Iterações
Objetivo Geral: Iterações em JavaScript
Conteúdo:
1. Compreender o uso de Iterações em JavaScript 2. Utilizar Iterações em Códigos JavaScript
Metodologia e Estratégia:
1.Aula expositiva dialogada com apoio de tutorial; 2.Exercícios de aplicação.

Iterações

A iteração é um recurso muito utilizado dentro da programação e não é à toa. Muitas vezes precisamos fazer um processamento que tem uma estrutura repetida e que só varia quanto ao valor que ele manipula.

Não é raro termos listas com cerca de 1000 registros para serem processados da mesma forma. Imagine se tivéssemos que repetir um trecho de código 1000 vezes só porque o valor muda. Não é nada viável isso. Ainda mais que a lista, na maioria das vezes, varia de tamanho - uma hora 1000 registros, outra hora 580, outra 150 e por aí vai.

Para que você aprenda a lidar com esse tipo de situação, foram criados os tópicos desse capítulo. Um falando do iterador for - mais conhecido como laço “for” – e outro para falar do laço ”while”.

Iterando com o laço “while”

O laço while - ou enquanto, em português - é utilizado para iterações onde o que importa é alcançar uma determinada condição para que o mesmo se encerre. Veja a estrutura dele:

```
while ( ) {
// Bloco do while
}
```

Entre parênteses é o lugar onde colocamos a condição. Enquanto a condição retornar o valor true, as iterações continuam.

Como condição podemos usar qualquer coisa que resulte em um valor booleano.

Veja exemplos:

```
while (espacoEmDiscoDisponivel > 0) { // Expressão lógica.
// Bloco de código do while.
/*
Nesse while, de alguma forma, o bloco precisaria atualizar o valor da variável espacoEmDiscoDisponivel fazendo com que essa variável se aproxime, a cada iteração, do valor 0. E, quando ela chegar ao valor 0 ou abaixo dele, então as iterações se encerram.
*/
}
```

```
while (tentarNovamente) { // Utilizando variável
// Bloco de código do while.
/* Aqui a intenção é que exista alguma regra que, em algum momento, torne o valor da variável tentar novamente como false para que as iterações se encerrem. */
}
```

Agora veja um exemplo comum e um pouco mais prático sobre o uso do laço while:

```
var iteracao = 0;
while (iteracao < 10) {
document.write("Iteração de número " + iteracao + ".");
iteracao = iteracao + 1;
}
```

No exemplo acima, serão impressas as iterações de 0 até 9, pois, quando a variável iteração chega ao valor 10, então as iterações se encerram.

Iterando com o laço “for”

O laço for - ou para, em português - serve para que possamos lidar com as iterações onde o número de vezes que se precisa iterar é conhecido. Veja como é a estrutura de um laço for:

```
for ( ; ; ) {
// Bloco do laço for
}
```

Repare que entre os parênteses do for, temos 3 posições separadas por ponto e virgula. A primeira posição é utilizada para incluirmos uma expressão qualquer para iniciar nossas iterações. Veja o exemplo mais comum de se incluir nessa primeira posição:

```
var i = 0;
```

Note acima que o nome da variável ficou apenas como i. Destaco isso, pois, nomear variáveis dessa maneira - usando apenas uma vogal - foi uma coisa que desaconselhei que você fizesse. Só que, no caso do laço for, isso é uma coisa tão comum que não tem perigo de causar confusão. Você pode ver essa letra “i” como uma abreviação para a palavra “iteração”, pois, é justamente isso que ela vai fazer nesse caso nosso e na grande maioria do que vemos dentro dos programas: controlar qual é o número da iteração corrente.

Na segunda posição, nós colocamos uma expressão que precisa retornar um valor booleano. Essa é a posição onde, na maioria das vezes, colocamos uma relação entre o número da iteração corrente e a quantidade de iterações desejada.

Exemplo:

```
i < 10;
```

Com a expressão acima nós avisamos ao laço for que, enquanto a variável i for menor que 10, então pode iterar, ou seja, vai iterar com o i variando de 0 até 9, e quando i chegar ao valor 10, as iterações irão parar.

A terceira posição do laço serve para colocarmos uma expressão que será avaliada toda vez que uma iteração acabar. Chamamos ela de ”expressão de iteração”. Ocupando essa posição, na maioria das vezes, o que você verá é simplesmente a variável i aumentando uma unidade. Exemplo:

```
i = i + 1;
```

E, no Java, nós podemos substituir a expressão acima por:

```
i++
```

… que, nesse caso, é a mesma coisa.
Observe o for completo agora:

```
for (var i = 0; i < 10; i++) {
System.out.println("Iteração número: " + i);
}
```

O for acima irá resultar em 10 voltas em nosso laço - lembre que, na primeira posição do for, o i começa com 0 e, a cada volta, ele é incrementado uma unidade.

Agora vamos para um exemplo mais prático. Imagine que você esteja trabalhando em um algoritmo que vai funcionar em um software de loja virtual. Mais especificamente, vamos supor que esteja criando um algoritmo que precise fazer a soma do preço dos produtos que um usuário da loja colocou no carrinho.

Para realizar a tarefa acima é conveniente usar o laço for, pois o número de iterações é conhecido, ou seja, o número de iterações é a mesma quantidade de itens que o carrinho de compras possui.

Importante notar que esse número não tem que ser conhecido para nós, seres humanos. Ele deve ser conhecido pelo seu algoritmo, ou seja, caso existir alguma variável no seu algoritmo que contenha o número de iterações, então podemos dizer que essa quantidade é conhecida.

Quando utilizar o laço “for” ou o laço “while”

Cheguei a mencionar que o for serve para quando o número de iterações é conhecido e que o “while” itera até que determinada condição seja atendida.

Mas, uma curiosidade interessante sobre esses laços é que, tecnicamente, é possível fazer as mesmas coisas com os dois. Claro, que vai mudar a estrutura da iteração, mas, no final das contas, existem as mesmas possibilidades para eles.

No final das contas, é mais uma questão de semântica do que de possibilidades.

Para tomar a sua decisão de qual vai usar pode usar a dica que já mencionei.

Resumindo:

- Número de iterações é conhecido? Então use o for;
- A iteração deve acontecer até que se chegue a uma determinada condição? Use o while.

Exemplos Laço de Repetição.

Utilizar os Exercícios de Vetor agora com os laços de repetição, While e For.

Exercícios com Vetores

1. Faça um Programa que leia um vetor de 5 números inteiros e mostre-os.
2. Faça um Programa que leia um vetor de 10 números reais e mostre-os na ordem inversa.
3. Faça um Programa que leia 4 notas, mostre as notas e a média na tela.

Exercícios:

4. Faça um Programa que leia um vetor de 10 caracteres, e diga quantas consoantes foram lidas. Imprima as consoantes.
5. Faça um Programa que leia 20 números inteiros e armazene-os num vetor. Armazene os números pares no vetor PAR e os números IMPARES no vetor impar. Imprima os três vetores.
6. Faça um Programa que peça as quatro notas de 10 alunos, calcule e armazene num vetor a média de cada aluno, imprima o número de alunos com média maior ou igual a 7.0.
7. Faça um Programa que leia um vetor de 5 números inteiros, mostre a soma, a multiplicação e os números.
8. Faça um Programa que peça a idade e a altura de 5 pessoas, armazene cada informação no seu respectivo vetor. Imprima a idade e a altura na ordem inversa a ordem lida.
9. Faça um Programa que leia um vetor A com 10 números inteiros, calcule e mostre a soma dos quadrados dos elementos do vetor.