



# switch ... case

Nos casos em que existem muitas condições nas quais testas a mesma variável pode ser mais proveitoso utilizar um switch ... case

A sintaxe desta estrutura de controlo de fluxo é:

```
switch (/* condição */) {  
    case valor_1:  
        // código a executar caso a variável tenha o valor_1  
        break;  
    case valor_2:  
        // código a executar caso a variável tenha o valor_2  
        break;  
    // ...  
    default:  
        // código a executar independentemente do valor  
        // excepto se houver breaks em todos os case  
}
```



Switch Case

# Vamos praticar



# Exercício:

- Implementa um código com `mesesDoAno` que recebe um número que representa um mês do ano, e retorna o mês correspondente.

Exemplo:

```
mesesDoAno(1) // Janeiro  
mesesDoAno(10) // Outubro
```

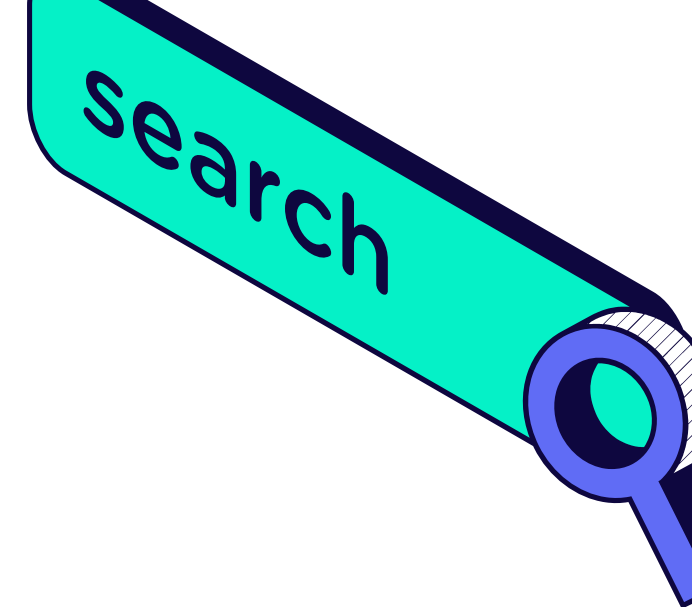
# Exercício:

1. Crie um código para os dias da semana em que um número de 1 a 7 como argumento e retorna o nome do dia correspondente.

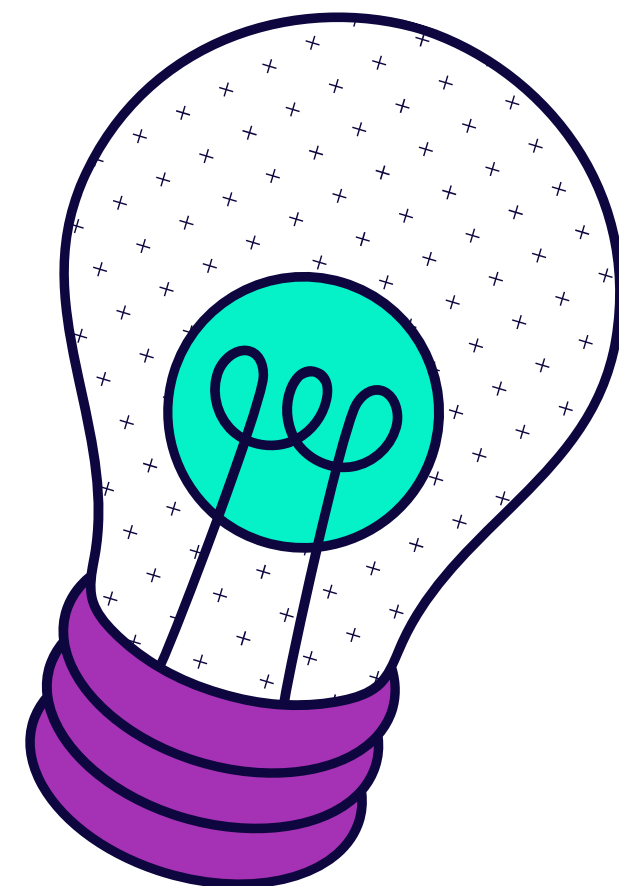
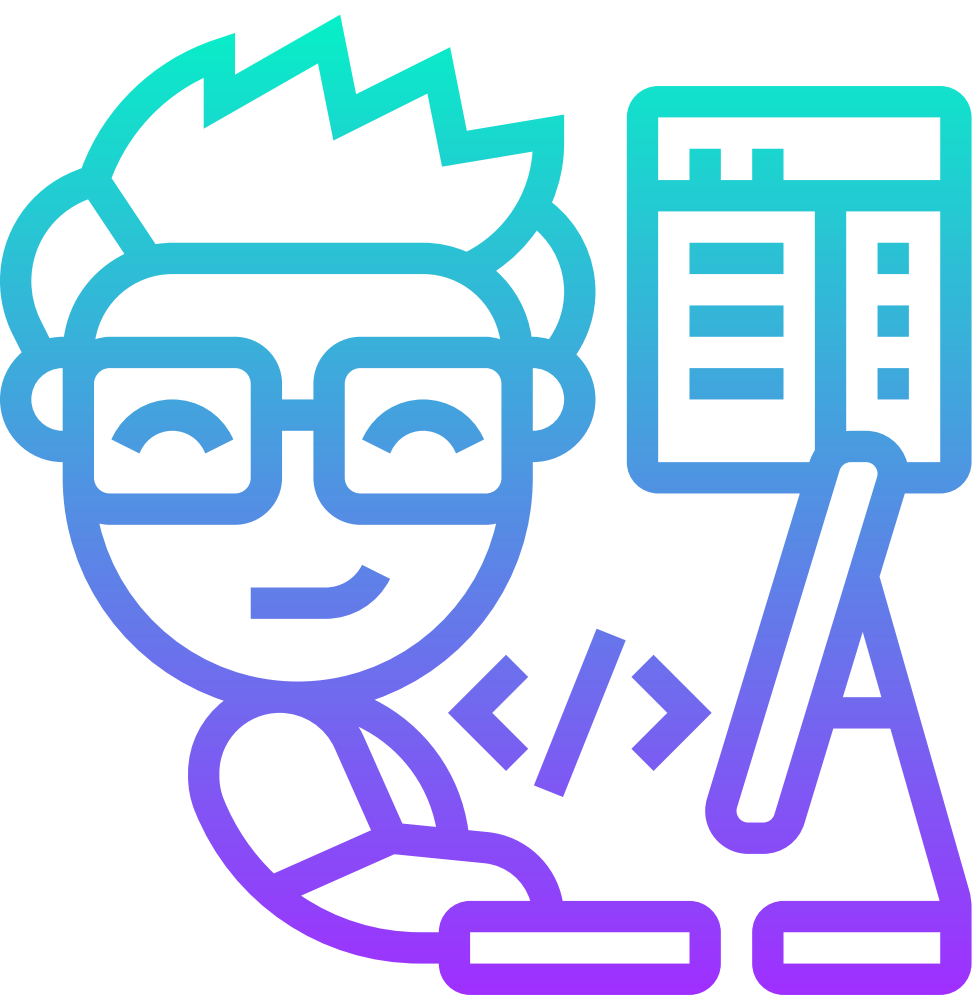
Utilize uma estrutura **switch-case** para implementar essa lógica.

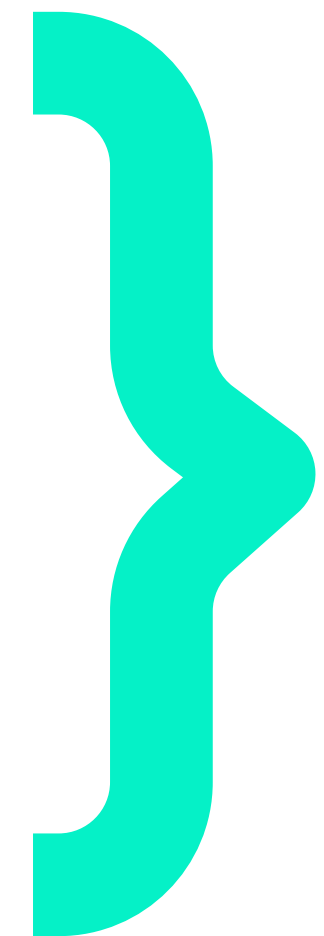
Exemplo:

```
console.log(diaDaSemana(1)); // Deve exibir "Domingo"  
console.log(diaDaSemana(3)); // Deve exibir "Terça-feira"  
console.log(diaDaSemana(8)); // Deve exibir "Dia inválido"
```



# QUIZ!!!





**CICLOS**



# Ciclos

Muitos algoritmos exigem que uma determinada **instrução seja executada um determinado número de vezes**.

Os "**ciclos**" referem-se a estruturas de controlo que permitem a repetição de um conjunto de instruções várias vezes.

# } while

O ciclo "**while**" é usado quando você deseja **repetir um conjunto** de instruções enquanto uma **condição específica for verdadeira**.

```
let i = 0;           // inicialização da variável  
                    // utilizada para a condição  
while (i < 10) { // Condição de continuação  
    // Instruções a executar enquanto i for menor que 10  
    i++;           // Incrementar i em 1  
}
```



Ciclo WHILE



# } for

O ciclo "**for**" é usado quando você sabe exatamente quantas vezes deseja repetir um conjunto de instruções.

```
for (inicialização; condição; incremento) {  
    // Código a ser repetido enquanto a condição for verdadeira  
}
```

```
// Repara que todo o controle do ciclo  
// está na primeira linha  
for (let i = 0; i < 10; i++) {  
    // Instruções a executar enquanto i for menor que 10  
}
```



Ciclo FOR

# } for

**Enunciado: Tendo um array com números inteiros, descubra o valor da soma de todos os números.**

```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];
let soma = 0;

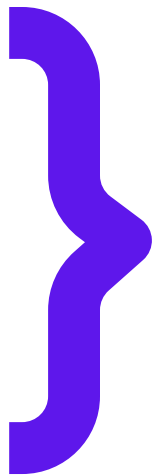
      expressão      condição      incrementação
for( let i = 0; i < arrayInteiros.length; i++ ) {

    soma += arrayInteiros[ i ];

}
```

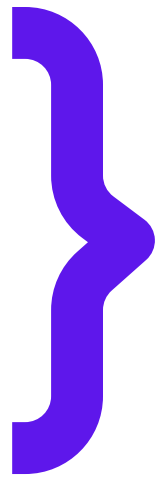


Ciclo FOR



# for

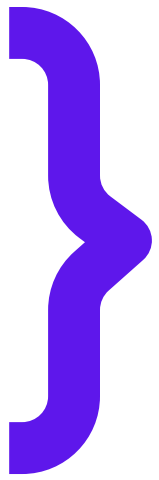
```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];  
let soma = 0;  
  
    // i < 9  
for( let i = 0; i < arrayInteiros.length; i++ ) {  
  
    soma += arrayInteiros[i];  
  
}
```



# for

```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];  
let soma = 0;  
  
for( let i = 0; i < arrayInteiros.length; i++ ) {  
  // soma += arrayInteiros[i];  
  0 += 1;  
}
```

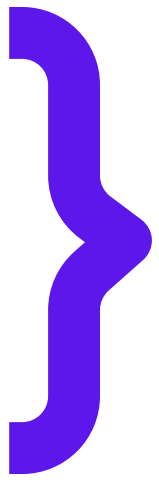
A diagram illustrating the execution of a for loop. A red arrow points down to the first element '1' in the array [1, 39, 2392, 92, 163, 2, 62, 27, 29]. Two red curved arrows above the for loop indicate the flow of the loop: one from 'i = 0' to 'i < arrayInteiros.length' and another from 'i < arrayInteiros.length' to 'i++'. A third red curved arrow below the loop body points from '0 += 1;' back to the condition 'i < arrayInteiros.length'.



# for

```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];  
let soma = 0;  
  
for( let i = 1; i < arrayInteiros.length; i++ ) {  
  // soma += arrayInteiros[1];  
  1 += 39;  
  
}
```

A diagram illustrating the execution of a for loop. A red arrow points down to the first element '1' in the array initialization. Two red curved arrows show the flow of the loop: one from the condition 'i < arrayInteiros.length' to the increment 'i++', and another from 'i++' back to the condition. A third red curved arrow points from the increment 'i++' to the first element '1' in the body statement '1 += 39;'. The code is color-coded: 'let' is purple, 'arrayInteiros' is orange, '1' is yellow, '39' is orange, '2392' is yellow, '92' is orange, '163' is yellow, '2' is orange, '62' is yellow, '27' is orange, '29' is yellow, 'soma' is orange, '0' is yellow, 'for' is purple, 'let' is purple, 'i' is yellow, '1' is yellow, 'i <' is blue, 'arrayInteiros.length' is orange, 'i++' is yellow, '{' is yellow, '//' is green, 'soma += arrayInteiros[1];' is green, '1 += 39;' is yellow, and '}' is yellow.



# for

```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];  
let soma = 0;  
  
for( let i = 2; i < arrayInteiros.length; i++ ) {  
    // soma += arrayInteiros[2];  
    40 += 2392;  
}
```

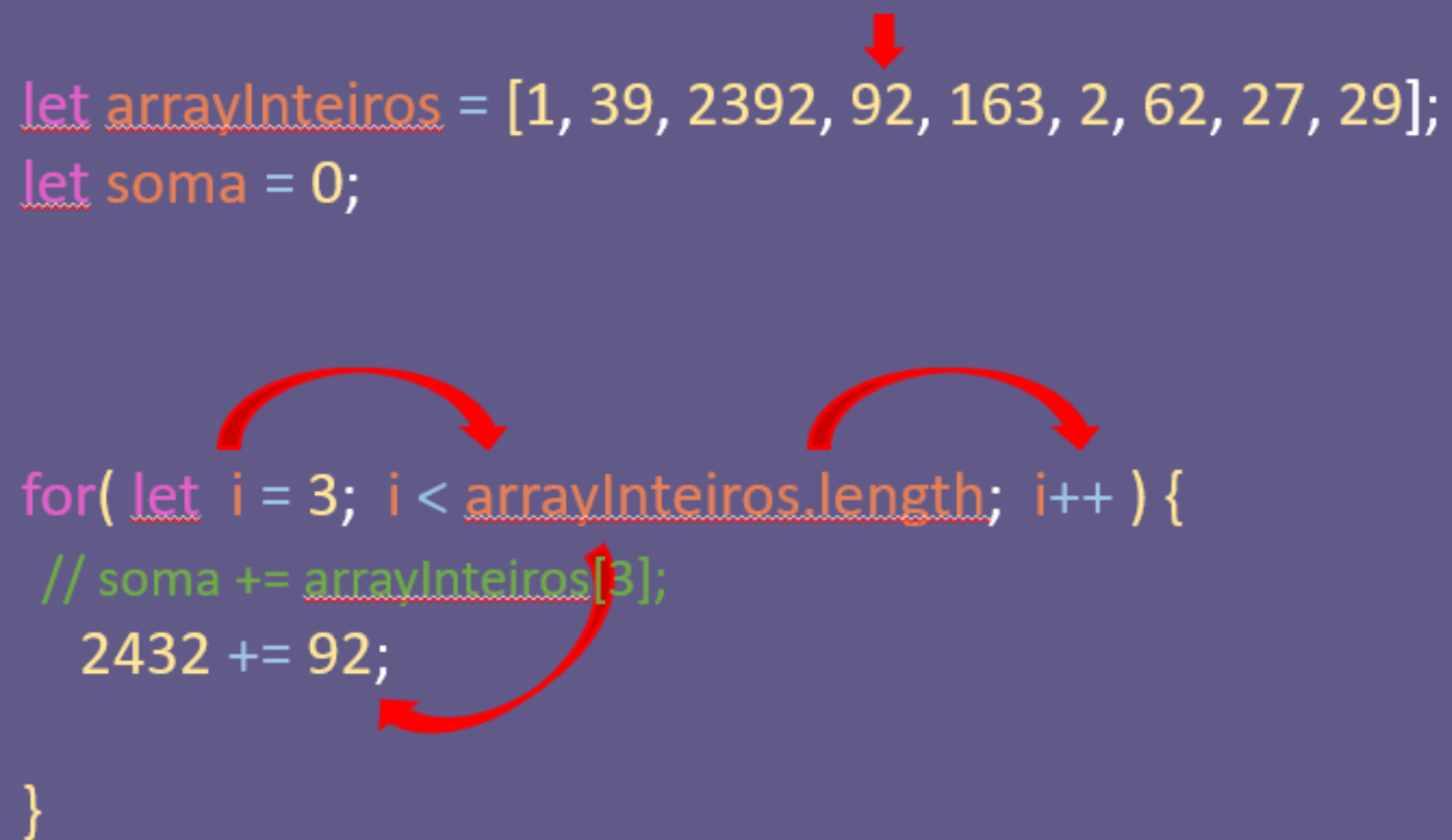
A diagram illustrating the execution of a for loop. A red arrow points to the value 2392 in the array initialization. Two red curved arrows show the loop iteration: one from 'i = 2' to 'i < arrayInteiros.length' and another from 'i++' back to 'i = 2'. A third red curved arrow points from the comment '// soma += arrayInteiros[2];' to the calculation '40 += 2392;'.

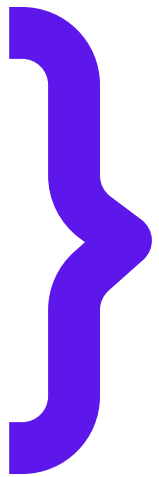
}

for

```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];
let soma = 0;

for( let i = 3; i < arrayInteiros.length; i++ ) {
  // soma += arrayInteiros[3];
  2432 += 92;
}
```



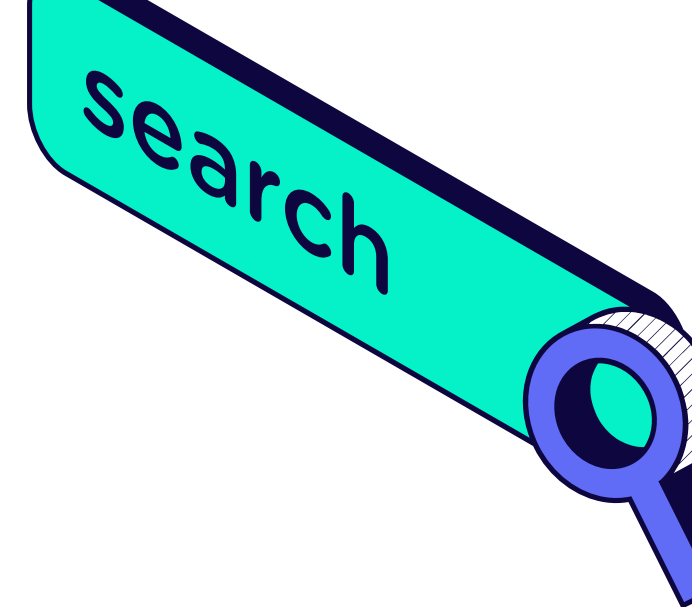


# for

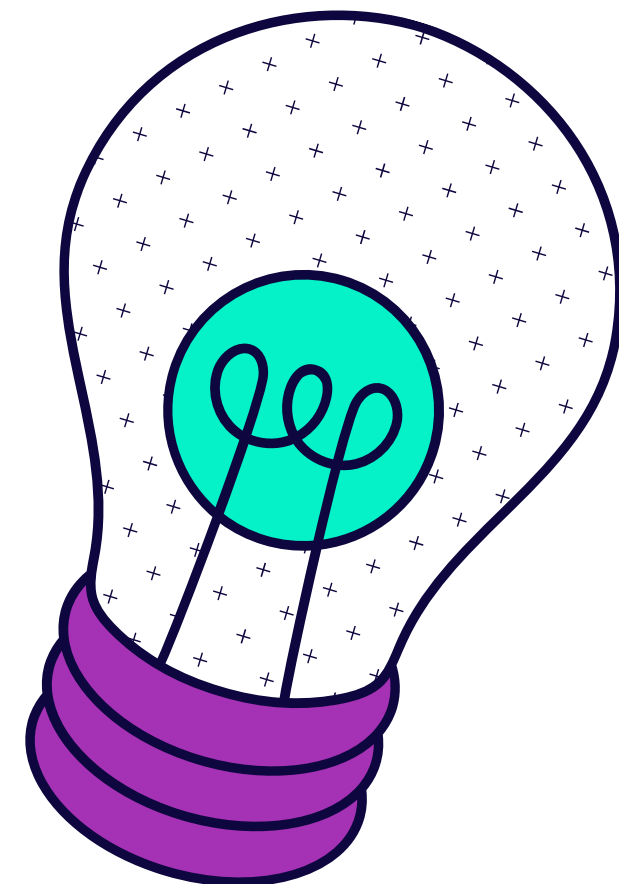
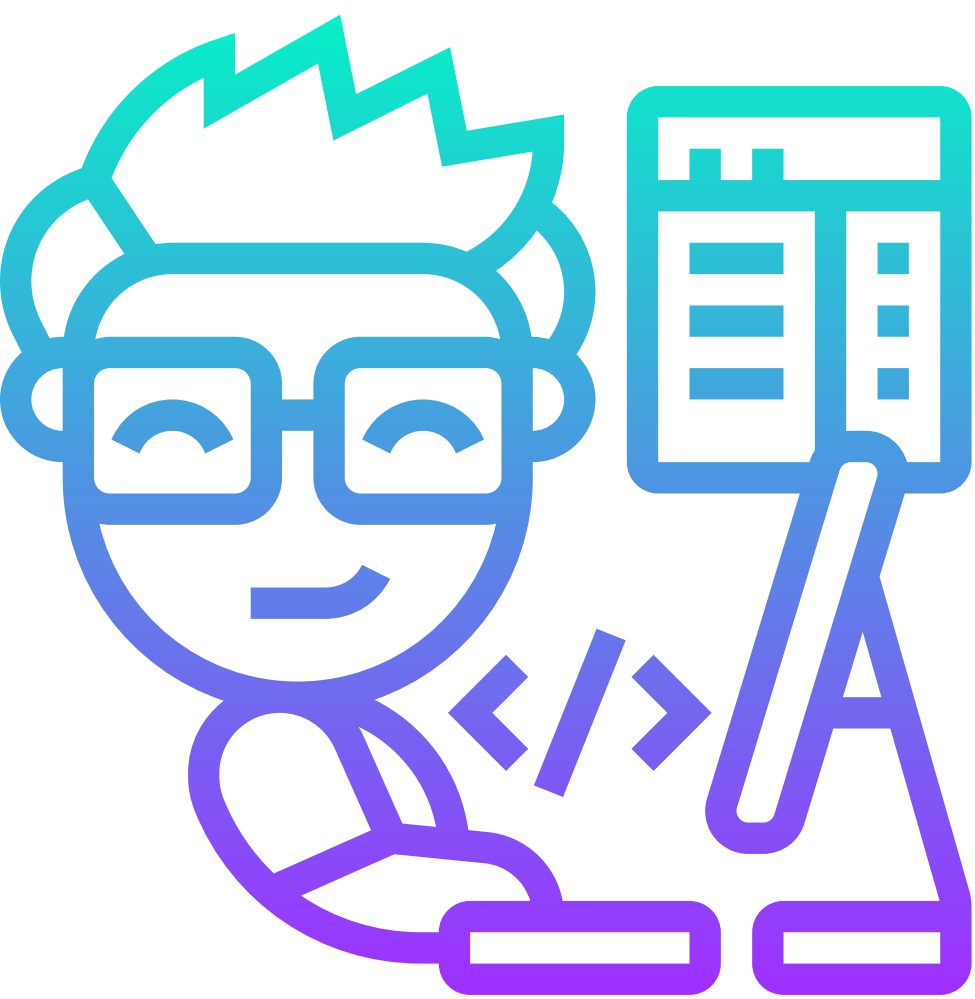
```
let arrayInteiros = [1, 39, 2392, 92, 163, 2, 62, 27, 29];  
let soma = 0;  
  
for( let i = 4; i < arrayInteiros.length; i++ ) {  
  // soma += arrayInteiros[4];  
  2524 += 163;  
}
```

A diagram illustrating the execution of a for loop. A red arrow points from the value 163 in the array to the variable i in the loop condition. Two red curved arrows show the increment of i from 4 to 5 and then to 6, corresponding to the values 163 and 2 in the array.





# QUIZ!!!



# Vamos praticar



# Exercício:

1. Cria um código que recebe um número inteiro positivo e, em seguida, utiliza um **while loop** para contar de 1 até o número fornecido.
2. A função deve imprimir cada número no console à medida que conta.

Por exemplo, se a função for chamada com o argumento 5, a saída esperada no console seria:

```
1
2
3
4
5
```

# Exercício:

1. Repita o código que recebe um número inteiro positivo como argumento, só que utiliza um **loop for** para contar de 1 até o número fornecido.
2. Deve imprimir cada número no console à medida que conta.

Por exemplo, se a função for chamada com o argumento 5, a saída esperada no console seria:

```
1  
2  
3  
4  
5
```