**Repositório com exercícios e slides**

**https://github.com/lira1705/mentoria-typescript**

**TypeScript**

**O que é?**

* superset do Javascript
* **linguagem construída por cima do JS**
* adiciona features à linguagem original
* tem tudo o que o Javascript tem
* compila para JS

**Por que usar TS?**

* adiciona definição estática de tipos:
  + funcionalidades do código mais explícitas
  + maior segurança durante o desenvolvimento
  + detecta erros durante a compilação que só seriam percebidos em execução
* compila um Javascript mais resiliente
* ajuda a desenvolver boas práticas de programação

npm install --save-dev typescript

npm install lite-server

npm start – compilou e starta o arquivo. utilizar no terminal do VSCode

  "scripts": {

    "teste":"echo \"Error: no test specified\" && exit 1",

    "start": "lite-server",

    "watch": "tsc --watch",

    "tsc": "tsc"

Adicionar esse script tsc para conseguir rodar. **npm run tsc -- --init**

let buttonTeste = document.getElementById('button');

buttonTeste?.addEventListener('click', () => {

    console.log('funcionou')

})

A interrogação funciona quase como se tivesse um “if”. Se tiver o buttonTeste, ativará o evento. No JavaScript, o código ficará com a seguinte forma:

let buttonTeste = document.getElementById('button');

buttonTeste === null || buttonTeste === void 0 ? void 0 : buttonTeste.addEventListener('click', () => {

    console.log('funcionou');

});

let button = document.getElementById('button');

let input1 = document.getElementById('input1') as *HTMLInputElement*;

let input2 = document.getElementById('input2') as *HTMLInputElement*;

function adicionarNumero(*numero1*: *number*, *numero2*: *number*, *devePrintar*: *boolean*, *frase*: *string*) { //no typescript, é preciso especificar o que o valor é (número, string)

    let resultado = *numero1* + *numero2*;

    if (*devePrintar*){

        console.log(*frase* + resultado);

    }

    return *numero1* + *numero2*

}

let devePrintar = true;

let frase: *string*;

frase = 'O valor é: '

if (button) {

    button.addEventListener('click', () => { //se botão existir, chamamos esse evento

        if (input1 && input2){

            console.log(adicionarNumero(*Number* (input1.value),*Number* (input2.value), devePrintar, frase));

        }

    })

}

**Features:**

* node
* typescript
* recomendado usar o VSCode

**Listando objeto, enum e interface**

const pessoa = {

    nome: 'Renan',

    idade: 30,

    profissao: 'desenvolvedor'

}

pessoa.idade = 31;

const leticia: {nome: *string*, idade: *number*, profissao: *string*} = { //o "=" é para igualar o objeto

    nome: 'Letícia',

    idade: 27,

    profissao: 'medica'

}

const mayara: {nome: *string*, idade: *number*, profissao: *string*} = { //o "=" é para igualar o objeto

    nome: 'Douglas',

    idade: 30,

    profissao: 'medica'

}

enum *Profissao* {

    Professora,

    Atriz,

    Medica,

    Desenvolvedor

}

interface *Pessoa* {

    nome: *string*,

    idade: *number*,

    profissao?: *Profissao* //criamos um tipo e um "enum", que é basicamente um grupo de constantes. no caso, é Profissao

    //?  é para dizer que pode ou não receber um valor desse elemento

}

interface *Estudante* extends *Pessoa* {

    materias: *string*[] //"[ ] " significa "lista" em typescript

}

const chico: *Pessoa* = {

    nome: 'Chico',

    idade: 29,

    profissao: *Profissao*.Desenvolvedor

}

const felipe: *Pessoa* = {

    nome: 'Felipe',

    idade: 30,

    profissao: *Profissao*.Desenvolvedor

}

const guilherme: *Estudante* = {

    nome: 'Guilherme',

    idade: 26,

    profissao: *Profissao*.Desenvolvedor,

    materias: ['Matemática discreta', 'programação']

}

const douglas: *Estudante* = {

    nome: 'Douglas',

    idade: 26,

    materias: ['Matemática discreta', 'programação']

}

function listar(*lista*: *string*[]) {

    for (const item of *lista*) {

        console.log('- ', item);

    } //vai iterar essa lista para cada constante "item" de lista, ela vai dar o console.log em "- " com o item

}

listar(douglas.materias);

**Any**

* geralmente é uma **má prática** utilizar ele
* recebe **qualquer tipo de arquivo** e **pode ser qualquer tipo de objeto**, inclusive string

let valorAny: *any*; //any é um tipo que pode receber qualquer tipo. any pode ser qualquer tipo de objeto, inclusive uma string

valorAny = 3; //geralmente é uma má prática utilizar o any

valorAny = 'ola';

valorAny = true;

let valorString: *string* = 'teste'

valorString = valorAny;

let valorString2: *string* = 'testao';

valorString2 = valorAny;

function somarString(*string1*: *string*, *string2*: *string*) {

    console.log(*string1* + *string2*);

}

somarString(valorString, valorString2);

somarString('ola', ', como vai?');

**Tipos**

type *input* = *number* | *string*;

function somarValores(*input1*: *input*, *input2*: *input*) {

    if (typeof *input1* === "string" || typeof *input2* === 'string') {

        return *input1*.toString() + *input2*.toString(); //colocando o toString, o valor retornado será o mesmo, MAS se um deles for string e o outro número, retornará apenas a string

    } else {

        return *input1* + *input2*;

    }

}

console.log(somarValores(1,5));

console.log(somarValores('ola', ' , tudo bem?'));

console.log(somarValores('que dia é hoje? ', 5));

**Definindo o retorno de uma função**

function somarValoresNumericos(*numero1*: *number*, *numero2*: *number*): *number* { //especificamos que o tipo de numero 1 e 2 é número. //o number no final GARANTE que o resultado será numerico

    return *numero1* + *numero2*;

}

console.log(somarValoresNumericos(1, 3));

**Void**

function printaValoresNumericos(*numero1*: *number*, *numero2*: *number*): *void* { //console.log não retorna de fato nada para o usuário. se utiliza "void" quando não se quer que a função devolva algo

    console.log(*numero1* + *numero2*);

}

**Callback**

* call-back é uma função passada como argumento para outra função. Ele faz com que a função seja executada logo em seguida, deixando ela assíncrona.

function printaValoresNumericos(*numero1*: *number*, *numero2*: *number*): *void* {

    console.log(*numero1* + *numero2*);

}

function somarValoresNumericosETratar(*numero1*: *number*, *numero2*: *number*, callback: (*numero*: *number*) => *number*): *number* {  //callback é uma função passada como argumento para outra função. faz com que a função seja executada após uma ação específica

    let resultado = *numero1* + *numero2*; //let cria uma variável, mas de escopo de bloco

    return callback(resultado);

}

function aoQuadrado (*numero*: *number*): *number* {

    return *numero* \* *numero*;

}

console.log(somarValoresNumericosETratar(1, 3, aoQuadrado))

//vai fazer 1 + 3 = 4. 4 ao quadrado = 16

**tsconfig**

* target – versão
* outdir – dizer para onde vão os arquivos .js criados
* strictNullChecks – se mudar para false, quando aparecer algo com resultado Null, não aparecerá mensagem de erro, mas não é recomendado fazer isso