



For e While



Resumo

Instruções FOR

Nesta seção, apresentarei as instruções “for” em JavaScript, uma ferramenta poderosa para realizar iterações em arrays. O “for” é amplamente utilizado na programação e oferece uma maneira eficiente de percorrer e manipular elementos de um array.

A sintaxe do “for” é composta por três partes: a condição de inicialização, a condição de teste e o incremento/decremento. Essas partes são colocadas entre parênteses e separadas por ponto-e-vírgula. O bloco de código a ser executado em cada iteração é delimitado por chaves.

Vamos começar com um exemplo prático utilizando um array de carros. Cada carro é representado por um objeto contendo informações como ID, modelo, marca, preço e data de criação. Nosso objetivo será percorrer esse array e obter o preço de cada carro.

Para isso, utilizaremos uma variável de controle chamada “i”. Inicializaremos essa variável com o valor zero e definiremos a condição de teste: enquanto “i” for menor que o tamanho do array de carros, continuaremos a iteração. A cada iteração,

adicionaremos 1 ao valor de “i” usando o operador de incremento

“”
+



Dentro do loop, teremos acesso ao preço de cada carro e poderemos realizar diversas ações com esses valores, como somar ao total ou exibir na tela. Ao final das iterações, teremos obtido o total dos preços dos carros ou realizado outras operações desejadas.

Além disso, exploraremos outras formas de iteração, como o “forin” e o “forof”. Essas instruções oferecem abordagens diferentes para acessar os elementos do array, proporcionando mais flexibilidade e facilitando o trabalho com estruturas de dados complexas.

É importante ressaltar que não existe uma única forma correta de utilizar o “for”. A escolha da abordagem adequada dependerá do contexto e dos objetivos do seu código. Por isso, é fundamental compreender as diferentes opções e saber aplicá-las conforme a situação.

Esta introdução às instruções “for” em JavaScript é apenas o começo. Em seguida, aprofundaremos ainda mais esse tema, abordando exemplos práticos e explorando outras funcionalidades dessa poderosa estrutura de controle.

Fique atento para os próximos conteúdos, onde continuaremos a explorar os recursos e possibilidades das instruções “for”. Não deixe de praticar e experimentar por conta própria, pois isso ajudará a aprimorar suas habilidades em programação com JavaScript.”


Agora, vou apresentar alguns exemplos de código para ilustrar as instruções “for”:



Exemplo 1: Iteração em um array de carros

```
const carros = [  
  
  { id: 1, modelo: "Corsa", marca: "Chevrolet", preco: 45000,  
    dataCriacao: "01/01/2022" },  
  
  { id: 2, modelo: "Punta", marca: "Fiat", preco: 35000, dataCriacao:  
    "01/02/2022" },  
  
  { id: 3, modelo: "Gol", marca: "Volkswagen", preco: 40000,  
    dataCriacao: "01/03/2022" }  
];  
  
let totalPrecos = 0;  
  
for (let i = 0; i < carros.length; i) {  
  
  const carro = carros[i];  
  
  totalPrecos += carro.preco;  
  
}  
  
console.log("Total dos preços dos carros:", totalPrecos);
```

Neste exemplo, temos um array de carros representado pela variável `carros`. Utilizamos um loop “for” para percorrer cada elemento do array e acessar a propriedade `preco` de cada carro. A cada iteração, somamos o preço ao valor da variável

totalPrecos. No final, exibimos o total dos preços dos carros no console. 

Exemplo 2: Iteração usando “forin” e “forof”

```
const carros = ["Corsa", "Punta", "Gol"];

console.log("Iteração usando forin:");

for (let indice in carros) {

    console.log("Índice:", indice, "Valor:", carros[indice]);

}

console.log("Iteração usando forof:");

for (let carro of carros) {


    console.log("Valor:", carro);

}
```

Neste exemplo, utilizamos o loop “forin” para percorrer os índices do array carros e exibir tanto o índice quanto o valor correspondente. Em seguida, usamos o loop “forof” para iterar diretamente sobre os valores do array, sem a necessidade de acessá-los pelos índices.

Esses são apenas alguns exemplos de como utilizar as instruções “for” em JavaScript. Lembre-se de praticar e experimentar por conta própria para aprimorar suas habilidades nessa estrutura de controle.

Mão na massa usando FOR


Mergulharemos no uso do “for” em JavaScript e colocaremos a mão na massa para compreender seu funcionamento na .ica. O “for” é uma ferramenta poderosa que nos permite percorrer e manipular elementos em um array de maneira eficiente. Através de exemplos e exercícios, você poderá experimentar o uso do “for” em situações reais e aprimorar suas habilidades de programação.

Parte 1: Iteração com “forin”:

Vamos começar explorando o “forin”, uma forma de iteração que nos permite acessar os atributos de um objeto. Veja o exemplo abaixo:

```
const carro = {  
  
  modelo: 'Audi A3',  
  
  marca: 'Audi',  
  
  ano: 2020  
  
};  
  
for (let caracteristica in carro) {  
  
  console.log(caracteristica + ': ' + carro[caracteristica]);  
  
}
```

Nesse código, utilizamos o “forin” para percorrer cada atributo do objeto “carro” e exibir seu valor no console. Isso nos permite acessar e manipular os dados do objeto de forma conveniente.

Experimente adicionar mais atributos ao objeto e observe como o “forin” percorre cada um deles. 

Parte 2: Adicionando Objetos a um Array:

Agora, vamos aprender a trabalhar com arrays de objetos e iterar sobre eles usando o “forin”. Considere o exemplo a seguir:

```
const carros = [  
  
  { modelo: 'Audi A3', marca: 'Audi', ano: 2020 },  
  
  { modelo: 'Jeep Compact', marca: 'Jeep', ano: 2021 }  
  
];  
  
for (let indice in carros) {  
  
  console.log('Carro ' + (parseInt(indice) + 1) + ':');  
  
  for (let atributo in carros[indice]) {  
  
    console.log(atributo + ':' + carros[indice][atributo]);  
  
  }  
  
  console.log('-----');  
  
}
```

Nesse caso, temos um array chamado “carros” que contém dois objetos representando carros diferentes. Utilizamos o “forin” para percorrer cada objeto do array e exibimos suas características no console. Observe como o “forin” nos permite acessar cada atributo dos objetos individualmente. Percebemos como essa

estrutura de controle nos permite percorrer e manipular objetos e arrays com facilidade.



Instrução while e do while

Agora, vamos abordar o while e o do-while em JavaScript, duas instruções fundamentais que devemos conhecer. É importante ressaltar que esses conceitos se aplicam não apenas ao JavaScript, mas também a outras linguagens de programação. Ao compreendermos o funcionamento do while e do do-while, estaremos adquirindo conhecimentos valiosos que poderão ser aplicados em diversas situações. Vamos entender a sintaxe e a lógica por trás dessas estruturas de controle.

1 - O while

O while é uma instrução de repetição que executa um bloco de código enquanto uma condição especificada for verdadeira. Vejamos a sintaxe do while:

```
while (condicao) {  
  
    // bloco de código a ser executado  
  
}
```

No exemplo a seguir, temos a condição $e < 10$. Se essa condição for verdadeira, o bloco de código será executado repetidamente. Observe como o valor da variável `e` é incrementado a cada iteração, gerando uma sequência de números até que a condição seja falsa.

```
let e = 0;
```

```
while (e < 10) {
```



```
    let texto = 'O número é ' + e;
```

```
    console.log(texto);
```

```
    e++;
```

```
}
```

No código acima, começamos com $e = 0$ e, a cada iteração, incrementamos o valor de e em 1. Enquanto e for menor que 10, o bloco de código dentro do `while` será executado. Esse loop continuará até que e seja igual a 10, momento em que a condição se torna falsa e a execução é encerrada.

2 - O do-while

O `do-while` é uma variação do `while`. A diferença fundamental é que o bloco de código é executado pelo menos uma vez, independentemente da condição ser verdadeira ou falsa. Vejamos a sintaxe do `do-while`:

```
do {
```

```
    // bloco de código a ser executado
```

```
} while (condicao);
```

No exemplo abaixo, o bloco de código será executado pelo menos uma vez antes que a condição seja avaliada. Se a condição for verdadeira, o bloco será executado repetidamente. Caso contrário, a execução será encerrada.

```
let resultado = "";
```



```
let e = 0;
```



```
do {
```

```
    resultado += 'Número ' + e + ', ';
```

```
    e++;
```

```
} while (e < 5);
```


```
console.log(resultado);
```

No código acima, independentemente da condição ser verdadeira ou falsa, o bloco de código será executado pelo menos uma vez. No exemplo, adicionamos o valor de `e` ao resultado a cada iteração e incrementamos seu valor. Nesse caso, o bloco é executado até que `e` seja igual a 5.

Compreendemos a lógica e a sintaxe do `while` e do `do-while` em JavaScript. Essas estruturas de controle nos permitem criar loops com base em condições específicas, proporcionando flexibilidade e controle no fluxo de execução do programa. Continuaremos explorando o uso do `while` e do `do-while` e veremos exemplos práticos e situações mais complexas.

Mão na massa usando `while` e `do while`

Nesta etapa da nossa jornada de aprendizado, chegou o momento de nos aprofundarmos no IEW do IEW em JavaScript. É hora de colocarmos em prática o que aprendemos até agora e desenvolver nossas habilidades codificando. Vamos aproveitar ao máximo essa oportunidade e começar a programar na nossa IDE. Antes de mergulharmos no IEW, faremos uma breve explicação

sobre o uso do For Off, que nos permite acessar cada elemento de um array. Vamos dar uma olhada! 


O For Off:

O For Off é uma forma conveniente de percorrer os itens de um array em JavaScript. Podemos utilizar essa estrutura para realizar tarefas específicas em cada elemento do array. A sintaxe é simples:

```
for (const item of array) {  
  
    // bloco de código a ser executado  
  
}
```

No exemplo a seguir, vamos imprimir o ano de cada carro presente no array de carros. Ao utilizar `for (const carro of carros)`, temos acesso a cada item do array e, em seguida, podemos acessar a propriedade desejada. Veja o código:

```
const carros = [  
  
    { modelo: 'Audi A3', marca: 'Audi', ano: 2020 },  
  
    { modelo: 'Compact', marca: 'Jeep', ano: 2021 }  
  
];  
  
for (const carro of carros) {  
  
    console.log(carro.ano);  
  
}
```

No exemplo acima, utilizamos a notação `carro.ano` para acessar a propriedade “ano” de cada objeto do array. A cada iteração  ano do carro correspondente é impresso no console. Essa é uma forma eficiente de acessar propriedades específicas de cada item do array.

Agora, vamos explorar o do-while:


O do-while é uma estrutura de controle semelhante ao while, com a diferença de que o bloco de código é executado pelo menos uma vez, independentemente da condição ser verdadeira ou falsa. Vejamos a sintaxe:

```
do {  
  
    // bloco de código a ser executado  
  
} while (condicao);
```

No exemplo a seguir, utilizaremos o do-while para imprimir os valores de C enquanto C for menor ou igual a 10. A cada iteração, o valor de C é incrementado em 1.

```
let C = 1;  
  
do {  
  
    console.log(C);  
  
    C++;  
  
} while (C <= 10);
```

No código acima, o bloco de código dentro do do é executado primeiro e, em seguida, a condição é verificada. Enquanto C for

menor ou igual a 10, o loop continuará a ser executado. Quando C atingir o valor 11, a condição se tornará falsa e a execução  será encerrada.

Nesta parte do nosso estudo, exploramos o IEW do IEW e conhecemos o do-while em JavaScript. Ao colocarmos as mãos na massa e praticarmos a codificação, ganhamos confiança e habilidade nessa linguagem. Continuem praticando e aprofundando seus conhecimentos, pois ainda temos muito a aprender juntos.

Conteúdo Bônus

Título do livro: '**JavaScript - Guia do Programador**: Guia completo das funcionalidades de linguagem JavaScript'

Autor: Maurício Samy Silva

O objetivo dessa indicação é fornecer uma referência abrangente em formato de livro para aprofundar o conhecimento sobre programação em JavaScript, incluindo as instruções "while" e "do-while". O livro "JavaScript - Guia do Programador" oferece explicações detalhadas, exemplos práticos e insights valiosos sobre as funcionalidades da linguagem JavaScript. Ao utilizar esse livro como conteúdo adicional, você aprofundará seus conhecimentos sobre programação em JavaScript, compreenderá as instruções "while" e "do-while" e explorará outras funcionalidades da linguagem para desenvolver habilidades de programação mais avançadas.

Referência Bibliográfica



FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!**: programação JavaScript. 1ª Ed. São Paulo: Alta Books, 2016.

ATIVIDADE PRÁTICA

Título da Prática: Estrutura de repetição

Objetivos: Compreender a implementação de repetição


Materiais, Métodos e Ferramentas: Para realizar esta prática vamos utilizar o Visual Studio Code

Prática

Imagine que você está desenvolvendo um programa em JavaScript para calcular a soma dos elementos de um array de números. Sua tarefa é escrever um código que utilize a estrutura 'do while' em conjunto com o método 'reduce' para realizar essa soma.

O programa deve começar recebendo um array de números como entrada. Em seguida, será necessário declarar uma variável chamada 'sum' e inicializá-la com o valor zero. Essa variável será responsável por armazenar a soma acumulada dos elementos do array.

Após a inicialização da variável 'sum', você precisará definir um índice inicial, que será utilizado no loop 'do while'. O índice deve ser inicializado com zero,

indicando que a primeira iteração do loop começará pelo primeiro elemento do array. 

Dentro do loop, você utilizará o método 'reduce' para realizar a soma acumulativa. Esse método recebe uma função de callback que, a cada iteração, adiciona o valor do elemento atual ao valor acumulado. A função de callback também verifica se o índice atual é menor ou igual ao índice definido anteriormente, para garantir que apenas os elementos até o índice atual sejam somados.

Após cada iteração do loop, o índice é incrementado em uma unidade. O loop continuará até que o índice seja igual ao comprimento total do array, o que indica que todos os elementos foram somados.

Por fim, o programa exibirá a mensagem 'A soma dos números é;', seguida pelo valor final da variável 'sum', que representa a soma total dos elementos do array.

Em resumo, seu código deve utilizar a estrutura 'do while' e o método 'reduce' para calcular a soma dos números em um array. Essa solução permite uma abordagem eficiente e concisa para resolver esse problema específico.

Boa sorte!

Resolução

```
// Array de números
```

```
const numbers = [1, 2, 3, 4, 5];
```

```
// Variável para armazenar a soma
```

```
let sum = 0;
```

// Índice inicial para o reduce



```
let index = 0;
```

```
// Executa o reduce utilizando do while
```

```
do {
```

```
    sum = numbers.reduce((accumulator, currentValue, currentIndex) => {
```

```
        if (currentIndex <= index) {
```

```
            return accumulator + currentValue;
```

```
        } else {
```

```
            return accumulator;
```

```
        }
```

```
    }, sum);
```

```
    index++;
```

```
} while (index < numbers.length);
```

```
console.log('A soma dos números é:', sum);
```

Ir para exercício