Programação Estruturada

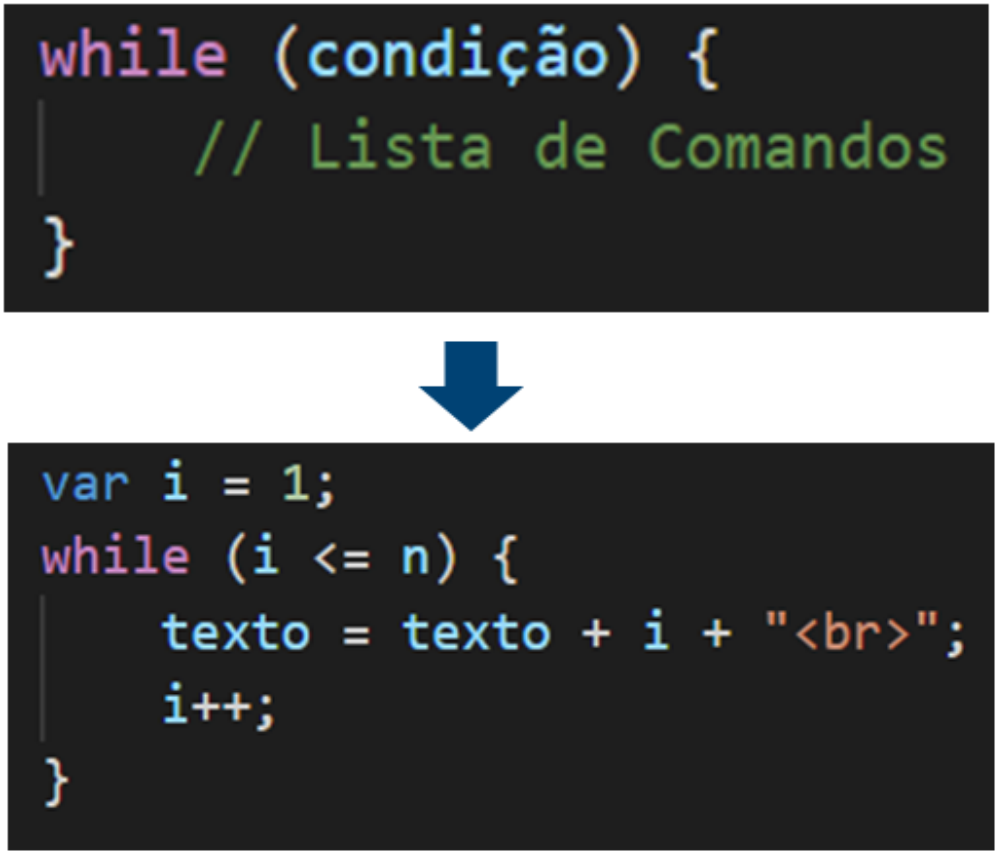
Aula 10 - Comandos de Iteração

Videoaula 03 - Laços While

Videoaula 03 - Laços While

O próximo comando de iteração que você irá aprender é o laço `while`, que tem a forma apresentada no slide. Assim como o laço `for`, o laço `while` é utilizado para repetir a execução de um bloco de comandos. Eles podem ser utilizados no lugar do laço `for`, como podemos ver no slide, o qual apresenta um programa que concatena os números de 1 até o número dado como entrada em um texto a ser exibido na página HTML, exatamente como fizemos nos exemplos anteriores. Vejamos como funciona esse comando.

Figura 1 - Estrutura do Laço `while`



Primeiro, a execução do `while` depende de uma condição, que no caso do exemplo é `i<=n`. Essa condição deve ser uma expressão booleana e seu valor é utilizado para determinar se o corpo do laço deve ser executado ou não. Se o valor da expressão for `true`, o corpo do laço é executado. Caso contrário, o corpo do laço não é executado e a execução passa para a próxima linha de código do programa após o corpo do laço. Você pode utilizar o arquivo **10_6 Contagem While.html** para testar o uso do laço `while` no nosso exemplo.

Código 1 - 10_6 Contagem While.html

```html
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 10</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Contagem</h1>

    N: <input type="number" id="N1" value="">
    <button onclick="contagem('N1')">OK</button>
    <p id="resultado"></p>

    <script src='script.js'></script>
  </body>
</html>

```

```javascript
function contagem(entrada1) {
  var n = Number(document.getElementById(entrada1).value);

  var texto = "";

  /*
  while (condição) {
    // Lista de Comandos
  }
  */

  for (i = 1; i <= n; i++) {
    texto = texto + i + "<br>";
  }

  var i = 1;
```

```
    while (i <= n) {
        texto = texto + i + "<br>";
        i++;
    }

    document.getElementById("resultado").innerHTML = texto;
}

```

Apesar do `while` e do `for` poderem ser utilizados de forma alternativa, cada um é mais apropriado para determinadas situações. Por exemplo, qual implementação do nosso exemplo você achou mais interessante, a versão que utilizou o `for` ou a versão que utilizou o `while`? Em geral, para situações como a do nosso exemplo, as pessoas preferem a implementação utilizando o `for`. Isso porque laços implementados com o comando `for` são interessantes quando existe uma contagem. Em geral, sabe-se quantas vezes o laço será executado e a condição normalmente é um contador que é usado para informar isso.

Já o `while` é bastante interessante quando não se sabe inicialmente quantas vezes o laço será executado, pois a condição pode ser algo diferente de um contador e o código continuará a executar o laço enquanto essa condição for verdade. Esse tipo de laço é bastante útil para validar a entrada de dados do usuário. Por exemplo, imagine uma página a qual, ao clicarmos no botão INICIAR, sorteia um número `n` entre 1 e 10 e pergunta qual é esse número para o usuário até que ele acerte. Isso é feito de forma relativamente natural, usando-se o comando `while`, como veremos no exemplo a seguir.

Código 2 - 10_7 Sorteio While.html

```
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Programação Estruturada - Aula 10</title>
    </head>
    <body>
        <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

        <h1>Sorteio</h1>

        <button onclick="sorteio()">INICIAR</button>
```

```
12      <p id="resultado"></p>
13
14      <script src='script.js'></script>
15    </body>
16  </html>
17
```

```
1  function sorteio() {
2    var n = Math.floor(Math.random() * 10) + 1;
3    var num;
4    var contagem = 1;
5
6    num = prompt("Escolha um número entre 1 e 10?");
7    while (num != n) {
8      num = prompt("Escolha um número entre 1 e 10?");
9      contagem++;
10   }
11   document.getElementById("resultado").innerHTML =
12     "O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";
13 }
14
```

Nesse exemplo tem-se um HTML simples para realizar o sorteio. Nele, há um botão, escrito INICIAR, e um parágrafo para escrever algum resultado no final.

Ao clicar no botão INICIAR, como aparece na linha 11, será chamada a função sorteio que está no arquivo JavaScript, conforme código 3.

Código 3 - 10_7 Sorteio While.js

```
1  <html>
2    <head>
3      <meta charset="UTF-8" />
4      <title>Programação Estruturada - Aula 10</title>
5    </head>
6    <body>
7      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9      <h1>Sorteio</h1>
10
11     <button onclick="sorteio()">INICIAR</button>
12     <p id="resultado"></p>
13
14     <script src='script.js'></script>
15   </body>
16 </html>
```

```
17
```

```
1  function sorteio() {
2     var n = Math.floor(Math.random() * 10) + 1;
3     var num;
4     var contagem = 1;
5
6     num = prompt("Escolha um número entre 1 e 10?");
7     while (num != n) {
8        num = prompt("Escolha um número entre 1 e 10?");
9        contagem++;
10    }
11    document.getElementById("resultado").innerHTML =
12       "O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";
13 }
14
```

Essa função inicialmente atribui a `n` o resultado da expressão que aparece na linha 2 do código 3, ou seja, o `Math.floor` do `Math.random` vezes 10 mais 1. O `Math.random` dará um número aleatório entre 0 e 1, mas não inclui o 1, ou seja, dará zero ponto alguma coisa. Esse valor aleatório entre 0 e 1 será multiplicado por 10 e passará a ser outro valor. Então, pegando o `floor`, o menor inteiro desse número, a ele será somado 1. Em outras palavras, essa expressão dará um número aleatório, entre 1 e 10, incluindo o 1 e o 10.

Há duas variáveis, `num` e `contagem`. A variável `num` será usada para receber valores e será usado o prompt do navegador, como apresentado nas linhas 6 e 8 do código acima.

Será exibido "Escolha um número entre 1 e 10?" e esperará que seja inserido um valor. Esse valor será atribuído à variável `num`.

Enquanto esse valor inserido na página HTML for diferente do `n`, que é o valor aleatório da linha 2, será exibida novamente a mensagem "Escolha um número entre 1 e 10?" e pedirá um novo valor. Em outras palavras, o que ele faz? Ele pergunta o outro valor e verifica se esse valor é diferente do `num`.

`

Se não for é porque acertou, tá? Caso contrário, se for diferente vai perguntar novamente, e vai incrementar a contagem. E no final informa: "O número era" e o `n` que sorteei e "Você acertou em", contagem de "tentativas". Está certo? Assim, enquanto o número que o usuário inserir no Prompt não for aquele número sorteado, ele vai pedir o número, e vai incrementar a contagem.

Na página HTML, ao clicar em iniciar, aparece a pergunta, como mencionado anteriormente, pedindo para escolher um número entre 1 e 10. E, indo na sequência, digitando o 1, aparece a pergunta novamente, seguindo para o 2, a mesma coisa acontece com o 3, 4, 5, 6, 7 e 8. Ao digitar o 9, descobriremos que esse era o número, pois aparece a mensagem: "O número era 9. Você acertou em 9 tentativas!!!".

Observe que ele foi validando e, enquanto não acertava o número sorteado, que era o número 9, ia perguntando. Ia permanecer fazendo isso até acertar e, nesse caso, acertou em nove tentativas. Então, é assim que você pode usar o laço `while` para validar a entrada do usuário, ok?

Não sei se você notou, mas, no nosso exemplo existe uma repetição do trecho de código marcado em amarelo no slide. Essa duplicação deve-se ao fato de que é preciso ter um valor inicial para ser testado na primeira vez em que o laço é executado. Nessas situações, em que sabemos que o bloco de código do laço precisa ser executado pelo menos uma vez, fazemos uso do comando `do/while`.

Figura 2 - Repetindo trechos de código

```
num = prompt("Escolha um número entre 1 e 10?");
while (num != n) {
    num = prompt("Escolha um número entre 1 e 10?");
    contagem++;
}
```

Veja agora esse mesmo programa com o `do/while`, no qual o código dentro do laço sempre será executado pelo menos uma vez.

Código 4 - 10_8 Sorteio_Do-While.html

```html
<html>
   <head>
      <meta charset="UTF-8" />
      <title>Programação Estruturada - Aula 10</title>
   </head>
   <body>
      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

      <h1>Sorteio</h1>

      <button onclick="sorteio()">INICIAR</button>
      <p id="resultado"></p>

      <script src='script.js'></script>
   </body>
</html>

```

```javascript
function sorteio() {
   var n = Math.floor(Math.random() * 10) + 1;
   var num;
   var contagem = 0;

   do {
      num = prompt("Escolha um número entre 1 e 10?");
      contagem++;
   } while (num != n);
   document.getElementById("resultado").innerHTML =
      "O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";
}

```

Aqui será implementado o mesmo exemplo do sorteio, só que ao invés de usar o `while`, será utilizado o `do/while`. Para isso, a página HTML será a mesma. No código JavaScript, as linhas 2 e 3 também continuarão com a mesma informação anterior, ou seja, o sorteio do número aleatório e a declaração da variável `num`. Entretanto, na linha 4, como é possível visualizar no Código 5, veja que a contagem é igual a 0, então alterou um pouco do exemplo anterior.

Código 5 - 10_8 Sorteio Do-While.js

```html
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 10</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Sorteio</h1>

    <button onclick="sorteio()">INICIAR</button>
    <p id="resultado"></p>

    <script src='script.js'></script>
  </body>
</html>

```

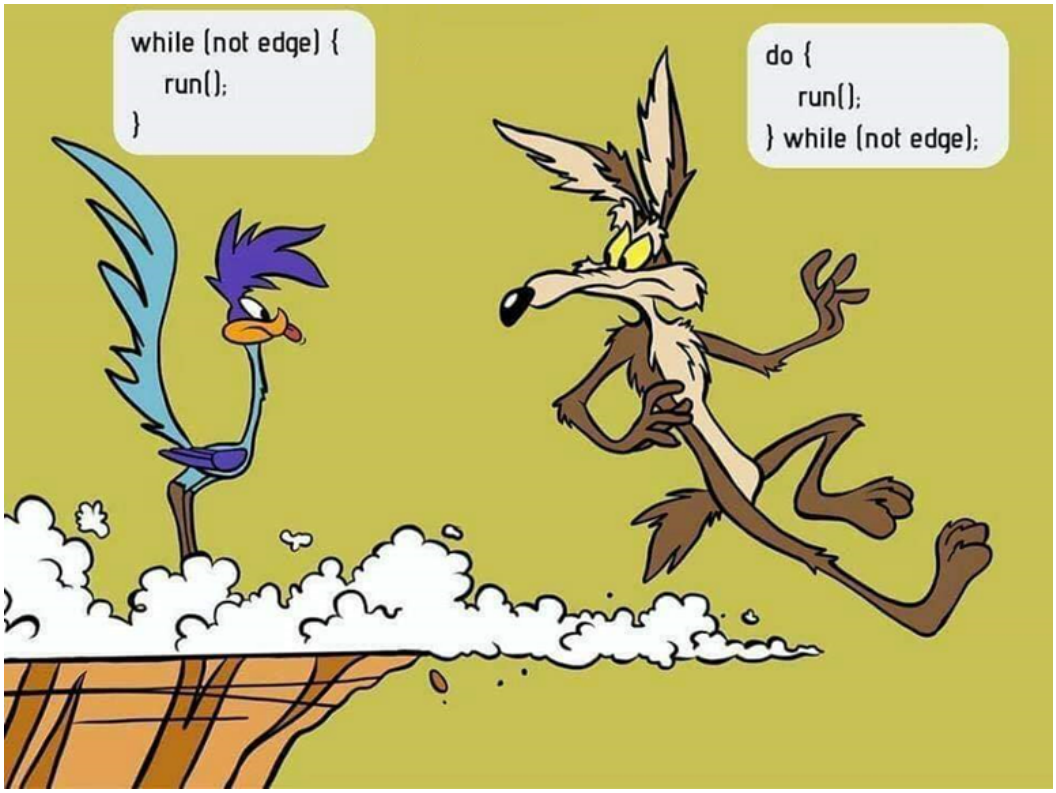
```
function sorteio() {
  var n = Math.floor(Math.random() * 10) + 1;
  var num;
  var contagem = 0;

  do {
    num = prompt("Escolha um número entre 1 e 10?");
    contagem++;
  } while (num != n);
  document.getElementById("resultado").innerHTML =
    "O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";
}

```

Essa contagem é igual a zero porque agora vai, necessariamente, perguntar o número dentro do laço `do/while`, como consta nas linhas 6 a 9. Na linha 7, pergunta e na 8, incrementa a contagem. Então, inicialmente pergunta um número ao usuário e já incrementa a contagem, passando a contagem para 1. E isso será feito enquanto o número que o usuário colocou for diferente de `n`, ok? Perceba que é um pouquinho diferente, pois há a garantia de que vai executar pelo menos uma vez essa solicitação e incremento, tá certo? E essa alteração do texto que vai ser colocada na página HTML é exatamente a mesma vista no exemplo anterior.

Então, na página HTML, clica em INICIAR, para começar o sorteio. Começando mais uma vez pelo 1, em seguida 2, 3, 4, 5, 6 e, finalmente, o 7. Dessa vez foi em 7 tentativas, ou seja, o número era 7. Assim, enquanto inserir o valor e incrementar a contagem, mas esse valor inserido não foi igual ao valor sorteado, que nesse caso era 7, ele foi pedindo novamente o valor e incrementando a contagem.

Logo, diferente do outro exemplo, tem-se o `do/while` com a garantia de que vai executar as linhas 7 e 8 pelo menos uma vez. Ok?

Figura 3 - Comparando `while` com `do/while`



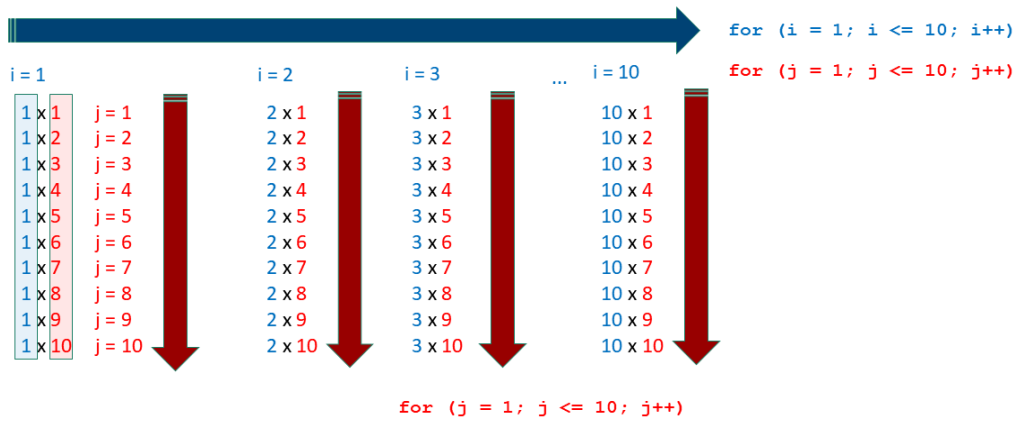
Pois é, mais uma vez o coiote se deu mal!!!

Note a diferença entre os comandos do papa-léguas e do coiote. O papa-léguas executou o `while`. Por esse motivo, ele primeiro verificou se estava na beira do precipício antes de correr, desse modo, permaneceu parado. Por outro lado, o coiote executou o `do/while`. Por isso, ele inicialmente correu para só depois verificar se estava na beira do precipício. Ups!!! Verificou tarde demais...

Antes de concluir esta aula, vou te apresentar uma utilização muito comum de laços, principalmente daqueles que utilizam o `for`, são os laços encadeados. Nesta utilização, temos laços acontecendo dentro de laços. Vamos, por exemplo, imaginar como poderíamos escrever um programa que exibisse na página as tabuadas de 1 até 10.

Neste caso, temos que, para cada número, digamos `i`, entre 1 e 10, precisamos fazer a multiplicação de `i` por outro número, digamos `j`, que também deve variar entre 1 e 10. Para resolver esse problema, teremos um laço para o `i` que variará entre 1 e 10, destacado em azul no slide. Para cada iteração do laço azul, teremos um outro laço dentro dele para o `j`, que também variará entre 1 e 10. Cada iteração desse laço mais interno terá um valor diferente para `i` e `j` juntos, e vamos usar esses valores para escrever a linha correspondente a `i x j`.

Figura 4 - Laços Encadeados



Olhe novamente para o slide! Quando tivermos, no laço mais externo azul o `i` igual a 1, o laço interno vermelho fará a iteração de `j` variando de 1 até 10. Ou seja, teremos `1 x 1`, `1 x 2`, `1 x 3`, até `1 x 10`. Depois disso, o laço interno vermelho termina e o laço externo azul alterará o `i` para 2. Novamente, o laço interno vermelho fará a iteração de `j` variando de 1 até 10. Ou seja, teremos `2 x 1`, `2 x 2`, `2 x 3`, até `2 x 10`. Depois disso, o laço interno vermelho termina e o laço externo azul alterará o `i` para 3. Mais uma vez, o laço interno vermelho fará a iteração de `j` variando de 1 até 10. Ou seja, teremos `3 x 1`, `3 x 2`, `3 x 3`, até `3 x 10`. Isso continuará se repetindo até

que o laço externo azul altere `i` para 10. Mais uma vez, o laço interno vermelho fará a iteração de `j` variando de 1 até 10. Ou seja, teremos `10 x 1`, `10 x 2`, `10 x 3`, até `10 x 10`. Após isso, o laço interno vermelho terminará sua execução, pois o `j` chegou em 10. Da mesma maneira, o laço externo azul terminará sua execução, pois o `i` também chegou em 10. Isso termina a execução do programa.

Vamos ver o código resultante?

Código 6 - 10_9 Tabuada.html

```html
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 10</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Cronômetro</h1>

    <button onclick="tabuada()">OK</button>
    <p id="resultado"></p>

    <script src='script.js'></script>
  </body>
</html>

```

```javascript
function tabuada() {
  var texto = "";
  var i, j;

  for (i = 1; i <= 10; i++) {
    texto = texto + "----------<br>";
    texto = texto + "Tabuada de "+ i +"<br>";
    texto = texto + "----------<br>";
    for (j = 1; j <= 10; j++) {
      texto = texto + (i + " x ") +  (j + " = ") + (i*j) + "<br>";
    }
    texto = texto + "----------<br>";
  }

  document.getElementById("resultado").innerHTML = texto;
}

```

Nesse exemplo, o HTML também é bem simples, há um botão escrito OK nele. Ao clicar nesse botão será chamada a função `tabuada` e tem o `resultado`, que é um parágrafo que será atribuído a um texto que será construído ao chamar a função.

O que há na função `tabuada`? Um `texto` que começa como vazio, e a ideia é construir esse `texto` que vai ser exatamente as tabuadas de 1 a 10. Ou seja, a tabuada de 1, a tabuada de 2, até a tabuada de 10. Para isso, terá um laço `for` para o `i`, na linha 5, como aparece no código abaixo, que é a tabuada que está sendo construída, ou seja, a tabuada de 1, de 2, de 3, de 4, etc.

Código 7 - 10_9 Tabuada.js

```
1  <html>
2    <head>
3      <meta charset="UTF-8" />
4      <title>Programação Estruturada - Aula 10</title>
5    </head>
6    <body>
7      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9      <h1>Cronômetro</h1>
10
11     <button onclick="tabuada()">OK</button>
12     <p id="resultado"></p>
13
14     <script src='script.js'></script>
15   </body>
16 </html>
17
```

```
1  function tabuada() {
2    var texto = "";
3    var i, j;
4
5    for (i = 1; i <= 10; i++) {
6      texto = texto + "----------<br>";
7      texto = texto + "Tabuada de "+ i +"<br>";
8      texto = texto + "----------<br>";
9      for (j = 1; j <= 10; j++) {
10       texto = texto + (i + " x ") +  (j + " = ") + (i*j) + "<br>";
11     }
12     texto = texto + "----------<br>";
13   }
14
15   document.getElementById("resultado").innerHTML = texto;
```

```
16 }
17
```

Então, para cada tabuada foi inserida uma linha especial (vide linha 6) para marcar, apenas com hífens e a quebra de linha. Na linha 7, escrevi "Tabuada de " para aparecer de quem é a tabuada, usando o `i` e novamente foi inserida outra linha apenas com hífens na linha 8 para então começar a escrever a Tabuada.

Para iniciar, na tabuada de 1 terá que multiplicar o 1 por 1, 2, 3, 4, 5, até 10, ou seja, preciso de outro laço para fazer essa outra parte. Então, terá outro laço descrito na linha 9, com o `j` variando de 1 até 10 também e o texto recebe valor de `i` mais a *string* com o símbolo de vezes, descrito na linha 10. Isso vai transformar o `i` em texto, como visto em aulas anteriores. Em seguida, vou concatenar ele com o `j` mais o igual, ou seja, aqui também o `j` vai ser convertido em texto. Então, o resultado do trecho na linha 10 é escrever o texto com o texto `i x j`.

Inicialmente, o `i` sendo 1 e o `j` sendo 1, vai aparecer "`1 x 1`". Então, iterando nesse laço interno, vai aparecer `1 x 2`. E depois de cada um desses, será acrescentado o valor, e é importante colocar entre parênteses o valor da multiplicação de `i` por `j`, como aparece na linha 10. Logo, aparecerá: `1 x 1` é igual a 1. Em seguida, `1 x 2` é igual a 2, e assim por diante, tá certo?

No final também será inserida uma linha para informar que foi encerrada a tabuada de 1 e as demais, isso está descrito no comando que aparece na linha 12. É importante perceber que a variável de controle do laço externo é o `i`, que está descrito na linha 5, e a variável do laço interno é o `j`, descrito na linha 9. É muito incomum que esses laços usem a mesma variável de controle. Portanto, se a variável `i` fosse usada para controlar o laço interno teríamos um erro.

Assim, há o laço externo que vai dizer quais serão os valores da tabuada de 1, 2, 3, 4, 5 e há também para cada uma delas um laço interno. Para a tabuada de 1, tem-se um laço interno que faz: `1 x 1`, `1 x 2`, `1 x 3`, `1 x 4`, ..., `1 x 10`. Depois que passa o laço externo para 2, e o laço interno fica: `2 x 1`, `2 x 2`, `2 x 3`, `2 x 4`, e assim por diante.

Logo, na página HTML, ao clicar em OK, aparecerá a tabuada de 1, que faz de `1 x 1` até `1 x 10`. Em seguida, a Tabuada de 2, que faz `2 x 1` até `2 x 10`, e assim por diante, até o laço externo chegar no `i` igual a 10. Quando isso acontecer, vai fazer o laço interno do `10 x 1` até o `10 x 10`. E quando fizer o `10 x 10` que é igual a 100, o laço interno colocará uma última linha de hífens no final e acabará porque o `j` chegou em 10, e o laço externo também porque do mesmo modo o `i` chegou em 10. Finalmente, os loops encadeados se encerram e atribuem o texto que foi construído ao parágrafo `resultado`, e é por isso que esse texto é exibido na página HTML.

Tá joia?

Chegamos ao final desta aula sobre comandos de iteração de JavaScript. Lembre-se que o comando de laço `for` é o mais adequado quando o programador quer determinar previamente o número de vezes que o trecho será executado. Já o comando `while`, executa uma iteração enquanto uma determinada condição for verdadeira, testando essa condição no início de sua execução. Por fim, o comando `do/while` é semelhante ao `while`, porém testa a condição no final de sua execução, garantindo, dessa forma, pelo menos uma execução do bloco do código do laço.

Mais uma vez, deixo uma lista de atividades. É muito importante praticar bastante o uso dos comandos de iteração, pois eles, assim como os comandos de seleção, são extremamente importantes na programação. Não deixe de fazê-los! Até a próxima aula, tchau!