# Tema 08 – Laços de Repetição

## Introdução

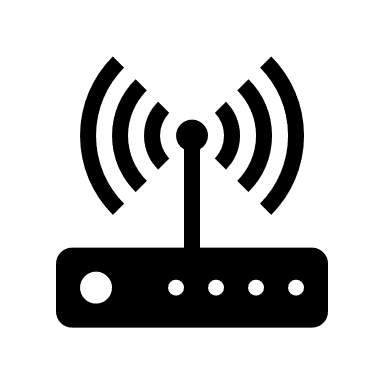
Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá como implementar laços de repetição.

## Criação do Projeto inicial

Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 08\_JS\_Lacos.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>Laços de Repetição</title>      </head>      <body>          <script src="./js/main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas iremos completar a implementação do código JavaScript.
3. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**).

## Laços de repetição

Os laços de repetição são comandos utilizados para executar instruções mais de uma vez, ou seja, cria loops de repetição de uma ou mais instruções com um número limitado de vezes. Eles possuem com variáveis de controle/acumuladora.

Para trabalhar com laços de repetições, sempre será necessário testar uma condição que irá resultar em um valor booleano (true ou false). Por isso, quase sempre, fazemos usar de operadores de comparação e/ou lógicos:

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores de comparação | == | Igual |
| != | Diferente |
| < | Menor que |
| <= | Menor ou igual |
| > | Maior que |
| >= | Maior ou igual |
| === | Triplo igual |

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores lógicos | && | Lógica “and” ou “e”, que retorna verdadeiro se todos os operandos forem verdadeiros. |
| || | Lógica “or” ou “ou”, que retorna verdadeiro se pelo menos um operando for verdadeiro. |
| ! | Lógica “not” ou “não”, que inverte o valor lógico se é verdadeiro, retorna falso e se é falso retorna verdadeiro. |

Uma estrutura de repetição permite especificar que um BLOCO ( { } ) de instruções ou o programa todo deverá ser repetido enquanto alguma condição permanecer verdadeira. O número de repetições pode ser fixo ou atender alguma condição. Vantagens: O algoritmo passa a ter um número menor de linhas e de instruções repetidas devido a estrutura de repetição e é possível aumentar a amplitude de processamento sem alterar o tamanho do código.

### Laço while

O laço **while** é uma repetição com teste no início do comando. Assim, o bloco de instruções será repetidamente executado enquanto a condição for verdadeira. Quando a condição for falsa, a execução do bloco de comandos será interrompida:

A picture containing light, way, dark, highway

Description automatically generated

A sintaxe da estrutura **while** é:

|  |
| --- |
| while ( condição )  {  instrucao1;  instrucao2;  instrucao3;  instrucaoN;  }  proximaInstrucao; |

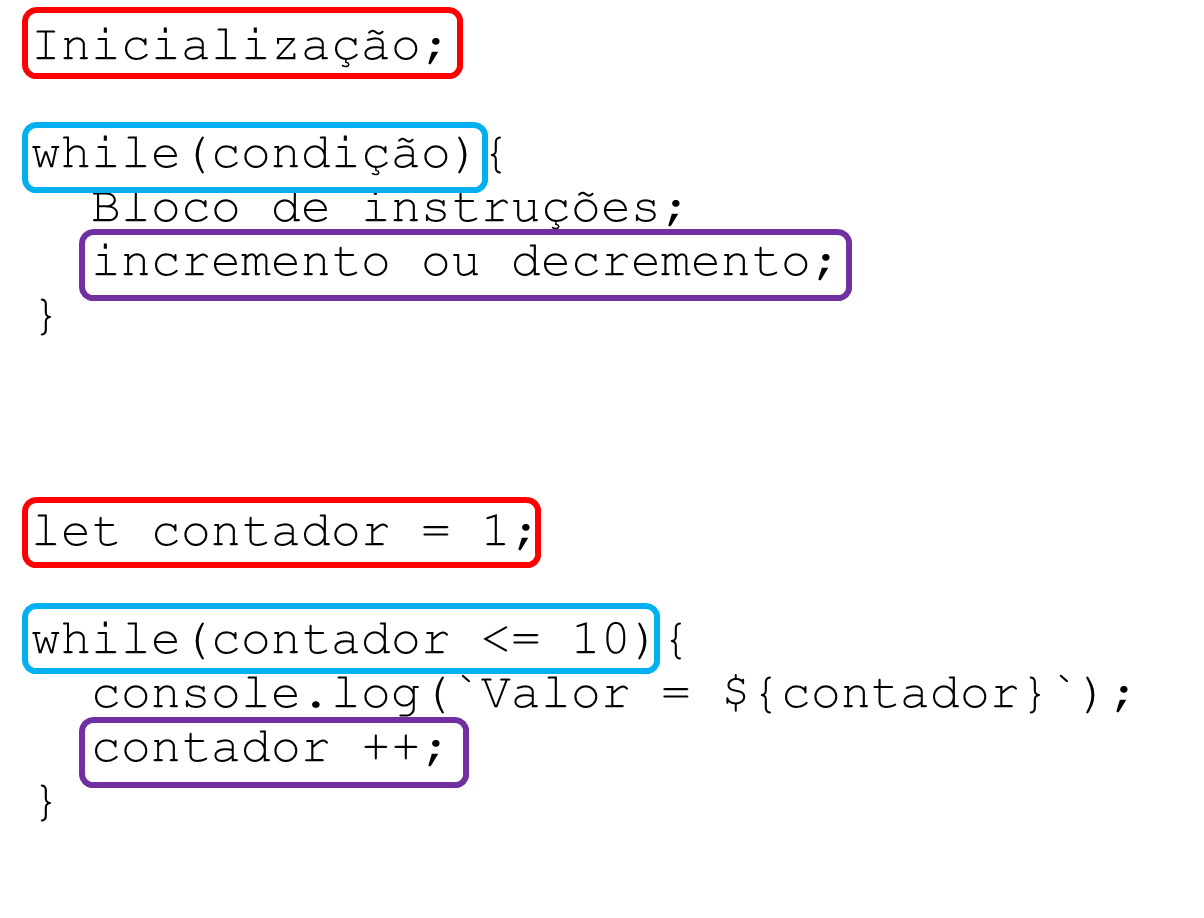
O bloco de comandos é executado enquanto a condição for verdadeira. As chaves podem ser ocultadas se o comando contiver apenas uma instrução:

|  |
| --- |
| while ( condição ) instrucao1;  proximaInstrucao; |

No while, a condição é verificada antes de entrar no laço e enquanto o resultado da condição for verdadeiro – executa o bloco de instruções. Quando a condição se tornar falsa, o laço é encerrado e o fluxo vai para a próxima instrução do algoritmo. Portanto, a condição é testada a cada iteração (volta).

Importante: Laço é um bloco de instruções ( { } ) que será executado repetidas vezes e que está contido em uma estrutura de repetição.

Todo laço de repetição pode ter um contador com variável de controle da condição. Um contador é uma variável do tipo inteiro usada para contar a quantidade de vezes que um bloco de instruções é repetido e deve ser inicializado antes de ser utilizado.



Observe, a inicialização fora do comando **while**, e o incremente ou decremente do contado dentro do **while**. Caso você não garantir que a condição em algum momento se torne falsa, o programa ficará em um loop infinito.

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 1 | // while  let contador = 1;  while (contador <= 10) {      console.log(`Valor = ${contador}`);      contador++;  } |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

1. Vamos ver o resultado mostrado no console.

Graphical user interface, text, application

Description automatically generated

Observe que a variável com o nome **contador** inicia com o valor 1, que é o valor inicializado na linha 1, portanto a condição (contador <= 10) é verdadeira e será impresso o valor pela primeira vez no console:



Em seguida, a variável é incrementada em uma unidade (linha 5), a condição é testada novamente e o resultado continua verdadeiro. Portanto o segundo valor é impresso no console:



Isso vai ser repetir até a variável assumir o valor 11, que então faz a condição retornar falso e encerrar a execução do laço.

1. Podemos implementar um laço com decremento também.
2. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 9 | // while com decremento  console.clear();  let cont = 10;  while (cont >= 0) {      console.log(`Valor = ${cont}`);      cont--;  } |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |

1. O resultado é mostrado no console do navegador:

Graphical user interface

Description automatically generated with medium confidence

Observe que a variável com o nome **contador** inicia com o valor 10, que é o valor inicializado na linha 11, portanto a condição (contador >= 0) é verdadeira e será impresso o valor pela primeira vez no console:



Em seguida, a variável é decrementada em uma unidade (linha 5), a condição é testada novamente e o resultado continua verdadeiro. Portanto o segundo valor é impresso no console:

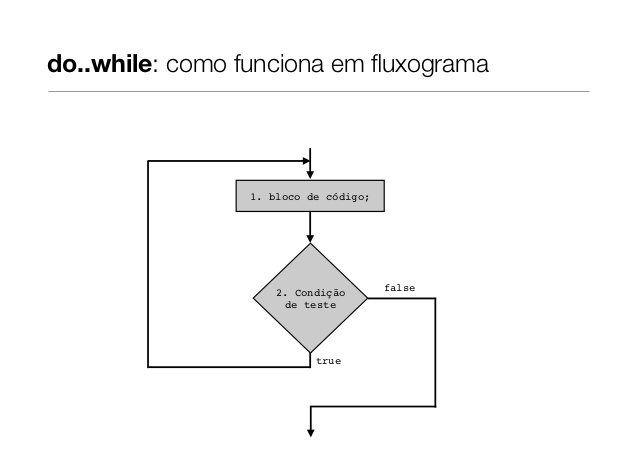


Isso vai ser repetir até a variável assumir o valor -1, que então faz a condição retornar falso e encerrar a execução do laço.

Note, também, número da linha da instrução executada é sempre é 14, isso significa que a instrução da linha 14 está sendo repetida por diversas vezes.

### Laço do-while

Esse tipo de estrutura de repetição é caracterizado por fazer o teste de controle no final do bloco de comandos. Os comandos repetidos são executados pelos menos uma vez antes da condição ser testada. A condição é sempre testada no final do laço e após a execução do bloco de comandos dele.



A diferença básica para o comando while é que o bloco a ser repetido sempre executa ao menos uma vez. **A sintaxe do do-while é:**

|  |
| --- |
| do{  instrucao1;  instrucao2;  instrucao3;  instrucaoN;  } while( condição );  proximaInstrucao; |

O bloco de comandos é executado enquanto a condição for verdadeira.

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 18 | // do-while  console.clear();  let i = 0,      text = '';  do {      text += `O número é ${i}\n`;      i++;  } while (i < 10);  console.log(text); |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

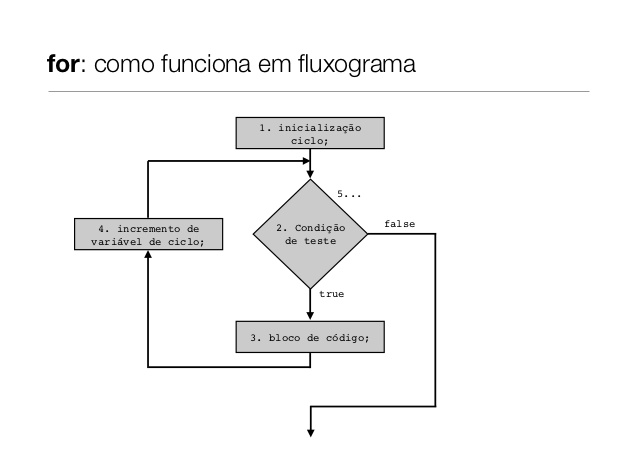
Observe que o while executa o bloco pelo menos uma vez, por exemplo se você altera o valor de i para 10. Ele irá imprimir uma vez no console e ao testar a condição, ela retornará falso, encerando, assim, o laço.

Graphical user interface, text, application, email

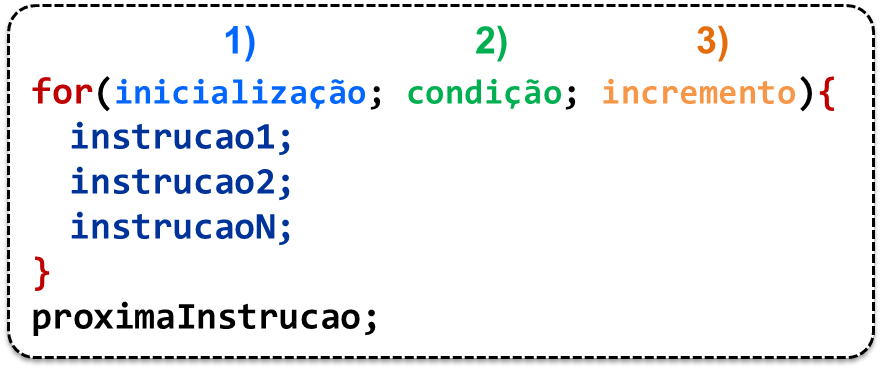
Description automatically generated

### Laço For

O comando **for** também realiza o teste lógico no início do laço. Ele tem uma estrutura um pouco diferente do **while**, mas funciona da mesma maneira. O programa não executará nenhuma repetição (ações programadas) sem antes testar a condição:



A sintaxe do comando **for** é:



**1)** A **inicialização** é uma atribuição e é executada uma **única vez** antes do laço ser iniciado

**2) Condição** que controla o laço. Repete o bloco (**{}**) enquanto a condição for verdadeira

**3)** O **incremento** é sempre a última instrução do laço (ex: cont++)

As chaves podem ser ocultadas se o comando contiver apenas uma instrução:

|  |
| --- |
| for(inicializacao; condicao\_testada; incremento ou decremento)  instruçao; |

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

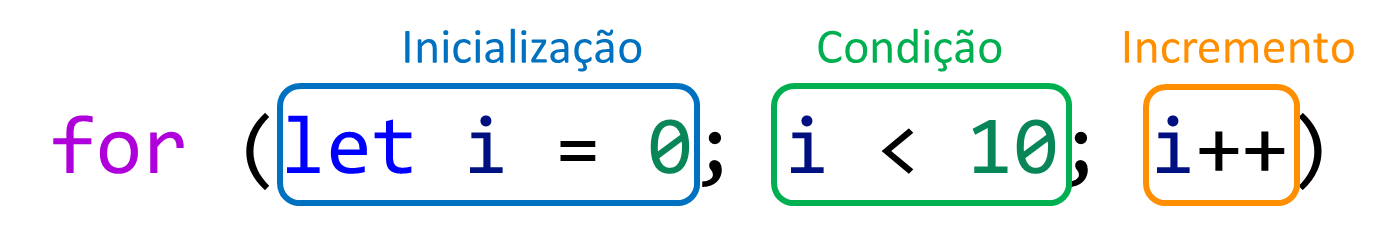
|  |  |
| --- | --- |
| 28 | // For  console.clear();  for (let i = 0; i < 10; i++) console.log(`Laço For número: ${i}`); |
| 29 |
| 30 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application

Description automatically generated

No comando **for** temos:



O valor da variável i é inicializada com 0, desse modo a condição a condição é verdadeira e mensagem é impressa pela primeira vez:



Em seguida, a variável i é incrementada de uma unidade, a condição é testada e continua verdadeira, imprimindo para mensagem pela segunda vez:



Em seguida, a variável i é incrementada de uma unidade, a condição é testada e continua verdadeira, imprimindo para mensagem pela terceira vez:



E isso vai se repetindo até a variável i assumir o valor 10, que torna a condição falsa, encerrando assim o laço.

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

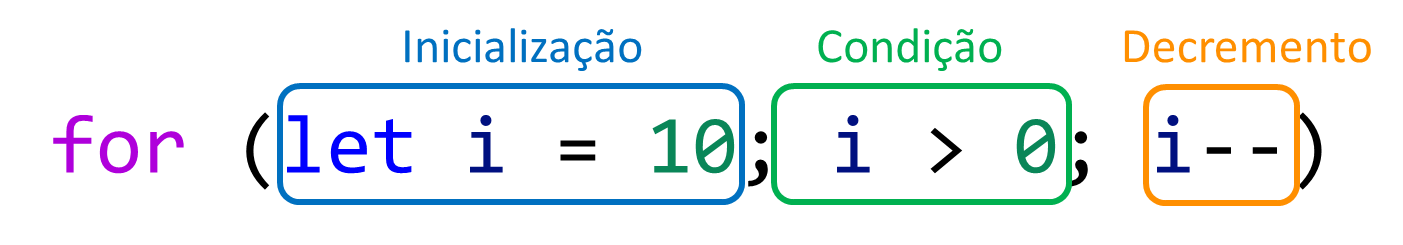
|  |  |
| --- | --- |
| 32 | // For - Decremento  console.clear();  for (let i = 10; i > 0; i--) console.log(`Laço For número: ${i}`); |
| 33 |
| 34 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Esse for faz a impressão contrário do anterior usado o decremento da variável de controle.



Graphical user interface, text, application, email

Description automatically generated

## Laços de repetição e arrays

Utilizamos muito laços de repetição para preencher ou percorrer array ou objetos de arrays vamos ver os exemplos:

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 36 | // Laços de arrays  const frutas = ['apples', 'oranges', 'pears', 10, true];  console.clear();  for (let j = 0; j < frutas.length; j++) console.log(`Nome: ${frutas[j]} \n`); |
| 37 |
| 38 |
| 39 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

Como você pode ver, o laço percorreu todos os elementos do array para imprimir cada um no console.

1. Outro exemplo, no arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 41 | // Array de objetos e laços  console.clear();  const todos = [      {          id: 1,          text: 'Take out trash',          isCompleted: true,      },      {          id: 2,          text: 'Meeting with boss',          isCompleted: true,      },      {          id: 3,          text: 'Dentist appt',          isCompleted: false,      },  ];  // For overloop  for (let t of todos) {      console.log(t);      console.log(t.text);      console.log(t.id);  } |
| 42 |
| 43 |
| 44 |
| 45 |
| 46 |
| 47 |
| 48 |
| 49 |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |
| 56 |
| 57 |
| 58 |
| 59 |
| 60 |
| 61 |
| 62 |
| 63 |
| 64 |
| 65 |
| 66 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

Nesse exemplo, usamos o chamado for overloop, que na inicialização, condição e incremento temos apenas a instrução **let t of todos**. Essa instrução do JavaScript é útil para percorrer os índices de arrays. Ela cria uma variável (let t), que irá armazenar um objeto por vez de execução do loop e vai começar no primeiro objeto do array, ir incrementando até chegar no último e encerrar o laço. Tudo isso é feito pelo interpretador do JS de forma automática.

Importante: essa é uma forma de percorrer um array que está sendo cada vez mais substituída pelos métodos de alto nível. Mas é interessante você ver isso para saber as diferenças e semelhanças de usar cada umas das técnicas.

## Conclusão

Procure sempre fazer mais do que é passado para você em sala de aula. Por exemplo, você pode alterar as condições, utilizar comparações múltiplas nos laços, criar novos laços para ver o resultado que será gerado.

Seguem alguns links para você estudar e aprender mais:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/while>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/do...while>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/for>