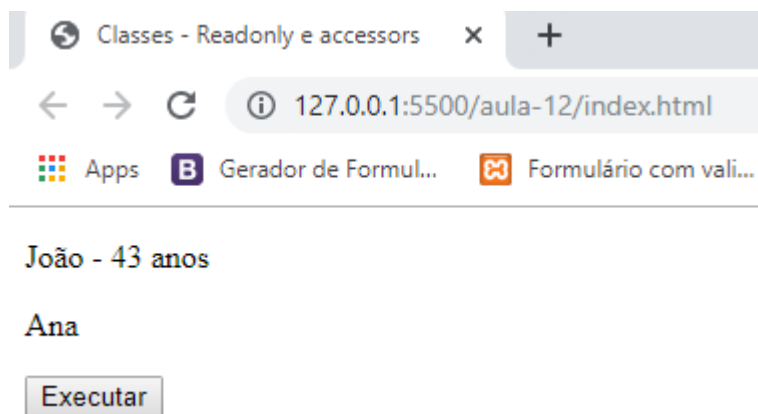
let m = new mulher();
m.Nome = 'x';
// m.Nome = 'Ana';

document.getElementById("info1").innerHTML = h.nome + " - " + h.idade + " anos";
document.getElementById("info2").innerHTML = m.Nome;

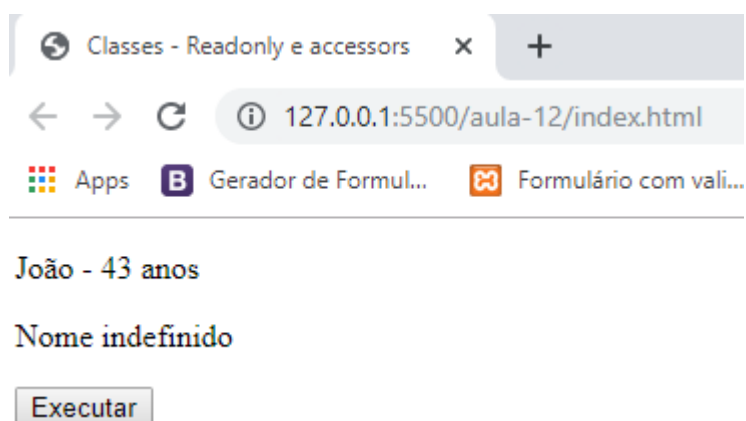
}

```

**tsc -w code.ts**



- Se nome de mulher é 'x':



# Aula 13 - Propriedade static e classes abstratas

## aula-13\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Propriedade static e classes abstratas</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-13\code.ts

```
function executar(){

  // classe - static properties

  class familia_kennedy {

    private _nome: string;
    static _sobrenome: string = "Kennedy";

    constructor(n: string){
      this._nome = n + ' ' + familia_kennedy._sobrenome;
    }

    get Nome(){
      return this._nome;
    }

  }

  // abstract classes

  abstract class humano{
    nome: string;
    idade: number;

    abstract falar(t: string): string;
  }

  class homem extends humano{
    falar(t: string){
```

```

        return "Eu falei " + t + " e sou homem.";
    }
}

class mulher extends humano{
    falar(t: string){
        return "Eu falei " + t + " e sou mulher.";
    }
}

let familiar1 = new familia_kennedy('John F. ');
let familiar2 = new familia_kennedy('Jacqueline F. ');

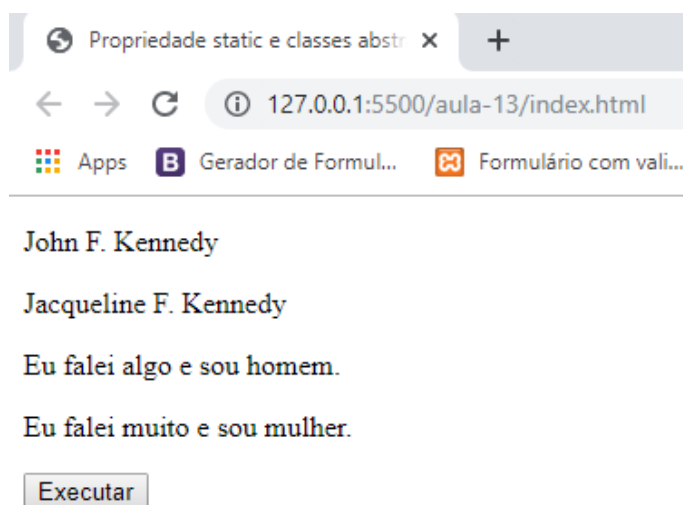
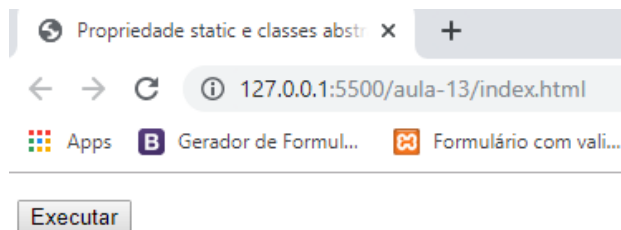
let h = new homem();
let m = new mulher();

document.getElementById("info1").innerHTML = familiar1.Nome;
document.getElementById("info2").innerHTML = familiar2.Nome;
document.getElementById("info3").innerHTML = h.falar('algo');
document.getElementById("info4").innerHTML = m.falar('muito');

}

```

**tsc -w code.ts**



# Aula 14 - Functions

## aula-14\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Functions</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-14\code.ts

```
function executar(){

  // funções (functions)

  function multiplicar(a: number, b: number){
    return a * b;
  }

  // funções anônimas

  let f = function(c: number, d: number){
    return c + d;
  }

  function nomeCompleto(nome: string, sobrenome?: string): string {
    let n = "";

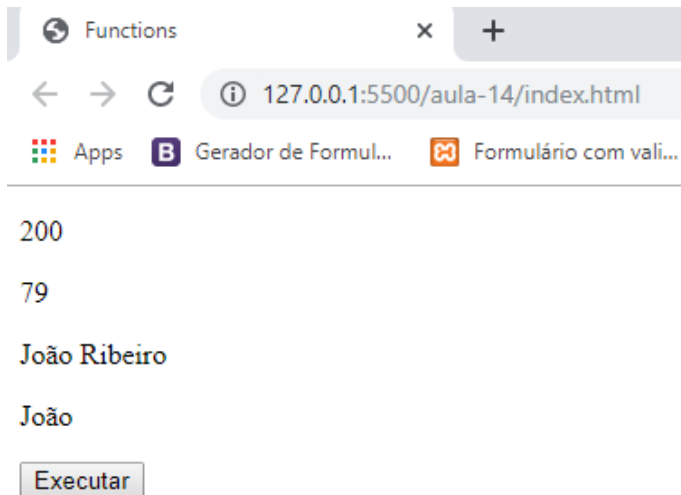
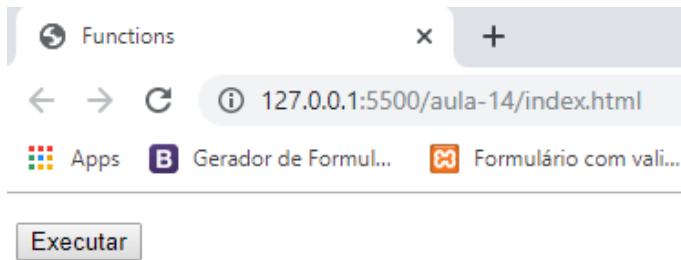
    if(sobrenome){
      n = nome + ' ' + sobrenome;
    } else {
      n = nome;
    }

    return n;
  }

  let resultado = multiplicar(10, 20);
  let soma = f(13, 66);
```

```
document.getElementById('info1').innerHTML = resultado.toString();
document.getElementById('info2').innerHTML = soma.toString();
document.getElementById('info3').innerHTML = nomeCompleto('João', 'Ribeiro');
document.getElementById('info4').innerHTML = nomeCompleto('João');
}
```

**tsc -w code.ts**





# Aula 15 - Functions (Rest parameters e Overloads)

## aula-15\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Functions</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-15\code.ts

```
function executar(){

  // funções (functions)

  // rest parameters

  function listaNomes(nome: string, ...restantes: string[]): string{
    return nome + ", " + restantes.join(", ");
  }

  let nomes = listaNomes('João', 'Carlos', 'Ana', 'Antonio');

  document.getElementById('info1').innerHTML = nomes;

  // overload

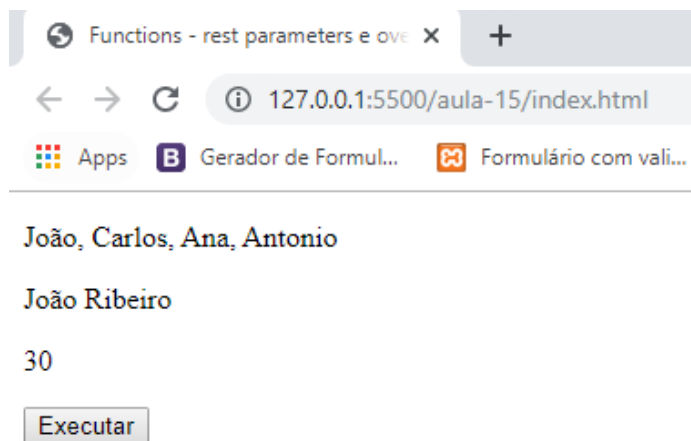
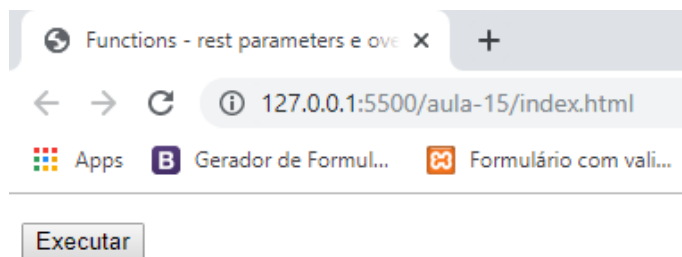
  function juntar(x: any, y:any): any {
    return x + y;
  }

  let nome = juntar('João ', 'Ribeiro');
  let resultado = juntar(10, 20);

  document.getElementById('info2').innerHTML = nome;
  document.getElementById('info3').innerHTML = resultado;

}
```

**tsc -w code.ts**



# Aula 16 - Generics

## aula-16\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Generics</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-16\code.ts

```
function executar(){

  // function apresentar(valor: any): any{
  //   return valor;
  // }

  // document.getElementById('info1').innerHTML = apresentar(5);

  // generics

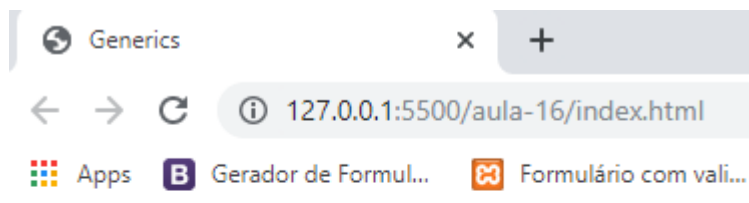
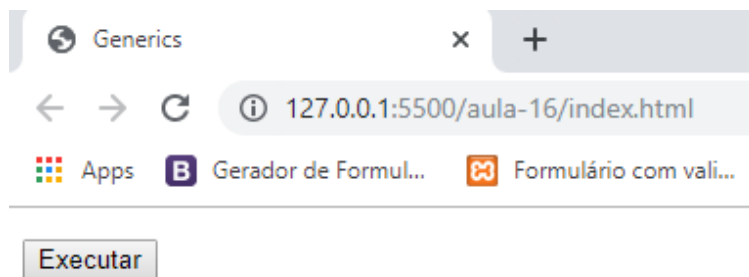
  function apresentar<T>(valor: T): T{
    return valor;
  }

  let teste1 = apresentar<string>('João');
  let teste2 = apresentar<number>(15);

  document.getElementById('info1').innerHTML = teste1;
  document.getElementById('info2').innerHTML = teste2.toString();

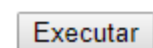
}
```

tsc -w code.ts



João

15



# Aula 17 - Enums

## aula-17\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Enums</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-17\code.ts

```
function executar(){

  // enums
  enum direcoes{
    cima, baixo, esquerda, direita
  }

  let sentido1 = direcoes.cima; // 0
  let sentido2 = direcoes[2]; // esquerda

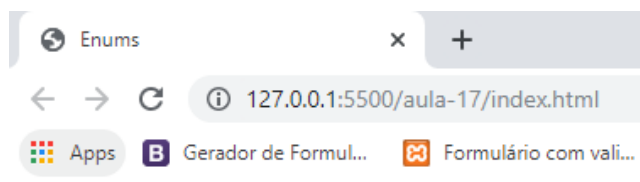
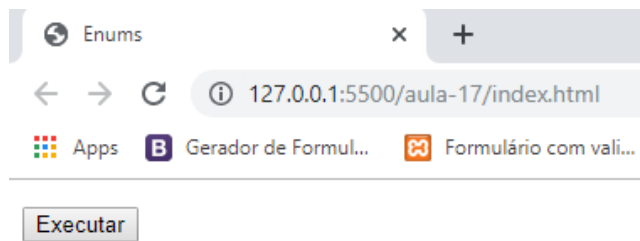
  function mover(sentido: direcoes){
    if(sentido == direcoes.cima){
      return "Direção para cima";
    }
  }

  enum partidos{
    pt=13, psl=17, pmdb=25, psdb=45
  }

  let eleicoes = partidos.psdb;
  document.getElementById('info1').innerHTML = sentido1.toString();
  document.getElementById('info2').innerHTML = sentido2;
  document.getElementById('info3').innerHTML = mover(0);
  document.getElementById('info4').innerHTML = eleicoes.toString();

}
```

tsc -w code.ts



0

esquerda

Direção para cima

45

Executar

# Aula 18 - Tipos e compatibilidade de tipos

## aula-18\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Tipos e compatibilidade de tipos</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <p id="info5"></p>
  <p id="info6"></p>
  <p id="info7"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-18\code.ts

```
function executar(){

  // tipos e compatibilidade de tipos
  let nome = 'João';
  let sobrenome = "Ribeiro";
  let numero = 100;
  let status = true;

  enum direcoes {
    cima, baixo, esquerda, direita
  }

  let sentido = direcoes.cima;
  let x = direcoes[3];
  let y = [1, "João", 3];

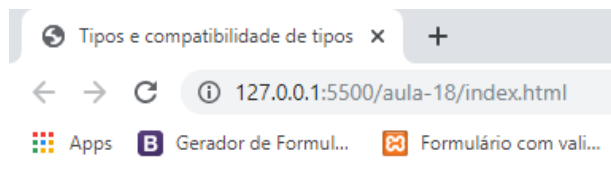
  document.getElementById('info1').innerHTML = typeof(nome);
  document.getElementById('info2').innerHTML = typeof(sobrenome);
  document.getElementById('info3').innerHTML = typeof(numero);
  document.getElementById('info4').innerHTML = typeof(status);
  document.getElementById('info5').innerHTML = typeof(sentido);
  document.getElementById('info6').innerHTML = typeof(x);
  document.getElementById('info7').innerHTML = typeof(y);

}
```

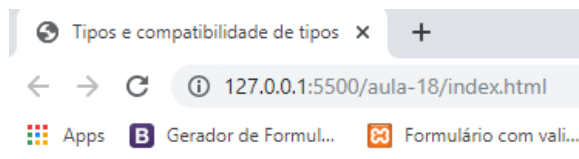
tsc -w code.ts

## aula-18\code.js

```
function executar() {  
  // tipos e compatibilidade de tipos  
  var nome = 'João';  
  var sobrenome = "Ribeiro";  
  var numero = 100;  
  var status = true;  
  var direcoes;  
  (function (direcoes) {  
    direcoes[direcoes["cima"] = 0] = "cima";  
    direcoes[direcoes["baixo"] = 1] = "baixo";  
    direcoes[direcoes["esquerda"] = 2] = "esquerda";  
    direcoes[direcoes["direita"] = 3] = "direita";  
  })(direcoes || (direcoes = {}));  
  var sentido = direcoes.cima;  
  var x = direcoes[3];  
  var y = [1, "João", 3];  
  document.getElementById('info1').innerHTML = typeof (nome);  
  document.getElementById('info2').innerHTML = typeof (sobrenome);  
  document.getElementById('info3').innerHTML = typeof (numero);  
  document.getElementById('info4').innerHTML = typeof (status);  
  document.getElementById('info5').innerHTML = typeof (sentido);  
  document.getElementById('info6').innerHTML = typeof (x);  
  document.getElementById('info7').innerHTML = typeof (y);  
}
```



Executar



string

string

number

boolean

number

string

object

Executar



# Aula 19 - Iterators for of e for in

## aula-19\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Iterators for of e for in</title>
</head>
<body>
  <p id="info1"></p>
  <p id="info2"></p>
  <p id="info3"></p>
  <p id="info4"></p>
  <p id="info5"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-19\code.ts

```
function executar(){

  // iterators

  let nomes1 = ['João', 'Ana', 'Carlos', 'Daniela', 'Sofia'];
  let elemento1 = document.getElementById("info1");
  elemento1.innerHTML = "";

  // for
  for(let i = 0; i < nomes1.length; i++){
    elemento1.innerHTML += `${nomes1[i]} <br />`;
  }

  // =====

  let nomes2 = ['Francisco', 'Maria', 'Sérgio', 'Miriam', 'Aline'];
  let elemento2 = document.getElementById("info2");
  elemento2.innerHTML = "";

  // foreach(array) - 1ª versão
  nomes2.forEach(function(e){
    elemento2.innerHTML += `${e} <br />`;
  })

  // =====

  let nomes3 = ['Paulo', 'Joana', 'Pedro', 'Nair', 'Rita'];
  let elemento3 = document.getElementById("info3");
```

```

elemento3.innerHTML = "";

// for arrow function

nomes3.forEach(e => {
  elemento3.innerHTML += `${e} <br />`;
})

// =====

let nomes4 = ['Antonio', 'Beatriz', 'Roberto', 'Cecília', 'Deise'];
let elemento4 = document.getElementById("info4");
elemento4.innerHTML = "";

// for ... of

for(let e of nomes4){
  elemento4.innerHTML += `${e} <br />`;
}

// =====

let nomes5 = ['Rafael', 'Priscila', 'Daniel', 'Michelle', 'Caroline'];
let elemento5 = document.getElementById("info5");
elemento5.innerHTML = "";

// for ... in

for(let e in nomes5){
  elemento5.innerHTML += `${e} <br />`;
}

// =====

}

```

tsc -w code.ts

## aula-19\code.js

```
function executar() {
  // iterators
  var nomes1 = ['João', 'Ana', 'Carlos', 'Daniela', 'Sofia'];
  var elemento1 = document.getElementById("info1");
  elemento1.innerHTML = "";
  // for
  for (var i = 0; i < nomes1.length; i++) {
    elemento1.innerHTML += nomes1[i] + " <br />";
  }
  // =====
  var nomes2 = ['Francisco', 'Maria', 'Sérgio', 'Miriam', 'Aline'];
  var elemento2 = document.getElementById("info2");
  elemento2.innerHTML = "";
  // foreach(array) - 1ª versão
  nomes2.forEach(function (e) {
    elemento2.innerHTML += e + " <br />";
  });
  // =====
  var nomes3 = ['Paulo', 'Joana', 'Pedro', 'Nair', 'Rita'];
  var elemento3 = document.getElementById("info3");
  elemento3.innerHTML = "";
  // for arrow function
  nomes3.forEach(function (e) {
    elemento3.innerHTML += e + " <br />";
  });
  // =====
  var nomes4 = ['Antonio', 'Beatriz', 'Roberto', 'Cecília', 'Deise'];
  var elemento4 = document.getElementById("info4");
  elemento4.innerHTML = "";
  // for ... of
  for (var _i = 0, nomes4_1 = nomes4; _i < nomes4_1.length; _i++) {
    var e = nomes4_1[_i];
    elemento4.innerHTML += e + " <br />";
  }
  // =====
  var nomes5 = ['Rafael', 'Priscila', 'Daniel', 'Michelle', 'Caroline'];
  var elemento5 = document.getElementById("info5");
  elemento5.innerHTML = "";
  // for ... in
  for (var e in nomes5) {
    elemento5.innerHTML += e + " <br />";
  }
  // =====
}
```

Iterators for of e for in x +

← → ↺ 127.0.0.1:5500/aula-19/index.html

Apps Gerador de Formul... Formulário com vali...

Executar

Iterators for of e for in x +

← → ↺ 127.0.0.1:5500/aula-19/index.html

Apps Gerador de Formul... Formulário com vali...

João  
Ana  
Carlos  
Daniela  
Sofia  
  
Francisco  
Maria  
Sérgio  
Miriam  
Aline  
  
Paulo  
Joana  
Pedro  
Nair  
Rita  
  
Antonio  
Beatriz  
Roberto  
Cecília  
Deise  
  
0  
1  
2  
3  
4

Executar

# Aula 20 - Modules export e import - Parte 1

## aula-20\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code.js"></script>
  <title>Modules export e import</title>
</head>
<body>
  <p id="info1"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-20\codigo.ts

```
export interface pessoa{
  nome: string,
  idade: number
}
```

## aula-20\code.ts

```
import { pessoa } from './codigo';

function criarPessoa(){
  let eu: pessoa;
}
```

## Aula 21 - Modules export e import - Parte 2

### aula-21\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code1.js"></script>
  <title>Modules export e import - Parte 2</title>
</head>
<body>
  <p id="info1"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

### aula-21\code1.ts

```
// import { pessoa, humano, enumSexo } from './code2';
import * as code from './code2';
import { animal } from './outro/codigoAnimal'
```

```
let eu: code.pessoa;
let animal: animal;
let voce = new code.humano();

voce.sexo = code.enumSexo.feminino;
```

### aula-21\code2.ts

```
interface pessoa {
  nome: string;
  idade: number;
}

enum enumSexo{
  masculino, feminino
}

class humano {
  nome: string;
  sexo: enumSexo;

  andar(){

  }

  saltar(){
```

```
    }  
    falar(){  
    }  
}  
  
export {pessoa, humano, enumSexo};
```

### **aula-21\outro\codigoAnimal.ts**

```
export interface animal{  
    especie: string;  
}
```

# Aula 22 - Namespaces

## aula-22\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src="code1.js"></script>
  <title>Modules export e import - Parte 2</title>
</head>
<body>
  <p id="info1"></p>
  <button type="button" onclick="executar()">Executar</button>
</body>
</html>
```

## aula-22\code1.ts

```
namespace terrestre {
  export interface animal {
    nome: string;
    especie: string;
  }
}
export { terrestre }
```

## aula-22\code2.ts

```
namespace maritimo {
  export interface animal {
    nome: string;
    especie: string;
  }
}
export { maritimo }
```

## aula-22\code.ts

```
import { terrestre } from './code1';
import { maritimo } from './code2';

let a: maritimo.animal;
let b: terrestre.animal;
```