

# Variáveis e Lógica Booleana

FUTURE4

# Por que estudar variáveis e lógica?

- Variáveis são fundamentais para a estruturação e legibilidade dos programas. Reduz a repetição e facilita a manutenção.
- Uma boa compreensão da lógica booleana e lógica aplicada à programação é fundamental para a formação de um bom desenvolvedor(a).
- Os conhecimentos vistos aqui se aplicam para a maioria das linguagens de programação e é através dessa conhecimento ferramental genérico que conseguimos migrar entre linguagens se adaptando somente as especificidades e mantendo a base lógica.

- Já que vimos a responsabilidade de cada frente no desenvolvimento de sites, vamos agora nos aprofundar em cada uma das frentes, e para entender a parte lógica das 3 linguagens/frentes a mais apropriada para isso é o JS.

- **Variáveis são como gavetas** onde podemos guardar valores, por mais que você mude o conteúdo da gaveta, ela continua como um espaço onde você pode guardar qualquer coisa.

# Operações de Atribuição

# Estrutura da atribuição



// É composta de 4 partes

```
const cor = "azul"
```

//	const	cor	=	"azul"
//	^	^	^	^
//	declaração	nome	atribuição	valor

Quick tip:



```
let variavelLegal
```



# Operações de Declaração

# Const

Usamos **const** para valores que assumimos que não mudarão



```
const corDoCeu = "azul"; // Algo que não muda
```

Outro exemplo de algo que não muda:



```
// um const não pode ser declarado sem valor explícito  
// então sua declaração e atribuição ser feitas ao mesmo tempo.  
const meuNome = "José Pereira";
```

# Erros mais comuns ao usar const!

Não pode declarar uma const **sem defini-la**.



```
const meuNome;|
```

Não pode mudar const **depois** de declarada.



```
const meuNome = "José Pereira";  
meuNome = "José Arruda";
```



Usamos **let** para valores que sabemos que podem mudar



```
let diaDaSemanaAgora = "Segunda-feira"; // Algo que pode mudar
diaDaSemanaAgora = "Terça-feira"; // mesma variável, outro valor
```

Outro exemplo de algo  
que **pode mudar**:



```
let horaAtual = 10;
horaAtual = 11;
horaAtual = 12;
horaAtual = 13;
```

Declaração e definição  
separadamente:



```
// Declaramos primeiro


let numeroDeVoltasTotal;

// Deram uma volta!
// Atribuímos quando precisarmos, e quantas vezes quisermos.

numeroDeVoltasTotal = 1;

// Deram outra volta!

numeroDeVoltasTotal = 2;
```



```
const A = 1 + 1
const B = "1" + "1"
const C = "1" + 1|
```

**Quais são os valores de A, B e C respectivamente?**

- a) 2, "11", "11"
- b) 11, "2", "11"
- c) "2", "1", 11

**Pausa pra respirar :)**

# Tipos de valores em JS e suas características

**String** (Exemplos ⇒ "Júpiter", "Saturno", "a", "42", "")

Propriedade de concatenação

- Como juntamos duas strings separadas?



```
const primeiroNome = "Yuri"
const ultimoNome = "Gagarin"

const nomeCompleto = primeiroNome + ultimoNome
imprimir(nomeCompleto)
// Resultado: "YuriGagarin"
```

**Number** (Exemplos  $\Rightarrow$  42, 3,14, 0, -1)

Propriedades algébricas:

```
// Soma e Subtração
```

```
const contaDeAgua = 50.30;  
const contaDeLuz = 20.32;  
const contaDeInternet = 80.00;  
const desconto = 20.00;
```

```
const total = contaDeAgua + contaDeLuz + contaDeInternet - desconto;  
imprimir(total)  
// Resultado: 150.62|
```

# Number (Exemplos $\Rightarrow$ 42, 3,14, 0, -1)

## Propriedades algébricas:

```
// Multiplicação e Divisão
```

```
const valorDaPizza = 120;  
const percentagemDaGorjeta = 1.10;  
const pessoas = 5;
```

```
const contaPorPessoa = (valorDaPizza * percentagemDaGorjeta) / pessoas;  
imprimir(contaPorPessoa);  
// Resultado: 24.00
```

```
// Módulo é o nome do operador que usamos para obter o resto da divisão  
const cinco = 5;  
const restoPorDois = 5 % 2;  
imprimir(restoPorDois)  
// Resultado: 1  
// 5 dividido 2 é igual a 2 e sobra 1
```

# Boolean (Exemplos $\Rightarrow$ true, false)

Negação, utilizado para inverter o valor do booleano



```
const estaDeNoite = true
const estaDeDia = !estaDeNoite // Ou seja, sempre que estiver de noite 'estaDeDia' será falso e vice versa
```

And: Combinar duas preposições



```
// AND é representado em JS por "&&"
true && true // true
true && false // false
false && true // false
false && false // false
```

Or: Pelo menos um verdadeiro




```
// em JS OR é representado por || (pipe)
true || true // true
true || false // true
false || true // true
false || false // false
```



**Array** (Exemplos  $\Rightarrow$  [1,2,3,3,2,1], ["a","b","c"], ["Apolo", 1969, true, true])

Um Array é uma estrutura de dados que nos permite guardar valores sequencialmente, esses valores podem ser de qualquer tipo e podem repetir.



```
// Arrays contam o seu elementos começando do 0
const meses = ['Janeiro', 'Fevereiro', 'Março', 'Abril'];

imprime(meses[0]) // Resultado: 'Janeiro'
imprime(meses[3]) // Resultado: 'Abril'
imprime(meses[2]) // Resultado: 'Março'
```

# Object

## Exemplos:

```
{  
  nome: "Neil Armstrong",  
  especie: "humana",  
}
```

```
{  
  nome: "Laika",  
  especie: "canina",  
  idade: 3,  
}
```

## Declarando e acessando valores:

```
const astronauta = {  
  nome: "Neil Armstrong",  
  especie: "humana",  
  idade: 82  
}  
  
// as duas formas a seguir são válidas para acessar propriedades de objetos.  
imprime(astronauta.nome) || imprime(astronauta['nome'])  
// Resultado: "Neil Armstrong"
```

# Function

Uma função é uma estrutura que nos permite facilitar a repetição de uma atividade e organizar grandes bases de código de forma mais estruturada

```
function bomDia() {  
    return "Bom dia!!"  
}
```

```
function checaParidade(numero) {  
    const numeroEPar = numero % 2 === 0;  
    return numeroEPar;  
}  
  
checaParidade(2); // Resultado: true  
checaParidade(655); // Resultado: false
```

```
function cumprimenta(nome) {  
    return "Seja bem-vindo(a) " + nome + "!"  
}  
  
cumprimenta("Valentina Tereshkova"); // Resultado: "Seja bem-vindo(a) Valentina Tereshkova!"  
cumprimenta("Bill Gates"); // Resultado: "Seja bem-vindo(a) Bill Gates!"  
cumprimenta("Steve Jobs"); // Resultado: "Seja bem-vindo(a) Steve Jobs!"
```

**typeof** é um comando que podemos usar para checar o tipo do valor de variáveis e o uso é o seguinte:



```
typeof 4 // Resultado: "number"  
typeof "oi" // Resultado: "string"  
typeof true // Resultado: "boolean"  
typeof undefined // Resultado: "undefined"
```

# Tipos de valores especiais

- **Undefined:** Representa a falta de valor de uma variável, ou seja, a indefinição de valor
- **Null:** Também representar a falta de valor de uma variável, a única diferença é que, para que uma variável tenha o valor de null precisamos explicitamente associa-la com esse valor.

```
let minhaVariavel;  
typeof minhaVariavel // Resultado: "undefined"
```

```
let nomeDeDesconhecido = null;  
const idadeDeDesconhecido = null;  
  
imprime(nomeDeDesconhecido); // Resultado: null  
imprime(idadeDeDesconhecido); // Resultado: null
```

# Introdução ao conceito de escopos e blocos

- Blocos são criados todas as vezes que cercamos um trecho de código com colchetes `{ }`
- As variáveis que declaramos dentro de blocos só existem nos blocos que foram criadas:

```
{  
  let nome = "Juliana Ferreira";  
  imprime(nome); // Resultado: "Juliana Ferreira"  
}  
  
imprime(nome); // Resultado: undefined
```

# Blocos aninhados

- Blocos aninhados (que estão dentro de outros blocos) têm acesso às variáveis de blocos superiores, exemplo:

```
{ // Primeiro Bloco
  let nome = "Juliana Ferreira";
  imprime(nome); // Resultado: "Juliana Ferreira"

  { // Segundo Bloco
    let idade = 10;
    imprime(nome); // Resultado: "Juliana Ferreira" <- conseguimos acessar por estarmos em um bloco aninhado
    imprime(idade) // Resultado: 10
  }

  imprime(idade) // Resultado: undefined
}

imprime(nome); // Resultado: undefined
imprime(idade); // Resultado: undefined
```

**Pausa pra respirar :)**



# Coding Together

# Exercícios:

Escreva um programa que recebe manualmente a idade atual da pessoa e retorna o ano em que ela nasceu.

- Primeira mudança, introdução do prompt()
- Além do ano, agora o programa precisa retornar também:
  - quantos dias (aproximados) de idade a pessoa tem
  - quantas horas (aproximadas) de idade a pessoa tem
  - quantos segundos (aproximadas) de idade a pessoa tem

# Exercícios:

## Cálculo de IMC

- O IMC de uma pessoa pode ser calculado com a seguinte fórmula:  $\text{peso} \div \text{altura} \times \text{altura}$
- Faça um programa que recebe o peso e a altura do usuário e retorna seu IMC

# Revisão

- **Valores e seus tipos**
- **Operações**
  - **Algébricas**
    - **Matemáticas**
    - **Booleanas**
  - **Concatenação de String**
- **Coerção**
  - **Explícita**
  - **Implícita**

Obrigado!

# Dúvidas?