**TIPOS:**Podemos fazer a tipagem de uma variável de duas maneiras. Ou podemos tipar diretamente após o nome da declaração da variável, exemplo:

let nome: string = ‘Felipe’;

Ou podemos usar a palavra reservada type, exemplo:

type nome = string;

let nome: nome = ‘Felipe’;

Vamos listar os tipos mais comuns de dados:

* string;
* boolean (true ou false);
* number (int ou float);
* number[] ou Array<number> (array de números);
* string[] ou Array<string> (array de strings);
* tuple (maneira de tipar um array que contém diferentes tipos de dados dentro dele (exemplo, string, number e boolean) e que já sabemos a quantidade de itens dentro do array);
* null e undefined;
* object (recomendado usar type, para o código ficar mais limpo);
* enum (conjunto de chave e valor);
* any (qualquer coisa) //não aconselhável;
* void (tipar funções sem retorno);

Exemplo de um tuple:

const nome: [string, number, boolean] = [‘Felipe’, 19, true]

Exemplo de uma tipagem para um objeto:

type pessoa {

nome: string,

idade: number,

cidade: string

}

let pessoa: pessoa{

nome: ‘Felipe’,

idade: 19,

cidade: ‘Bragança Pta’

}

Exemplo de enum:

enum cores = {

white = ‘#FFF’,

black = ‘#000’

}

**INTERFACES:**

É uma outra maneira de tipar um objeto. Exemplo:

interface pessoa {

nome: string,

idade: number,

cidade: string

}

let pessoa: pessoa = {

nome: ‘Felipe’,

idade: 19,

cidade: ‘Bragança Pta’

}

**Union & Aliases:**

O union serve para indicar que um determinado dado vai ser de um tipo ou de outro tipo. Ele é representado pela barra reta |. Exemplo:

interface ocupacao {

trabalho: string,

local: string

}

interface pessoa {

nome: string,

idade: number

}

let pessoa: pessoa | ocupacao = {

nome: ‘Felipe’,

idade: 19

}

let pessoa: pessoa | ocupacao = {

trabalho: ‘Programador,

local: ‘Remoto’

}

Já o alias serve para unir dois tipos diferentes. Ele é representado pelo ‘e’ comercial &. Exemplo:

interface ocupacao {

trabalho: string,

local: string

}

interface pessoa {

nome: string,

idade: number

}

type dados = ocupacao & pessoa;

let pessoa {

nome: ‘Felipe’,

idade: 19,

trabalho: ‘Programador’,

local: ‘Remoto’

}

Nesses exemplos, devemos utilizar todas as chaves e valores (nome, idade, trabalho e local), pois se algum faltar dará erro. Para deixar alguma chave como opcional, podemos usar o ponto de interrogação no momento da tipagem. Exemplo:

interface ocupacao {

trabalho: string,

local?: string

}

Nesse caso, o local é opcional e se não for usado não haverá erros.