

O que são laços de repetição?

Laços são estruturas que permitem **executar um bloco de código repetidamente, enquanto uma condição for verdadeira.**

Laços de repetição em JavaScript:

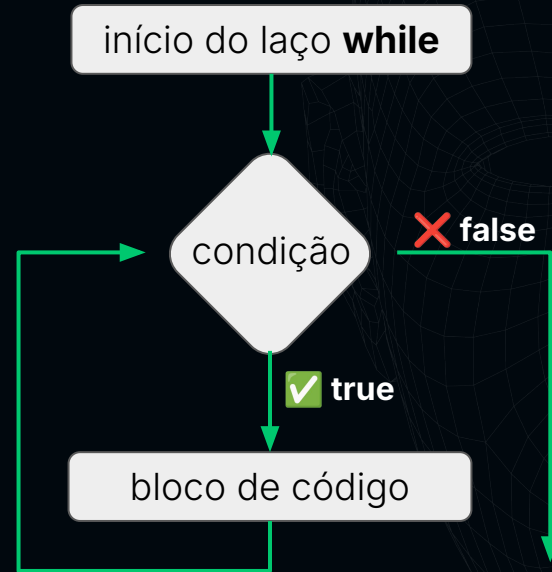
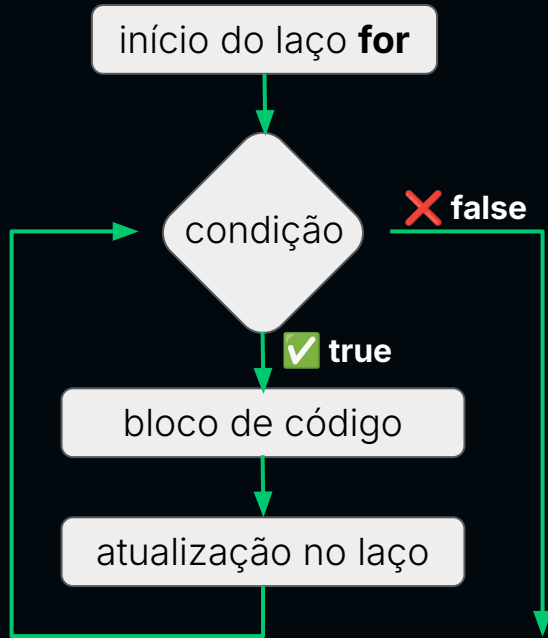
- For
- While
- Do...While

Outros tipos de instrução e laços:

- For...of
- For...in
- Métodos de array, string, objeto

O que são laços de repetição?

A leitura de um laço pode ser interpretada como: **repita até que** a condição se torne falsa.



Sintaxe do laço FOR

```
for ( inicialização ; condição ; atualização ) {  
    }  
}
```

Executada uma única vez antes do laço começar.

Avaliada antes de cada iteração. Se for **true**, o laço continua; se for **false**, o laço para.

Executada após cada iteração do laço.

Exemplo de código FOR

1. Calcular a tabuada de um número armazenado na variável **numero**;
2. Para cada “loop”, imprimir na tela o número, o multiplicador e o resultado.

```
const numero = 5;  
  
for (let i = 1; i <= 10; i++) {  
  const resultado = numero * i;  
  console.log(`${numero} x ${i} = ${resultado}`);  
};
```

Diagrama de anotações no código:

- inicialização**: `let i = 1;`
- condição**: `i <= 10;`
- atualização**: `i++`

Exemplo de código FOR

```
const numero = 5;  
  
for (let i = 1; i <= 10; i++) {  
  → const resultado = numero * i;  
  → console.log(`${numero} x ${i} = ${resultado}`);  
};
```

iteração 1
valor de i: 1
5 x 1 = 5

iteração 2
valor de i: 2
5 x 2 = 10

iteração 3
valor de i: 3
5 x 3 = 15

iteração 4
valor de i: 4
5 x 4 = 20

iteração 5
valor de i: 5
5 x 5 = 25

iteração 6
valor de i: 6
5 x 6 = 30

iteração 7
valor de i: 7
5 x 7 = 35

iteração 8
valor de i: 8
5 x 8 = 40

iteração 9
valor de i: 9
5 x 9 = 45

iteração 10
valor de i: 10
5 x 10 = 50

Condições de execução e parada do FOR

```
for (let i = 1; i <= 10; i++) {
```

Expressão executada **apenas** na inicialização do laço. "i" é um **identificador padrão** para variáveis contadoras em laços *for*.

A **condição** é avaliada antes do início de cada iteração. Em caso de **false** o laço é interrompido.


Esta expressão é sempre executada **ao final** de cada iteração.

[for no MDN](#)

Loop Infinito

Um loop infinito ocorre quando um laço continua a **executar sem parar**, porque a condição de saída nunca ocorre.

```
const numero = 5;  
  
for (let i = 1; i <= 10; ) {  
  const resultado = numero * i;  
  console.log(`${numero} x ${i} = ${resultado}\n`);  
};
```



Neste exemplo, o laço continuará imprimindo o resultado da primeira iteração, porque a condição $i \leq 10$ **nunca** mudará para falsa.

Loop Infinito

Um loop infinito ocorre quando um laço continua a **executar sem parar**, porque a condição de saída nunca ocorre.

```
const numero = 5;
for (let i = 1; i >= 0; i++) {
  const resultado = numero * i;
  console.log(`${numero} x ${i} = ${resultado}\n`);
};
```

Neste exemplo, o laço nunca vai parar de incrementar, porque a **i já inicia com valor 1** e **i >= 0 nunca será false**.



Após iniciado o laço, o programa só continua executando o restante do código **após finalizar as iterações do laço**.

Break

A instrução **break** interrompe um laço **for** ou **while**, mesmo que a condição para continuar seja **verdadeira**.

```
let numero = 0;
let contador = 0;

for (i = 1; i <= 50; i++) {
  numero = Math.floor(Math.random() * (50 - 1 + 1)) + 1;
  contador++;
  if (numero === 15) {
    console.log(`${numero} em ${contador} tentativas`);
    break
  };
};
```

O laço irá gerar 50 números aleatórios de 1 a 50.

Se o número 15 for gerado em até 50 tentativas, o laço é interrompido antes de completar as 50 iterações.

Continue

A instrução **continue** “pula” para a próxima iteração do laço, ignorando o restante do código na iteração atual.

```
let numero = 0;
let contador = 0;

for (i = 1; i <= 20; i++) {
  numero = Math.floor(Math.random() * (50 - 1 + 1)) + 1;
  if (numero % 5 === 0) {
    continue
  };
  contador++;
};

console.log('contador', contador);
```

Neste exemplo, quando é gerado um número divisível por 5, **a iteração termina** e o **restante das instruções é ignorado**.

Antes de passar para a próxima iteração, o laço faz o incremento e a checagem da condição normalmente.

Sintaxe do laço WHILE

```
while ( condição ) {  
    bloco de código  
}
```

A condição é qualquer **expressão** booleana (**true/false**) ou que possa ser avaliada como **truthy/falsy**, utilizando **operadores** de comparação e operadores lógicos.

Exemplo: `while (numero < 30)`

[Operadores no MDN](#)

Exemplo de código WHILE

Código que gera números aleatórios até “acertar” o número de referência.

```
let numeroSecreto = 8;  
let numeroAleatorio = 0;  
let contador = 0;  
  
while (numeroSecreto !== numeroAleatorio) {  
  numeroAleatorio = Math.floor(Math.random() * (50 - 1 + 1)) + 1;  
  contador++;  
}  
  
console.log(`adivinhou em ${contador} tentativas`);
```

[while no MDN](#)

Exemplo de código WHILE

```
let numeroSecreto = 8;  
let numeroAleatorio = 0;  
let contador = 0;  
  
while (numeroSecreto !== numeroAleatorio) {  
    numeroAleatorio = Math.floor(Math.random() * (50 - 1 + 1)) + 1;  
    contador++;  
}
```

O laço while verifica a condição. **Enquanto essa expressão for true**, o bloco de código dentro do while será executado.

Condição de parada: o código tem que dar condições de saída do loop, caso contrário gerará um **loop infinito**.

Sintaxe do laço DO...WHILE

O laço **do...while** tem uma sintaxe similar a do **while**, visto anteriormente, sendo útil quando não se sabe exatamente quantas iterações serão necessárias. A sua diferença é que ela faz com que o bloco seja executado **pelo menos uma vez**, mesmo que a condição seja **falsa**.

do {

bloco de código

} while (condição); <.....

A condição é qualquer **expressão** que resulte em um **valor booleano** (*true/false*) ou seja avaliada como **truthy/falsy**.

Exemplo: ***while (numero < 30).***

Exemplo de código DO...WHILE

Código que gera números aleatórios até retornar um valor par.

```
let numeroRandom;  
  
do {  
  numeroRandom = Math.floor(Math.random() * (50 - 1 + 1)) + 1;  
} while (numeroRandom % 2 !== 0);  
  
console.log(numeroRandom);
```

[do..while no MDN](#)

Exemplo de código DO...WHILE

Código que gera números aleatórios até retornar um valor par.

```
let numeroRandom;  
  
do {  
  numeroRandom = Math.floor(Math.random() * (50 - 1 + 1)) + 1;  
} while (numeroRandom % 2 !== 0);  
  
console.log(numeroRandom);
```

No laço **do...while** o bloco de código é executado **pelo menos uma vez**.

Se o primeiro número gerado já for par, a primeira iteração não é feita e o processo é encerrado.

- A propriedade **.length** retorna o comprimento de um array ou string. É possível saber quantas iterações devem ser feitas em uma string ou array.

```
const palavra = "papagaio";  
  
//    0    1    2    3    4    5    6    7  
// ["p", "a", "p", "a", "g", "a", "i", "o"];  
  
for (i = 0; i < palavra.length; i++) {  
  console.log(palavra[i]);  
};
```

Strings e **arrays** em JS possuem a propriedade **.length**.

É possível iterar blocos de texto usando **for** e **while**.

Listas em JS iniciam em 0 e é possível acessar um índice da lista com a sintaxe **lista[i]**.

Exemplo: **palavra[3]** se refere à letra "a",
palavra[6] se refere à letra "i".

Diferenças e casos de uso

for	while	do...while
número definido de iterações	número indefinido de iterações	ao menos uma iteração
possui instruções de incremento/contador	incremento/contador gerenciado externamente	incremento/contador gerenciado externamente

Compartilhe um resumo de seus novos
conhecimentos em suas redes sociais.

[#aprendizadoalura](#)

alura



Escola Programação