[CT JS] Aula 3

♣ Conteúdos

Laços de repetição - While

While:

O `while` é uma estrutura de controle de fluxo em JavaScript que permite executar um bloco de código repetidamente enquanto uma condição especificada for verdadeira. A estrutura básica é:

```
while (condicao) { // Código a ser executado enquanto a condição for verdadeira }
```

- O bloco de código dentro do `while` é repetido enquanto a condição dentro dos parênteses for verdadeira.
- Antes de cada execução, a condição é avaliada. Se a condição for falsa, a execução do bloco é interrompida.

Exemplo:

```
let contador = 1; while (contador <= 5) { console.log(`Contagem: ${contador}`); contador++; }
```

Neste exemplo, o bloco de código dentro do `while` é executado enquanto o contador for menor ou igual a 5.

> 💡 **Quando usar While?**
>
> - Você quer repetir um bloco de código apenas se a condição inicial for verdadeira.
> - Pode ser que o bloco de código nunca seja executado se a condição inicial já for falsa.

Do-While:

O `do-while` é semelhante ao `while`, mas com uma diferença crucial: o bloco de código é executado pelo menos uma vez, mesmo se a condição for falsa. A estrutura básica é:

```
do { // Código a ser executado } while (condicao);
```

- O bloco de código é executado uma vez antes da avaliação da condição.
- Se a condição for verdadeira, o bloco é repetido. Caso contrário, a execução é interrompida.

Exemplo:

```
let numero; do { numero = prompt("Digite um número maior que 5:"); } while (numero <= 5); console.log(`Número digitado: ${numero}`);
```

Neste exemplo, o usuário é solicitado a digitar um número maior que 5. O bloco de código é executado pelo menos uma vez, mesmo que o usuário digite um número menor ou igual a 5, pois a condição é avaliada após a primeira execução.

💡 **E quando usar do-while?**

- Você quer garantir que o bloco de código seja executado pelo menos uma vez, independentemente da condição.
- A condição é avaliada após a execução do bloco de código.

▼ Laços de repetição - For

O `for` é uma estrutura de controle de fluxo em JavaScript que permite criar loops, ou seja, repetir um bloco de código várias vezes. Ele é especialmente útil quando você sabe antecipadamente quantas vezes deseja repetir uma ação.

A estrutura básica de um loop `for` é composta por três partes:

```
for (inicialização; condição; incremento/decremento) { // Código a ser executado a cada iteração }
```

- **Inicialização:** É a parte onde você inicializa a variável de controle do loop. Geralmente, é aqui que você define a variável que será usada para contar as iterações.
- **Condição:** É a parte que verifica se o loop deve continuar executando. Enquanto a condição for verdadeira, o código dentro do loop é executado.
- **Incremento/Decremento:** É a parte onde você atualiza a variável de controle, geralmente incrementando ou decrementando seu valor. Isso é feito para evitar que o loop seja executado indefinidamente.

Exemplo Simples:

Vamos criar um loop `for` que imprime os números de 1 a 5:

```
for (let i = 1; i <= 5; i++) { console.log(i); }
```

for-of:

O `for-of` é utilizado para iterar sobre os elementos de **objetos iteráveis**, como arrays e strings.

Sintaxe:

```
for (const elemento of iterable) { // Código a ser executado para cada elemento }
```

Exemplo com Array:

```
const numeros = [1, 2, 3, 4, 5]; for (const numero of numeros) {
console.log(numero); } // Saída: 1, 2, 3, 4, 5
```

Exemplo com String:

```
const texto = "Hello"; for (const letra of texto) { console.log
(letra); } // Saída: H, e, l, l, o
```

for-in:

O `for-in` é utilizado para iterar sobre as **propriedades enumeráveis de um objeto**, incluindo propriedades herdadas. Ele não é recomendado para iterar sobre arrays.

Sintaxe:

```
for (const chave in objeto) { // Código a ser executado para cad
a propriedade }
```

Exemplo com Objeto:

```
const pessoa = { nome: "João", idade: 25, cidade: "Exemplo" }; f
or (const chave in pessoa) { console.log(`${chave}: ${pessoa[cha
ve]}`); } // Saída: nome: João, idade: 25, cidade: Exemplo
```

Nota Importante: Ao usar `for-in` em arrays, pode haver comportamentos inesperados, pois ele itera sobre as propriedades do protótipo do array, além dos índices. Portanto, em arrays, é preferível usar `for-of` para iterar sobre os valores diretamente.

💡 **Quando usar?**

`for:`

- Para iterar sobre índices de arrays ou sequências numéricas fixas.
- Quando você precisa de maior controle sobre o índice de iteração.
- Em casos em que a ordem de iteração é crucial.

`for-in:`

- Para iterar sobre as propriedades enumeráveis de um objeto, incluindo propriedades herdadas.
- Quando a ordem específica das chaves não é relevante.

`for-of:`

- Para iterar sobre valores de objetos iteráveis como arrays, strings, mapas, conjuntos, etc.
- Quando você está mais interessado nos **valores** do que nos **índices**.

▸ **Busca simples em vetores e matrizes**

♧ Exercicios

> Link para alunos
> [CT JS] Exercícios Aula 3

While

1. **Contagem Regressiva**

   Você está desenvolvendo um temporizador e precisa criar uma contagem regressiva a partir de um número inicial. Crie um código que utilize um loop para imprimir os números de 10 a 1.

```
//Resolução let contador = 10; while (contador >= 1) { conso
le.log(`Contagem regressiva: ${contador}`); contador--; }
```

2. **Validador de número para jogo**

   Você está criando um trecho de jogo e quer garantir que o jogador insira uma resposta válida. Crie um código que use um loop para solicitar ao jogador que insira um número entre 1 e 5 até que insira uma resposta válida (um valor entre 1 e 5).

```
//Resolução let resposta; do { resposta = prompt("Insira um
número entre 1 e 5:"); resposta = parseInt(resposta); // Con
verter para número inteiro } while (isNaN(resposta) || respo
sta < 1 || resposta > 5); console.log(`Parabéns! Você inseri
u um número válido: ${resposta}`);
```

3. **Calculando o saldo da conta**

   Você está desenvolvendo um programa de finanças e precisa calcular o saldo acumulado a partir de uma lista de transações. Crie um programa que use um loop para calcular o saldo acumulado.

```
//Resolução let transacoes = [50, -20, 30, -10, 40]; let sal
do = 0; let indice = 0; while (indice < transacoes.length) {
saldo += transacoes[indice]; indice++; } console.log(`Saldo
acumulado: ${saldo}`);
```

4. **Lista de números primos**

   Você deseja gerar uma lista de números primos no intervalo de 1 a 20. Crie um programa que utilize um loop para imprimir os números primos dentro desse intervalo.

```
//Resolução let numero = 2; do { let ehPrimo = true; let div
isor = 2; while (divisor < numero) { if (numero % divisor ==
= 0) { ehPrimo = false; break; } divisor++; } if (ehPrimo) {
console.log(`Número primo: ${numero}`); } numero++; } while
(numero <= 20);
```

For

1. **Lista de compras inteligente**

   Você está fazendo uma lista de compras e quer garantir que não esquecerá nada. Retorne uma lista de compras única sem elementos duplicados.

```
const lista1 = ["Maçã", "Banana", "Leite", "Ovos"]; const li
sta2 = ["Banana", "Ovos", "Arroz", "Feijão"];
```

```
//Resolução const lista1 = ["Maçã", "Banana", "Leite", "Ovo
s"]; const lista2 = ["Banana", "Ovos", "Arroz", "Feijão"]; c
onst listaUnica = lista1.slice(); for (const elemento of lis
ta2) { if (!listaUnica.includes(elemento)) { listaUnica.push
(elemento); } } console.log(listaUnica.sort(););
```

2. **Calculando média de notas**

   Você é um professor e deseja calcular a média das notas dos seus alunos. Retorne a média das notas recebidas no array.

```
const notas = [8, 7, 6, 9, 5];
```

```
//Resolução const notas = [8, 7, 6, 9, 5]; let soma = 0; for
(let i = 0; i < notas.length; i++) { soma += notas[i]; } con
st media = soma / notas.length; console.log(media);
```

3. **Refazendo a lista de números primos**

   Você deseja refatorar o programa anterior que gera uma lista de números primos no intervalo de 1 a 20. Reescreva seu código utilizando o `for` para imprimir os números primos dentro desse intervalo.

```
let numero = 2; do { let ehPrimo = true; for (let i = 2; i < numero; i++) { if (numero % i === 0) { ehPrimo = false; break; } } if (ehPrimo) { console.log(`Número primo: ${numero}`); } numero++; } while (numero <= 20);
```

4. **Saldo de transações**

   Você está construindo um sistema financeiro e precisa calcular a soma total de uma lista de transações. Crie um programa que utilize `for-of` para iterar sobre os elementos de um array de transações e calcule a soma total.

```
const transacoes = [120, -50, 30, -20, 40];
```

```
//Resolução const transacoes = [120, -50, 30, -20, 40]; let somaTotal = 0; for (const valor of transacoes) { somaTotal += valor; } console.log(`Soma total das transações: ${somaTotal}`);
```

5. **Listando o usuário**

Busca simples em vetores

1. **Encontrando Usuário por Nome e Índice de Usuário por Email**

Você está construindo um sistema de gerenciamento de usuários. Todos os usuários são adicionados em um array `usuarios`, e cada objeto representa um desses usuários. Cada objeto possui as propriedades `nome`, `email` e `idade`. Escreva um código que verifique se o usuário `Alice` existe e qual o índice do usuário que possui o email `carlos@example.com`

```
const usuario = { nome: "Alice", idade: 28, email: "alice@example.com" };
```

```
const usuarios = [ { nome: "João", email: "joao@example.com", idade: 25 }, { nome: "Alice", email: "alice@example.com", idade: 30 }, { nome: "Carlos", email: "carlos@example.com", idade: 22 } ];
```

```
//Resolução const usuarios = [ { nome: "João", email: "joao@example.com", idade: 25 }, { nome: "Alice", email: "alice@example.com", idade: 30 }, { nome: "Carlos", email: "carlos@example.com", idade: 22 } ]; const usuarioEncontrado = usuarios.find(user => user.nome === "Alice"); console.log("Usuário Encontrado:", usuarioEncontrado); const indiceUsuario = usuarios.findIndex(user => user.email === "alice@example.com"); console.log("Índice do Usuário:", indiceUsuario);
```

2. **Verificando Idade Mínima e Filtrando Usuários Menores de 25 Anos**

Você está desenvolvendo um sistema de verificação de idade. Você recebe um array de usuários com informações sobre idade. Verifique se todos os usuários têm pelo menos 18 anos e retorne uma lista de usuários com