Capítulo 3
Condições

Programação com JavaScript & TypeScript

Antes de começar

Instalação

```
# Instale o Node.js (nodejs.org)
$ node --version
v20.x.x

# Para TypeScript
$ npm install -g tsx
```

Executando programas

```
# JavaScript
$ node meu_programa.js

# TypeScript
$ npx tsx meu_programa.ts
```

Saída de dados

```
console.log("Olá, mundo!");
console.log("Resultado:", 2 + 3);
console.log(`A soma é ${2 + 3}`);   // template literal
```

Lendo entrada do teclado

Como ler dados do usuário no terminal (como scanf em C)

JavaScript (Node.js)

```
const rl = require("readline")
  .createInterface({
    input: process.stdin,
    output: process.stdout
  });

rl.question("Nome: ", (nome) => {
  console.log(`Olá, ${nome}!`);
  rl.close();
});
```

TypeScript (com tsx)

```
import * as readline from "readline";

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.question("Nome: ", (nome) => {
  console.log(`Olá, ${nome}!`);
  rl.close();
});
```

Dica: salve o arquivo e rode com `node entrada.js` ou `npx tsx entrada.ts`

Estrutura condicional: if...else

Permite executar blocos de código com base em uma condição

Simples

```
if (condição) {
  // comandos
}
```

if...else

```
if (condição) {
  // se verdadeiro
} else {
  // se falso
}
```

if...else if

```
if (cond1) {
  // bloco 1
} else if (cond2) {
  // bloco 2
} else {
  // bloco 3
}
```

Espaço para exemplo ao vivo

Operadores relacionais

Usados para comparar valores — retornam true ou false

Operador	Significado	Exemplo
`==`	Igual (valor)	`5 == "5"  → true`
`!=`	Diferente (valor)	`5 != "5"  → false`
`===`	Estritamente igual	`5 === "5" → false`
`!==`	Estritamente diferente	`5 !== "5" → true`
`>`	Maior que	`10 > 5    → true`
`<`	Menor que	`3 < 1     → false`
`>=`	Maior ou igual	`7 >= 7    → true`
`<=`	Menor ou igual	`4 <= 3    → false`

Operadores lógicos

&&

AND (E)

Verdadeiro quando
TODAS as condições
são verdadeiras

```
idade >= 18 && idade <= 65
```

||

OR (OU)

Verdadeiro quando
AO MENOS UMA
condição é verdadeira

```
cor == "Azul" || cor ==
"Cinza"
```

!

NOT (NÃO)

Inverte o resultado
da condição

```
!logado  // se NÃO logado
```

Espaço para exemplo ao vivo

Tabela verdade

&& (AND)

p	q	p && q
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**F**

|| (OR)

p	q	p || q
V	V	**V**
V	F	**V**
F	V	**V**
F	F	**F**

! (NOT)

p	!p
V	**F**
F	**V**

Espaço para exemplo ao vivo — combinando operadores

Switch...case

Ideal quando temos muitas alternativas baseadas em uma variável

Sintaxe

```
switch (variavel) {
  case "valor1":
    // comandos
    break;
  case "valor2":
  case "valor3":
    // mesmo bloco
    break;
  default:
    // caso padrão
}
```

Pontos-chave

`break`

Encerra o bloco do case.
Sem ele, executa o próximo case!

`default`

Executado quando nenhum case corresponde

`Agrupar`

Vários cases sem break compartilham o mesmo bloco

Operador ternário

Forma abreviada do if...else para atribuição de valores

```
const resultado = condição ? valorSeTrue : valorSeFalse;
```

Com ternário

```
const cat = idade >= 18 ? "Adulto" :
"Juvenil";
```

Com if...else

```
let cat;
if (idade >= 18) {
  cat = "Adulto";
} else {
  cat = "Juvenil";
}
```

Espaço para exemplo ao vivo

Resumo — Capítulo 3

`if...else`	Estrutura condicional clássica
`Operadores relacionais`	==, !=, ===, !==, >, <, >=, <=
`Operadores lógicos`	`&& || !`
`switch...case`	Múltiplas alternativas para uma variável
`Operador ternário`	condição ? valorV : valorF