Laços

Laços ou loops são formas de repetir instruções um determinado número de vezes sem que haja a necessidade de repetir essas instruções. Em pseudocódigo nós usamos a palavra **ENQUANTO**, essa palavra indicava que os passos que estivessem descritos em seguida seriam repetidos até a nossa condição se tornar falsa. Qual condição? Não conseguimos simplesmente mandar o código se repetir sozinho, para esse laço acontecer é necessário existir uma condição para avaliar se o laço vai ser executado ou não, do mesmo jeito que fazemos com o **SE/SENÃO**.

A estrutura básica seria:

```
ENQUANTO CONDIÇÃO FOR VERDADEIRA FAÇA
    Instruções a serem repetidas
FIM do ENQUANTO
```

Em Javascript este 'enquanto' se torna **WHILE** e funciona exatamente da mesma forma que o enquanto no pseudocódigo. A sintaxe dele no javascript é a seguinte:

```
while(CONDICAO){
    Código a ser repetido se a condição for verdadeira
}
```

Ao escrever um while é importante notar que no seu código é necessário criar uma variável fora do laço para essa condição, normalmente chamamos de contador, e em algum momento a sua condição deve ser atualizada, caso contrário o seu loop se torna infinito e no nosso caso trava o navegador. O que eu quero dizer com atualizar a condição? É necessário que o nosso loop tenha um fim e para isso a nossa condição tem que se tornar falsa.

Existem outras formas de se fazer um laço e uma delas é o **FOR**. Essa forma usa outra sintaxe, o for precisa saber onde é o início desse laço, o fim e o passo.

Início - onde que o laço começa
Fim - até onde o laço roda (condição de fim)
Passo - o passo para a próxima rodada

O interessante do for é que na própria estrutura dele nós já fazemos uma atualização do valor que é usado na nossa condição de fim.
A sintaxe do for no javascript é:

```
for(inicio; fim; passo){
    Código a ser repetido enquanto o critério de fim estiver válido
}
```

Além do for tradicional que foi esse que acabamos de citar, temos outros dois derivados: o **for in** e o **for of**. Esses dois tipos de **for** são usados para listas. Como são derivados do **for**, a ideia dele muda na hora de escrever a sintaxe. No ***for in*** nós criamos uma variável responsável por receber o valor do índice da nossa lista. Enquanto no ***for of*** criamos uma variável responsável por receber o valor do **elemento** da lista, isso de passar elemento por elemento de uma lista nós chamamos de iterar uma lista.

Abaixo nós temos a sintaxe dos dois for:

```
for(let indice  in lista){
    Código
}

for(let elemento of lista){
    código
}
```

Exemplos práticos:

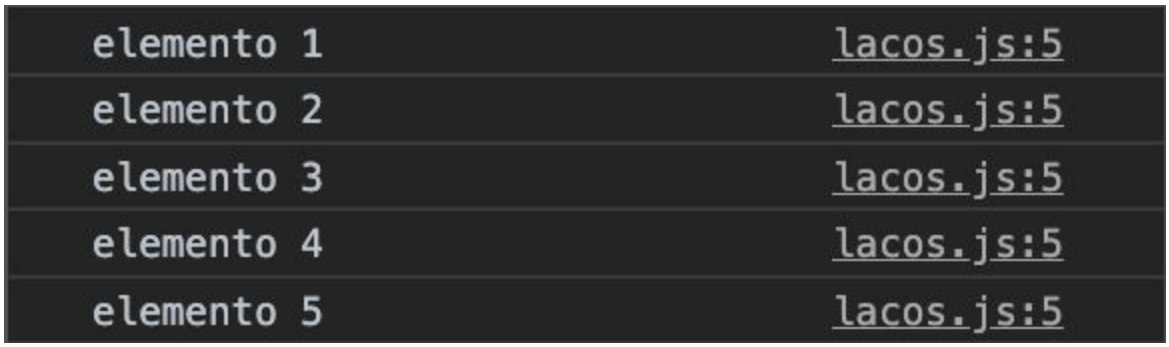
```
1   let lista = ["elemento 1", "elemento 2", "elemento 3", "elemento 4", "elemento 5"];
```

Vamos usar essa lista para todos os exemplos para entender o que está acontecendo em cada um dos laços.

Primeiro o **while**

```
2   let contador = 0;
3
4   while(contador < lista.length){
5       console.log(lista[contador]);
6       contador++;
7   }
```

Console:



Nesse exemplo, criamos uma variável de contador externa ao laço e essa variável é usada na condição para executar o laço.
A linha 4 é lida assim: *ENQUANTO o contador for menor que o tamanho da lista*, a condição é verdadeira e o **console.log** é executado.
Dentro do **console.log** nós colocamos o nosso elemento da lista que está na posição do nosso contador. E logo em seguida nós atualizamos o valor do contador, somando um ao valor do contador.

For

```
9   for(let indice = 0; indice < lista.length; indice++){
10      console.log(lista[indice]);
11  }
```

Console:

Nesse laço criamos uma variável chamada índice que é o nosso contador com o seu valor de início que é zero. No segundo valor indicamos que o nosso loop vai ser executado enquanto o valor do índice for menor que o tamanho da lista. E por fim, no último valor indicamos qual é o passo do nosso loop, neste caso é somado um ao índice no final de cada rodada do laço.

For in

```
13  for(let indice in lista){
14      console.log(lista[indice]);
15  }
```

Console:

Nesse caso, nós estamos criando uma variável responsável por assumir o valor de cada índice da nossa lista. Sendo assim, para mostrar no console cada item da nossa lista é preciso acessar elemento por elemento usando a lista[indice].

For of

```
4  for(let elemento of lista){
5      console.log(elemento);
6  }
```

Console:

Nesse caso, criamos uma variável responsável por assumir valor de cada elemento da nossa lista. Assim, para poder mostrar cada item da nossa lista no console, precisamos usar somente o próprio elemento. É comum que o nome de listas estejam no plural por serem listas de um conjunto de coisas, nesse caso o nome da variável criada para assumir o valor de cada elemento costuma ter o nome da lista no singular. Exemplo: a lista chama filmes, a variável criada no for of chama-se filme.

Teste de Mesa

Teste de Mesa é uma forma de se validar uma lógica de algoritmo e entender o que está acontecendo em cada linha do seu código.

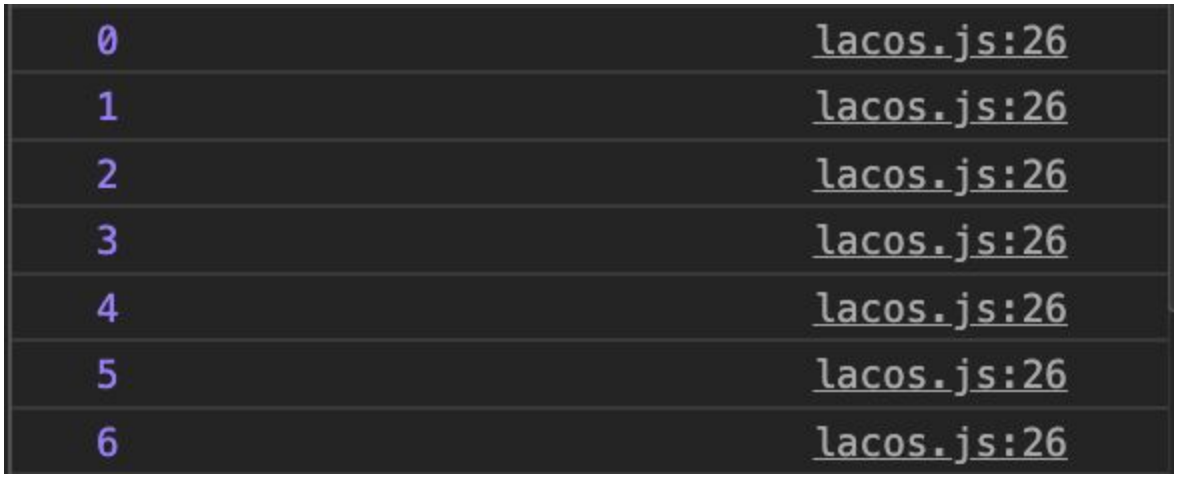
Para montar o teste de mesa, precisamos das variáveis envolvidas no algoritmo de teste, às outras colunas podem mudar os nomes um pouco, neste caso coloquei rodada porque estamos testando um laço, mas em outros casos poderia ser o número da linha do código. A coluna marcada como for pode ser considerada também como saída do algoritmo.

Nosso objetivo aqui é imprimir o nome de cada anime da nossa lista um em cada linha. Para isso vamos fazer um passo a passo e utilizar o teste de mesa para ilustrar o funcionamento do nosso algoritmo a cada rodada de iteração.

Primeiro vamos entender o que é o **i** e o que acontece com ele em cada rodada do nosso laço:

```
23  let animes = ["Boku no Hero", "Naruto", "Death Note", "Kimi no na wa", "Cavaleiros do
    Zodíaco", "Dragon Ball Z", "Super Campeões"];
24
25  for(let i = 0; i < animes.length; i++){
26      console.log(i);
27  }
```

rodada	animes	i	for (console.log(i))	passo(i++)
-> inicio	é a própria lista			
1	é a própria lista	0	console.log(0)	i=0+1=1
2	é a própria lista	1	console.log(1)	i=1+1=2
3	é a própria lista	2	console.log(2)	i=2+1=3
4	é a própria lista	3	console.log(3)	i=3+1=4
5	é a própria lista	4	console.log(4)	i=4+1=5
6	é a própria lista	5	console.log(5)	i=5+1=6
7	é a própria lista	6	console.log(6)	i=6+1=7



Note que a lista de animes não tem seu valor alterado em nenhum momento no nosso código, por isso o valor dela permaneceu o mesmo em toda a coluna.
No nosso estado inicial do for nós temos apenas a lista criada. Quando chegamos na linha 25 do código, temos o nosso **for** e a partir daí começamos a primeira rodada do laço. Nessa linha temos a criação da variável i com valor inicial zero. Dentro dessa mesma rodada temos a execução do nosso console.log do valor de **i**, que no caso é 0. Ao final do for temos o nosso passo, que indica de quanto em quanto nosso contador vai passar, nesse caso somamos um ao valor de **i**. E assim vamos fazendo para cada rodada do nosso loop.

Percebam que o valor de **i** é exatamente o valor do índice de cada elemento da nossa lista de animes. Então se já temos os valores dos índices, precisamos agora somente mostrar o valor de cada elemento para atingirmos nosso objetivo. Para isso utilizamos o nome da lista e o índice da posição. Sendo assim o código passa a ser o seguinte:

```
29  for(let i = 0; i < animes.length; i++){
30      console.log(animes[i]);
31  }
```

rodada	animes	i	for (console.log(animes[i]))	passo(i++)
-> inicio	é a própria lista			
1	é a própria lista	0	console.log(animes[0])	i=0+1=1
2	é a própria lista	1	console.log(animes[1])	i=1+1=2
3	é a própria lista	2	console.log(animes[2])	i=2+1=3
4	é a própria lista	3	console.log(animes[3])	i=3+1=4
5	é a própria lista	4	console.log(animes[4])	i=4+1=5
6	é a própria lista	5	console.log(animes[5])	i=5+1=6
7	é a própria lista	6	console.log(animes[6])	i=6+1=7

Exemplo de **for in** desse mesmo exercício:

```
33  for(let indice in animes){
34      console.log(animes[indice]);
35  }
```

Exemplo do exercício com **for of**:

```
37  for(let anime of animes){
38      console.log(anime);
39  }
```