

Título: **Introdução aos Aplicativos TypeScript**
Professor: Eleandro Maschio
Ano: 2023
Disciplina: Fundamentos de Programação
Curso: Sistemas para Internet
Câmpus: Guarapuava
Instituição: Universidade Tecnológica Federal do Paraná
Como citar: [Referência](#)

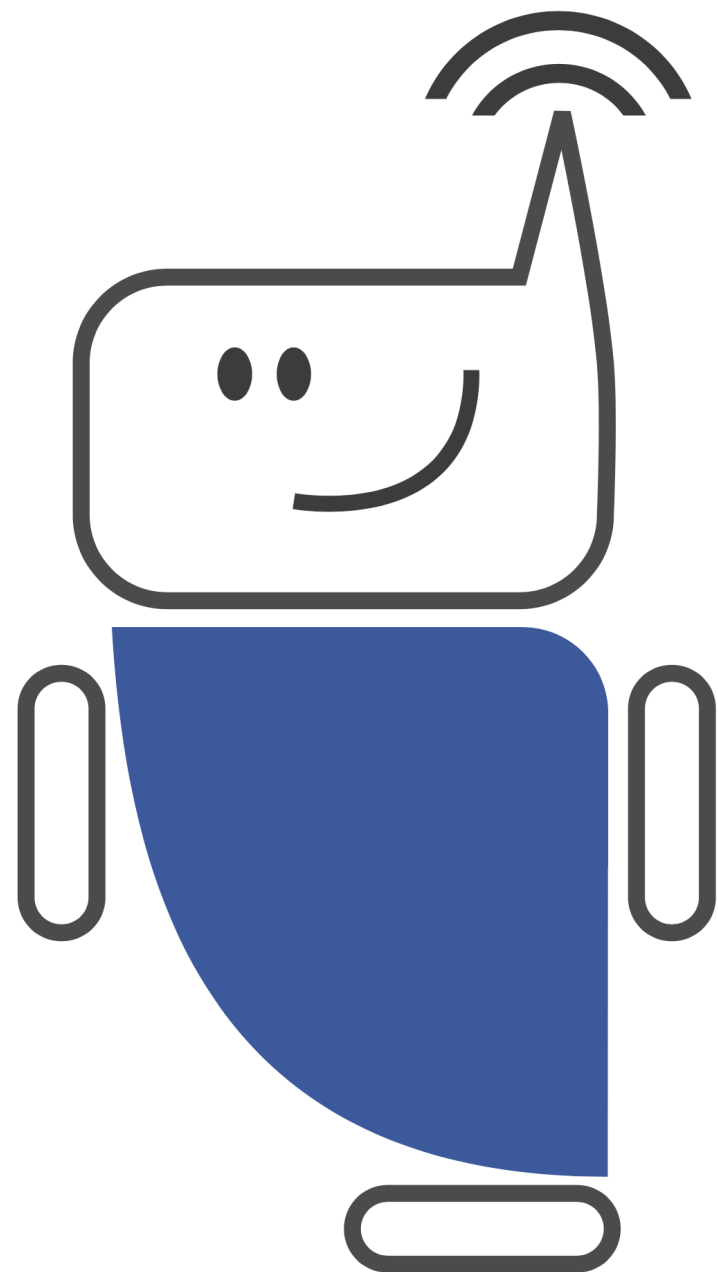


INTRODUÇÃO AOS

APLICATIVOS TYPESCRIPT

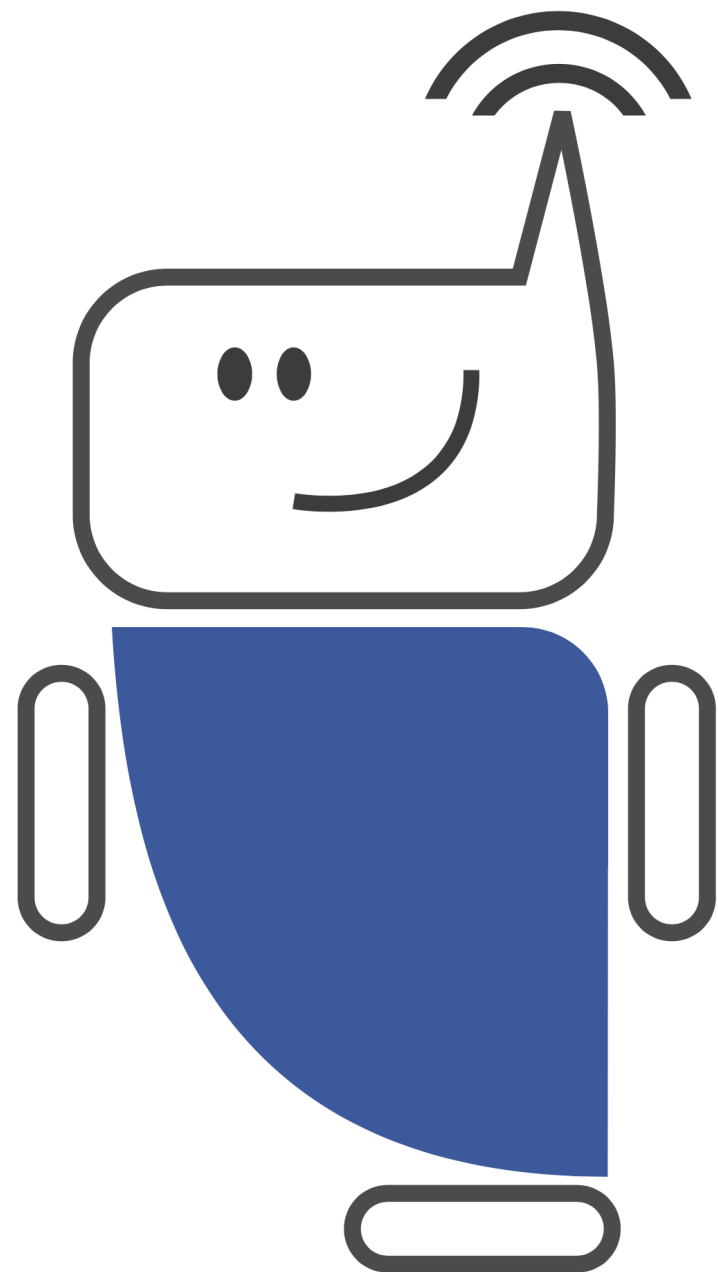


PROF. ELEANDRO MASCHIO



ROTEIRO

- ▶ Representação de Algoritmos
- ▶ Estrutura de um Código-Fonte
- ▶ Variáveis
- ▶ Atribuição
- ▶ Operadores Aritméticos
- ▶ Instrução de Entrada
- ▶ Instrução de Saída
- ▶ Comentários
- ▶ Advertências



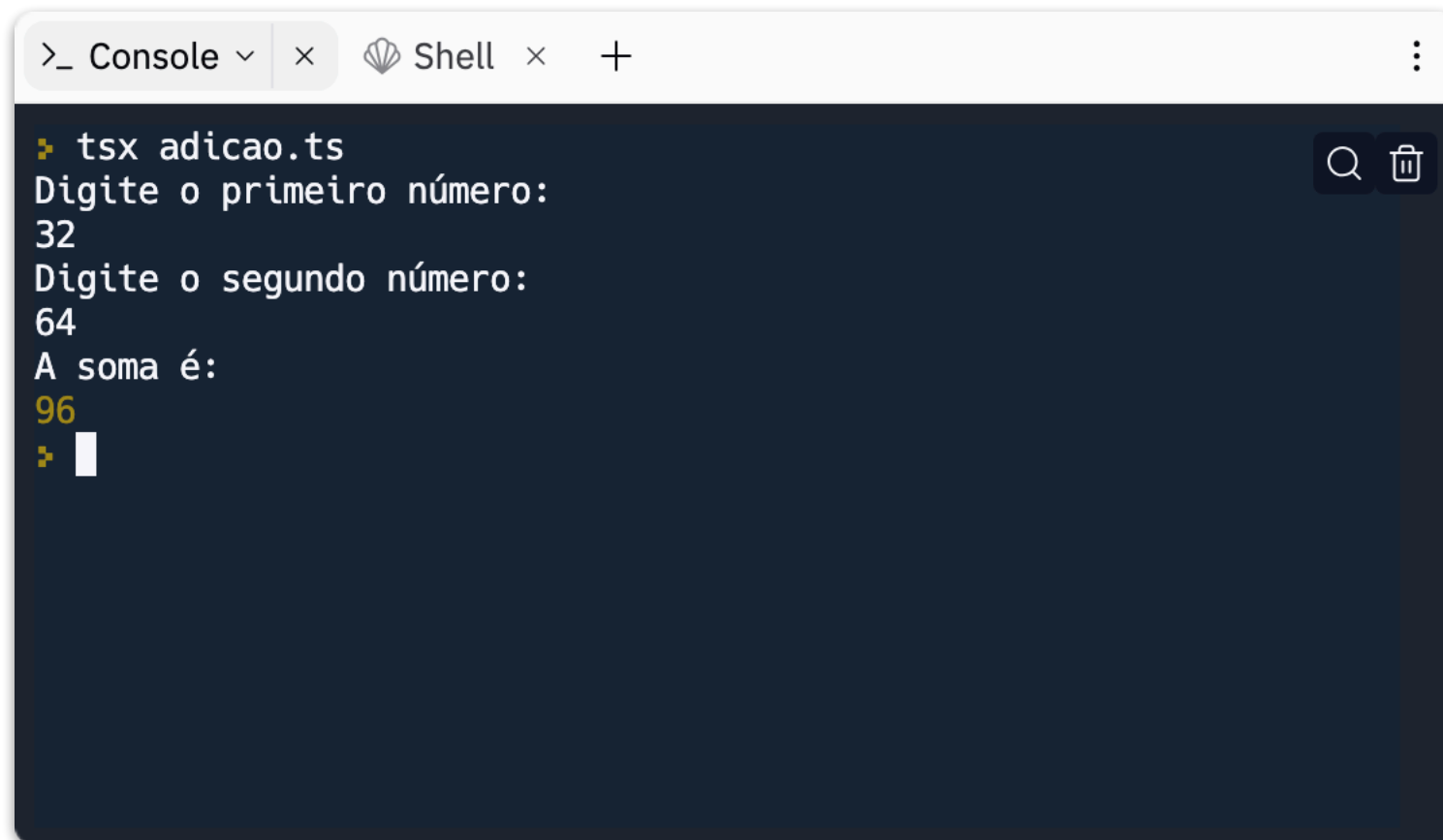
REPRESENTAÇÃO DE ALGORITMOS

EXEMPLO CANÔNICO

Tome, como exemplo canônico, o seguinte problema:

**REALIZAR A SOMA DE
DOIS NÚMEROS INTEIROS DADOS**

APLICATIVO TYPESCRIPT EM EXECUÇÃO



A screenshot of a terminal window with a dark blue background. The window has a title bar with tabs labeled ">_ Console" and "Shell", along with close, maximize, and search icons. The terminal content shows the execution of a TypeScript application. It starts with a prompt character ">" followed by the command "tsx adicao.ts". The application then prompts the user to "Digite o primeiro número:" and receives the input "32". It then prompts "Digite o segundo número:" and receives the input "64". Finally, it outputs "A soma é:" followed by the result "96" in yellow text. A new prompt character ">" is visible at the bottom, indicating the application is ready for further input. In the top right corner of the terminal area, there are icons for search and trash.

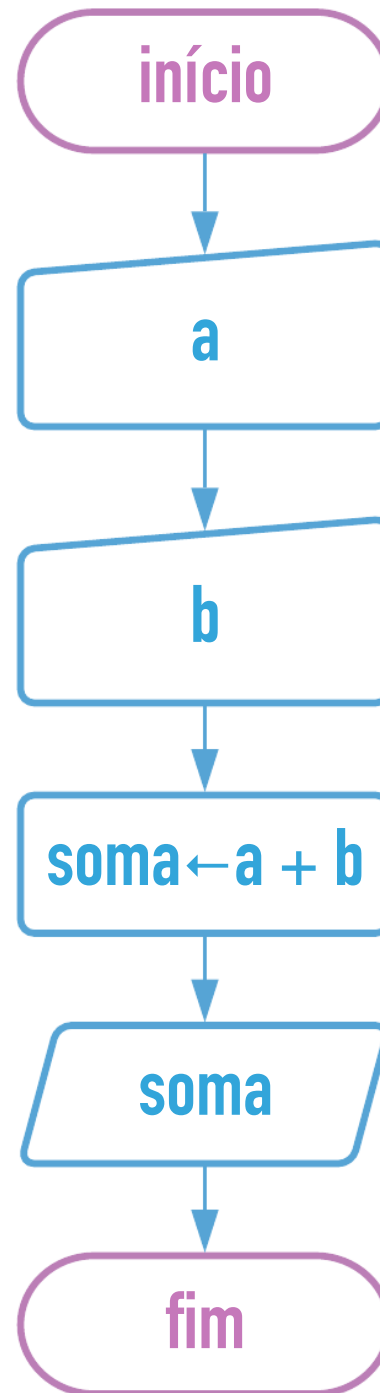
```
>_ Console × Shell × +  
➤ tsx adicao.ts  
Digite o primeiro número:  
32  
Digite o segundo número:  
64  
A soma é:  
96  
➤
```

EM FORMA DESCRITIVA OU NARRATIVA

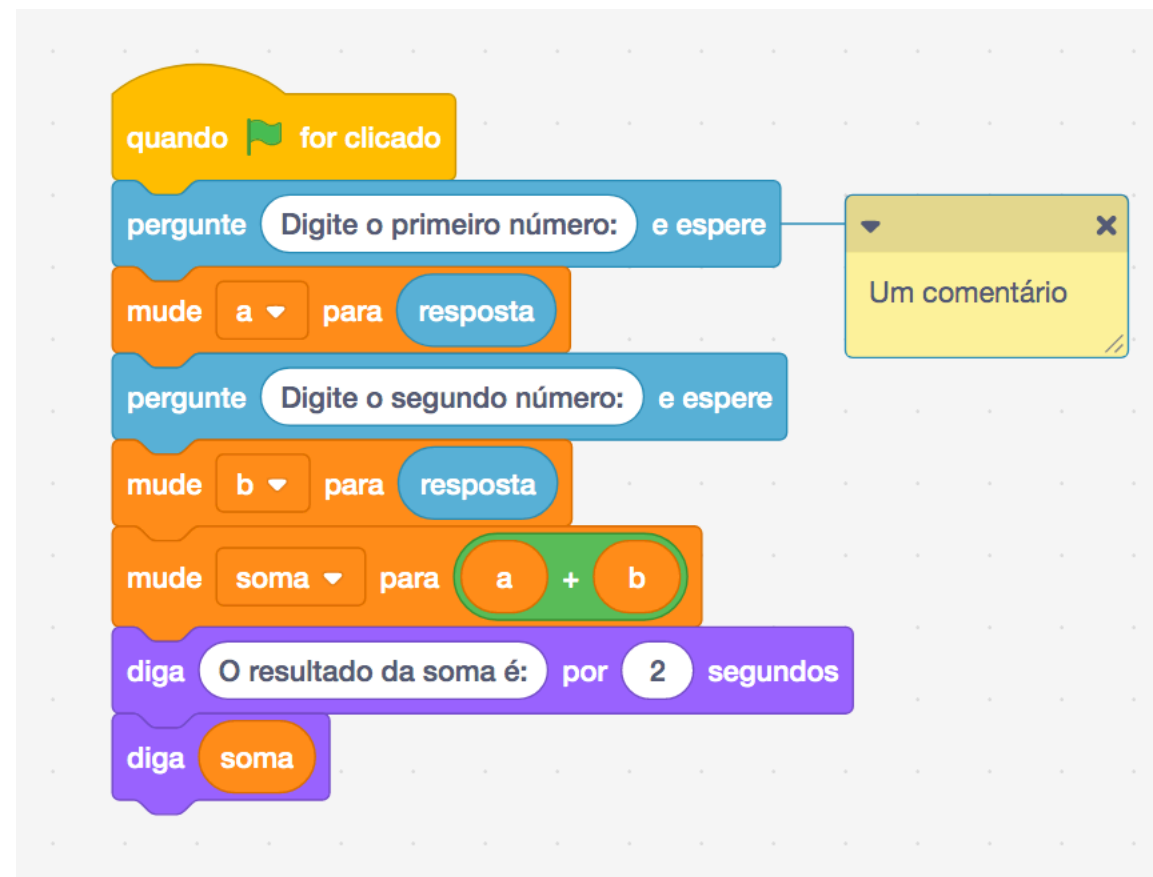
Algoritmo para **somar dois números inteiros**:

1. peça que o usuário forneça dois números inteiros.
2. calcule a soma desses dois números.
3. exiba o resultado ao usuário.

EM FLUXOGRAMA



EM BLOCOS



EM PORTUGOL, PSEUDOCÓDIGO OU PORTUGUÊS ESTRUTURADO

```
programa
{
    funcao inicio()
    {
        inteiro a, b, soma
        a = 0
        b = 0    // um comentário

        escreva("Digite o primeiro número: ")
        leia(a)

        escreva("Digite o segundo número: ")
        leia(b)

        soma = a + b

        escreva("O resultado da soma é: ")
        escreva(soma)
    }
}
```

IMPLEMENTAÇÃO NA LINGUAGEM PASCAL

```
program adicao;  
  
var  
    a, b, soma : integer;  
  
begin  
    a := 0;  
    b := 0;  
    soma := 0;    // um comentário  
  
    writeln('Digite o primeiro número:');  
    readln(a);  
  
    writeln('Digite o primeiro número:');  
    readln(b);  
  
    soma := a + b;  
  
    write('A soma é ');  
    writeln(soma);  
end.
```

IMPLEMENTAÇÃO NA LINGUAGEM JAVA

```
import java.util.Scanner;

public class Adicao
{
    public static void main(String[] args)
    {
        Scanner entrada = new Scanner(System.in);

        int a, b, soma;    // um comentário
        a = 0;
        b = 0;
        soma = 0;

        System.out.println("Digite o primeiro número:");
        a = entrada.nextInt();

        System.out.println("Digite o segundo número:");
        b = entrada.nextInt();

        soma = a + b;

        System.out.print("A soma é: ");
        System.out.println(soma);
    }
}
```

IMPLEMENTAÇÃO NA LINGUAGEM TYPESCRIPT

```
let teclado = require("readline-sync");

let a: number = 0,
    b: number = 0,
    soma: number = 0;    // um comentário

console.log("Digite o primeiro número:");
a = teclado.questionInt();

console.log("Digite o segundo número:");
b = teclado.questionInt();

soma = a + b;

console.log("A soma é:");
console.log(soma);
```

LINGUAGEM JAVASCRIPT

- ▶ Sintaticamente simples. Instruções de controle não muito diferentes de Java, C e PHP.
- ▶ Linguagem da qual não se pode fugir no curso (mais usada na web).
- ▶ Dificilmente pode ser evitada na programação web.
- ▶ Antecipa a formação para os estágios.
- ▶ Bastante aderente ao perfil do egresso (nacional e mundial).
- ▶ Crescentes materiais de JavaScript como primeira linguagem.
- ▶ Aceita pela maioria dos juízes on-line.



LINGUAGEM JAVASCRIPT

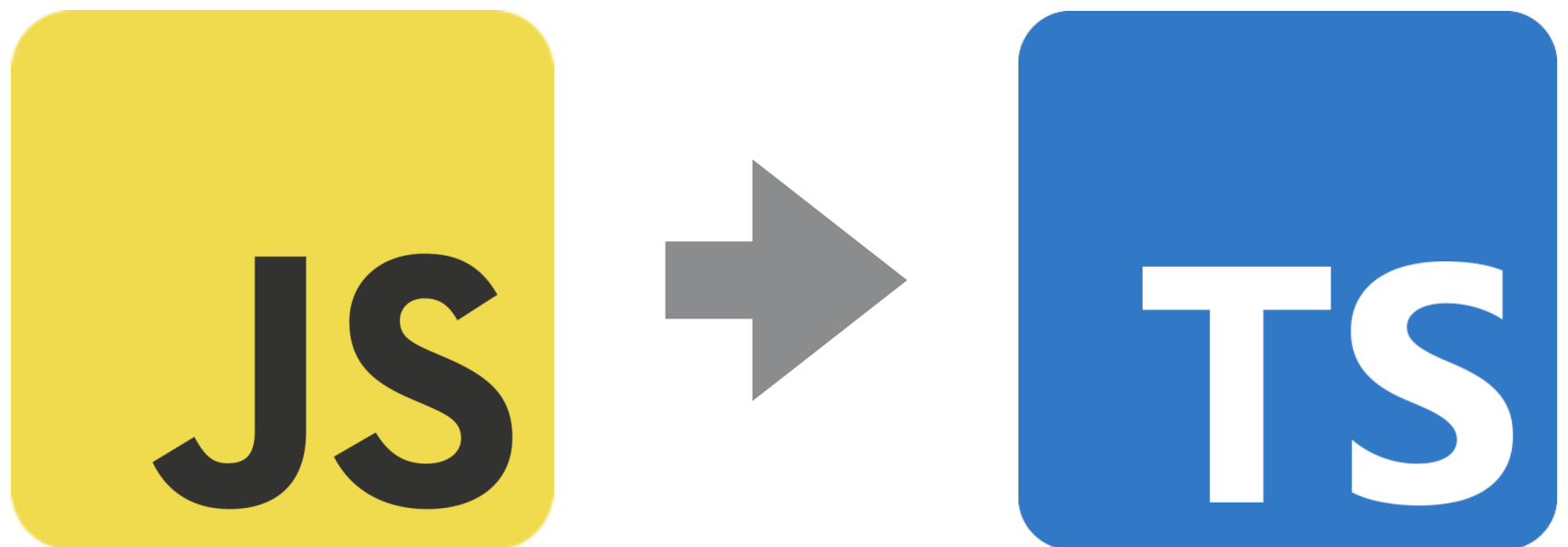
- ▶ Por que Stanford trocou Java por JavaScript?
 - ▶ Pela simplicidade!
 - ▶ Objetivo de usar uma linguagem menos complexa para introduzir os fundamentos de programação para iniciantes.
 - ▶ Usaram Java por 15 anos como primeira linguagem.

**SIM... MAS A DISCIPLINA USA TYPESCRIPT,
NÃO JAVASCRIPT...**



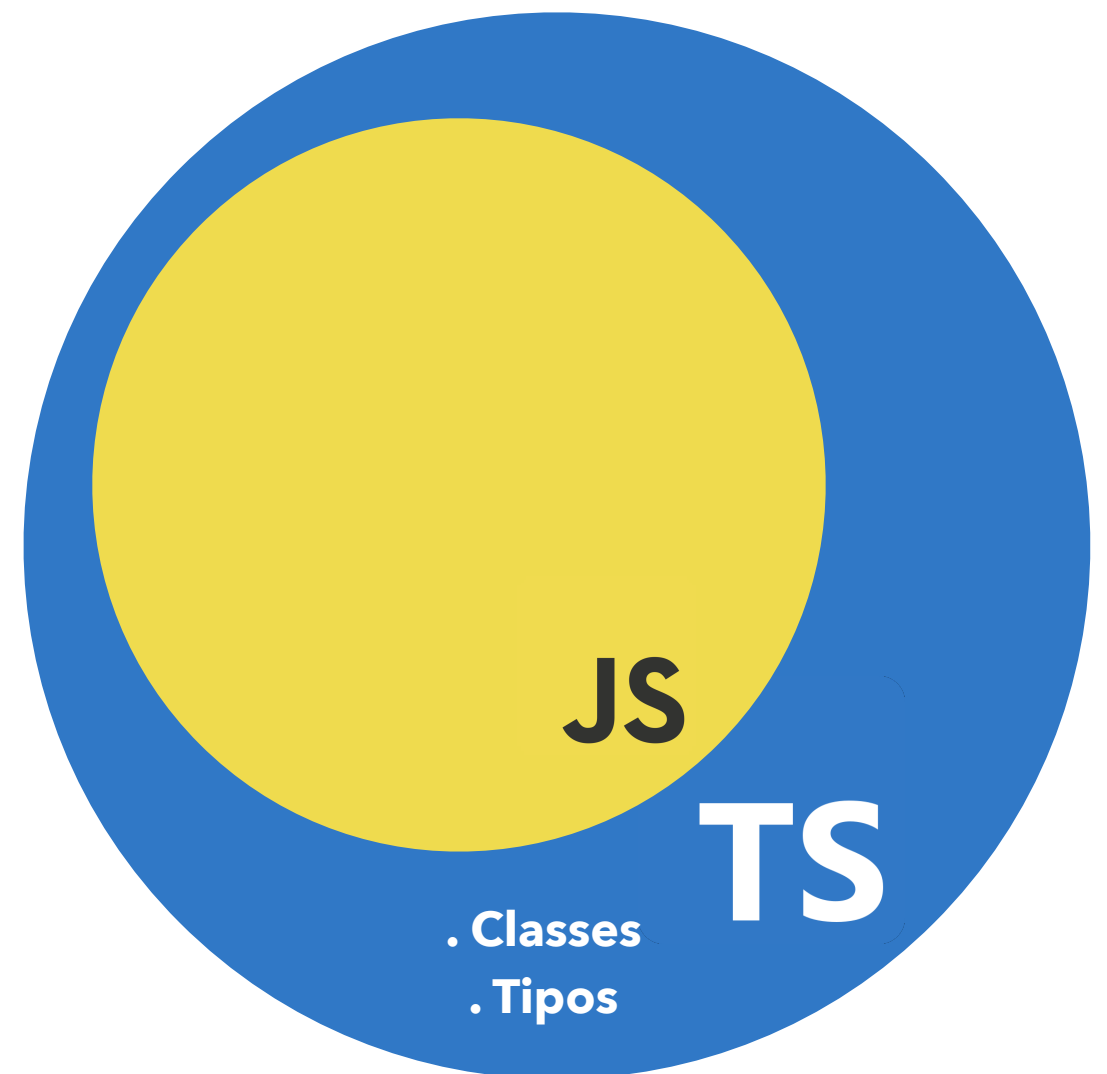
LINGUAGEM TYPESCRIPT

- ▶ Linguagem de programação de código aberto desenvolvida pela Microsoft.
- ▶ Na verdade, é um **superconjunto** sintático estrito de JavaScript.
- ▶ Divulgada em 2012, após dois anos de desenvolvimento.
- ▶ Influenciada por C#, Java e JavaScript.
- ▶ Pode ser usada para desenvolver aplicações no lado tanto do cliente quanto do servidor.



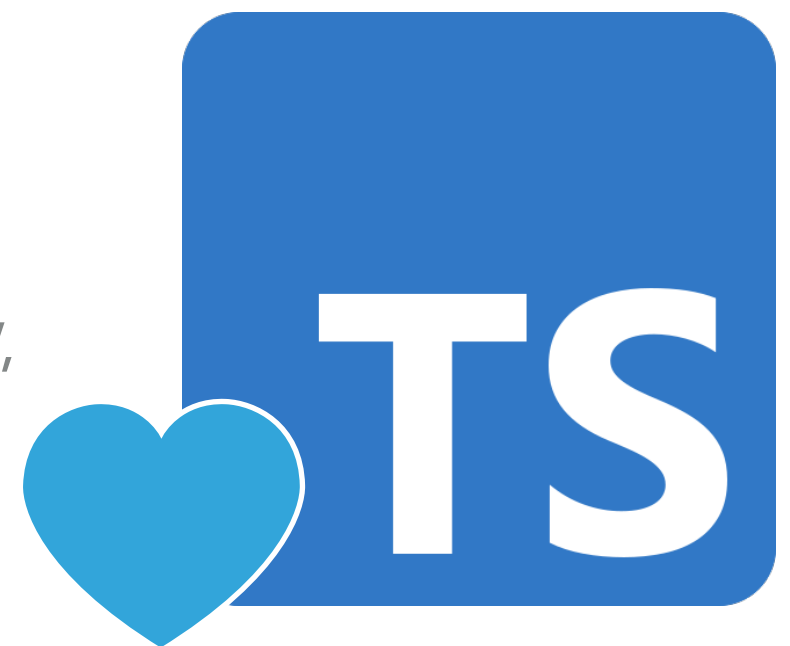
LINGUAGEM TYPESCRIPT

O QUE MAIS NOS
INTERESSA...



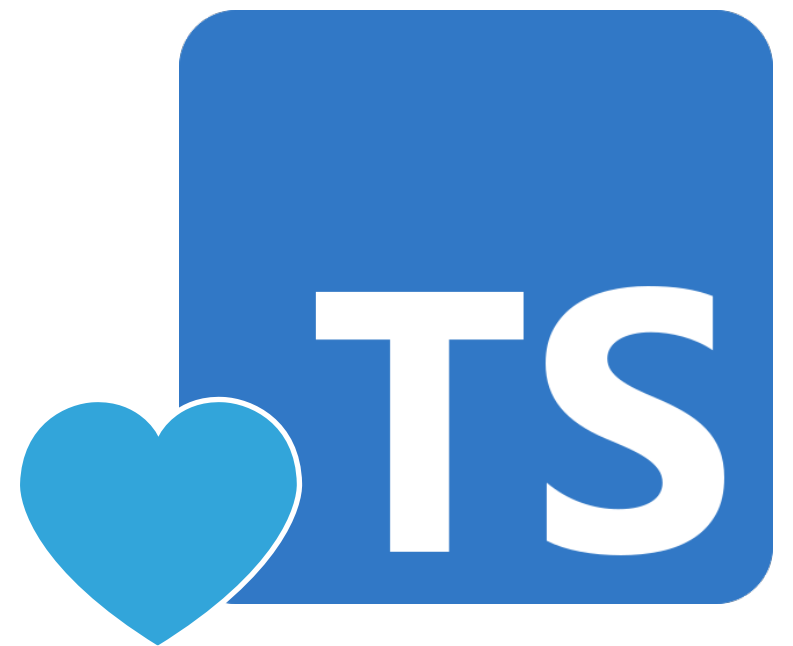
LINGUAGEM TYPESCRIPT

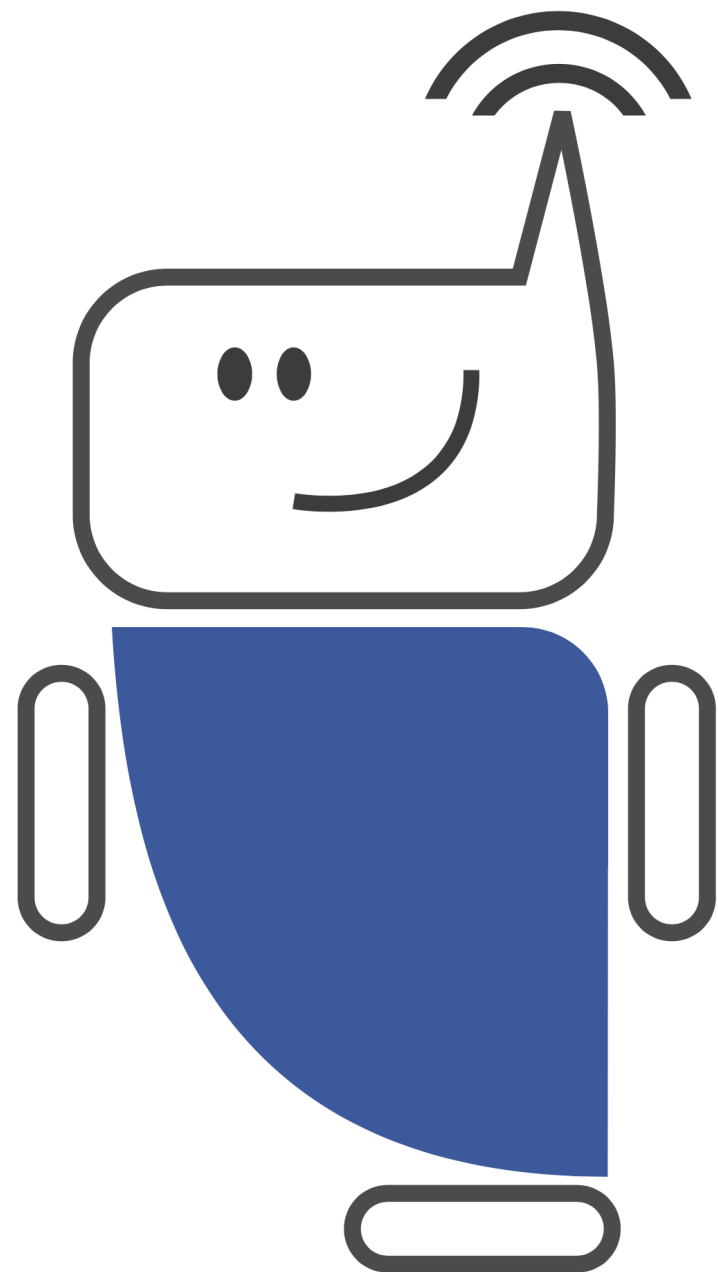
- ▶ Algumas vantagens:
 - ▶ Fácil migração para JavaScript propriamente dita.
 - ▶ Programas JavaScript existentes também são programas TypeScript válidos.
 - ▶ Adiciona tipagem estática opcional à linguagem.
 - ▶ Beneficia o entendimento dos tipos de dados.
 - ▶ Didaticamente, proporciona uma codificação mais regrada.
 - ▶ Maior possibilidade de descobrir erros durante a codificação.
 - ▶ Muitos consideram uma das linguagens “mais amadas”, mas cuidado com isso.



LINGUAGEM TYPESCRIPT

```
console.log("Olá, mundo!");
```



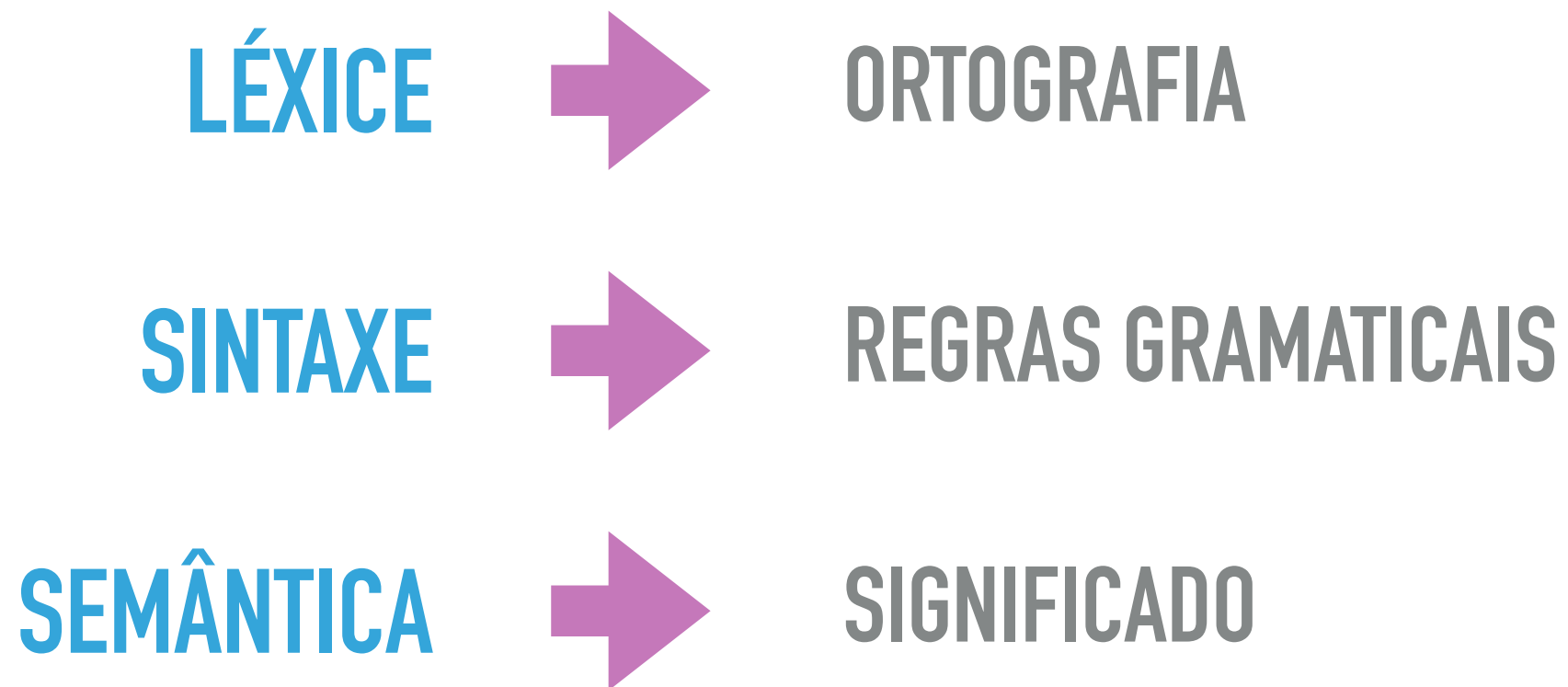


ESTRUTURA DE UM

**CÓDIGO-FONTE
EM TYPESCRIPT**

MAS... SINTAXE?

Uma linguagem de programação, assim como qualquer outra língua (idioma), possui:



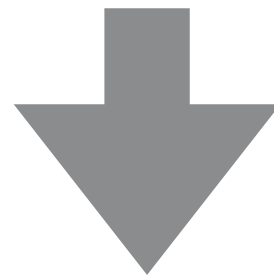
COMPARAÇÃO COM OUTRAS LINGUAGENS

- ▶ A maioria das outras linguagens exigem que o código siga uma estruturação muito bem definida.
- ▶ Em Java, por exemplo, um "Olá, Mundo!", para ser executado, precisa:
 - ▶ Estar dentro de um método `main()` com a devida assinatura.
 - ▶ Esse método precisa estar dentro de uma classe (`class`).
 - ▶ Tudo isso deve ser delimitado por chaves.
 - ▶ E salvo em um arquivo `.java`.
- ▶ Do contrário, o aplicativo não é executado como se espera.
- ▶ Em TypeScript, isso é muito mais simples...

COMPARAÇÃO COM OUTRAS LINGUAGENS

```
public class OlaMundo
{
    public static void main(String[] args)
    {
        System.out.println("Olá, Mundo!");
    }
}
```

olaMundo.java



```
console.log("Olá, Mundo!");
```

olaMundo.ts

PONTO E VÍRGULA

```
let teclado = require("readline-sync");

let a: number = 0,
    b: number = 0,
    soma: number = 0;    // um comentário

console.log("Digite o primeiro número:");
a = teclado.questionInt();

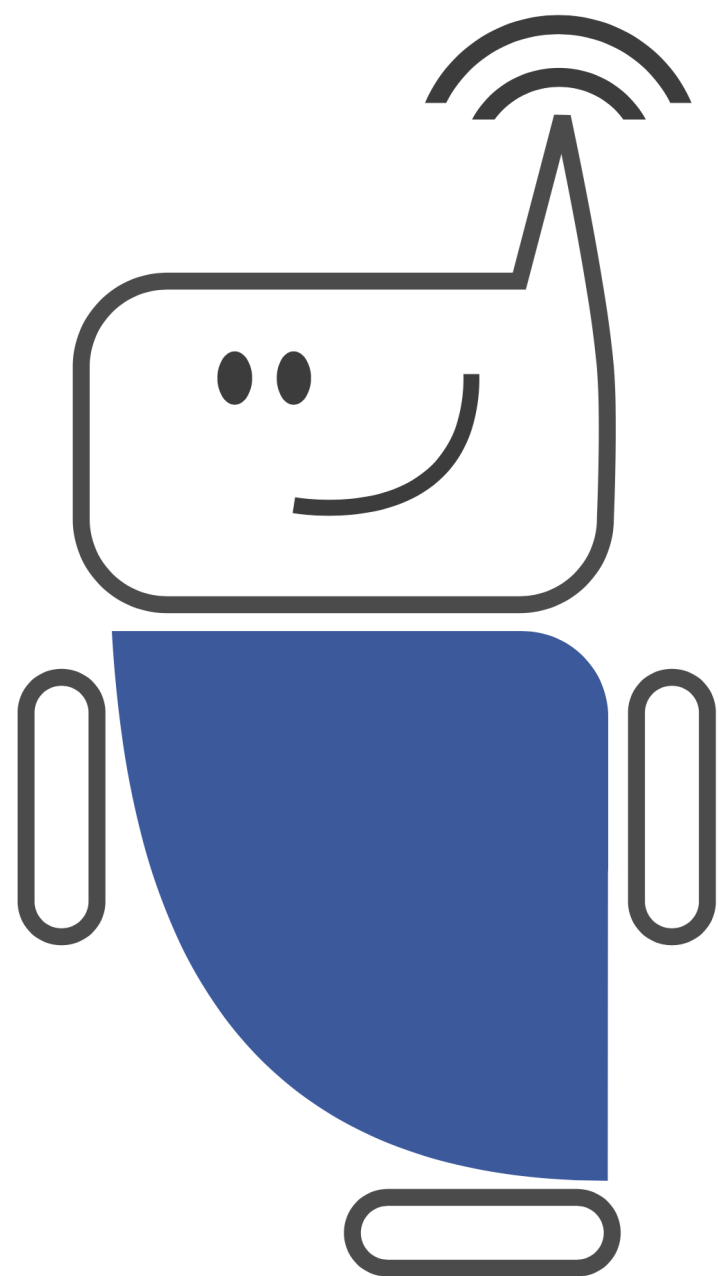
console.log("Digite o segundo número:");
b = teclado.questionInt();

soma = a + b;

console.log("A soma é:");
console.log(soma);
```


PONTO E VÍRGULA

- ▶ O ponto e vírgula indica o **fim de uma instrução** em TypeScript.
- ▶ É útil tanto para o programador quanto para o **compilador**.
- ▶ Embora seja facultativo na linguagem, trata-se de uma boa prática.
 - ▶ Principalmente em termos didáticos.
 - ▶ **Este formalismo será cobrado nas avaliações.**



VARIÁVEIS

DECLARAÇÃO DE VARIÁVEIS

```
let teclado = require("readline-sync");
```

```
let a: number,  
    b: number,  
    soma: number; // um comentário
```

```
a = 0;  
b = 0;  
soma = 0;
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

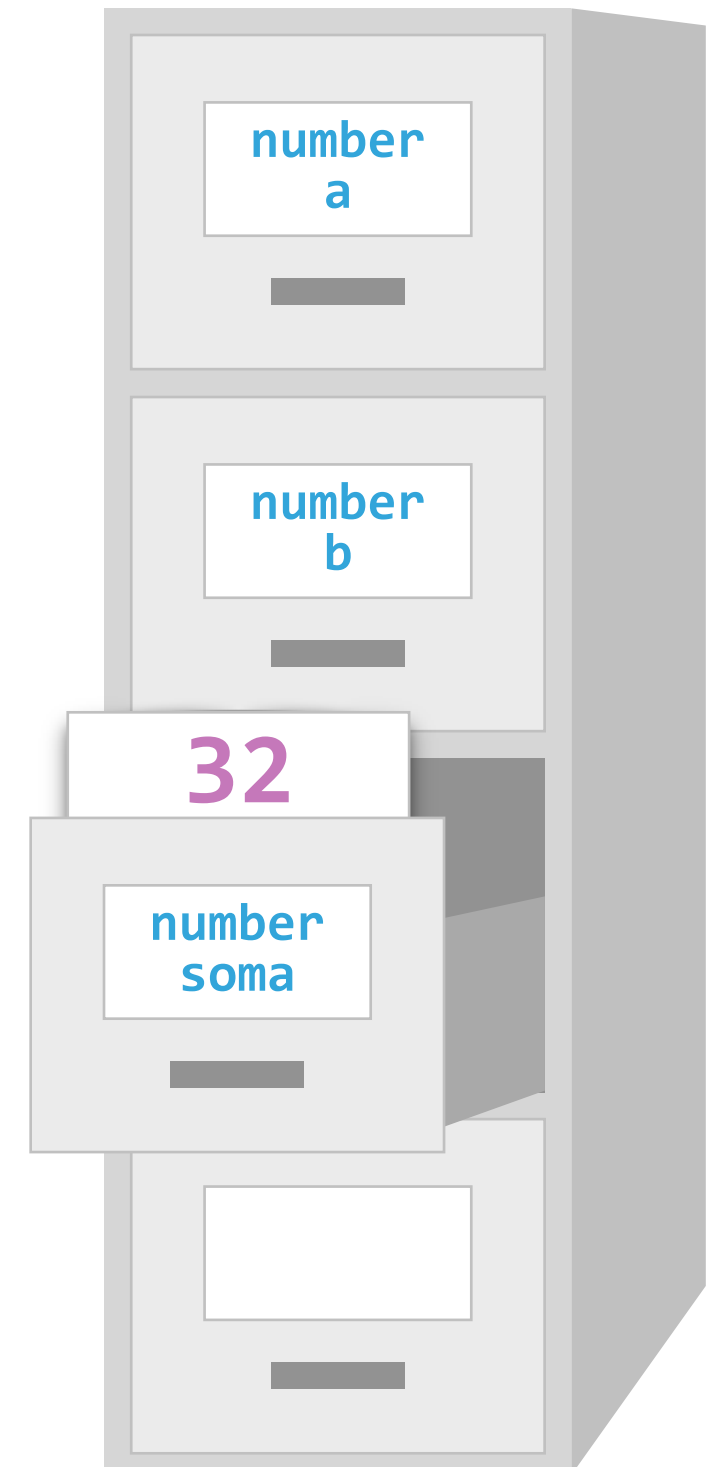
```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

DECLARAÇÃO DE VARIÁVEIS

- ▶ Equivale ao conceito matemático de variável e diz respeito a uma **incógnita**.
- ▶ É um valor desconhecido durante a codificação.
- ▶ Significa reservar uma área (endereço) da memória RAM para armazenar dados de determinado tipo.
- ▶ Para uma declaração, são necessárias duas informações:
 - ▶ o **tipo** da variável; e
 - ▶ o **identificador** ou nome dado a variável.
- ▶ Posteriormente, pode-se resgatar o valor armazenado em uma variável, por meio do identificador dela.



TIPOS DE DADOS BÁSICOS EM TYPESCRIPT

TIPO	EXPLICAÇÃO	EXEMPLO
number	número inteiro (sem casas decimais)	18
number	número real (com casas decimais)	3.1416
string	um caractere do teclado	"A"
string	cadeia ou sequência de caracteres	"palavra"
boolean	valor booleano (verdadeiro ou falso)	true / false

IDENTIFICADORES

Na maioria das linguagens, são impostas algumas restrições quanto aos nomes de variáveis.

Em TypeScript, entre **restrições** e **recomendações** iniciais, tem-se:

- ▶ Os nomes devem ser **únicos** (exclusivos).
- ▶ Nomes dados às variáveis não podem ser os mesmos de **palavras-reservadas** (`let`, `number`, `string`, `boolean`, `true`, `false`, `typeof`, `return`, etc.).
- ▶ O primeiro caractere deve ser uma **letra**.
- ▶ Os nomes devem ser formados por caracteres **numéricos** ou **alfabéticos**.
- ▶ Não devem conter **acentuação** ou **cedilha**.

IDENTIFICADORES

- ▶ Não pode haver **espaços em branco** entre os caracteres do identificador.
 - ▶ Se o identificador for composto por mais de uma palavra, use **lowerCamelCase**.
 - ▶ Ao invés de tudojunto ou com espacos `_em_` branco.



IDENTIFICADORES

- ▶ A linguagem **diferencia maiúsculas e minúsculas** (*case sensitive*).
 - ▶ São identificadores distintos: `soma`, `Soma` e `SOMA`.
- ▶ Os nomes devem ser **autoexplicativos**.
- ▶ Devem ser evitados nome muito **longos**.
 - ▶ Dificuldade de leitura.
 - ▶ Limite de caracteres significativos imposto pelos compiladores.
- ▶ Embora permitido, recomenda-se cautela com o **sublinhado** e o **cifrão**.
- ▶ Enquanto as classes começam com letras maiúsculas, **variáveis** e **métodos** são **iniciados com letras minúsculas**.

ONDE E QUANDO DECLARAR?

- ▶ De forma distinta de muitas linguagens, a declaração de variáveis em TypeScript pode ser feita em **qualquer momento do código**.
- ▶ A variável deve, obviamente, ser declarada **antes da respectiva utilização**.
- ▶ Há uma importante diferença de **escopo** (abrangência) se a variável é declarada dentro de um método específico, ou fora de qualquer método.
 - ▶ Essa distinção é vista no decorrer da disciplina.
- ▶ Para os primeiros exemplos e exercícios, convencionam-se que todas as variáveis são declaradas no corpo do código principal.

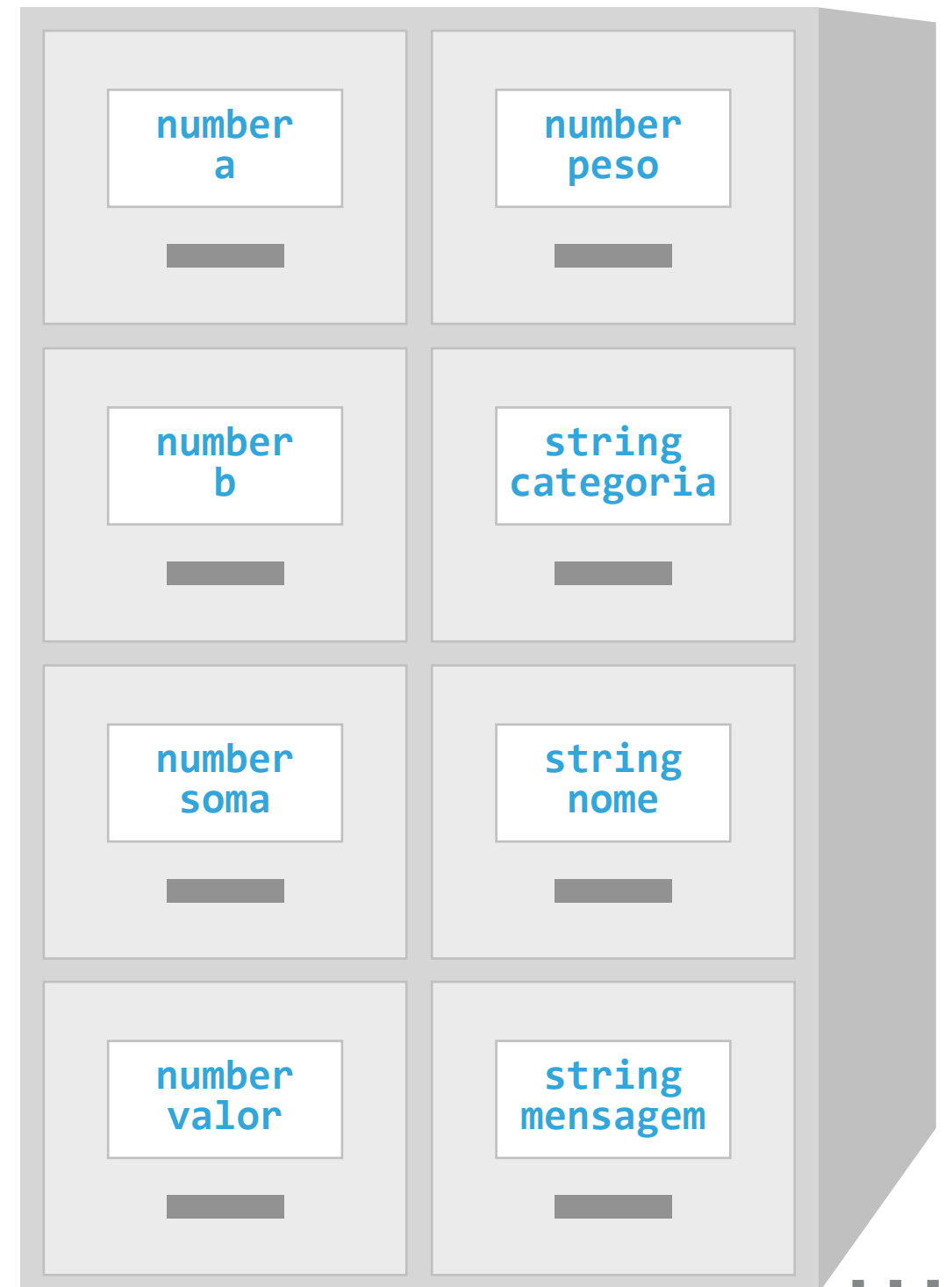
SINTAXE PARA DECLARAÇÃO

```
let variavel1: tipo,  
    variavel2: mesmoTipo,  
    variavel3: outroTipo;
```

EXEMPLOS

```
let a: number,  
    b: number,  
    c: number,  
    valor: number,  
    peso: number,  
    categoria: string,  
    nome: string,  
    mensagem: string,  
    aprovado: boolean;
```

```
// (...)
```



INICIALIZAÇÃO DE VARIÁVEIS

```
let teclado = require("readline-sync");
```

```
let a: number,  
    b: number,  
    soma: number;    // um comentário
```

```
a = 0;  
b = 0;  
soma = 0;
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

INICIALIZAÇÃO DE VARIÁVEIS

- ▶ Em termos teóricos, ao declarar uma variável, o conteúdo inicial dela é considerado **indefinido** ou, conforme a linguagem, comumente referido como **lixo de memória**.
- ▶ Linguagens de alto nível já contam com mecanismos para inicialização automática.
- ▶ Independentemente da linguagem, no começo da aprendizagem, trata-se de uma boa prática, ou medida de segurança, **inicializar as variáveis logo após declaradas**.
- ▶ Geralmente, atribuem-se valores que indiquem que a variável ainda não foi usada.
 - ▶ Tal como o **zero**, para tipos numéricos, ao invés de valores arbitrários.
- ▶ Variáveis não inicializadas em TypeScript recebem o valor de **undefined**.

```
let valor: number;  
console.log(valor);    // undefined
```

EXEMPLOS DE INICIALIZAÇÃO CONFORME O TIPO

number

```
a = 0;  
b = 0;  
soma = 0;  
valor = 0;  
peso = 0;
```

string

```
categoria = "";  
nome = "";  
mensagem = "";
```

// sem qualquer espaço entre aspas duplas

boolean

```
aprovado = true;  
matriculado = false;
```


SINTAXE PARA DECLARAÇÃO E INICIALIZAÇÃO IMEDIATA

```
let variavel1: tipo = expressao,  
    variavel2: mesmoTipo = expressao,  
    variavel3: outroTipo = expressão;
```

onde **expressao** obedece a sintaxe da linguagem.

EXEMPLOS

```
let a: number = 0,  
    b: number = 0,  
    soma: number = 0,  
    valor: number = 0,  
    peso: number = 0,  
    categoria: string = "",  
    nome: string = "",  
    mensagem: string = "",  
    aprovado: boolean = true,  
    matriculado: boolean = false;
```



```
let a: number;  
a = 0;
```

TIPAGEM ESTÁTICA OPCIONAL

- ▶ No TypeScript, a tipagem estática é opcional.
- ▶ A linguagem consegue inferir o tipo da variável na declaração, conforme o valor atribuído.

```
let valor = 0;  
console.log(typeof valor);    // number
```
- ▶ Explicitar o tipo auxilia muito na didática de uma primeira linguagem de programação.
 - ▶ Força-se o processo cognitivo de entender a tipagem de um valor.
 - ▶ **Este formalismo será cobrado nas avaliações.**
- ▶ Ajuda a detectar erros relacionados à tipagem durante o desenvolvimento.
- ▶ O operador `typeof` retorna uma cadeia de caracteres (`string`) com o tipo da variável **depois da inicialização**.

TIPAGEM ESTÁTICA OPCIONAL

- ▶ Depois de declarada, não se pode mudar o tipo de uma variável.

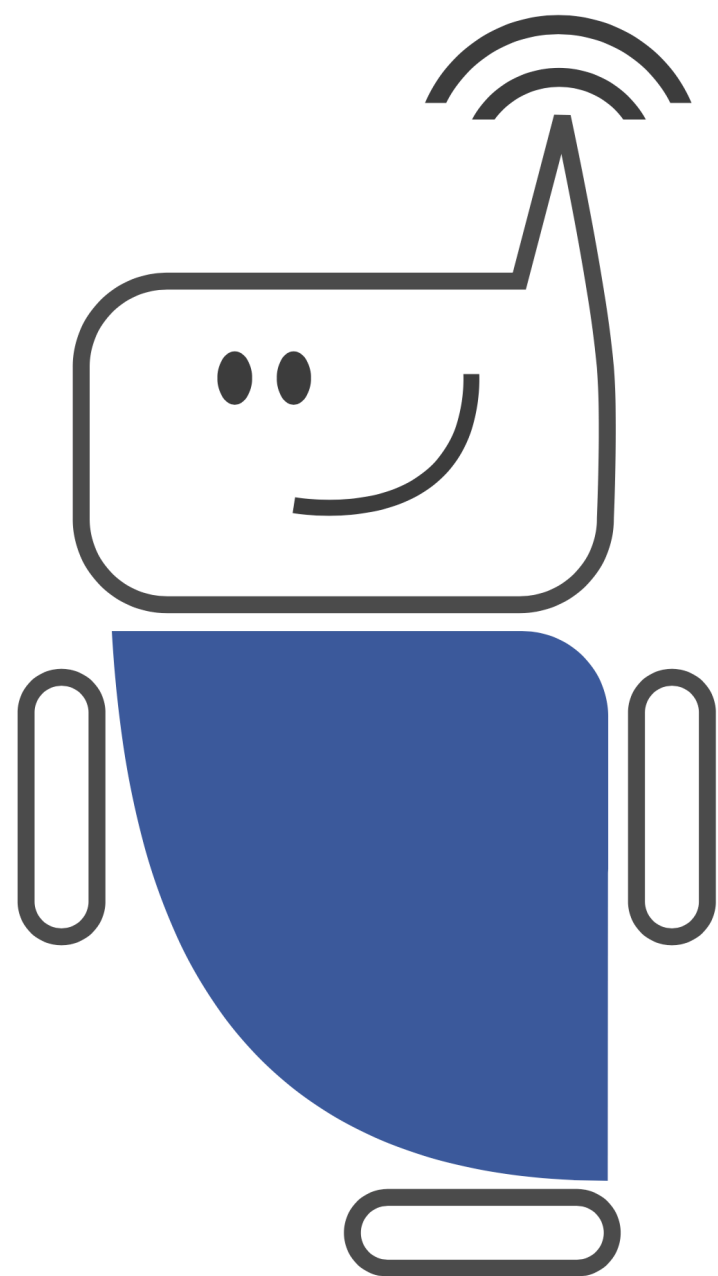
```
let valor: number = 0;  
valor = "oito";           // Type 'string' is not assignable to type 'number'.
```

- ▶ E também, não se consegue tipificar múltiplas variáveis com o uso de vírgulas.

```
let a, b : number;       // Apenas B foi declarada como number.  
a = "Um texto";  
b = 0;
```

- ▶ Agora, tanto a quanto b são do tipo number.

```
let a: number, b: number;  
a = 0;  
b = 0;
```

ATRIBUIÇÃO

OPERADOR DE ATRIBUIÇÃO

```
let teclado = require("readline-sync");
```

```
let a: number = 0,  
    b: number = 0,  
    soma: number = 0;    // um comentário
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

ATRIBUIÇÃO

- ▶ Armazena, no endereço de memória correspondente ao identificador, o resultado da expressão avaliada.

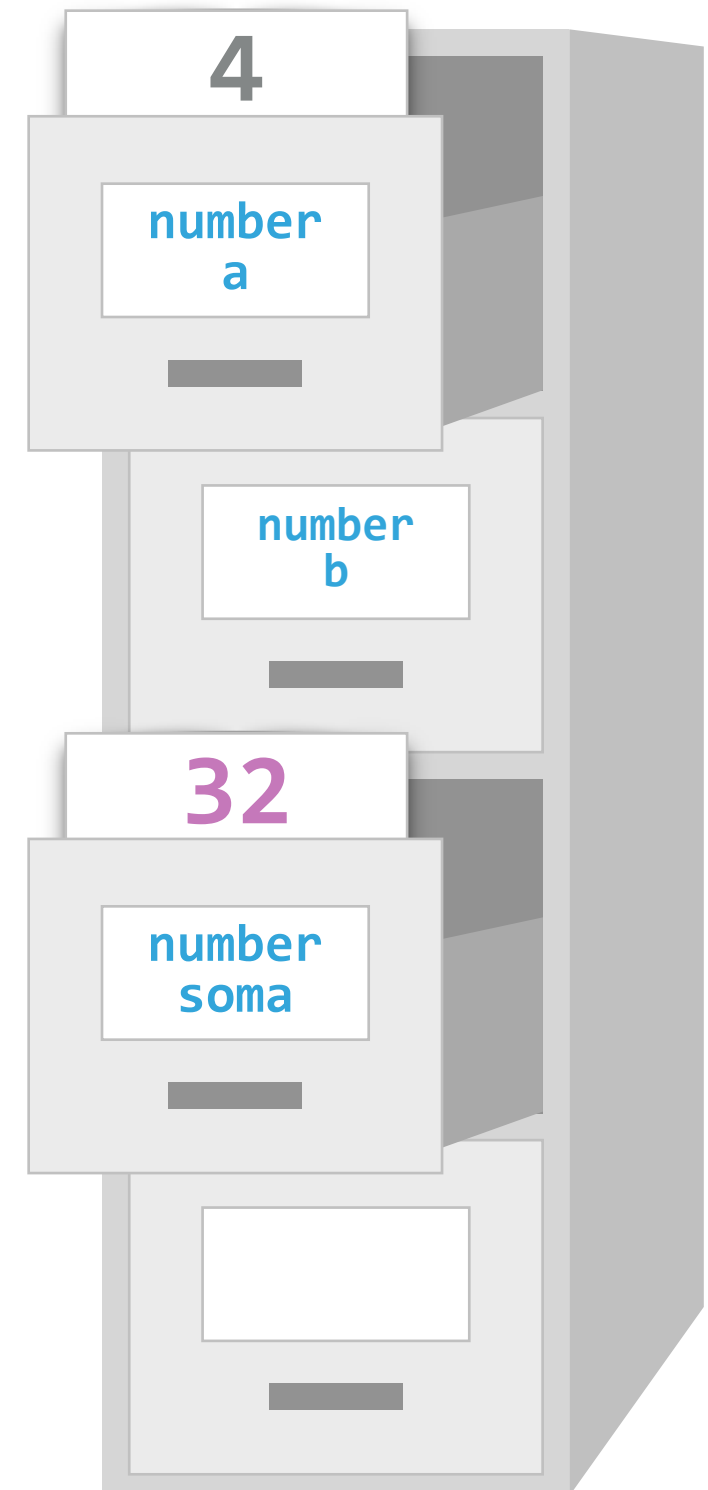
SINTAXE DA ATRIBUIÇÃO

`variavel = expressao;`

onde `variavel` é o identificador de uma variável declarada e a `expressao` obedece a sintaxe da linguagem.

EXEMPLOS

```
soma = 32;  
soma = 24 + 8;  
soma = 4 * 8;  
soma = 8 * a;
```



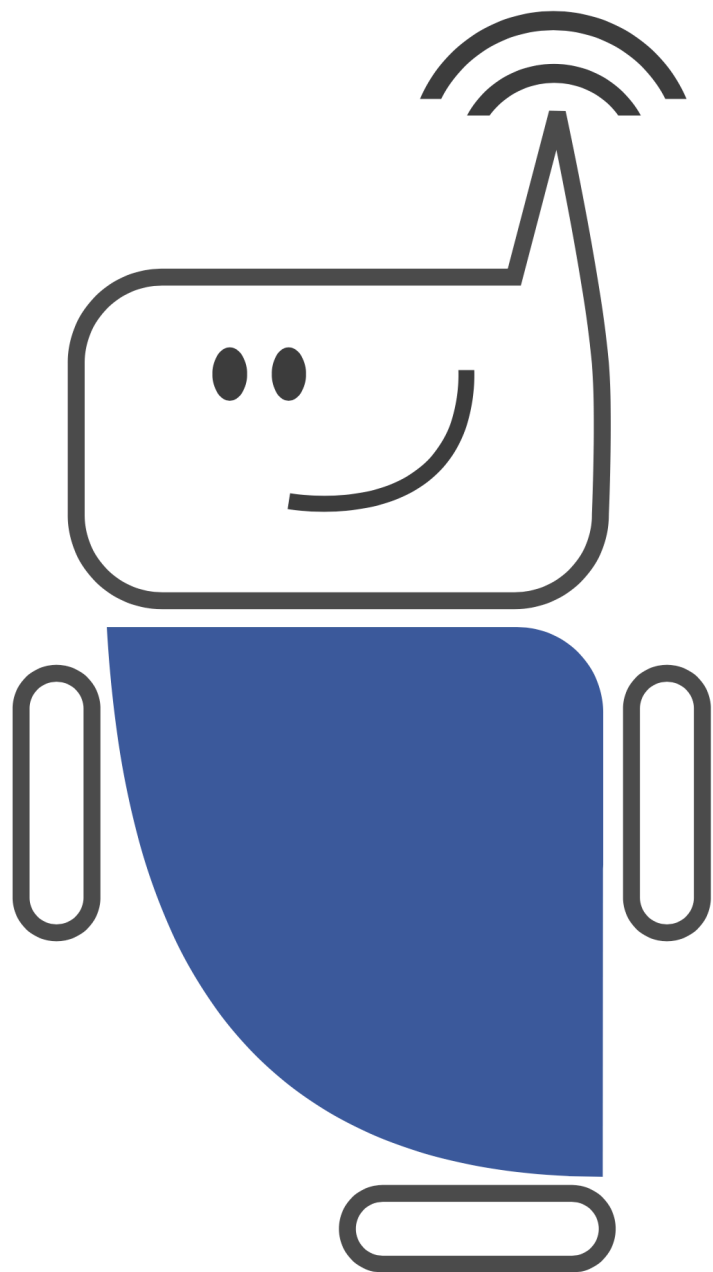
OUTROS EXEMPLOS DE ATRIBUIÇÃO

```
valor = 123.45;  
  
categoria = 'A';  
  
nome = "Sheldon Cooper";  
  
aprovado = true;
```

QUAIS SÃO OS TIPOS?

ATRIBUIÇÃO DOS TIPOS DE DADOS BÁSICOS EM TYPESCRIPT

TYPESCRIPT	TIPO	SINTAXE	EXEMPLO
number	inteiro	apenas o número	18
number	real	ponto	3.1416
string	caractere	asplas simples ou duplas	'A'
string	cadeia de caracteres	asplas simples ou duplas	"palavra"
boolean	booleano	palavras-reservadas	true / false



OPERADORES ARITMÉTICOS

OPERADORES ARITMÉTICOS EM TYPESCRIPT

OPERADOR	AÇÃO
-	subtração e sinal de negativo (menos unário)
+	adição
*	multiplicação
/	divisão
/	divisão inteira, truncar com <code>Math.trunc()</code>
%	resto da divisão inteira
**	potência

DIVISÃO INTEIRA E REAL

- ▶ O TypeScript não possui operador específico para divisão inteira.
- ▶ Deve-se executar a divisão real e truncar o valor.
 - ▶ `Math.trunc()`.

OPERAÇÃO	RESULTADO
<code>10 / 4</code>	2.5
<code>Math.trunc(10/4)</code>	2

DIVISÃO INTEIRA E RESTO

17

14

3

7


2

OPERAÇÃO	OPERADOR	EXPRESSÃO	RESULTADO
Divisão inteira	/	Math.floor(17 / 7)	2
Resto da divisão	%	17 % 7	3

PRECEDÊNCIA DOS OPERADORES ARITMÉTICOS

- ▶ Operadores de mesma precedência são avaliados na ordem em que se fizerem presentes, da esquerda para a direita.

PRECEDÊNCIA	OPERADOR
1°	()
2°	- (unário)
3°	* / %
4°	+ -



MAIOR

MENOR

ALTERAÇÃO DA PRECEDÊNCIA

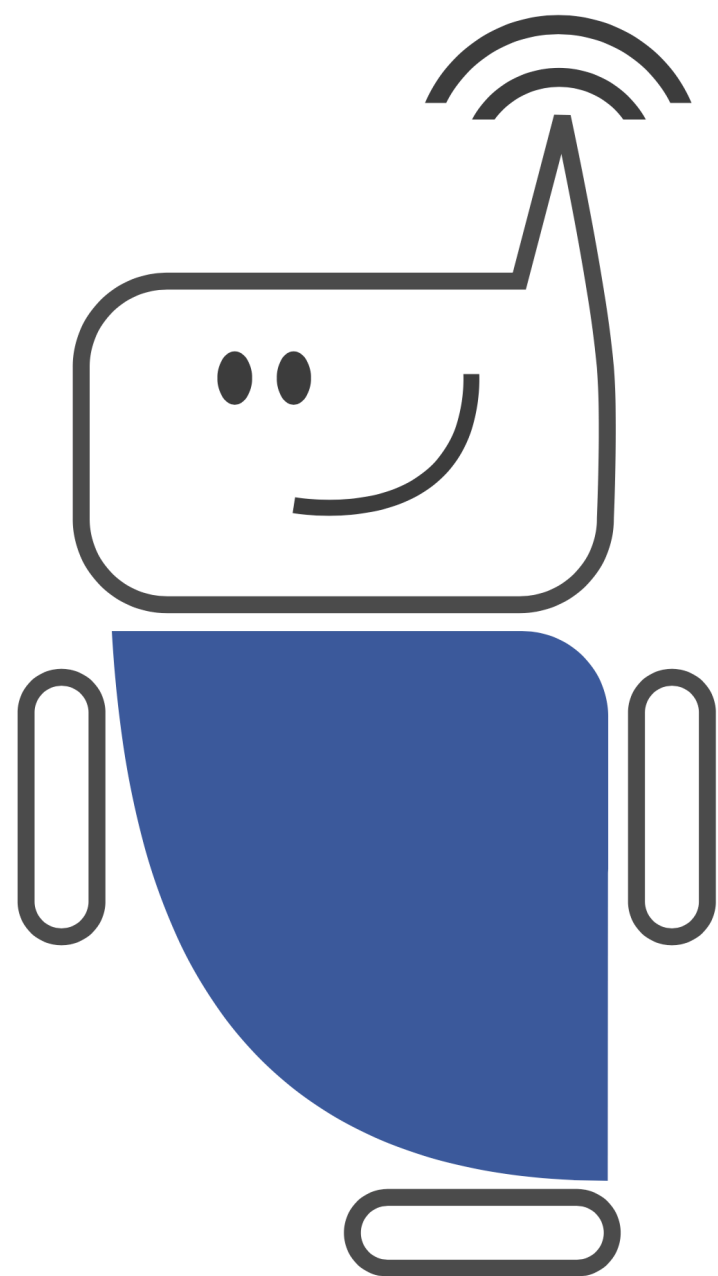
- ▶ Não se usa colchetes ou chaves, **somente parênteses**:

```
calculo = (5+2)*4 - ((3*9)+(15*5));
```

- ▶ Mesmo que não se deseje alterar a precedência, parênteses melhoram a **legibilidade**:

```
calculo = 5 + 2 * 4 - 3 * 9 + 15 * 5;
```

```
calculo = 5 + (2 * 4) - (3 * 9) + (15 * 5);
```



INSTRUÇÃO DE

ENTRADA

MÉTODO QUESTIONINT()

```
let teclado = require("readline-sync");

let a: number = 0,
    b: number = 0,
    soma: number = 0;    // um comentário

console.log("Digite o primeiro número:");
a = teclado.questionInt();

console.log("Digite o segundo número:");
b = teclado.questionInt();

soma = a + b;

console.log("A soma é:");
console.log(soma);
```

MÉTODO QUESTIONINT()

- ▶ Retorna um valor inteiro lido a partir do teclado.
- ▶ Ao ser atribuído a uma variável, armazena o valor lido no endereço de memória correspondente.

SINTAXE

```
variavel = teclado.questionInt();
```

onde **variavel** é o identificador de uma variável inteira previamente declarada.

EXEMPLO

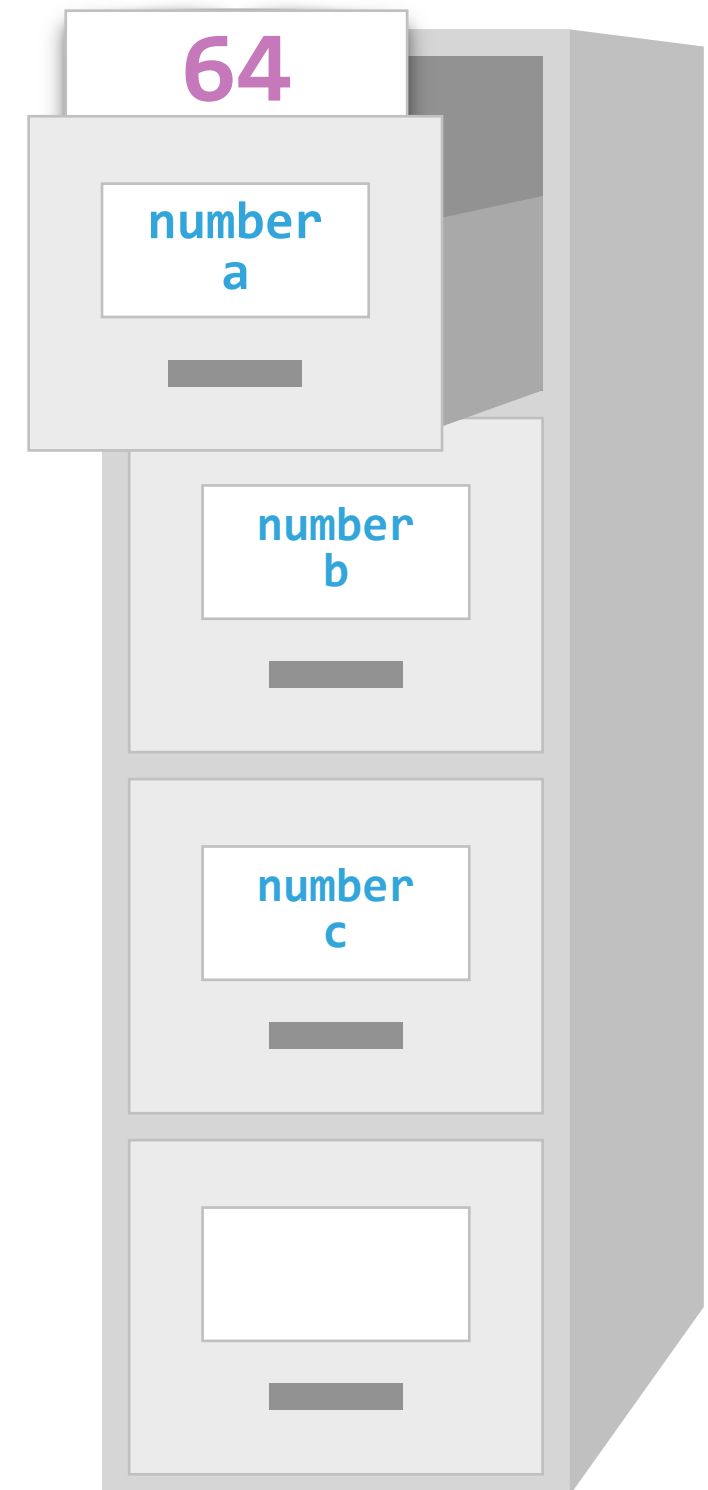
```
a = teclado.questionInt();
```

MÉTODO QUESTIONINT()

```
a = teclado.questionInt();
```



Onde 64 é um valor digitado no teclado e depois atribuído para a variável a.



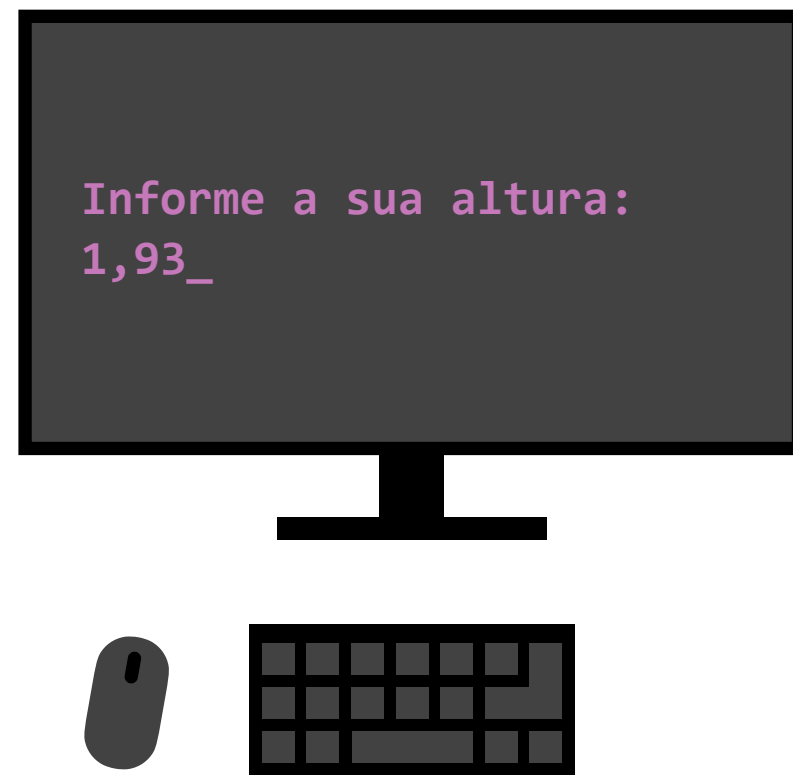
MÉTODOS PARA OUTROS TIPOS DE DADOS

TIPO	MÉTODO	ENTRADA
number	questionInt()	18
number	quesitonFloat()	3,1416
string	question().charAt(0)	A
string	question()	palavra
boolean	–	–

SEPARADOR DECIMAL

- ▶ As linguagens de programação têm o inglês como idioma padrão.
 - ▶ No código, usa-se o **ponto** como separador decimal.
- ▶ A entrada pelo teclado obedece o idioma do sistema operacional.
 - ▶ Em PT-BR (Português Brasileiro), usa-se a **vírgula** como separador decimal.

```
let altura: number = 1.93;
```



OBJETO RESPONSÁVEL PELA ENTRADA DE DADOS

```
let teclado = require("readline-sync");
```

```
let a: number = 0,  
    b: number = 0,  
    soma: number = 0;    // um comentário
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

CRIAÇÃO DO OBJETO PARA A ENTRADA DE DADOS

```
let teclado = require("readline-sync");
```

- ▶ A linha faz a declaração da variável `teclado`.
- ▶ Logo entenderemos que `const` é melhor do que `let`, nesse caso.
- ▶ O sinal de igual na mesma linha indica que a variável `teclado` é inicializada, na própria declaração, com o resultado da expressão à direita.
- ▶ Essa variável recebe o retorno dado por `require`, que será do tipo `object`.
- ▶ Com isso, a variável `teclado` pode ser usada para acessar atributos e métodos de `readline-sync`.

CARREGAMENTO DO READLINE-SYNC PELO REQUIRE

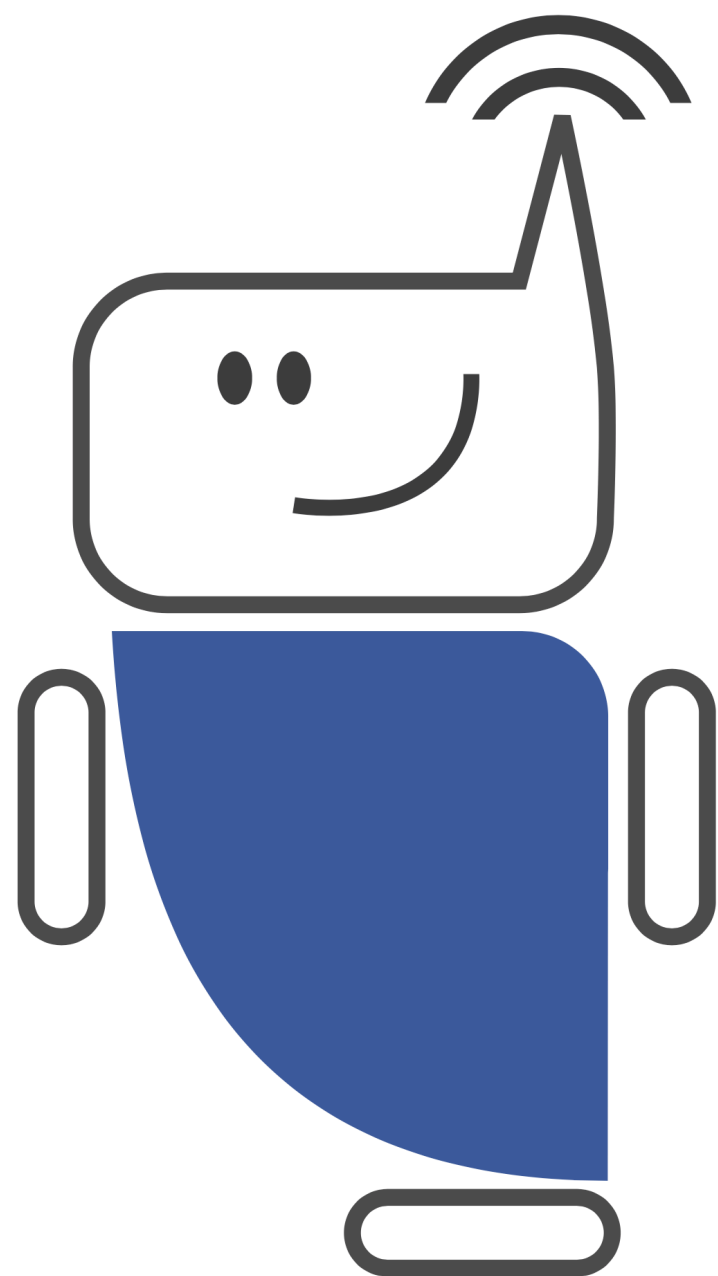
```
let teclado = require("readline-sync");
```

- ▶ A palavra-reservada **require** indica, ao compilador, o uso de um **módulo de arquivo externo** ou de um **pacote instalado globalmente**.
- ▶ TypeScript possui um vasto conjunto de **classes predefinidas**, tanto da Microsoft quanto de desenvolvedores independentes.
- ▶ Essas classes são agrupadas em **pacotes**.
- ▶ Sem falar nos pacotes do JavaScript e do Node.js.

CARREGAMENTO DO READLINE-SYNC PELO REQUIRE

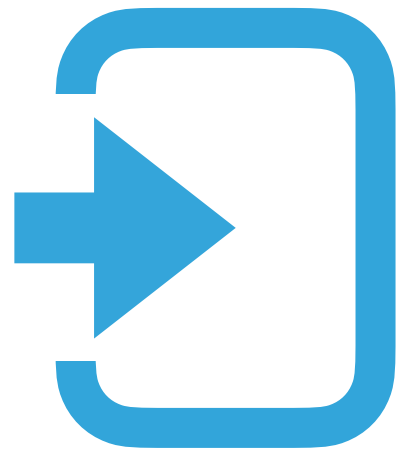
```
let teclado = require("readline-sync");
```

- ▶ Para a entrada de dados, é necessário importar o pacote `readlineSync` (Synchronous Readline).
- ▶ O Synchronous Readline permite a execução interativa (conversação com o usuário) via console.
- ▶ Permite que um programa leia os dados (por exemplo, números) para posterior utilização.
- ▶ Esses dados são provenientes da digitação pelo teclado.



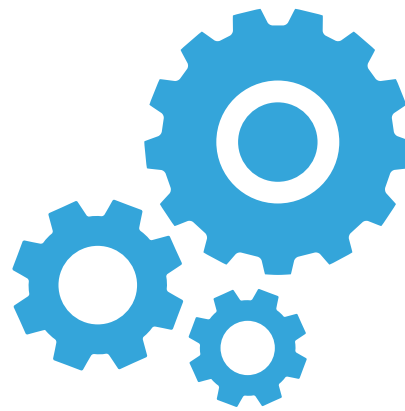
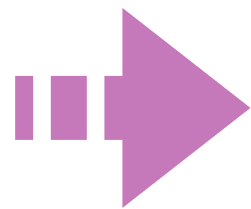
INSTRUÇÃO DE SAÍDA

ENTRADA, PROCESSAMENTO E SAÍDA



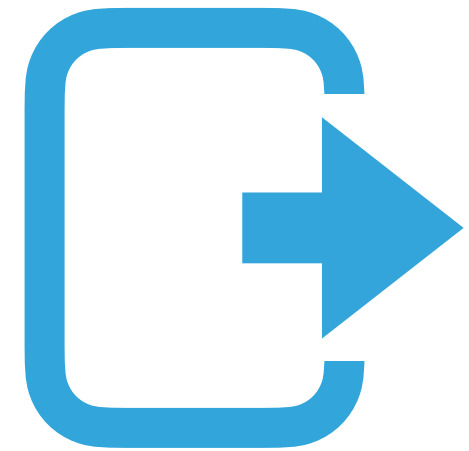
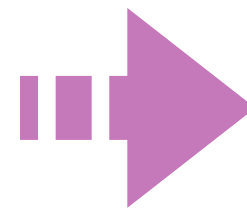
ENTRADA

O que é
fornecido?
(input)



PROCESSAMENTO

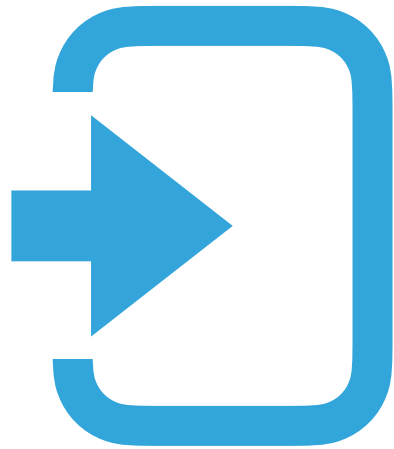
Como transformo
o que é fornecido
no que deve ser
retornado?



SAÍDA

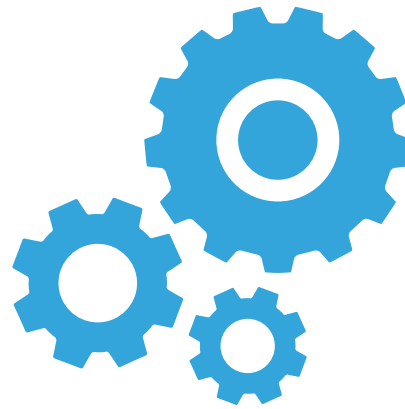
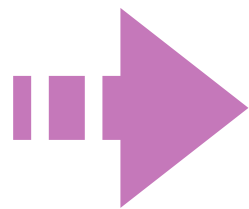
O que é
retornado?
(output)

ATÉ AGORA, ESSENCIALMENTE...



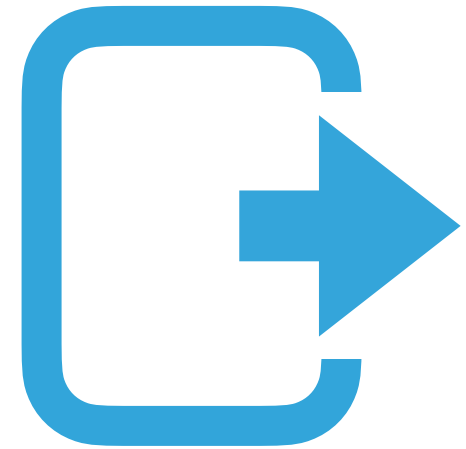
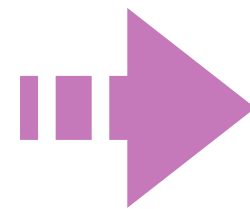
READLINE-SYNC

```
questionInt()  
questionFloat()  
question().charAt(0)  
question()
```



ATRIBUIÇÕES

Expressões
aritméticas



CONSOLE

```
log()
```


MÉTODO `console.log()`

```
let teclado = require("readline-sync");
```

```
let a: number = 0,  
    b: number = 0,  
    soma: number = 0;    // um comentário
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

MÉTODO `console.log()`

- ▶ Imprime (exibe) na saída o resultado da expressão repassada como parâmetro, seguido de uma **quebra de linha**.

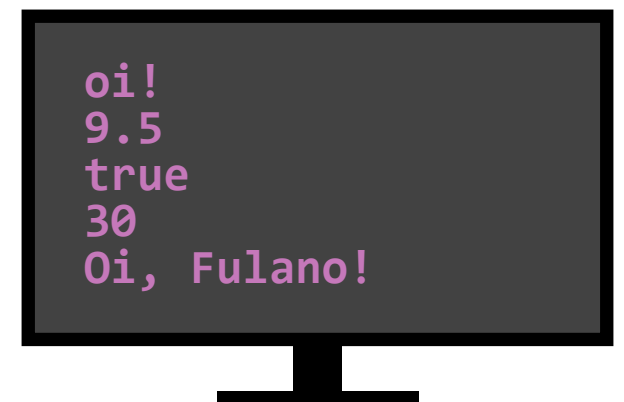
SINTAXE

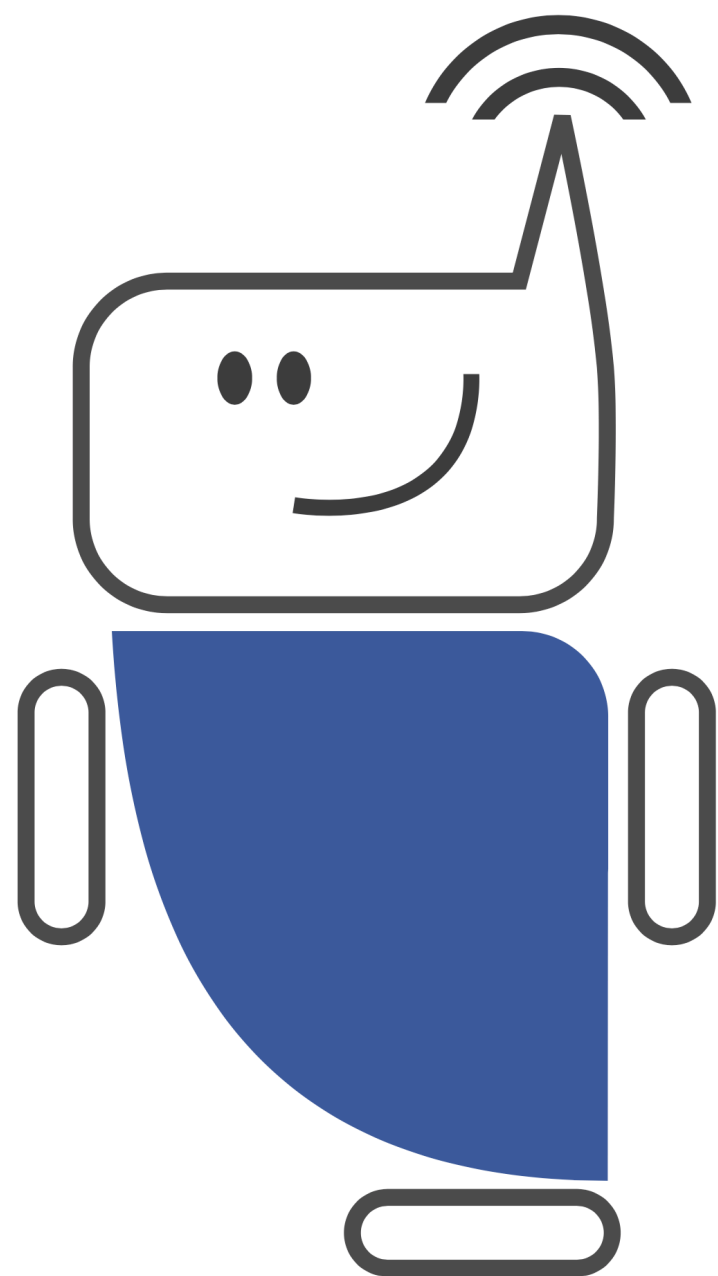
```
console.log(expressao);
```

onde `expressao` obedece a sintaxe da linguagem.

EXEMPLOS

```
console.log("oi!");  
console.log(nota);  
console.log(aprovado);  
console.log(a * 6);  
console.log("Oi,_" + nome + "!");
```





COMENTÁRIOS

COMENTÁRIOS DE LINHA ÚNICA

```
let teclado = require("readline-sync");
```

```
let a: number = 0,  
    b: number = 0,  
    soma: number = 0; // um comentário
```

```
console.log("Digite o primeiro número:");  
a = teclado.questionInt();
```

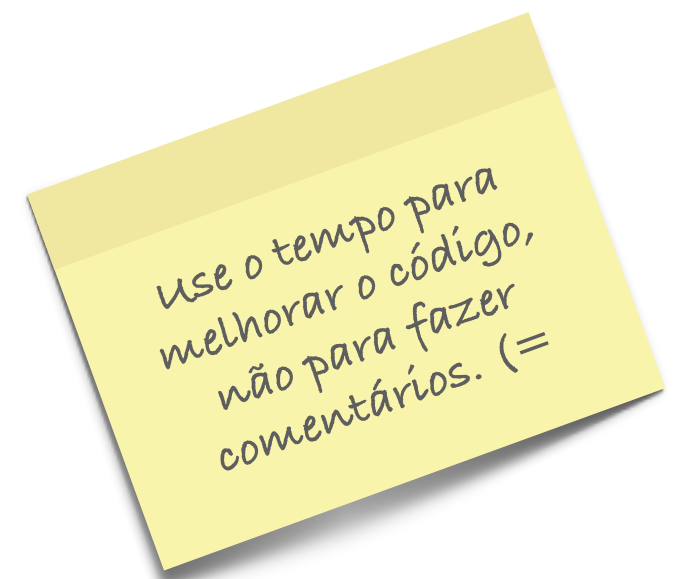
```
console.log("Digite o segundo número:");  
b = teclado.questionInt();
```

```
soma = a + b;
```

```
console.log("A soma é:");  
console.log(soma);
```

COMENTÁRIOS DE LINHA ÚNICA

- ▶ **Barras duplas** indicam o início de um comentário em TypeScript.
- ▶ Têm a finalidade de incluir **informações acessórias** ao código.
 - ▶ No início da aprendizagem é útil para observações, notas, pendências.
- ▶ Todo o restante da linha é **ignorado** pelo compilador.
- ▶ Perceba que os editores de código, inclusive, utilizam **cores mais esmaecidas** para indicar comentários (cinza, por exemplo).
- ▶ Idealmente, um código deve ser autoexplicativo!



COMENTÁRIOS DE MÚLTIPLAS LINHAS

```
/*
  Exemplo 1
  Realizar a soma de dois números inteiros dados.
*/

let teclado = require("readline-sync");

let a: number = 0,
    b: number = 0,
    soma: number = 0;    // um comentário

/*
  console.log("Digite o primeiro número:");
  a = teclado.questionInt();

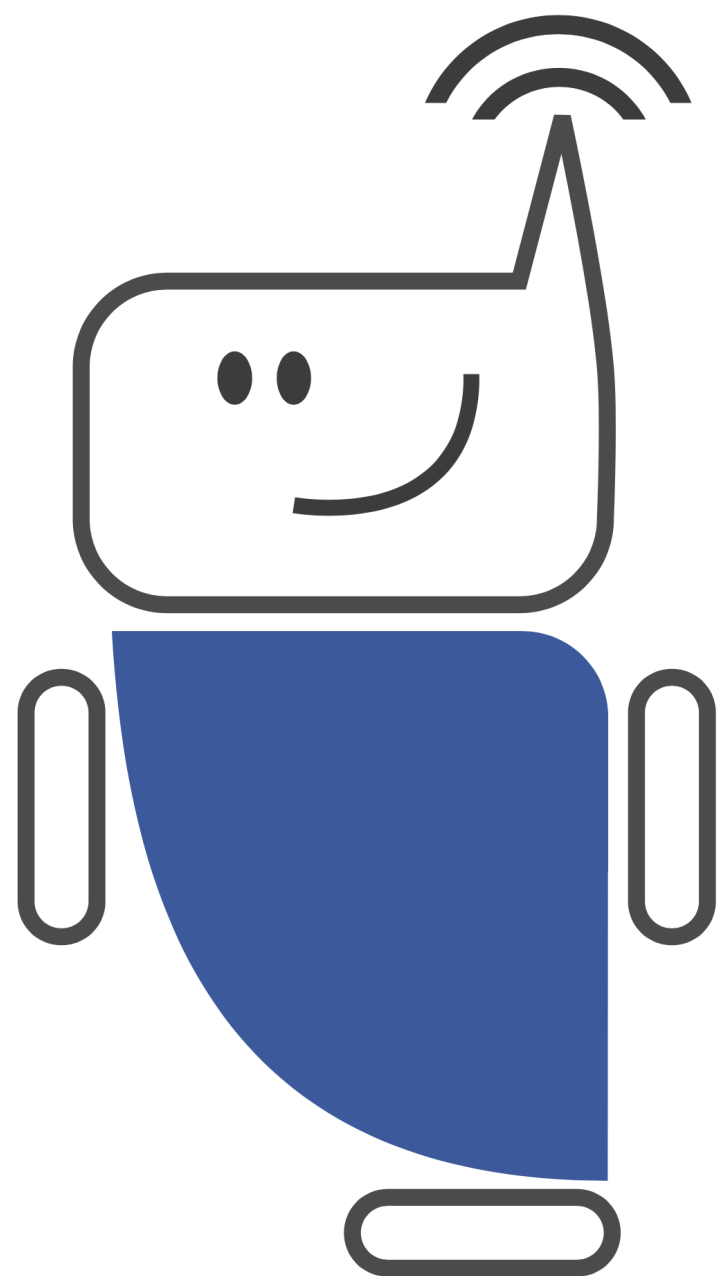
  console.log("Digite o segundo número:");
  b = teclado.questionInt();
*/

soma = a + b;

console.log("A soma é:");
console.log(soma);
```

COMENTÁRIOS DE MÚLTIPLAS LINHAS

- ▶ **Comentários com** mais de uma linha podem ser delimitados por `/*` e `*/`.
 - ▶ Ao invés de cada linha iniciar com as barras duplas.
- ▶ São úteis para **anular** trechos de código
- ▶ Os editores de código possuem **teclas de atalho** que permitem comentar / descomentar um trecho selecionado.



ADVERTÊNCIAS

POSSIBILIDADES MAIS AMPLAS

- ▶ A maioria dos elementos vistos têm possibilidades que vão muito além do que se apresentou.
- ▶ Apenas as **possibilidades mais básicas** foram mostradas neste início.
 - ▶ Não gerar sobrecarga cognitiva.
 - ▶ Cuidar com o excesso de novidades.
 - ▶ Permitir prática mais focada.
- ▶ O maior enfoque é **explorar os fundamentos de programação...**
 - ▶ ... ao invés das inúmeras possibilidades de açúcares sintáticos que a linguagem oferece.
 - ▶ Muito disso será aprendido naturalmente com o tempo na disciplina.

AINDA NÃO É PROGRAMAÇÃO ORIENTADA A OBJETOS

- ▶ Agora, o principal enfoque é o entendimento das **instruções primitivas isoladas**.
- ▶ Com isso, ainda **não praticamos**, ou se exige, muitos dos preceitos da Programação Orientada a Objetos.
- ▶ Embora TypeScript seja uma linguagem orientada a objetos, simplesmente **codificar em TypeScript não é sinônimo de programar orientado a objetos**.
- ▶ Elementos do **Paradigma Orientado a Objetos** serão adicionados à proporção do andamento das disciplinas.
- ▶ E serão reforçados na disciplina de **Programação Orientada a Objetos** (3º Período).
- ▶ Por enquanto, a ideia é favorecer uma **exploração** que proporcione o **domínio** dessas instruções primitivas.

CONHECIMENTOS IMPRESCINDÍVEIS

▶ **Pensamento Computacional e Linguagem de Programação Visual**

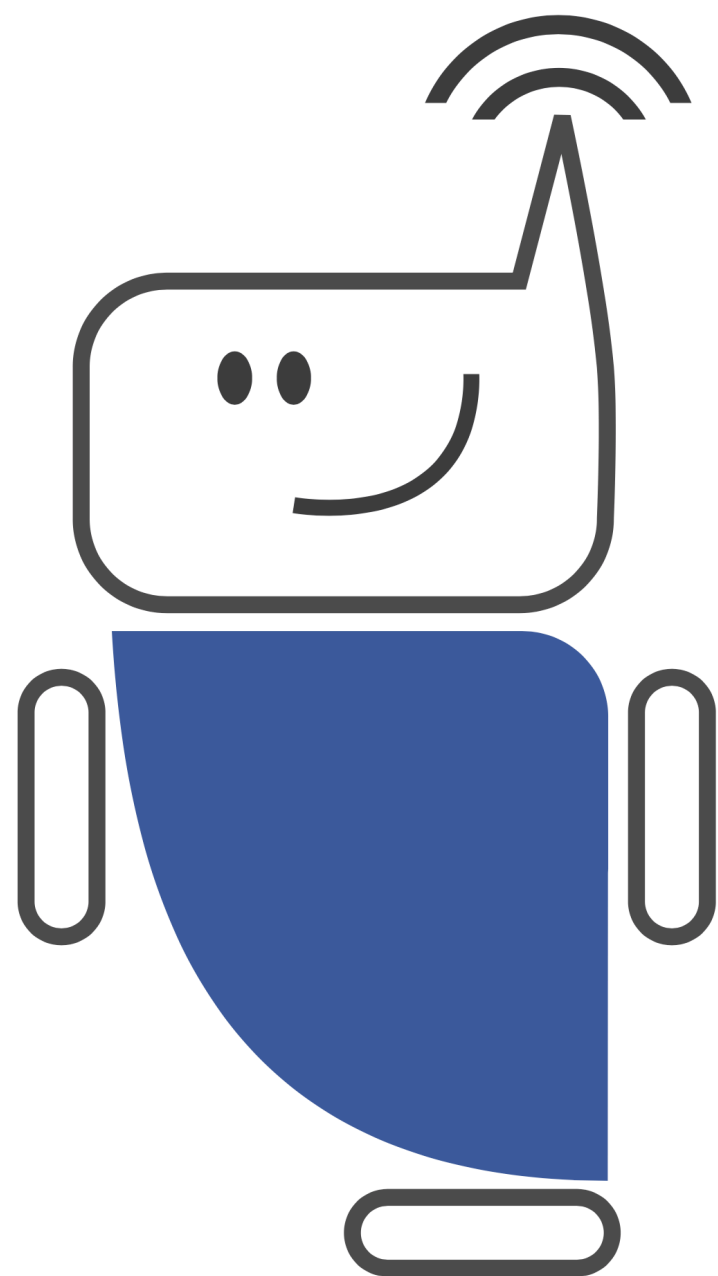
- ▶ Análise e a resolução de problemas diante dos quatro pilares do pensamento computacional.
- ▶ Equivale a ter aprendido: letras, sílabas, palavras e frases.

▶ **Fundamentos de Programação**

- ▶ Análise e a resolução de problemas computacionais por meio de uma linguagem de programação de alto nível e orientada a objetos.
- ▶ Equivale a aprender: parágrafos e redação.
- ▶ **O conhecimento anterior é fundamental.**

CONHECIMENTOS IMPRESCINDÍVEIS

**NÃO SE APRENDE A CORRER
SEM ANTES SABER ANDAR!**



REFERÊNCIAS

REFERÊNCIA

MASCHIO, Eleandro. **Introdução aos Aplicativos TypeScript**. Guarapuava: Universidade Tecnológica Federal do Paraná, 2023. 78 slides, color. Material didático da disciplina de Pensamento Computacional e Fundamentos de Programação.

CITAÇÃO COM AUTOR INCLUÍDO NO TEXTO

Maschio (2023)

CITAÇÃO COM AUTOR NÃO INCLUÍDO NO TEXTO

(MASCHIO, 2023)