



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 03 - LAÇO DE REPETIÇÃO I

O QUE IREMOS APRENDER

- 01** LOOPINGS
- 02** WHILE
- 03** LOOPING INFINITO
- 04** DO WHILE
- 05** BREAK E CONTINUE
- 06** WHILE TRUE
- 07** MÃOS NO CÓDIGO

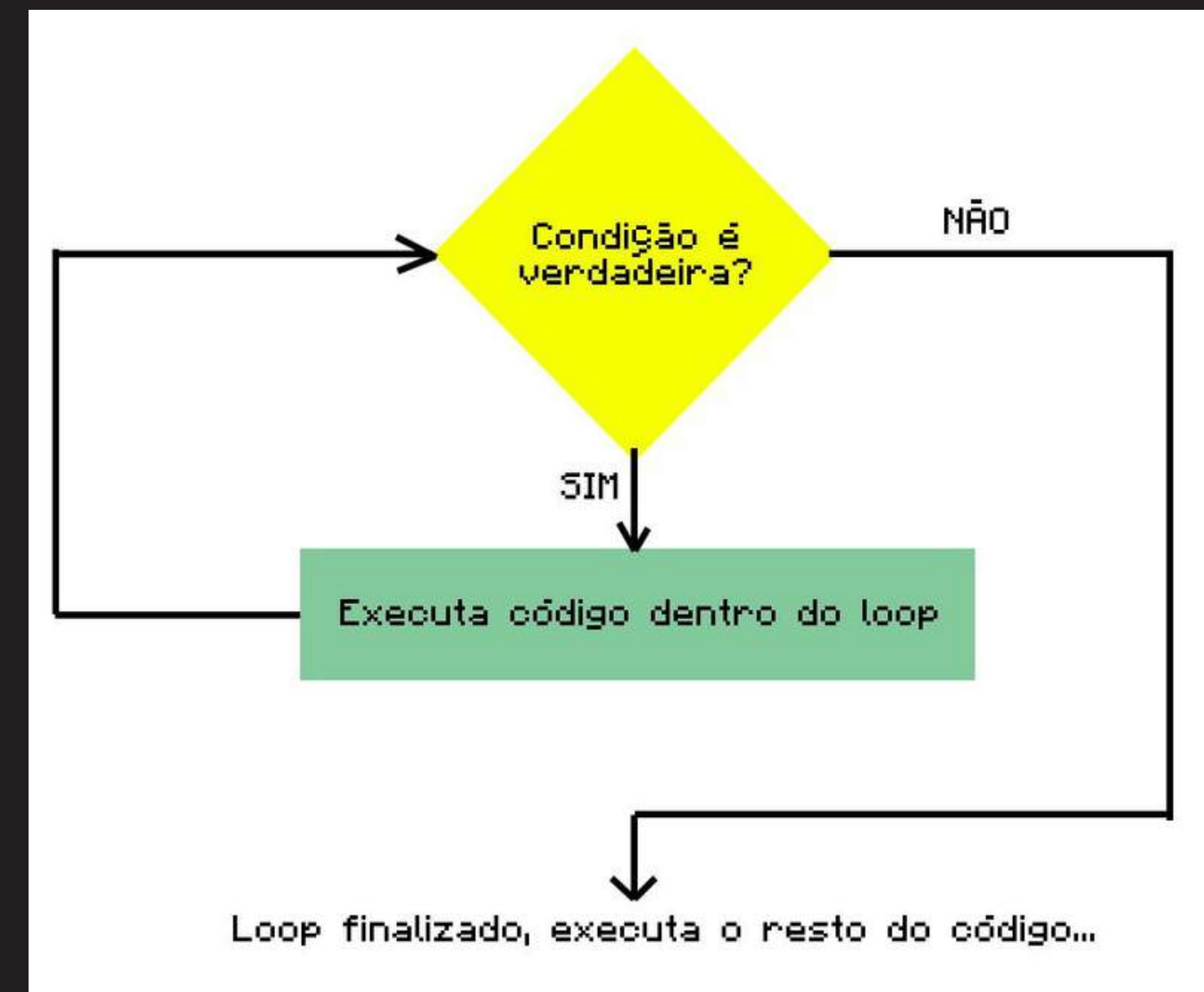
LOOPINGS

Um Laço de Repetição, ou loop, é uma estrutura em programação que repete uma sequência de instruções até que uma condição específica seja atendida. É um recurso que a linguagem de programação fornece para executar as mesmas instruções uma determinada quantidade de vezes.

O JavaScript nos fornece diferentes estruturas de repetição, nesta aula vamos abordar sobre a estrutura WHILE

EXEMPLO DE LOOPINGS

Suponha que você tenha uma caixa de entrada de e-mail e queira verificar cada e-mail para ver se há alguma mensagem importante de um determinado remetente. Você poderia usar um laço de repetição para percorrer todos os e-mails na caixa de entrada e tomar uma ação específica para cada e-mail relevante.



WHILE

O laço de repetição While (Enquanto) segue uma lógica bem simples:

Esse laço executará determinada instrução enquanto determinada condição for verdadeira, a partir do momento em que o teste retornar false, o laço de repetição não será mais executado.

O laço verifica a condição, se for verdadeira, ele executa as instruções que estão dentro das chaves, após isso, ele verifica a condição novamente

WHILE

EXEMPLO:

```
let contador = 0 // Variável inicializada com o valor 0

while (contador < 10) {
  // O laço se repetirá enquanto o contador for menor do que 10
  console.log(contador)
  contador++ //A variável contador está sendo atualizada a cada laço
}
```

LOOPING INFINITO

Quando estamos trabalhando com laços de repetição devemos nos atentar a não programar um looping infinito! Um looping infinito é um laço de repetição que se repete para sempre. Ele consumirá todo o recurso da máquina e irá travar a máquina. Ou então sempre repetirá as mesmas instruções e nunca conseguiremos sair dessas instruções.

LOOPING INFINITO

EXEMPLO:

```
let contador = 0 // Variável inicializada com o valor 0

while (contador < 10) {
    // O laço se repetirá enquanto o contador for menor do que 10
    console.log(contador)
}

/*
Perceba que nesse programa a variável contador
não é atualizada em nenhum momento, ou seja,
ela terá o valor 0 para sempre, portanto,
o while se repetirá para sempre!
*/
```


DO WHILE

Além da estrutura While, temos também o Do While. Sabemos que caso a condição retorne false no primeiro teste, as instruções dentro do while não irão ser executadas.

A estrutura Do While executa pelo menos uma vez as instruções, mesmo que a condição seja falsa, e após essa primeira execução, o laço trabalha como um laço While normalmente.

DO WHILE

A estrutura 'do while' é basicamente: Faça e depois verifique.

```
let contador = 1

do {
  console.log(`Essa instrução foi executada ${contador} vez(es).`)
} while (contador < 0)
```

```
/*
Perceba que ele retornará apenas uma vez
devido ao "do", pois se não houvesse ele
não retornaria nada, pois 0 não é maior que 1.
*/
```

BREAK E CONTINUE

Conforme vamos incrementando nossos programas, os laços de repetição vão ficando mais complexos também. Diante desse cenário, é importante que tenhamos mais controle do fluxo do programa. O JavaScript nos dá esse controle através das instruções Break e Continue.

O Break (quebrar) irá quebrar a estrutura de repetição, saindo dela.


O Continue irá apenas quebrar aquele laço e partir para o próximo.

BREAK

O comando `break` é usado para sair ou terminar imediatamente o loop. Ele permite interromper prematuramente a execução do loop e continuar com o código após o loop.

```
let contador = 0

while (contador < 10) {
  console.log(contador)
  if (contador === 5){
    // Quando o contador for igual a 5, o laço será quebrado
    break
  }
  contador ++
}
console.log(`O laço foi quebrado e contador parou com o valor ${contador}`)
```



```
0
1
2
3
4
5
O laço foi quebrado e contador parou com o valor 5
```

CONTINUE

A instrução continue permite quebrar o laço atual e pular para o próximo laço. Diferente do break, que sai da estrutura, o continue quebra apenas o laço atual, sem sair da estrutura de repetição, e vai para o próximo laço

```
let contador = 0

while (contador < 10) {
  contador ++
  if (contador === 5){
    // Quando o contador for igual a 5, o laço será quebrado
    continue
  }
  console.log(contador)
}
console.log(`O laço finalizou e contador parou com o valor ${contador}`)
//Perceba que não houve o print do valor 5
```

1
2
3
4
6
7
8
9
10

O laço finalizou e o contador parou com o valor 10

WHILE TRUE

O while true é uma construção comum em várias linguagens de programação para criar um loop infinito. Ele é utilizado quando se deseja que um bloco de código seja executado continuamente, sem parar, a menos que seja explicitamente interrompido. A expressão true é uma constante booleana que sempre avalia para verdadeiro, garantindo que a condição do while seja sempre satisfeita.

```
while (true) {  
    console.log("Este é um looping infinito!")  
}
```

WHILE TRUE

Nesse exemplo utilizamos o `while true` para construir um loop infinito. Solicitamos nomes para o usuário e verificamos se o nome é igual a 'raama'. Se for, iremos interromper o código e encerrar o programa.

```
while (true){  
  
  let nome = prompt('Digite um nome').toLowerCase()  
  console.log(nome)  
  
  if(nome == 'raama'){  
    console.log('Aqui vai quebrar')  
    break  
  }  
}
```


ATIVIDADE PRÁTICA

Atividade 01

Crie um programa que solicita ao usuário um número e, em seguida, conta regressivamente até zero, imprimindo cada número no console. O programa deve terminar quando atingir zero.

Atividade 02

Desenvolva um programa que solicita ao usuário que insira suas notas de uma série de disciplinas. O programa deve calcular e imprimir a média das notas, o usuário pode continuar adicionando notas até decidir parar.

ATIVIDADE PRÁTICA

Atividade 03

Peça ao usuário para inserir números continuamente e imprima o somatório dos números inseridos. O programa deve continuar executando até que o usuário insira zero.

Atividade 04

Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.

ATIVIDADE PRÁTICA

Atividade 05

Faça um programa que peça para 5 pessoas a sua idade, ao final o programa devera verificar se a média de idade da turma varia entre 0 e 25, 26 e 60 e maior que 60; e então, dizer se a turma é jovem, adulta ou idosa, conforme a média calculada.

Atividade 06

Calcule a soma dos números de 1 a 50

ATIVIDADE PRÁTICA

Atividade 06

Crie um programa que solicite um número de 1 a 7, representando o dia da semana. Use uma estrutura switch (ou equivalente) para imprimir o nome do dia correspondente.

Atividade 07

Escreva um programa que solicite um número e determine se ele é um número primo.

DESAFIO PRÁTICO

Você foi desafiado a criar um simulador de caixa eletrônico em JavaScript. O programa deve permitir que o usuário interaja com sua conta bancária simulada, realizando operações como verificar o saldo, fazer saques, fazer depósitos e sair do programa.

Aqui estão os requisitos do programa:

O programa deve começar com um saldo inicial de \$1000.

Utilize um loop while para manter o programa em execução até que o usuário escolha sair.

DESAFIO PRÁTICO

A cada iteração do loop, o programa deve
exibir um menu com as seguintes opções:

Ver Saldo

Fazer Saque

Fazer Depósito

Sair

Se o usuário escolher "Ver Saldo", o programa
deve exibir o saldo atual.

DESAFIO PRÁTICO

Se o usuário escolher "Fazer Saque", o programa deve solicitar o valor do saque. Certifique-se de validar se o valor é numérico, maior que zero e não excede o saldo disponível. Se a validação for bem-sucedida, atualize o saldo e exiba uma mensagem indicando o sucesso da transação.

DESAFIO PRÁTICO

Se o usuário escolher "Fazer Depósito", o programa deve solicitar o valor do depósito. Certifique-se de validar se o valor é numérico e maior que zero. Se a validação for bem sucedida, atualize o saldo e exiba uma mensagem indicando o sucesso da transação.

Se o usuário escolher "Sair", encerre o programa exibindo uma mensagem de despedida.

Se o usuário escolher uma opção inválida, exiba uma mensagem indicando isso e permita que o usuário faça uma nova escolha.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 04 DE JAVASCRIPT.
LAÇOS DE REPETIÇÃO II.



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

FOR

Assim como o while, o for também nos permite repetir um trecho de código enquanto uma condição for atendida. Mas utilizando o while, sempre que precisamos de um contador em nosso laço de repetição para executar um número específico de vezes, precisamos inicializar uma variável antes de começar o nosso while. O for vem para resolver esse problema, nos proporcionando um controle muito maior sobre a quantidade de vezes que aquele código irá rodar.



FOR

Exemplos de **laço de repetição For** em JavaScript:

```
for (let i = 0; i < 3; i++) {  
    console.log(i) // Imprime o valor de i no console  
    console.log('Passei por um ciclo') // Imprime a mensagem no consol  
e  
}
```



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 03 – LAÇO DE REPETIÇÃO I