Tornando o seu programa um robô incansável

Na última aula nós pudemos entender como funciona e como criar uma estrutura de repetição básica utilizando o `while`.

Entendemos que precisamos ficar atentos a condição de parada e do contador.

Também entendemos que podemos utilizar o valor do contador para ditar o passo do *looping* e realizar operações com seu valor.

E nesse sentido que a estrutura de repetição `for` se encaixa perfeitamente.

Se você reparar, no `while` sempre precisamos:

1. Criar a variável para contagem externamente: `var x = 0`
2. Definir a condição de parada: `x < 10`
3. Incrementar o contador internamente: `x++`

```
var x = 0;          // 1

while(x < 10) {     // 2
  console.log(x);
  x++;              // 3
}
```

Com o `for` podemos definir tudo isso em apenas uma linha, veja o exemplo abaixo:

```
for(x = 0;x < 10;x++) { // 1, 2 e 3
    console.log(x);  // comando de execução
}
```

Sendo:

1. Variável e seu valor inicial: `x = 0`
2. Condição de parada: `x < 10`
3. Passo de contagem: `x++`

E como você percebe cada definição é separada por `;`.

A grande vantagem do `for` é que deixa o seu código mais enxuto, já que grande parte das vezes você precisa utilizar o valor do contador e também precisa definir o valor de parada. Perfeito!

Operadores do FOR

O `for` permite praticamente utilizar todos os operadores de comparação que o `if` e `while` utilizam, que são:

- Diferente (`!=`)
- Menor (`<`) ou Menor igual (`<=`)
- Maior (`>`) ou Maior igual (`>=`)

Por exemplo:

```
for(x = 0;x <= 15;x++) {
    console.log(x);
}
```

O contador do FOR

Como você já pode perceber o `for` utiliza o contador na sua estrutura compacta (`x++`). E podemos:

- `x++` - incrementar de 1 em 1.
- `x--` - decrementar de 1 em 1.
- `x+=2` - incrementar de 2 em 2 (ou de *n*);
- `x-=2` - decrementar de 2 em 2 (ou de *n*);
- `x*=2` - multiplicar de 2 em 2 (ou de *n*);

Por exemplo este `for` decrescente (perceba que a lógica de comparação precisa estar invertida):

```
for(x = 10;x > 0;x--) {
    console.log(x);
}
```

Looping infinito com o FOR

Normalmente podemos ver com mais frequência o uso do `while` como *looping* infinito. De qualquer modo com o `for` também é possível:

```
for(;;) {
    // comando...
}
```

Quando precisamos utilizar o `for` com *looping* infinito simplesmente deixamos de declarar na sua estrutura a variável, condição e contador.

E para a condição de parada o `break` também é válido:

```
for(;;) {
    // comando...

    minutos = ler_tempo();
    if(minutos > 60) {
```

```
            break; // <<-- parada hipotética por tempo...
        }
    }
```

Desafio

Utilizando o `for` crie um programa que calcule a função log `Math.log(x)` para cada valor do contador.

Este programa deve solicitar os valores iniciais, finais e de incremento. E também escrever na tela o resultado.

Resposta

```
var vi = Number(prompt("Digite o valor inicial"));
var vf = Number(prompt("Digite o valor final"));
var vc = Number(prompt("Digite o valor de incremento"));

var res = 0;

for(x = vi;x <= vf;x+=vc) {
    res = Math.log(x);
    console.log(res);
}
```