
# Lógica de Programação - Aula 6

Santander Coders 2024

# Laço de Repetição "for"

O for é uma estrutura de controle de fluxo em programação que é utilizada para criar loops ou laços de repetição. Ele é especialmente útil quando você precisa executar um bloco de código várias vezes de maneira controlada.

```
for (inicialização; condição; expressão de incremento/decremento) {
    // Código a ser executado durante cada iteração
}
```

**Inicialização**: É onde você define a variável de controle e atribui um valor inicial a ela. Geralmente, utiliza-se **let i = 0** para iniciar um contador.
**Condição**: É a expressão booleana que determina se o bloco de código dentro do **for** deve ser executado. Enquanto essa condição for verdadeira, o loop continuará.
**Expressão de Incremento/Decremento**: Determina como a variável de controle é modificada a cada iteração. i++ significa "incrementar i por 1" e i-- significa "decrementar i por 1".

```
// Exemplo: Imprimir os números de 0 a 4 no console
for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

O for-in é uma estrutura de controle em JavaScript projetada especificamente para iterar sobre as propriedades enumeráveis de um objeto. Ele fornece uma maneira conveniente de acessar e manipular os pares chave-valor de um objeto.

```
for (let propriedade in objeto) {
    // Código a ser executado para cada propriedade do objeto
}
```

**propriedade:** Uma variável que receberá o nome da propriedade a cada iteração.
**objeto:** O objeto sobre o qual você está iterando.

```
const carro = {
    marca: 'Toyota',
    modelo: 'Corolla',
    ano: 2020
};

for (let propriedade in carro) {
    console.log(propriedade + ': ' + carro[propriedade]);
}
```

ada

O for-of é uma estrutura de controle introduzida no ECMAScript 6 (ES6) que simplifica a iteração sobre elementos de objetos iteráveis, como arrays, strings, mapas, sets, entre outros. Ao contrário do for-in, que itera sobre as propriedades de um objeto, o for-of percorre os valores diretamente.

```
for (let elemento of iteravel) {
    // Código a ser executado para cada elemento do iterável
}
```

**elemento:** Uma variável que receberá o valor do elemento a cada iteração.
**iteravel:** O objeto iterável sobre o qual você está iterando.

```
const frutas = ['Maçã', 'Banana', 'Morango'];

for (let fruta of frutas) {
    console.log(fruta);
}
```

# Bora Praticar

## Desafio 1

Utilizando um loop **for**, calcule a soma dos números de 1 a 10 e imprima o resultado.

(1+2=3+3=6+4=10+5=15)

## Desafio 2

Crie um objeto representando uma pessoa com **nome, idade e cidade**.
Utilize um loop **for-in** para imprimir no console todas as propriedades e os valores do objeto

## Desafio 3

Crie um **array** com pelo menos 5 nomes de frutas.
Utilize um loop **for-of** para imprimir no console cada nome de fruta.

## Desafio 4

Escolha um número e imprima sua tabuada de 1 a 10 utilizando um loop **for**.

Saída: N X 1 = 10

## Desafio 5

Crie um objeto com valores numéricos e utilize um loop **for-in** para calcular e imprimir a soma desses valores.

**{ valor1: 10, valor2: 20, valor3: 30, valor4: 40 }**

## Desafio 6

Faça um mecanismo de **busca** dentro de um array usando **for**.

[1,2,3,4,5,6,7,8,9,10]

Achei o número 7

# Desafio 7

Ache a lógica: **1 1 2 3 5 8 13**
Agora, sua tarefa é criar um programa que recebe um número inteiro N como entrada e retorna os primeiros N termos da sequência.

Exemplo:
Entrada: 6 Saída: [1 , 1, 2, 3, 5, 8]
Entrada: 14 Saída: [1, 1, 2, 3, 5, 8, 13, …, 377]

# Desafio 8

Crie um programa que pede ao usuário para inserir uma palavra e conta quantas vogais (a, e, i, o, u) ela contém. Utilize um loop **for** e estruturas condicionais **if** para realizar a contagem.

(Não precisa considerar acentos. Ex.: aviao)

# Desafio 9

Criar um algoritmo de ordenação de arrays;

novoArray = [1, 7, 2, 8, 3, 4, 5, 0, 9]

**09.01 -** Método que faz isso (Achar o método que ordena os arrays)

## Super Desafio

**10 -** Fazer um simulador de rolagem de dados, que receba como input N dados de **6 lados** e mostre as rolagens individuais e a soma dos valores

Entradas:

Quantidade de dados: 2

```
"Rolagens Individuais:"
"Dado 1: 3"
"Dado 2: 5"
"Soma dos Valores: 8"
```

## Super Desafio +

**11 -** Fazer um simulador de rolagem de dados, que receba como input N dados e **N lados** e mostre as rolagens individuais e a soma dos valores?

Entradas:

Quantidade de dados: 2
Quantidade de lados: 9

```
"Rolagens Individuais:"
"Dado 1: 3"
"Dado 2: 5"
"Soma dos Valores: 8"
```

## Super Desafio ++

**12 -** Fazer um simulador de rolagem de dados, que receba como input N dados e **N lados** e **quantidade de tentativas** e mostre as rolagens individuais e a soma dos valores

Entradas:

Quantidade de dados: 3
Quantidade de lados: 9
Quantidade de tentativas: 3

```
"
Tentativa 1:"
"Rolagens Individuais:"
"Dado 1: 6"
"Dado 2: 3"
"Dado 3: 4"
"Soma dos Valores: 13"
"
Tentativa 2:"
"Rolagens Individuais:"
"Dado 1: 8"
"Dado 2: 4"
"Dado 3: 6"
"Soma dos Valores: 18"
"
Tentativa 3:"
```

# Proposta de Projeto 1

- **O que?**

Desenvolver, utilizando os conceitos abordados ao longo do módulo, uma aplicação de **lista de tarefas** (ToDo List).

- **Requisitos**

Dentre as funcionalidades, espera-se que seja possível:

- Adicionar uma tarefa (**C**reate)
- Listar todas as tarefas salvas (**R**ead)
- Editar uma tarefa salva (**U**pdate)
- Remover uma tarefa salva (**D**elete)
- Obter uma tarefa, através de um parâmetro (id)

- **Observações**

Não haverá a persistência das tarefas em banco de dados. Para isso, podem utilizar um array para armazenar a lista das tarefas cadastradas.

# Proposta de Projeto 2

- **O que?**

**Jogo da Velha**: implemente o clássico jogo da velha em JavaScript.

- **Requisitos**

Deve utilizar **funções**. Permita que **dois jogadores** se alternam para fazer suas jogadas e **determine o vencedor**.

- **Observações**

Não haverá a persistência de dados em banco. Para isso, podem utilizar arrays para armazenar o tabuleiro. Armazene também o nome dos jogadores.