

Curso: Front-End do Bem



AULA 27

100% Gratuita | Baixou... Estudou | Sem pedir e-mail

- O que é esse tal de Controle de Fluxo envolvendo loop?
 - For loop
 - Exemplos com For loop
- Sugestões de Prática para essa aula
 - O que vamos aprender na aula 28



O que é esse tal de Controle de Fluxo envolvendo loop

O controle de fluxo faz referência ao meio como o código será executado, atendendo condições e/ou eventos específicos.

Ou seja conforme a complexidade daquilo que estamos desenvolvendo forem aumentando existirão ocasiões dentro do nosso código em que precisaremos tomar uma decisão, e essa decisão será: Qual a ação seguinte que deverá ser executada.

Por exemplo: Temos um Array com os tipos de dados que aprendemos nas aulas anteriores, o array `typeDataLearned` ['numbers', 'strings', 'null', 'undefined', 'boolean', 'if', 'else', 'else if'] e gostaríamos de percorrer esse Array e executar algum código para cada item dentro desse array `typeDataLearned`, aqui usaríamos um loop. Um Loop assim como a condicional (IF, ELSE e ELSE IF) é um tipo de controle de fluxo em JavaScript.

O primeiro tipo de loop que veremos aqui.... é o For loop.

IMPORTANTE: Antes de partirmos para o exemplo na prática, eu gostaria de pedir uma gentileza para vocês. Eu peço que acompanhem cada item inserido que é composta a estrutura base do nosso brother For Loop. Ao terminarmos de inseri-los, será explicado passo a passo com direito a um resuminho maroto como esse loop funciona.

Então bora ver essa estrutura básica do for....



/igor-rebolla

For loop

Então abaixo do nosso comentário maroto // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal

Inseriremos um for, logo em seguida um abre e fecha parênteses (), e logo em seguida um abre e fecha chaves {} e pressionaremos o enter 1 ou 2x entre as chaves até ficar uma linha entre a abertura e o fechamento dessas chaves permitindo que ali dentro seja inserido algum código. Aqui dentro dos parênteses, declararemos uma (let i que receberá 0; i < 27; i++). Ficando assim:

```
for (let i=0; i < 27; i++) {  
  
}
```



Igor, Igor, Igor, Igor... e esse caractere 'i' ai dentro dos parênteses tem referência com o caractere 'i' de Igor ????

Não, esse caractere 'i' que aparece dentro dos parênteses foi inserido ali por convenção ou seja todo for loop que encontrarmos terá essa variável i.

Agora o número 27 ali dentro dos parentêses faz referência ao número dessa aula aqui.

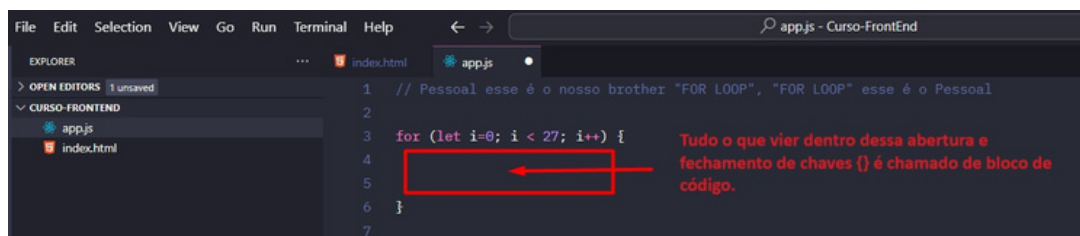


For loop

Com a estrutura base do nosso brother “For” realizada, eu gostaria que vocês ligassem as anteninhas para um detalhe importantíssimo, o objetivo de todo loop é efetuar um pedacinho de um determinado código várias vezes de forma repetida. É importantíssimo que tenha fixado e entendido esse detalhe.

Ah e mais um detalhe importante... essa abertura e fechamento de chaves {} e o que vem dentro dela chamamos de bloco de código, o código que tiver aqui dentro corresponderá a esse loop for (let i=0; i < 27; i++).... Ficando assim:

```
for (let i=0; i < 27; i++) {  
    //bloco de código (conforme imagem abaixo)  
}
```



Primeiro Resuminho Maroto... do que fizemos até o momento:

Criamos esse loop utilizando a palavra for, e na sequência entre esses parênteses (), foram declaradas 3 expressões distintas separadas cada uma por seu respectivo ponto e vírgula(;).

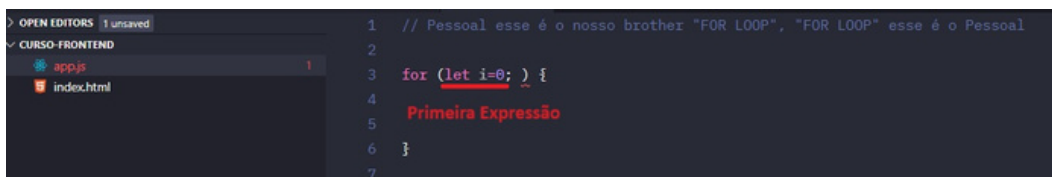
Bora “desmembrar” cada uma dessas expressões, para um melhor entendimento....



For loop

A primeira expressão que criamos foi a inicialização de uma variável, interpretaremos essa variável como um contador e esse contador contará quantas vezes esse loop já foi rodado (ou seja quantas vezes rodamos o código que colocaremos dentro desse bloco da abertura e do fechamento das chaves{}). Ficando assim:

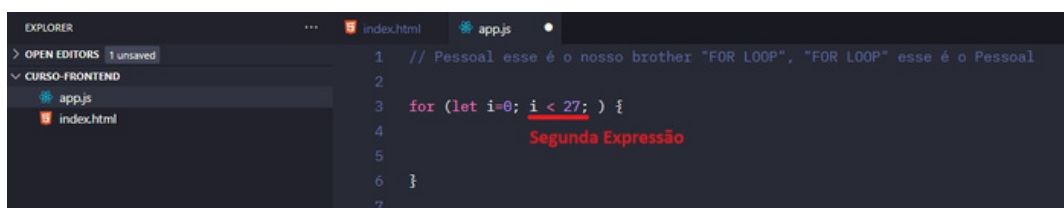
`let i=0` (Primeira Expressão)



```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; ) {
4     Primeira Expressão
5 }
6
7
```

A segunda expressão que criamos foi uma condição e essa condição resultará em falso ou verdadeiro, caso essa condição resulte em verdadeiro o código localizado dentro desse bloco das chaves {} será rodado, caso ela resulte em falso, o código localizado dentro desse bloco das chaves{} não rodará mais. Ficando assim:

`i < 27` (Segunda Expressão)

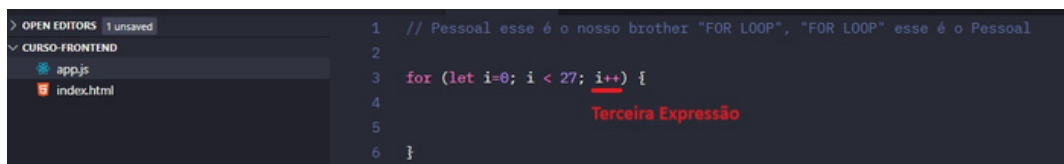


```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; i < 27; ) {
4     Segunda Expressão
5 }
6
7
```

For loop

E a terceira expressão que criamos foi a `i++`, isso representa um incremento, ou seja essa expressão será rodada ao fim de cada execução do código localizada dentro desse bloco das chaves `{}`. Ou seja sempre que a última linha do código dentro desse bloco for rodada, essa `let i` será incrementada. Ficando assim:

`i++` (Terceira Expressão)



```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; i < 27; i++) {
4
5     Terceira Expressão
6 }
```

Resuminho maroto do que fizemos:

A linha `for (let i=0; i < 27; i++)` iniciou a sua leitura e uma variável com o valor inicial 0 foi criada, logo em seguida a expressão `i < 27` foi feita a sua leitura, `i` é menor que 27?

Sim, pois o valor de `i` é 0, e como essa expressão resulta em verdadeiro o código localizado dentro das `{}` é rodado e quando esse código aqui finaliza sua execução, a terceira expressão `i++` é rodada ou seja a variável `i` que até o momento guarda 0 tem o seu valor incrementado, e agora ela passa a guardar 1.

For loop

Depois que o primeiro incremento é realizado, ficaremos de olho apenas nas expressões $i < 27$; $i++$. Igor, Igor, Igor, Igor. E porque isso acontece?

Pois a let i não será criada de novo, nós só a criamos para termos um ponto de inicio, como a variável i foi incrementada, essa condição $i < 27$ será verificada mais uma vez.. i continua menor que 27?

Sim, a i está guardando 1 e 1 é ainda é menor que 27, então essa expressão resultará em verdadeiro. E o que isso significa?

Isso significa que o código dentro das $\{\}$ será rodado mais uma vez

E ao terminar a sua execução a terceira expressão $i++$ é rodada mais uma vez, então a i receberá todo o valor que ela já tem + 1, então a partir de agora ela guardará 2. 2 ainda é menor que 27 ($i < 27$) ?

Sim, então o código localizado dentro das $\{\}$ será rodado e esse ciclo se repetirá até que a i guarde 27, ou seja:

Aqui ele repetiria o mesmo processo do número 3 até o número 27.....

E no momento em que i guardar 27?

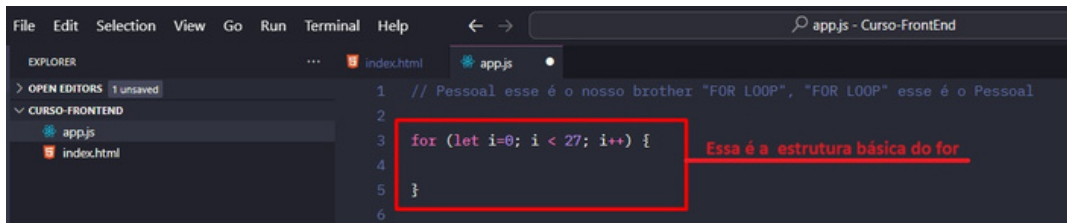
Será, retornado falso, então o bloco de códigos localizado dentro das chaves $\{\}$ não será mais rodado. Conforme a nossa condição $i < 27$, executamos o código localizado aqui dentro das chaves $\{\}$ diversas vezes até que a nossa condição $i < 27$ não seja mais verdadeira.

Bora ver como isso funciona com um exemplo maroto.....



For loop

Antes aproveitaremos a nossa estrutura base do for a qual foi criada nas páginas acima dessa aula (sem nenhuma modificação). Ficando assim:



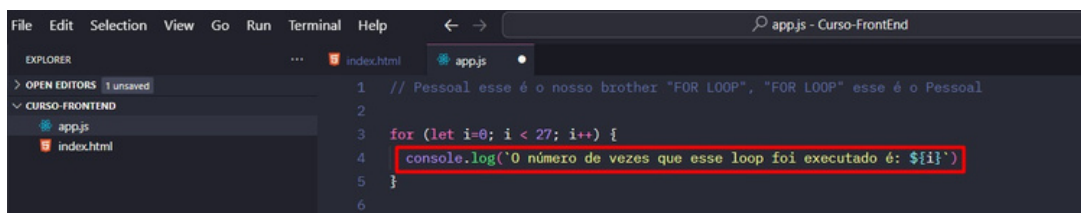
The screenshot shows a code editor with a file named 'app.js' open. The code contains a basic for loop structure:

```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; i < 27; i++) {
4
5 }
6
```

 A red box highlights the loop structure, and a red text annotation to the right says 'Essa é a estrutura básica do for'.

E dentro das {} declararemos um console.log que receberá uma template string `` com a seguinte mensagem: O número de vezes que esse loop foi rodado é: colocaremos uma interpolação com i.... `${i}`. Ficando assim:

```
for (let i=0; i < 27; i++) {
  console.log(`O número de vezes que esse loop foi executado é: ${i}`)
}
```



The screenshot shows the same code editor as before, but now the code includes a console.log statement inside the for loop:

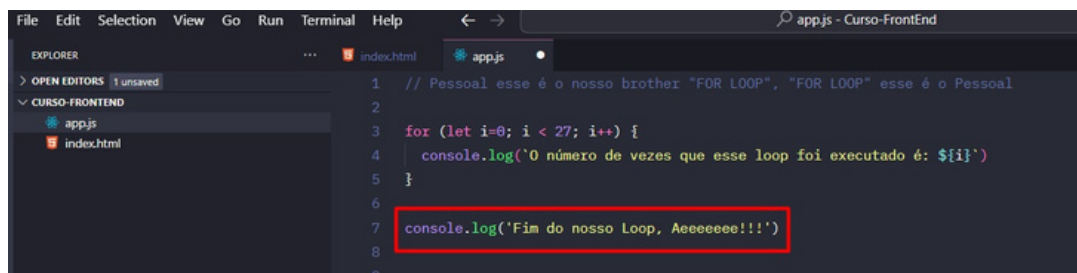
```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; i < 27; i++) {
4   console.log(`O número de vezes que esse loop foi executado é: ${i}`)
5 }
6
```

 A red box highlights the console.log statement.

Acrescentaremos mais um carinha (para que esse código faça mais sentido)... entenderemos o porque disso.... na próxima página....

For loop

Logo no final do fechamento das chaves, pressionaremos o enter 2x e na linha 7 dentro do nosso arquivo app.js, acrescentaremos mais um console.log passando a mensagem ('Fim do nosso Loop, Aeeeeeeee!!!'). Ficando assim:

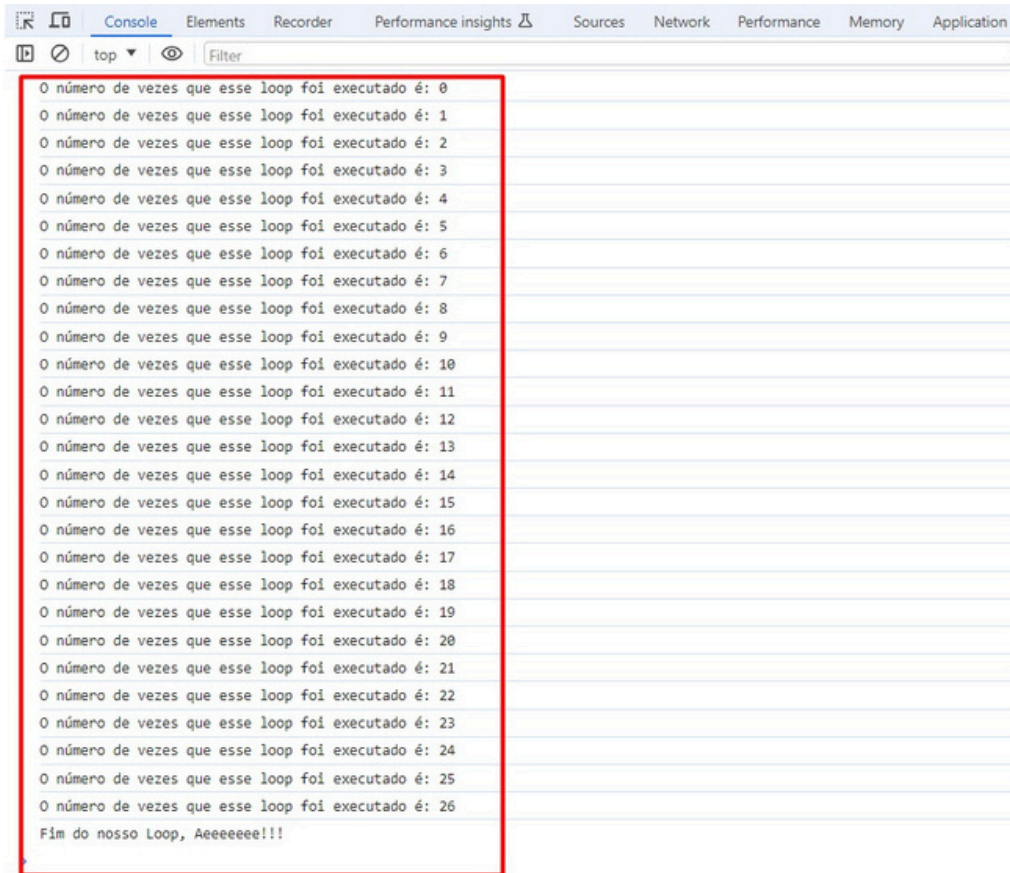


```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 for (let i=0; i < 27; i++) {
4   console.log('O número de vezes que esse loop foi executado é: ${i}')
5 }
6
7 console.log('Fim do nosso Loop, Aeeeeeeee!!!')
```

E ao salvarmos o arquivo app.js com aquele CTRL + S maroto podemos ver no console do Navegador(Chrome) que ele foi rodado diversas vezes, até que a let i guardasse 27. E ao ser guardado 27 o loop terminou e ai o nosso console.log localizado logo abaixo do loop foi chamado e como ele estava a postos, foi feita a sua leitura e o mesmo atuou com sucesso. Ficando assim:

Imagem na próxima página (aqui não coube... ficou grande devido a quantidade loops)

For loop



```
O número de vezes que esse loop foi executado é: 0
O número de vezes que esse loop foi executado é: 1
O número de vezes que esse loop foi executado é: 2
O número de vezes que esse loop foi executado é: 3
O número de vezes que esse loop foi executado é: 4
O número de vezes que esse loop foi executado é: 5
O número de vezes que esse loop foi executado é: 6
O número de vezes que esse loop foi executado é: 7
O número de vezes que esse loop foi executado é: 8
O número de vezes que esse loop foi executado é: 9
O número de vezes que esse loop foi executado é: 10
O número de vezes que esse loop foi executado é: 11
O número de vezes que esse loop foi executado é: 12
O número de vezes que esse loop foi executado é: 13
O número de vezes que esse loop foi executado é: 14
O número de vezes que esse loop foi executado é: 15
O número de vezes que esse loop foi executado é: 16
O número de vezes que esse loop foi executado é: 17
O número de vezes que esse loop foi executado é: 18
O número de vezes que esse loop foi executado é: 19
O número de vezes que esse loop foi executado é: 20
O número de vezes que esse loop foi executado é: 21
O número de vezes que esse loop foi executado é: 22
O número de vezes que esse loop foi executado é: 23
O número de vezes que esse loop foi executado é: 24
O número de vezes que esse loop foi executado é: 25
O número de vezes que esse loop foi executado é: 26
Fim do nosso Loop, Aeeeeeee!!!
```

For loop

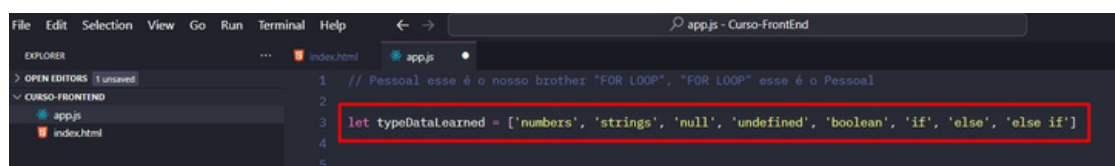
Até aqui vimos um exemplo didático para o entendimento do conceito e para um primeiro contato com o For Loop. Só que na maioria das casos (No dia a dia da nossa profissão) não saberemos quantas vezes teremos que rodar $i < 27$ (tomaremos como base essa aqui) um loop. E porque isso? Pois normalmente esse número de vezes que um loop for feito, será com base em algum dado que receberemos de alguém e ou de algum lugar.

Por exemplo: buscamos de um banco de dados um Array com uma quantidade considerável de tipos de dados e precisamos percorrer dentro desse Array pra que seja efetuada alguma coisa com cada item dentro do Array ou seja com cada tipo de dados, e devido ao tamanho da quantidade de dados, não tem muita lógica contabilizar isso item por item.

Então bora ver.... como podemos fazer isso:

Declararemos uma `let typeDataLearned` que receberá um Array com os tipos de dados que aprendemos nas aulas anteriores `['numbers', 'strings', 'null', 'undefined', 'boolean', 'if', 'else', 'else if']`. Ah.... eu gostaria que fizéssemos de conta (assim como quem não quer nada) que esse Array veio de um banco de dados (veremos isso mais a frente no curso) e não temos ideia de quantos itens tem dentro desse Array e esse Array tem que ser percorridooooooooooooooooo de alguma forma. Ficando assim:

```
let typeDataLearned = ['numbers', 'strings', 'null', 'undefined', 'boolean', 'if', 'else', 'else if']
```

A screenshot of a code editor interface. The top bar shows menu items: File, Edit, Selection, View, Go, Run, Terminal, Help. Below the menu is a search bar with the text 'app.js - Curso-FrontEnd'. The left sidebar shows a file explorer with 'EXPLORER' and 'CURSO-FRONTEND' folders. The main editor area shows a file named 'app.js' with the following code:

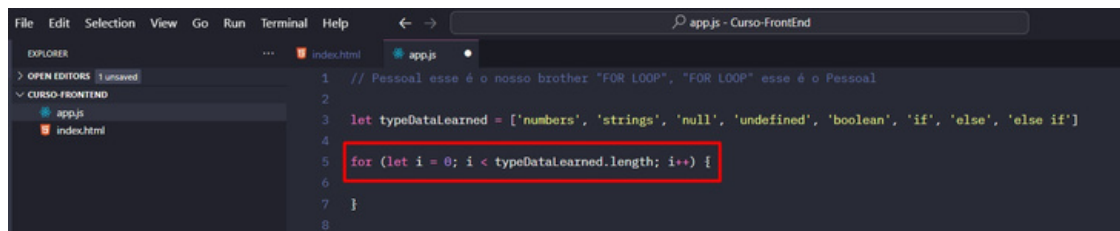
```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 let typeDataLearned = ['numbers', 'strings', 'null', 'undefined', 'boolean', 'if', 'else', 'else if']
4
5
```

The line 3 is highlighted with a red rectangular box.

For loop

Declararemos um for que terá uma let i que receberá 0 e na sequência especificaremos que enquanto i for menor que typeDataLearned.length, a i será incrementada. Ficando assim:

```
for (let i = 0; i < typeDataLearned.length; i++) {  
  
}
```



RESUMO MAROTO E IMPORTANTE:

A expressão `i < typeDataLearned.length` resulta em 8 que é a quantidade de itens dentro do nosso Array `typeDataLearned` e isso é o que garante que conseguiremos rodar algum código com base na quantidade de itens dentro desse Array. Bora ver isso...

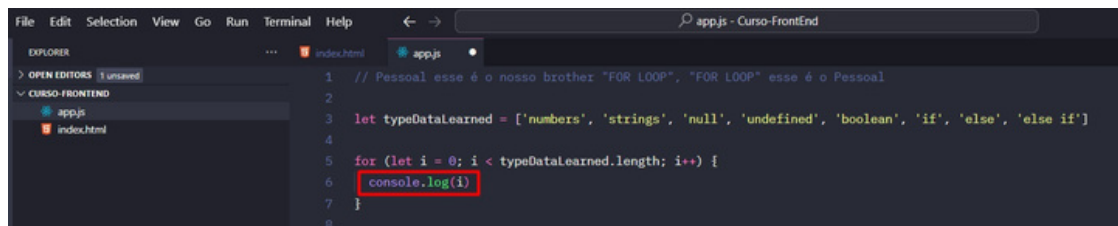
Dentro do for declararemos um `console.log` e inseriremos apenas a i dentro dos parênteses.

Ficando assim:

```
for (let i = 0; i < typeDataLearned.length; i++) {  
    console.log(i)  
}
```



For loop

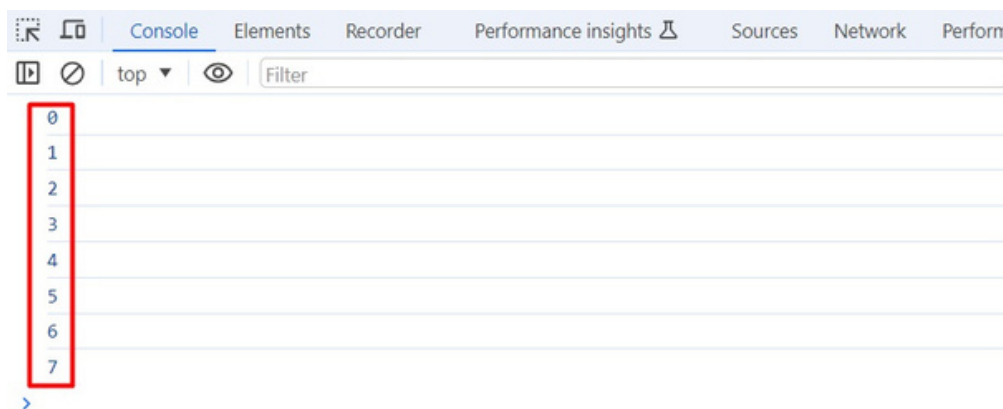


```
1 // Pessoal esse é o nosso brother "FOR LOOP", "FOR LOOP" esse é o Pessoal
2
3 let typeDataLearned = ['numbers', 'strings', 'null', 'undefined', 'boolean', 'if', 'else', 'else if']
4
5 for (let i = 0; i < typeDataLearned.length; i++) {
6   console.log(i)
7 }
8
```

E o que visualizaremos no console?

0, 1, 2, 3, 4, 5, 6 e 7 porque quando i guardar 8, a expressão $i < \text{typeDataLearned.length}$ resultará em falso então esse código localizado aqui dentro do `console.log(i)` não será rodado.

E ao salvarmos o arquivo `app.js` com aquele CTRL + S maroto podemos ver no console do Navegador(Chrome) que 0, 1, 2, 3, 4, 5, 6 e 7 foi mostrado.



Uma última observação: Notem que eu usei o verbo “rodar” ao invés do verbo “executar” durante essa aula, eu usei o rodar fazendo um paralelo com o loop... caso queiram substituir o rodar por executar durante os vossos estudos (sem problema algum).

Sugestões de prática para essa aula:

1. Refazer e Entender os exemplos dessa aula;
2. Usar valores definidos por vocês para praticar mais sobre o For loop com Strings e Numbers;
3. Notem que nos exemplos vistos nessa aula, utilizamos a let e se tivéssemos utilizado a const os resultados teriam sido(s) os mesmos? Fica mais essa sugestão para vocês realizarem!!!

Colocar isso em prática no VSCode e ver o resultado no console do Navegador.

Programação é prática, curiosidade e repetição!!!



/igor-rebolla

O que vamos aprender na aula 28:

Na vigésima oitava aula do curso (Décima Quarta aula de JavaScript) o que veremos: É surpresa – Parte 10 (Vem maisssssssss coisa boa por ai!!!)

Feedbacks dessa vigésima sétima aula são bem-vindos.

Nos vemos na vigésima oitava aula que será disponibilizada no dia 06/06/2024 às 8h30 (Horário de Brasília) – Dentro do meu perfil no LinkedIn.

