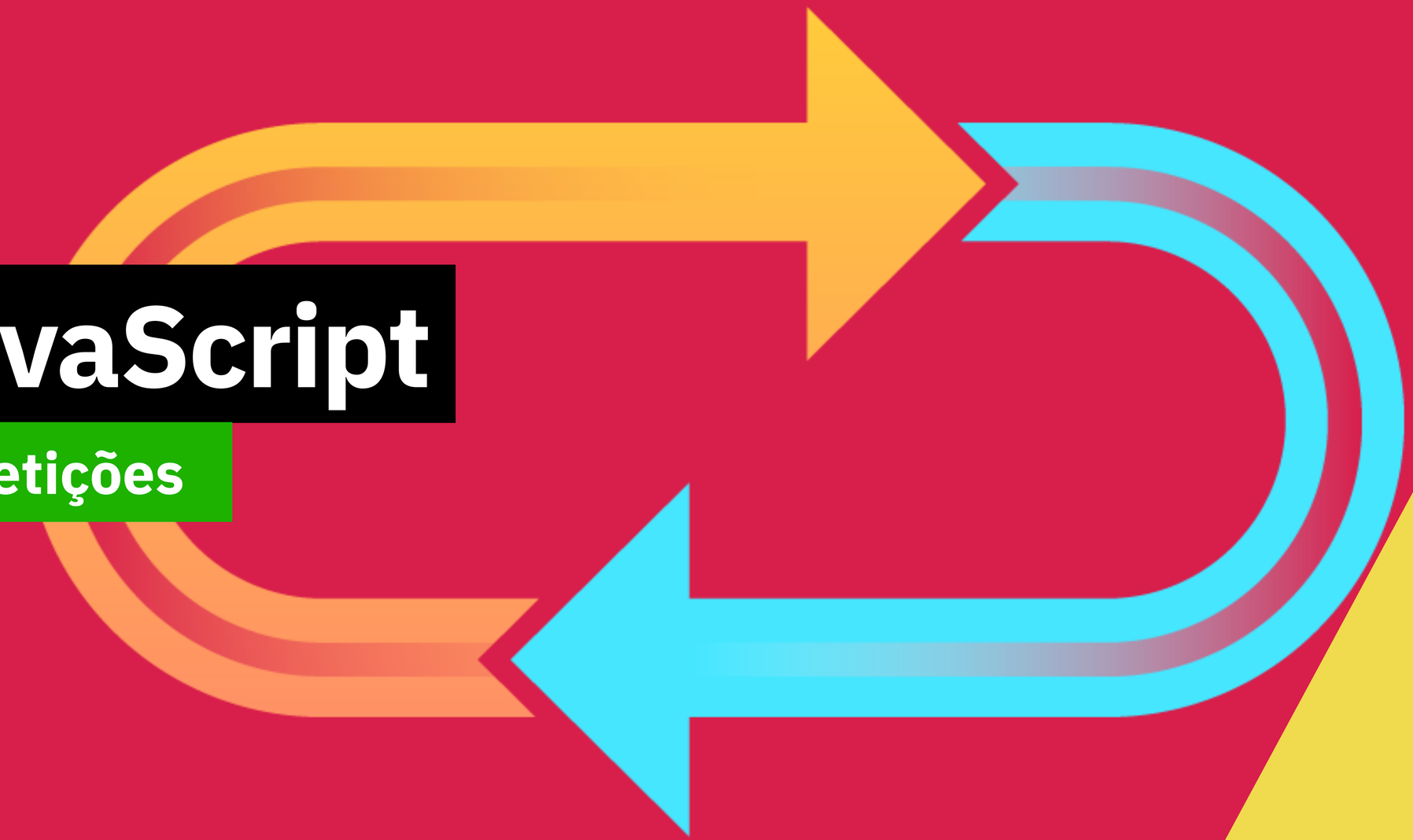
JavaScript

repetições



JS

JavaScript

parte 7

Alan Alencar, 2023

aprendendo Javascript

agenda

até aqui já aprendemos muito sobre a estrutura da linguagem.
Aprendemos sobre tipos primitivos, escopo e como criar condições específicas para o nosso algoritmo.
Agora e aprender a lidar com instruções repetitivas em JavaScript.

usando while

uma alternativa ao for

há uma segunda forma de criar laços de iteração usando o comando `while` no JavaScript. A principal diferença dele é que você só diz a condição. Todo o controle fica no seu algoritmo. Vejamos:

```
while (condição) {
  // Comandos aqui.
  ...
}
```

o ponto importante é que você tem que determinar a condição e dentro do bloco de comandos fazer com que esta condição seja alcançada ou não, e o importante é que ela normalmente, precisa ser inicializada antes.

```
let contador = 2;

while (contador <= 1000) {
  console.log(Math.pow(contador,3));
  contador += 2;
}

console.log(`O valor de contador é ${contador}`);
```

a variável normalmente é criada e inicializada antes e deve atender a condição. Já dentro do bloco `{` e `}` todo o controle é feito pelo programador. Aqui temos grandes chances de criarmos o famoso ***loop infinito***. Portanto todo cuidado é pouco.

usando do

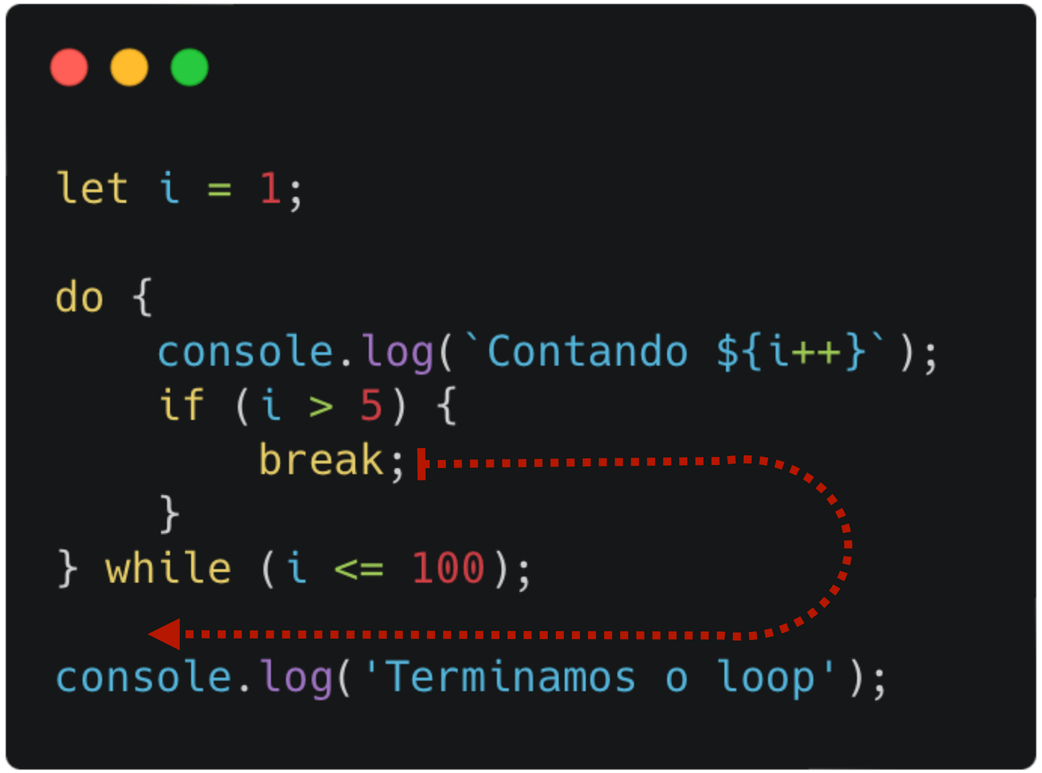
uma estrutura similar ao while

há principal diferença desta estrutura `do` com a estrutura `while` é que esta em específico sempre irá executar uma vez. Isto porque o teste de próxima iteração fica ao final da estrutura como podemos ver a seguir.

```
let i = 1;

do {
    console.log(`Contando ${i++}`);
} while (i <= 100);
```

o primeira iteração sempre vai ocorrer porque ela só irá testar se fica ou não para um próximo laço quando chega ao final do bloco do `do`.



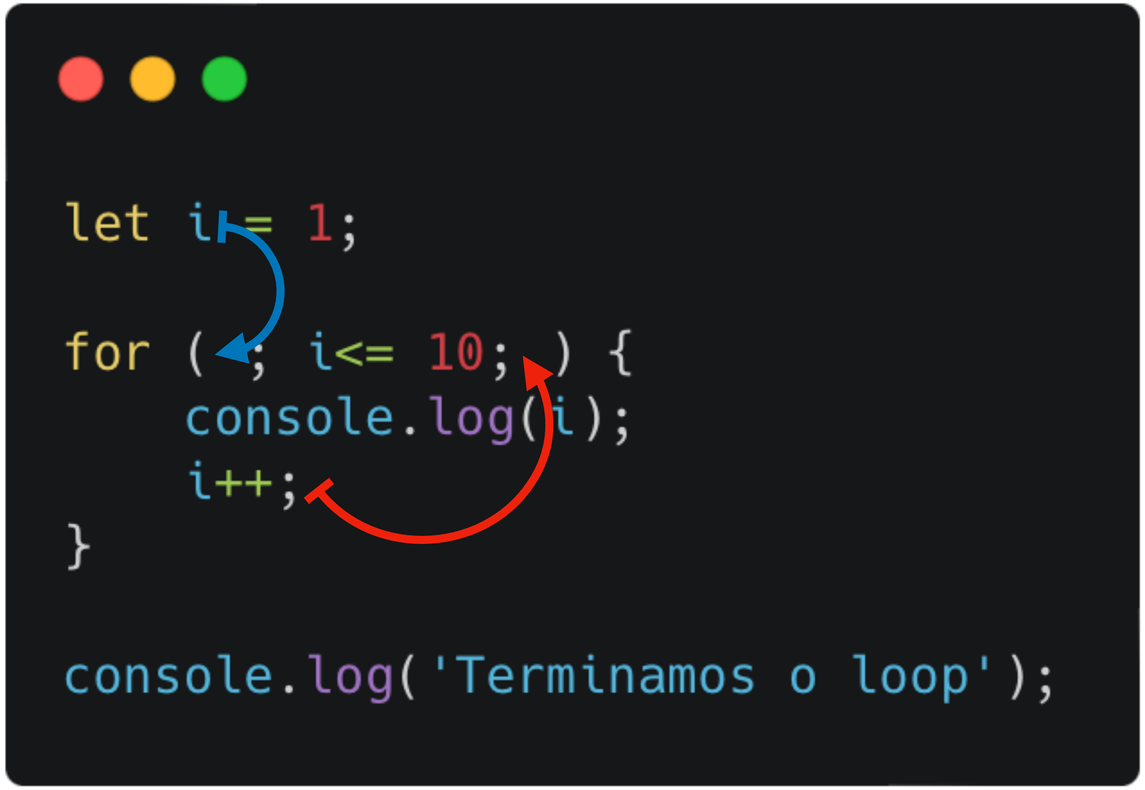
assim como no `for`, `while` e `do` também são afetados quando encontra um `break` no código, forçando seu término antes do teste da próxima iteração. Para fechar, ambos também são afetados quando encontra um `continue` retornando imediatamente para o topo do bloco. Tenha cuidado.


VAMOS PRATICAR
programming

parece bizarro mas não é

tudo depende do quanto você é capaz de abstrair

olhando bem as três estruturas de repetição do JavaScript dá para notar que elas tem uma grande similaridade. A diferença do `for` é a vantagem dele fazer o papel de incrementar nossa variável de iteração sempre que chegar ao fim do bloco. Mas na prática ele também pode deixar esse controle nas nossas mãos. Apesar de não ser recomendado fazer isto para não causar má interpretação de outros devs, sim, é possível escrever um for onde o incremento seja controlado dentro do bloco de maneira similar ao `do` e `while`. ***But ... don't do it***



revendo a prática

princípio básico

1. vimos que JavaScript é possível criar loops usando **for**, **while** e **do** para quaisquer condições.
2. não há uma melhor que a outra, mas sim como você implementa melhor no seu código.
3. **for** tem um controle melhor, mas pode ser totalmente controlado pelo programador dentro do bloco.
4. em todos os casos, **break** e **continue** tem uma ação importante que é preciso conhecer.