Boas-vindas!

Agora você faz parte da maior comunidade de aprendizagem online e ao vivo da América Latina!

Como você chega?

1

2

3

CODERHOUSE

Esta aula será

- gravada

Resumo
da aula anterior

- ✓ Definição de escopo
- ✓ Escopo global
- ✓ Escopo de função
- ✓ Escopo de bloco
- ✓ Declarações com let e const
- ✓ Entendendo o que é hosting

CORREÇÃO ATIVIDADE EXTRA

Descrição

Usando funções desenvolva um sistema que determine se dois número são múltiplos:

- ✓ Inserir números a serem verificados
- ✓ Verificar se um é múltiplo do outro
- ✓ Mostrar resultado.

Observação

- ✓ Não há entrega dessa atividade.

O tutor irá corrigir e tirar as dúvidas.

Dúvidas?

Aula 06. Javascript

Ciclos e Iterações

CODERHOUSE

Objetivos da aula

- Compreender o que é um ciclo (ou loop) e como nos permite repetir operações similares de maneira fácil.
- Conhecer os tipos de laços que podemos utilizar e quais as diferenças entre eles.
- Analisar como combinar operadores lógicos, laços e funções para resolver problemas.

Ciclos

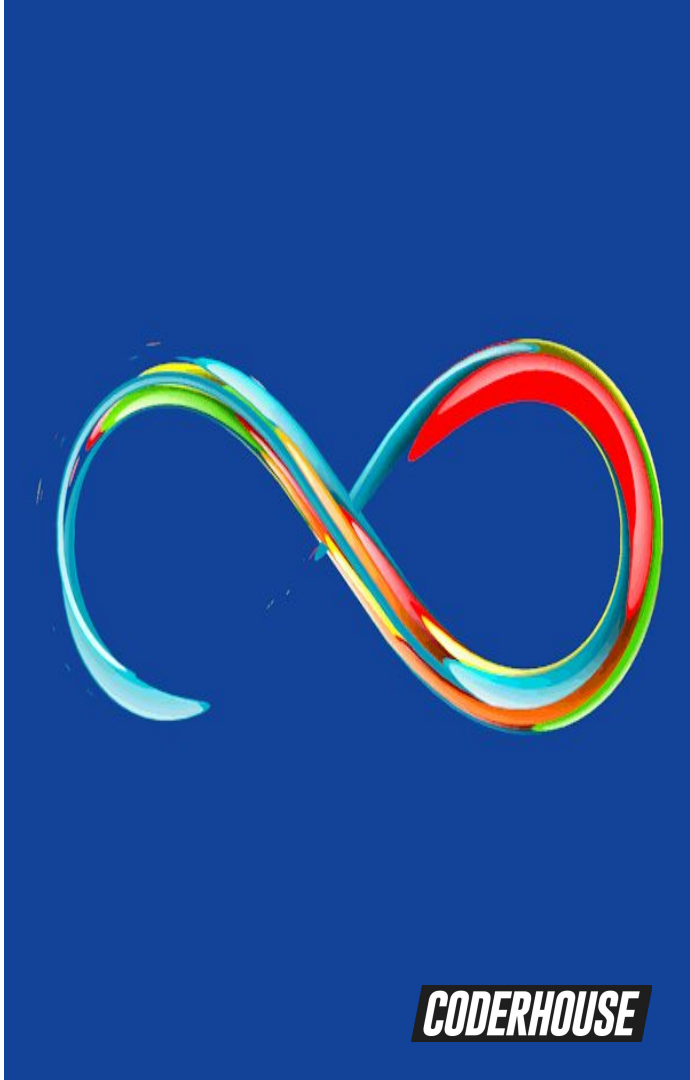
Ciclos no Javascript

Os ciclos, também conhecidos como laços, loops ou iterações, são **uma forma rápida e simples de fazer algo repetidamente.**
Se temos que fazer alguma operação no programa mais de uma vez, de forma consecutiva, usaremos as estruturas de loop do JavaScript: for, while ou **do...while.**

Tipos de loops

Laços por contagem
Repetem um bloco de código por um determinado número de vezes. Estrutura **for**.

Laços condicionais
Repetem um bloco de código enquanto a condição verificada for verdadeira. Estruturas **while e do...while**.



For

Sintaxe do for

O **“início”** é o local em que se estabelecem os valores iniciais das variáveis que controlam o loop.
A **“condição”** é o único elemento que decide se o que está contido no loop será repetido.
A **“atualização”** é o novo valor que se atribui às variáveis que controlam a repetição ao final de cada loop.

```
for (início; condição; atualização) {
    //o que está escrito aqui será executado
enquanto durar o loop
}
```

Exemplo prático

Neste exemplo, utilizamos um **for** para contar de 0 a 9.

```
for (let i = 0; i < 10; i++) {
    console.log(i);
}
```

Agora, usamos um **for** para contar de 1 a 10.

```
for (let i = 1; i <= 10; i++) {
    console.log(i);
}
```

Exemplo aplicado: Tabuada

Algoritmo para calcular a tabuada de um número.

```
// Solicitamos um valor ao usuário
let numero = parseInt(prompt("Inserir Número"));

// A cada repetição, calculamos o número inserido vezes o número
da repetição (i)
for (let i = 1; i <= 10; i++) {
    let resultado = numero * i;
    console.log(numero + " X " + i + " = " + resultado);
}
```

#FindTheBug

Vamos analisar o código para encontrar o que está causando um bug.

```
let numero = parseInt(prompt("Inserir Número"));

function produtorio(numero) {
    let resultado = 1;
    for (let i = 0; i <= numero; i++) {
        resultado = resultado * i;
    }
    console.log("Produtorio de " + numero + " = " + resultado);
}

produtorio(numero)
```

CODERHOUSE

Sentença Break

Às vezes, quando escrevemos uma estrutura **for**, precisamos que o laço se interrompa conforme certa condição.
Para isso, utilizamos a **sentença break**.
Ao escrever essa linha dentro de um laço **for**, ele será interrompido como se tivesse finalizado todas as repetições.

```
for (let i = 1; i <= 5; i++) {
    //Se a variável i for igual à 3, o for
será interrompido.
    if (i == 3) {
        break;
    }
    console.log(i);
}
```

CODERHOUSE

Sentença Continue

Às vezes, quando escrevemos uma estrutura **for**, precisamos que o laço, em certa condição, salte a repetição atual e siga para a próxima. Para isso, utilizamos a **sentença continue.**

```
for (let i = 1; i <= 5; i++) {
    //Se a variável i for 3, não se
interpreta essa repetição.
    if (i == 3) {
        continue;
    }
    console.log(i);
}
```

CODERHOUSE

Para praticar!

Duração: **10 minutos**

ATIVIDADE EM SALA

Para Praticar

Descrição
Utilizando o “for” crie uma função que:

- ✓ Receba um número de 1 a 10
- ✓ Exiba todos os número que são menores que ele

Exemplos:

Se for enviado o número 4, o retorno será 1, 2, 3
Se for enviado o número 7, o retorno será 1, 2, 3, 4, 5, 6

Break

5 minutos e voltamos!

Break

10 minutos e voltamos!

CODERHOUSE

While

While

Quando usamos **while,** assumimos que em algum momento a repetição será finalizada.
Se a comparação não se realiza adequadamente podemos gerar o chamado "loop infinito"

```
let repetir = true;
while (repetir) {
    console.log("Ao infinito
e...Além!");
}
```

```
17784 Ao infinito e...Além!
>
```

CODERHOUSE

Exemplo aplicado While: Sair

Algoritmo que solicita uma entrada ao usuário
até que ele(a) insira a palavra “sair”.

```
let entrada = prompt("Inserir um dado");
//Repetimos com while até que o usuário digite "sair"

while (entrada != "sair") {
    alert("O usuário inseriu " + entrada);
    //Solicitamos novamente um dado
    //Na próxima iteração, será verificado se não é "sair"
    entrada = prompt("Inserir outro dado");
}
```

Para praticar!

Duração: **10 minutos**

ATIVIDADE EM SALA

Para Praticar

Descrição
Utilizando o “while” crie uma função que:

- ✓ Recebe um número e calcula o seu fatorial
- ✓ O fatorial de um número é o produto dele pelos seus antecessores maiores que 0.

Exemplos:

- ✓ Se for enviado o número 4, o resultado será 1 x 2 x 3 x 4 = 24
- ✓ Se for enviado o número 7, o retorno será 1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040

Do...While

A estrutura **do...while** permite criar loops que são executados **uma ou mais vezes,** dependendo da condição indicada.
A diferença da estrutura *do...while* é que ela **garante que o bloco de código seja interpretado ao menos uma vez, porque a condição é verificada apenas no final.**

```
let repetir = false;
do {
    console.log("Só uma vez!");
} while (repetir)
```

CODERHOUSE

Exemplo aplicado Do...While: Número

Algoritmo que solicita uma entrada e se encerra quando não for um número.

```
let numero = 0;
do {
    //Repetimos com do...while enquanto o usuário não inserir um número
    numero = prompt("Inserir um número");
    console.log(numero);
    //O loop será interrompido quando não digital um número
} while (parseInt(numero));
```

Switch

Switch

A estrutura **switch** é especialmente projetada para administrar **múltiplas condições sobre a mesma variável** de forma simples (tecnicamente, também poderia ser resolvido com um **if**, mas o uso do **switch** é mais ordenado).
Sua definição formal pode parecer confusa, então vejamos a seguir um exemplo para entender sua simplicidade.

Sintaxe do Switch

Cada condição é verificada e, se for cumprida, será executado o que estiver especificado dentro de cada ***case***.
Normalmente, depois das instruções de cada *case*, é incluída a sentença **break** para interromper a execução do **switch**, mesmo que não seja obrigatório.

*O que acontece se nenhum valor da variável do switch corresponde aos valores definidos em algum dos **case**?*

Nesse caso, se utiliza o valor **default** para indicar as instruções que serão executadas quando nenhuma condição anterior for cumprida.

```
switch (numero) {
    case 5:
      ...
        break;

    case 8:
      ...
        break;

    case 20:
      ...
        break;

    default:
      ...
        break;
}
```

CODERHOUSE

Sintaxe do Switch

Algoritmo que exibe a mensagem de acordo com o nome de entrada.
A execução do bloco se encerra se a entrada for “sair”.

```
let entrada = prompt("Inserir um
nome");
//Repetimos até que "sair" seja
inserido.
while (entrada != "sair") {
    switch (entrada) {
        case "ANA":
            alert("OLÁ, ANA");
            break;
        case "JOÃO":
            alert("OLÁ, JOÃO");
            break;
        default:
            alert("QUEM É VOCÊ ? ")
            break;
    }
    entrada = prompt("Inserir um
nome");
}
```

CODERHOUSE

Exemplo aplicado While e Switch: Entradas

Algoritmo que exibe a mensagem de acordo com o nome de entrada.
A execução do bloco se encerra se a entrada for “sair”.

```
let entrada = prompt("Inserir um nome");
//Repetimos até que "sair" seja inserido.
while (entrada != "sair") {
    switch (entrada) {
        case "ANA":
            alert("OLÁ, ANA");
            break;
        case "JOÃO":
            alert("OLÁ, JOÃO");
            break;
        default:
            alert("QUEM É VOCÊ ? ")
            break;
    }
    entrada = prompt("Inserir um nome");
}
```

Para praticar!

Duração: **10 minutos**

ATIVIDADE EM SALA

Para Praticar

Descrição
Utilizando o “switch” crie uma função que:

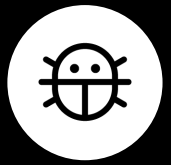
- ✓ Utilizando o “switch” crie uma função que:
- ✓ Recebe um número de 1 a 100
- ✓ Calcula em qual dezena ele está

Exemplos:

- ✓ Se for enviado o número 33, o resultado será “família do 30”
- ✓ Se for enviado o número 9, o resultado será “família do 10”
- ✓ Se for enviado o número 80, o resultado será “família do 80”

O mais importante!

Todas os temas que vimos (e os que vamos ver), podem e devem ser combinados entre si.
De forma que, dentro de uma função, possa existir um **condicional** contendo um **for**, e dentro desse for, um **while**... deixando a combinação infinita.
A partir deste ponto é que a programação JavaScript começa a ficar interessante!



#FindTheBug

Vamos analisar o código para encontrar o que está causando um bug.

```
let entrada = prompt("Inserir um nome");

while (entrada) {
    alert("Você digitou: ", entrada);
}
```

CODERHOUSE

Perguntas?

Como foi a aula?

1

Que bom

O que foi super legal na aula e podemos sempre trazer para as próximas?

2

Que pena

O que você acha que não funcionou bem e precisamos melhorar?

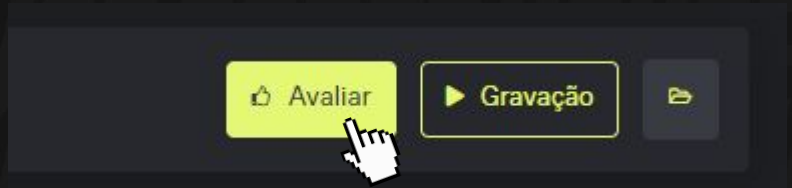
3

Que tal

Qual sugestão deveríamos tentar em próximas aulas?

CODERHOUSE

O que você achou da aula?



Seu feedback vale pontos para o Top 10!! 😎

Deixe sua opinião!

1. Acesse a plataforma
2. Vá na aula do dia
3. Clique em Avaliar

CODERHOUSE

Inserir um ciclo no simulador

DESAFIO COMPLEMENTAR

CODERHOUSE

DESAFIO COMPLEMENTAR

Inserir um ciclo no simulador

Descrição:

- ✓ Tomando como base os exemplos anteriores da estrutura for e while, criar um ciclo que repita um bloco de instruções no seu simulador.
- ✓ A cada repetição, é necessário efetuar uma operação ou comparação para obter uma saída por alert ou console.

Recomendação:

- ✓ Usamos a instrução “for” para realizar a repetição por um número fixo de vezes.
- ✓ Já o “while”, usamos quando quisermos repetir algo até que uma condição deixe de ser cumprida.

Exemplos:

- ✓ Solicitar a quantidade de parcelas e exibir o valor de cada uma, caso sejam parcelas simples.
- ✓ Simular um empréstimo e exibir o valor de parcelas decrescentes.
- ✓ Solicitar uma entrada através de prompt, manipular o valor a cada repetição, realizando uma saída para cada resultado, até que digite “sair”.
- ✓ Inserir valores, e exibir resultados de soma e média a cada novo valor inserido.

Formato:

- ✓ Link do código no GitHub e da publicação no GitHub Pages

Resumo
da aula de hoje

- ✓ O que é loop;
- ✓ Estrutura for;
- ✓ Estrutura while ;
- ✓ Estrutura do...while;
- ✓ Estrutura switch;
- ✓ Combinação de operadores lógicos, laços e funções.

Ainda quer saber mais?
Recomendamos o seguinte material

MATERIAL AMPLIADO

Recursos multimídia

Referências

- ✓ Laços em JavaScript **| Freecodecamp**
- ✓ “For” em Javascript **| DevMedia**
- ✓ Código em loop **| Developer Mozilla**

Obrigado por estudar conosco!