

# While e Do... While

## Objetivo da Aula

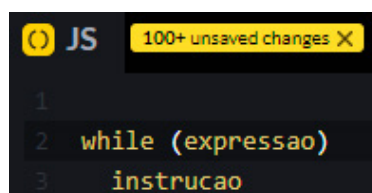
Implementar scripts com laços de repetições.

## Apresentação

Nas aulas anteriores aprendemos como implementar diferentes tipos de estruturas condicionais no nosso código. Nesta aula aprenderemos as estruturas de repetição **while** e **do...while**. Por que utilizar estruturas de repetição? Basicamente, para evitar repetição de trechos de código. Uma estrutura de repetição, seja ela qual for, permite que um trecho de código seja executado repetidamente até que ocorra condição de parada. A partir de agora conseguiremos solucionar problemas cada vez mais complexos.

### 1. While

A estrutura de repetição **while** (enquanto) funciona da seguinte forma: um bloco de código é executado até que o teste condicional se torne falso. Veja como é a sintaxe desta estrutura:



```
1  
2 while (expressao)  
3     instrucao
```

Onde **instrução** representa o bloco de instruções que será executado dentro do **while** e **expressão** representa o teste condicional para que o código seja executado. Por exemplo, criaremos um pequeno script para exibir na tela os números de 1 a 10.



The screenshot shows a code editor with a dark theme. On the left, there's a tab labeled 'JS' with a yellow icon and a notification '100+ unsaved changes X'. The code is as follows:

```

1 let contador = 1; //variável contadora
2
3 while (contador <= 10) {
4   console.log(contador);
5   contador++;
6 }
  
```

On the right side, there's a 'Console' panel showing the output of the code, which is the numbers 1 through 10, each on a new line.

Link para ter acesso ao código: <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/9a5c10f46e8567bd539be608ec74f6085be221c5/Unidade2Aula3a>

Optei por exemplificar apenas com 10 números para que o **print** da imagem coubesse na apostila, mas você pode acessar o código no link acima e testar com o número que quiser.

Agora vamos analisar o código. Na **linha 1**, temos uma variável chamada contador, esta variável dentro de uma estrutura de repetição se comporta como uma variável contadora. Calma! Já vamos entender o que isso significa. Na **linha 3**, temos a estrutura de repetição **while** testando a seguinte condição `contador <= 10`, podemos traduzir esta linha da seguinte forma: “enquanto a variável contador for menor ou igual a 10, faça.” Caso a condição seja verdadeira, as instruções dentro do **while** são executadas. No nosso caso, a condição é verdadeira, portanto, a **linha 4** será executada, ou seja, a variável contadora será exibida na tela. Em seguida, a **linha 5** será executada, nela temos `contador++` (é o mesmo que `contador = contador + 1`), ou seja, a variável contador está sendo incrementada, logo o valor da variável deixa de ser 1 e passa a ser 2. Na **linha 6**, temos o final do bloco de instruções representado pelo fechamento da `}`, ao se deparar com o final do bloco ele entende que deve voltar para a linha 3 e repetir todo o processo até que a condição seja falsa, por isso chama-se estrutura de repetição. Repare que em cada rodada a variável contador é incrementada, por esse motivo é considerada uma variável contadora. Vale ressaltar que, ser contadora não significa contar de um em um, podemos contar de dois em dois (`contador+=2` ou `contador=contador + 2`), de três em três (`contador+=3` ou `contador=contador + 3`) e assim sucessivamente.

Vamos melhorar o nosso exemplo exibindo também o somatório dos números de 1 a 10, para isso, utilizaremos uma variável que se comportará como uma variável acumuladora

dentro da nossa estrutura de repetição. Diferentemente da variável contadora que recebe ela mesma + 1 (caso você esteja contando de 1 em 1), a variável acumuladora recebe ela mesma + o que ela quer acumular. Veja:

```

JS 100+ unsaved changes X
1 let contador = 1; //variável contadora
2 let soma = 0;
3
4 while (contador <= 10) {
5     console.log(contador);
6     soma+=contador; //variável acumuladora (soma = soma + contador)
7     contador++;
8 }
9 console.log("O somatório é: " + soma);

```

Console

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
"O somatório é: 55"

Link para ter acesso ao código: <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/9a5c10f46e8567bd539be608ec74f6085be221c5/Unidade2Aula3a>

Agora vamos analisar o código. Inserimos aqui uma variável nova chamada soma (**linha 2**) e a mesma está recebendo o valor inicial 0. Dentro da nossa estrutura de repetição inserimos a nossa variável acumuladora (**linha 6**) soma+=contador (soma = soma + contador), que a cada rodada acumulará os valores da variável contador que varia de 1 a 10, ou seja, a nossa variável acumuladora somará 1+2+3+4+5+6+7+8+9+10. Ao final exibirá a seguinte

mensagem: "O somatório é: 55", através do método console.log (linha 9). Veja na tabela abaixo os valores assumidos pelas variáveis contador e soma em todas as rodadas do nosso script.

	contador	soma
Valor Inicial	1	0
Rodada 1	1	1
Rodada 2	2	3
Rodada 3	3	6
Rodada 4	4	10
Rodada 5	5	15
Rodada 6	6	21
Rodada 7	7	28
Rodada 8	8	36
Rodada 9	9	45
Rodada 10	10	55



### Você Sabia?

Esta técnica de montar uma tabelinha para testar programas de computador é chamada de teste de mesa ou chinês.

## 2. Do... While

Diferentemente do **while**, a estrutura de repetição **do... while** (faça... enquanto) testa a condição somente no final do bloco de instruções, com isso, mesmo que a condição seja FALSA, o bloco de instruções será executado ao menos uma vez. Veja como é a sintaxe desta estrutura:

```

1  do {
2
3      instrucoes
4
5  }while(condicao);
  
```

Nela as instruções são executadas uma vez e reexecutadas cada vez que a condição (linha 5) *for* verdadeira, quando a condição *for* falsa, a estrutura de repetição é abandonada. Vamos refazer os mesmos exemplos só que agora usando a estrutura *do... while*. Vejamos:

```

JS 100+ unsaved changes X
1 let contador = 1;
2
3 do {
4   console.log(contador);
5   contador++;
6 }while (contador <= 10);
  
```

Console

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

*Link para ter acesso ao código:* <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/9a5c10f46e8567bd539be608ec74f6085be221c5/Unidade2Aula3a>

Repare que trocamos de estrutura, mas obtivemos o mesmo resultado. Agora vamos ver como fica o exemplo do somatório.

```

JS 100+ unsaved changes X
1 let contador = 1; //variável contadora
2 let soma = 0;
3
4 do{
5   console.log(contador);
6   soma+=contador; //variável acumuladora (soma = soma + contador)
7   contador++;
8 } while (contador <= 10);
9 console.log("O somatório é: " + soma);
  
```

Console

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
"O somatório é: 55"

**Link para ter acesso ao código:** <https://github.com/GRANCodigo/PraticaDeProgramacao/blob/9a5c10f46e8567bd539be608ec74f6085be221c5/Unidade2Aula3a>

Reparou que a lógica é a mesma? A diferença é que o teste que antes era feito na **linha 4**, agora está sendo feito na **linha 8**. Outra diferença também é que a condição do **do... while** é seguida de ponto e vírgula (;).

## Considerações Finais

A estrutura repetição é utilizada quando queremos executar mais de uma vez um mesmo bloco de instruções. Nesta aula, conhecemos duas estruturas de repetição bem semelhantes, são elas: **while** e **do... while**. Você enquanto programador é que deve decidir qual delas irá utilizar. A diferença entre elas é bem sutil, porém, a estrutura **do... while** não é muito utilizada, pois não é comum escrevermos um código considerando que um bloco de instruções será executado mesmo que a condição testada seja falsa. Portanto, use parcimônia!

## Materiais Complementares

Tutorial **do... while**:

<https://youtu.be/y8°tiWoVfRI>

Tutorial **while**:

<https://youtu.be/buZQknMhPKo>

Laços e interações:

[https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Loops\\_and\\_iteration](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Loops_and_iteration)

## Referências

FLANAGAN, David. *JavaScript: O guia definitivo*. Porto Alegre, RS: Bookman, 2013.