Revendo algumas estruturas de loop, ou seja, laço de repetição

Loops while e do/while

Os loops while e do/while têm o funcionamento parecido com o loop for, mas a forma de escrevê-los é diferente, como vamos ver a seguir.

While (enquanto)

```
<script>
    var count = 0;// variável contadora

while (count < 5) {// enquanto a variável for menor que 5 executará:
    document.write(count+"<br>");
    count++;
}

</script>
```

do/while (faça enquanto)

O loop do/while funciona da mesma maneira, a diferença é que a avaliação da condição é feita apenas ao final da primeira passagem. Isso significa que os comandos serão executados pelo menos uma vez, mesmo que a condição retorne false.

```
<script>
var count = 1;
    do {
            document.write(count+"<br>");
            count++;
        } while (count < 5); //Enquanto o contador for menor que 5 faz
</script>
```

Observação quando a condição não chegar a ser atendida, mas como a avaliação é feita apenas no final, os comandos serão executados pelo menos uma vez.

Exemplo se mudar o valor da variável para 10 ele irá imprimir na tela uma vez o número 10.

Loops For e For/In

Loops são estruturas de repetição de comandos. Se quisermos, por exemplo, repetir o mesmo comando cinco vezes, não precisamos escrevê-lo as cinco vezes, podemos escrevê-lo apenas uma vez dentro de um loop.

Podemos usar também os loops para percorrer todos os elementos de arrays e objetos e realizar comandos sobre cada um deles. Estes comandos só precisarão ser escritos uma vez, não importa o tamanho do array ou do objeto. Isto nos abrirá incríveis possibilidades, conforme vamos ver logo mais.

Loop For

```
<script>
   for (var x = 0; x < 5 ; x++) {
        document.write(x+"<br>");
    }

</script>
```

Note que ao iniciar o loop for usamos três partes expr1; expr2; expr3 dentro dos parênteses, separados por ponto e vírgula. Entre as chaves { } escrevemos os comandos que queremos repetir, assim como fazemos com as funções.

Sobre os comandos que estão entre parênteses na estrutura do loop for:

- **Expr1: var a = 0**: O primeiro comando passado no loop for é executado uma vez antes de começar o loop. Neste caso criamos uma variável com valor zero.
- **Expr2: a < 5**: O segundo comando é uma condição que precisa ser atendida para que o loop continue sendo executado. Lembra-se da aula em que falamos de booleanos? No início deste loop, a variável *a* vale zero, portanto o teste a < 5 retorna true, então os comandos do loop são executados. Depois que os comandos forem executados, o loop será avaliado outra vez e assim repetidamente. Quando este teste retornar false, o loop é interrompido.
- **Expr3: a++**: O terceiro comando é executado sempre ao final de cada passagem no loop, por isso utilizamos ele para incrementar a variável *a*.

Usando o loop for para percorrer elementos de um array

Para percorrer arrays usando o loop for, usaremos a propriedade *length* do array para controlar quando deve acabar o loop.

```
<script>
var alunos = ['Pedro', 'Maria', 'José', 'João', 'Carlos'];

for (var i = 0; i < alunos.length ; i++) {
    document.write(alunos[i]+"<br>");
}

</script>
```

Loop For/In

O loop for/in serve para facilmente percorrermos elementos de um objeto, já que este tipo de dados não possui índices e portanto não pode ser percorrido com um loop for como os arrays.

O funcionamento dele é bem simples. Consideremos um objeto de alunos, cujas chaves são as matrículas e os valores são os nomes dos alunos.

Para iniciar o loop for/in criamos uma variável chamada *matr*. O loop então vai percorrer todos os elementos do objeto e a cada passagem a variável *matr* vai ser igual a chave do elemento. Sendo assim, temos acesso tanto à matrícula por meio desta variável quanto ao nome do aluno, usando a notação alunos[matr].

```
<script>
 var alunos = ['Pedro', 'Maria', 'José', 'João', 'Carlos'];

for (var i = 0; i < alunos.length ; i++) {
    document.write(alunos[i]+);
}

</script>
```