

**Estruturas de controle**

**Comandos de seleção**

Aprender a criar estruturas de condição é fundamental para programar a lógica de negócio de um sistema. Essas estruturas permitem a execução de diferentes linhas de código (diferentes fluxos de execução) a partir da avaliação de uma condição.

Os comandos de seleção da linguagem JavaScript, são fundamentais para especificar as decisões que um computador realizará.

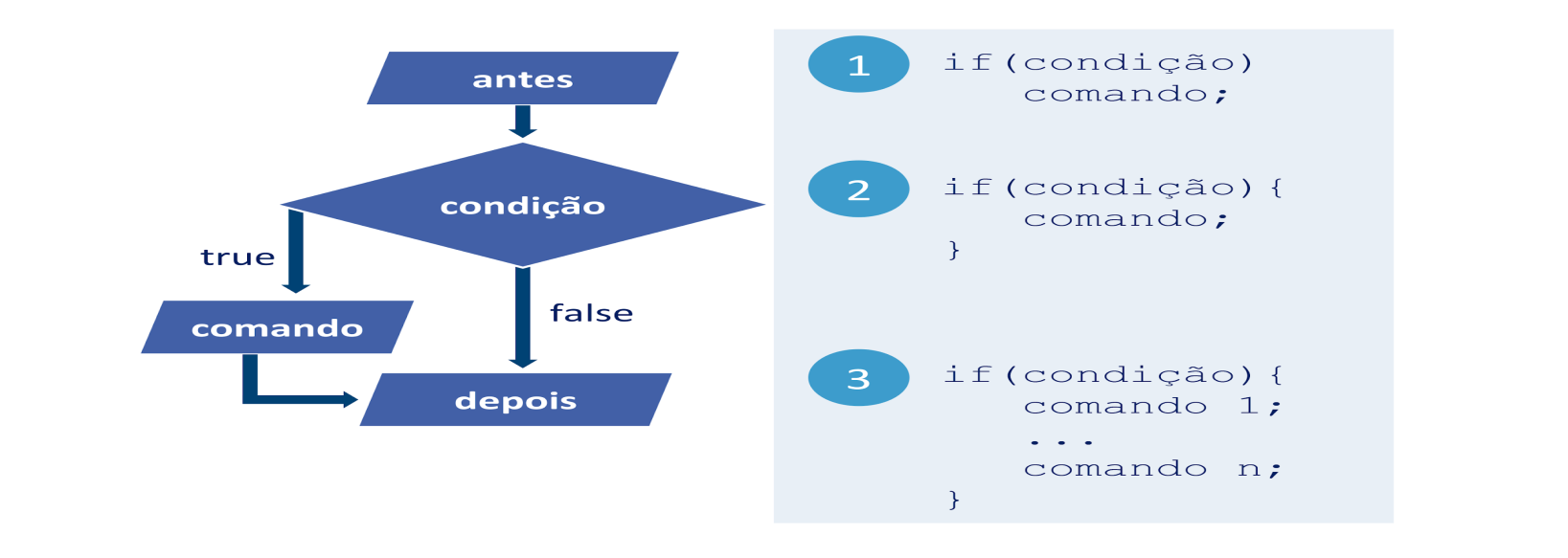
Entre os comandos de seleção, temos os comandos if-else (se/senão) e switch (escolha). Esse conteúdo é de grande importância para o desenvolvimento de programas em geral, não apenas em JavaScript, pois é através desses comandos que são feitos inúmeros testes condicionais no decorrer dos programas.

**if**

O comando if é o mais básico de todos os comandos de seleção e deve ser usado com as duas letras minúsculas. Ele determina que o programa deve executar um certo comando ou seção de código apenas se uma certa condição for verdadeira.

A forma geral do comando if está apresentada na Figura 1.

Figura 01 - Comando de Seleção if



Se desejarmos executar apenas um comando, caso a condição seja verdadeira, o uso das chaves é opcional, ou seja, podemos ter o comando sem o uso da chave, como vemos no exemplo 1, ou com o uso da chave, como vemos no exemplo 2. No entanto, se a condição verdadeira levar à execução de mais de um comando, faz-se necessário o uso das chaves abrindo e fechando o bloco de comandos, como vemos no exemplo 3.

As chaves, ou delimitadores de bloco, têm a função de agrupar um conjunto de instruções. O uso desses delimitadores é opcional caso haja apenas uma linha de código. Ainda assim, seu uso é recomendado, pois essa prática facilita a leitura e manutenção do código, tornando-o mais legível.

**Exemplo1:**

Os programas recebem os dados do usuário e têm de processar esses dados a fim de tomar decisões sobre as possíveis saídas do programa. Desse modo, precisaremos definir seqüências de execução diferentes de acordo com os dados de entrada. Para isso, fazemos uso dos comandos de seleção ou decisão.

O programa abaixo recebe como entrada as duas notas de um aluno e exibe na página HTML "true" apenas se a média das notas for maior ou igual a 7.

***Código 1 - Media de Notas.html e script.js***

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Comandos de Decisão </title>*

*</head>*

*<body>*

*<h1>Situação do Aluno</h1>*

*N1: <input type="number" id="N1" value="">*

*N2: <input type="number" id="N2" value="">*

*<button onclick="situacao('N1', 'N2', 'resultado')">OK</button>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*function situacao(entrada1, entrada2, saida) {*

*var x = Number(document.getElementById(entrada1).value);*

*var y = Number(document.getElementById(entrada2).value);*

*var media = (x + y)/2;*

*document.getElementById(saida).innerHTML =*

*"O resultado é " + resultado(media);*

*}*

*function resultado(m) {*

*return m >= 7;*

*}*

**Exemplo 2:**

E se desejássemos exibir na tela a palavra "Aprovado" se esta condição fosse satisfeita, e "Reprovado", caso contrário. Para essa situação, precisamos usar comandos de seleção de JavaScript. Assim, o nosso programa será capaz de calcular a média e decidir se o aluno está "Aprovado" ou "Reprovado".

Note que podemos trabalhar isoladamente na função resultado, fazendo com que ela retorne uma string ao invés do valor booleano resultante da comparação  m >= 7.

Vemos agora uma solução usando o comando de seleção if para retornar o texto "Aprovado" somente quando a média do aluno for maior ou igual a 7. Isso porque a expressão m >= 7 será verdadeira (true) apenas quando o valor da variável m for maior ou igual a 7, fazendo com que o comando resultado = "Aprovado" seja executado.

No caso de a expressão ser falsa, ela resultará no valor false e o comando resultado = "Aprovado" não será executado.

Caso o aluno seja aprovado, deverá ser exibido "Aprovado" ao invés de "true". Temos uma página HTML, dois campos de entrada N1 e N2 do tipo numérico.

Ao clicar no botão será invocada a função situacao, passando o nome dos dois campos de entrada N1 e N2 e do campo de saída resultado.

***Código 2 - Media de Notas.html e script.js***

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> Comandos de Decisão </title>*

*</head>*

*<body>*

*<h1>Situação do Aluno</h1>*

*N1: <input type="number" id="N1" value="">*

*N2: <input type="number" id="N2" value="">*

*<button onclick="situacao('N1', 'N2', 'resultado')">OK</button>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*function situacao(entrada1, entrada2, saida) {*

*var x = Number(document.getElementById(entrada1).value);*

*var y = Number(document.getElementById(entrada2).value);*

*var media = (x + y)/2;*

*document.getElementById(saida).innerHTML =*

*"O resultado é " + resultado(media);*

*}*

*function resultado(m) {*

*var resultado;*

*if (m >= 7) {*

*resultado = "Aprovado";*

*}*

*return resultado;*

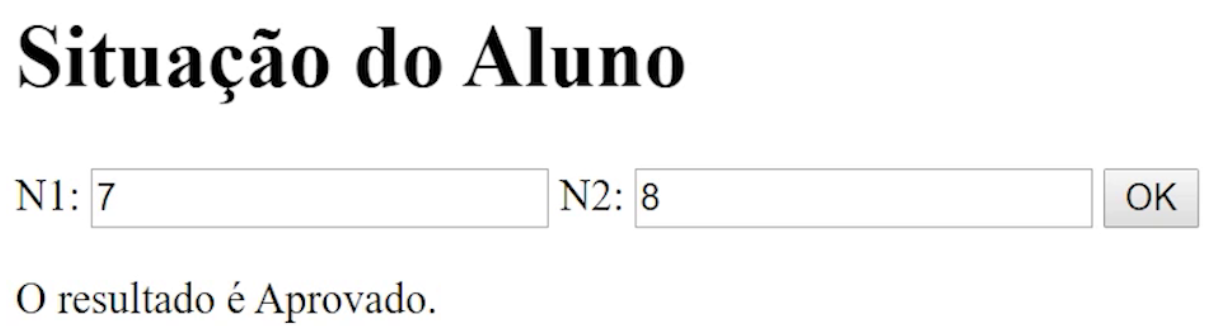
*}*

No JavaScript, veja que a função situacao recebe o nome dos campos de entrada, 1 e 2, e o campo de saída; recupera o valor numérico do campo de entrada1 e atribui o valor à variável x, e da mesma forma para a variável y, só que com entrada2; calcula a média desses dois valores e atribui a variável media; altera o valor do texto de saída com o texto: "O resultado é", e em seguida ele chama a função resultado, passando esse valor media que foi calculado na linha 04.

Observando a função resultado, ela recebe um valor numérico da média que foi calculada. E se o m >= 7, ele faz a atribuição de que resultado é igual a "Aprovado" e retorna esse resultado.

Apenas se o m > = 7 o resultado vai receber o texto "Aprovado"; dessa forma, ao digitar os valores 7 e 6, a média fica 6,5, que é um valor abaixo de 7, portanto, a linha de código 13 (resultado = "Aprovado") não será executada.

E como o valor de resultado é undefined, ele continuará undefined ao ser retornado, e isso será exibido na tela.



**if/else**

No exemplo, vimos que não estávamos exibindo "Reprovado" caso o aluno não atingisse a média 7. No entanto, considere que desejamos não somente exibir a mensagem "Aprovado" quando a média for igual ou superior a 7, mas também exibir a mensagem "Reprovado" no caso de sua média ser menor que 7. O que será preciso alterar, no exemplo mostrado anteriormente, para que isso passe a acontecer?

Uma alternativa é executar dois testes seqüencialmente, como podemos ver a seguir:

**Sintaxe do if/else:**

*if (expressão booleana)*

*{*

*// código 1*

*}*

*else*

*{*

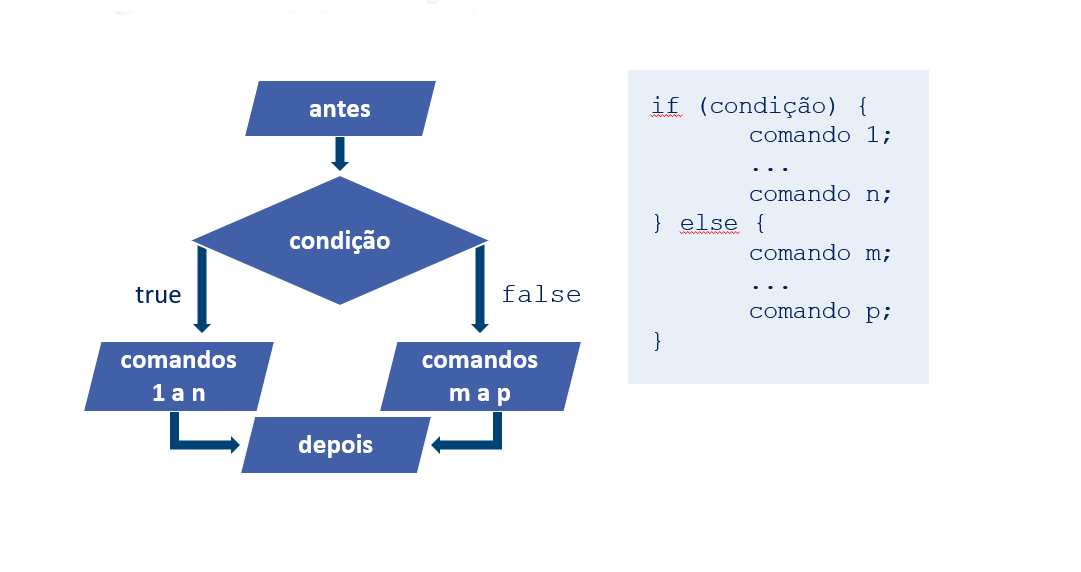
*// código 2*

*}*

Caso a expressão booleana seja verdadeira, as instruções entre chaves presentes no código 1 serão executadas; caso contrário, serão executadas as instruções presentes no código 2.

 Utiliza-se de duas condições complementares, ou seja, opostas ou contraditórias. Quando a primeira é verdadeira, a segunda é falsa e vice-versa. Nesse comando, temos que, se a condição for verdadeira, apenas os comandos do *if*  serão executados. Senão, ou seja, caso a condição seja falsa, apenas os comandos *else*  serão executados.

**Figura 06** - Comando de Seleção if-else



Apesar de existir a opção sem as chaves, apresentaremos apenas a versão recomendada, ou seja, a versão com as chaves. É importante notar que, assim como o if, o else também deve ser usado somente com letras minúsculas.

Analise como fica o exemplo ao fazer uma pequena alteração na função resultado utilizando o if e o else. Se (if) o m >= 7, será executada a linha de comando 13, sendo atribuído "Aprovado" ao resultado. Senão (else), caso o m não seja maior ou igual a 7, que é a mesma coisa que dizer que o m < 7, resultado vai receber o texto "Reprovado".

***Código 3 - Media de Notas.html e script.js***

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> Comandos de Decisão</title>*

*</head>*

*<body>*

*<h1>Situação do Aluno</h1>*

*N1: <input type="number" id="N1" value="">*

*N2: <input type="number" id="N2" value="">*

*<button onclick="situacao('N1', 'N2', 'resultado')">OK</button>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*function situacao(entrada1, entrada2, saida) {*

*var x = Number(document.getElementById(entrada1).value);*

*var y = Number(document.getElementById(entrada2).value);*

*var media = (x + y)/2;*

*document.getElementById(saida).innerHTML =*

*"O resultado é " + resultado(media);*

*}*

*function resultado(m) {*

*var resultado;*

*if (m >= 7) {*

*resultado = "Aprovado";*

*} else {*

*resultado = "Reprovado";*

*}*

*return resultado;*

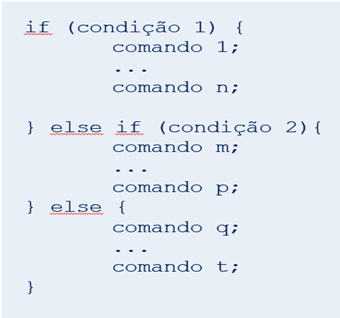
*}*

**if/else/if (encadeados)**

Até agora, vimos que o exemplo trabalha apenas com duas condições complementares. E se tivermos mais condições? Por exemplo, se desejarmos que o programa continue retornando "Aprovado" para médias maiores que 7, mas que ele retorne "Recuperação" para notas maiores ou iguais a 5 e menores que 7 e retorne “Reprovado” apenas para notas menores que 5.

Nesse caso, é indicado utilizarmos uso da instrução *if-else*  encadeada, cuja forma geral é apresentada abaixo. Nesse comando, temos que, se a condição 1 for verdadeira, apenas os comandos 1 a n serão executados. Caso a condição 1 seja falsa, verificamos a condição 2. Se ela for verdadeira, apenas os comandos m a p serão executados; caso contrário, ou seja, caso a condição 2 também seja falsa, apenas os comandos q a t serão executados. Assim, para que os comandos q a t sejam executados, será necessário que as condições 1 e 2 sejam falsas.

**Figura 08** - Comando de Seleção if-else Encadeado



É importante ressaltar que não existe um limite para o encadeamento de condições.

Veja no Código 4 que podemos alterar um pouco mais a função resultado e ter o *if-else* encadeado. Assim, se (if) o m for maior ou igual a 7, resultado recebe o texto "Aprovado". Senão (else), se o m não for maior ou igual a 7 e for maior ou igual a 5, resultado recebe o texto "Recuperação". Senão (else), resultado recebe o texto "Reprovado"

***Código 4 - Media de Notas.html e script.js***

*html >*

*<head>*

*<meta charset="UTF-8" />*

*<title> Comandos de Decisão </title>*

*</head>*

*<body>*

*<noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>*

*<h1>Situação do Aluno</h1>*

*N1: <input type="number" id="N1" value="">*

*N2: <input type="number" id="N2" value="">*

*<button onclick="situacao('N1', 'N2', 'resultado')">OK</button>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*function situacao(entrada1, entrada2, saida) {*

*var x = Number(document.getElementById(entrada1).value);*

*var y = Number(document.getElementById(entrada2).value);*

*var media = (x + y)/2;*

*document.getElementById(saida).innerHTML =*

*"O resultado é " + resultado(media);*

*}*

*function resultado(m) {*

*var resultado;*

*if (m >= 7) {*

*resultado = "Aprovado";*

*} else if (m >=5) {*

*resultado = "Recuperação";*

*} else {*

*resultado = "Reprovado";*

*}*

*return resultado;*

*}*

Temos agora o if-else encadeado, que verifica primeiro se o m >= 7, atribuindo o texto "Aprovado". Caso seja falso, será feito um novo teste para verificar se o m é maior ou igual a 5, atribuindo o texto "Recuperação" à resultado. Caso esta segunda condição também seja falsa, ou seja, o m é menor que 7 e menor que 5, entramos na condição else e o resultado recebe o texto "Reprovado".

Você pode ir à página e verificar que, se forem adicionados os valores numéricos 7 e 8, o aluno, nesse caso, estará aprovado. Agora, você verificará uma média entre os valores 5 e 7, digitando as notas 7 e 4. O aluno ficará com a média 5,5 e ao clicar OK aparecerá o resultado "Recuperação", entrando no caso intermediário. Faça um novo teste para as notas 5 e 4. O aluno ficará com a média 4,5 e ao clicar OK o resultado será "Reprovado", como exposto na Figura 9. Veja que você terá o if-else encadeado e, como já foi dito, não tem um limite para esse encadeamento.

**Figura 09** - Saída na Página HTML



JavaScript também oferece um operador condicional ternário que recebe uma condição e duas expressões.

Para isso, o formato utilizado é condição?expressão1:expressão2. Esse operador retorna a primeira expressão, caso a condição seja verdadeira; caso contrário, retorna a segunda expressão.

Ele é muito utilizado para fazer uma atribuição condicional, a qual corresponde à atribuição que vimos anteriormente usando if-else. Porém, note que fazemos a atribuição condicional com um único comando. Se m for maior ou igual a 7, a variável resultado receberá o valor "Aprovado". Caso contrário, ou seja, caso m seja menor que 7, a variável resultado receberá o valor "Reprovado".

***script.js***

*function situacao(entrada1, entrada2, saida) {*

*var x = Number(document.getElementById(entrada1).value);*

*var y = Number(document.getElementById(entrada2).value);*

*var media = (x + y)/2;*

*document.getElementById(saida).innerHTML =*

*"O resultado é " + resultado(media);*

*}*

*function resultado(m) {*

*var resultado = (m >= 7) ? "Aprovado":"Reprovado";*

*return resultado;*

*}*

Qualquer expressão que retorne um valor booleano pode ser usada em uma condição do comando if-else. Para isso, além dos valores primitivos true e false, também podemos usar os operadores de comparação para determinar a igualdade ou diferença entre expressões.

**Os operadores básicos de comparação são:**

e1 == e2, que retorna true apenas se o valor da expressão e1 for igual ao valor da expressão e2;

e1 === e2, também chamada de igualdade estrita, que retorna true apenas se o valor e o tipo da expressão e1 e e2 forem iguais;

e1 != e2, que retorna true apenas se o valor da expressão e1 for diferente do valor da expressão e2;

e1 !== e2, também chamada de desigualdade estrita, que retorna true apenas se o valor da expressão e1 for diferente do valor da expressão e2 ou se o tipo da expressão e1 for diferente do tipo da expressão e2;

e1 > e2, que retorna true apenas se o valor da expressão e1 for maior que valor da expressão e2;

e1 < e2, que retorna true apenas se o valor da expressão e1 for menor que valor da expressão e2;

e1 >= e2, que retorna true apenas se o valor da expressão e1 for maior ou igual ao valor da expressão e2;

e1 <= e2, que retorna true apenas se o valor da expressão e1 for menor ou igual ao valor da expressão e2;

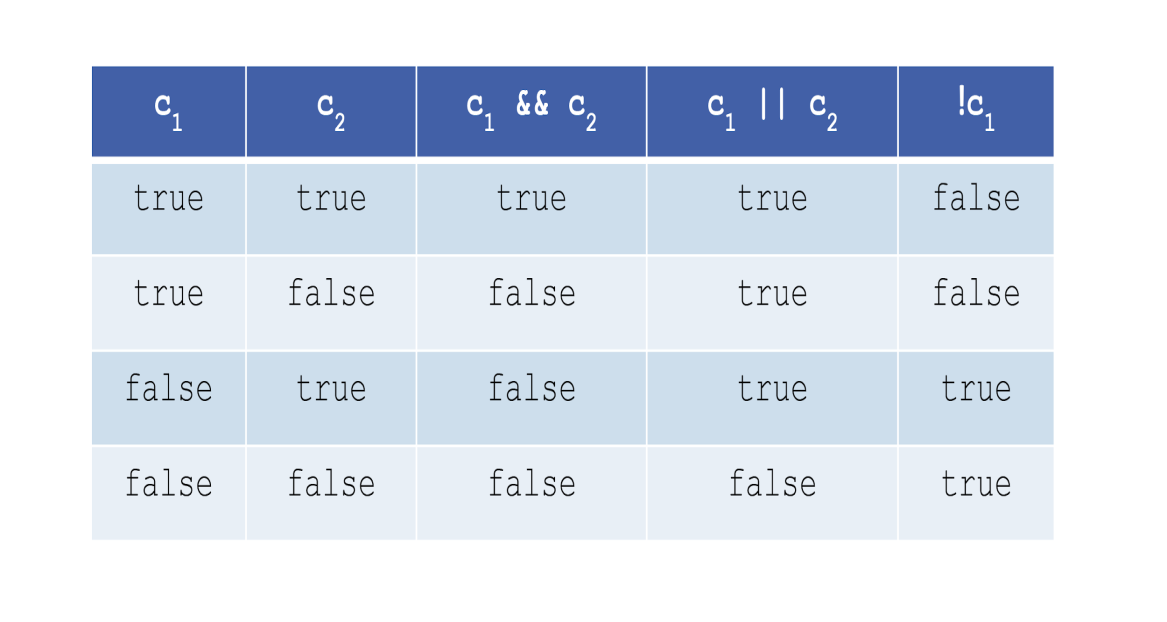
Além disso, podemos combinar expressões que usam esses operadores básicos por meio dos operadores lógicos. São eles e, ou e não. Na (Figura 2), podemos ver uma tabela que chamamos de tabela-verdade, a qual apresenta o resultado desses operadores.

**Dadas as duas condições c1 e c2:**

* Usamos c1 && c2 para obter o valor de c1 e c2
* Usamos c1 || c2 para obter o valor de c1 ou c2
* Usamos !c1 para obter o valor de não c1

No e-lógico temos que c1 && c2 é true apenas se ambas as condições forem true. Ou seja, temos que apenas true && true é true; todas as outras linhas são false. Por outro lado, no ou-lógico, temos que c1 || c2 é false apenas se ambas as condições forem false. Ou seja, temos que apenas false || false é false; todas as outras linhas são true. Por fim, é simples ver que !true é false e que !false é true.

**Figura 02** - Combinando Condições



A comparação de dados de diferentes tipos pode gerar resultados inesperados. Por exemplo, ao comparar uma string com um número, o JavaScript converterá, se possível, a string em um número e fará a comparação.

Uma string vazia é convertida em 0 e strings não numéricas são convertidas em NaN. Nesse caso, a comparação resultará sempre no valor false. Essa conversão também é chamada de coerção de tipos.

A coerção de tipos tem efeitos interessantes. Isso porque todo valor em JavaScript pode ser tratado como verdadeiro ou falso.

Basicamente, JavaScript considera tudo que tem valor como true e tudo que não tem um valor como false. Outra característica interessante é que também são considerados como verdadeiros objetos cujos valores são diferentes de null.

Sabe para que você pode usar isso? Para verificar, por exemplo, a existência de um elemento na sua página HTML.

Vamos ver isso na prática:

Inicialmente, nesse exemplo, será apresentado como é que alguns valores de outros tipos são interpretados quando são convertidos num valor booleano, se ele é true ou false. Dito de maneira bem simples, se a variável tem um valor, ela é true, caso contrário, ela é false.

No exemplo, é dada uma página bastante simples com um campo resultado que você utilizará para ficar escrevendo. Entre as linhas 20 a 25 do arquivo JavaScript, tem-se um if-else que simplesmente pega a variável b, e é a variável que você ficará atribuindo valores de outros tipos e verificará se ela é true ou false. Se for true, você pegará o texto que começou com o valor da variável b e acrescentará a ele o texto "transformou em true".

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> Comandos de Decisão </title>*

*</head>*

*<body>*

*<h1>Valores Verdadeiros e Falsos</h1>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*/ Valores Falsos*

*var b = false;*

*//var b = 0;*

*//var b = '';*

*//var b = 10 / 'score';*

*//var b;*

*// Valores Verdadeiros*

*//var b = true;*

*//var b = 1;*

*//var b = 10 / 5;*

*//var b = 'carrot';*

*//var b = 'true';*

*//var b = '0';*

*//var b = 'false';*

*//var b = null;*

*var texto = b;*

*if (b) {*

*texto = texto + " transformou em true";*

*} else {*

*texto = texto + " transformou em false";*

*}*

*// Procurando elementos na página*

*var nome\_elemento = "resultado";*

*if (document.getElementById(nome\_elemento)) {*

*document.getElementById(nome\_elemento).innerHTML = texto;*

*} else {*

*alert("Não encontrei o elemento "+ nome\_elemento)*

*}*

*// Note que o código abaixo não faz a mesma coisa*

*// pois document.getElementById(nome\_elemento) retorna um objeto*

*/\**

*if (document.getElementById(nome\_elemento) == true) {*

*document.getElementById(nome\_elemento).innerHTML = texto;*

*} else {*

*alert("Não encontrei o elemento "+ nome\_elemento)*

*}*

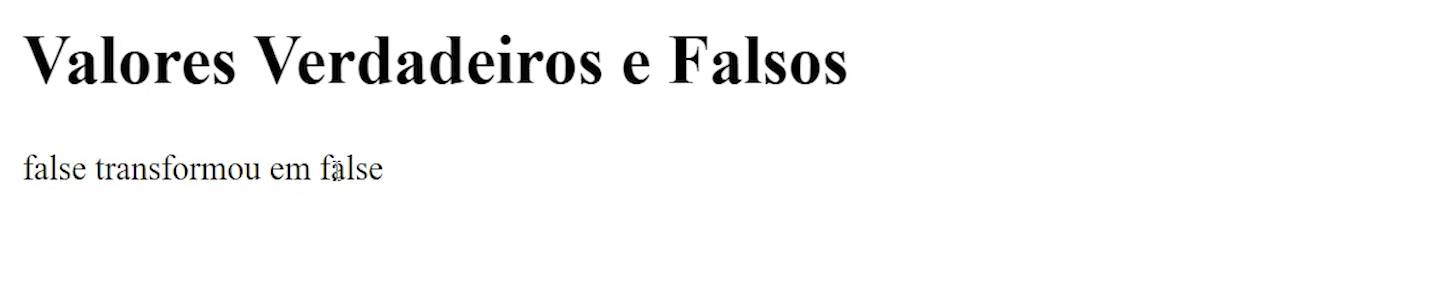
*\*/*

Ou seja, se passarmos um valor true, será exibido: "transformou em true", caso contrário, se for else, é porque o b transformado em booleano resultou em false, então você colocará: "transformou em false".

No exemplo, começa na linha 3 com a variável b recebendo false. E o que é esperado é que seja exibido "false transformou em false", afinal, false já é um valor booleano. Se você voltar pra página HTML, confirmará que é exibido: “false transformou em false".

Veja a Figura 3.

**Figura 03** - Exibindo valores booleanos



Anteriormente, foi mostrado o comando if-else, o qual pode ser utilizado para resolver problemas de seleção. Porém, em alguns casos, o uso desse comando, quando encadeado, torna a escrita e a compreensão do código mais trabalhosa.

Por exemplo, considere a função apresentada na (Figura 1) que retorna o nome do dia da semana passado como número. Lembre-se que em JavaScript os dias da semana são representados como números no intervalo que vai de 0 (para o domingo) até 6 (para o sábado). Nessa função, comparamos o valor recebido primeiramente com 0. Em caso afirmativo, atribuímos o texto "Domingo" à resposta. Caso contrário, comparamos o valor recebido com 1. Em caso afirmativo, atribuímos o texto "Segunda-feira" à resposta. Caso o valor recebido não seja 1, continuamos encadeando if’s até chegarmos na comparação com o 6.

**Figura 01** - Comando de Seleção if encadeado

**

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Condicional Encadeado</title>*

*</head>*

*<body>*

*<h1>Dia da Semana Usando if</h1>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*var data = new Date(1969, 06, 20);*

*var dia = data.getDay();*

*document.getElementById("resultado").innerHTML = dia\_da\_semana(dia);*

*function dia\_da\_semana(dia) {*

*var resposta = "Dia Inválido";*

*if (dia == 0) {*

*resposta = "Domingo";*

*} else if (dia == 1) {*

*resposta = "Segunda-feira";*

*} else if (dia == 2) {*

*resposta = "Terça-feira";*

*} else if (dia == 3) {*

*resposta = "Quarta-feira";*

*} else if (dia == 4) {*

*resposta = "Quinta-feira";*

*} else if (dia == 5) {*

*resposta = "Sexta-feira";*

*} else if (dia == 6) {*

*resposta = "Sábado";*

*}*

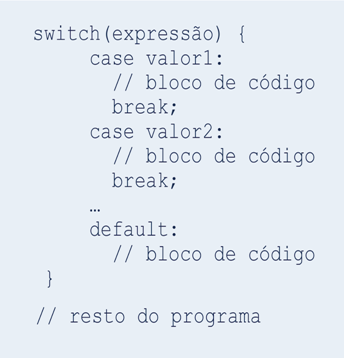
*return resposta;*

*}*

**switch**

O comando switch facilita a escrita de trechos de programa como este em que a seleção deve ser feita entre várias alternativas. O switch é chamado de comando interno de seleção múltipla, ele compara o valor de uma expressão com uma lista de valores. Na (Figura 2), podemos ver a estrutura geral desse comando, o qual funciona da seguinte maneira: a expressão passada para o comando é avaliada uma vez. Esse valor é comparado, em ordem, com os valores das constantes especificadas nos comandos case. Quando ocorrer uma condição em que a expressão é igual, a sequência de comandos associada ao case em questão será executada até chegar ao comando break, que para a execução do case em questão e salta para a próxima linha de código após o bloco do switch. O comando default será apenas executado se nenhum valor for igual ao valor da expressão passada para o comando. O comando default é opcional. Se ele não existir, nenhuma ação será realizada caso todos os testes falhem.

**Figura 02** - Comando switch



Vamos ver como fica a mesma função usando o switch:

Nesse exemplo (ver Código 2), usaremos novamente uma página muito simples em que temos apenas um parágrafo cujo identificador é "resultado" e escreveremos nesse resultado, ok? Você aprendeu na aula sobre datas que a primeira vez que o homem pisou na lua foi no dia 20 de julho de 1969, era um domingo. Usaremos esse dia histórico no nosso exemplo.

**Código 2** - Dia da Semana com switch.html

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Switch</title>*

*</head>*

*<body>*

*<h1>Dia da Semana Usando switch</h1>*

*<p id="resultado"></p>*

*<script src="script.js"></script>*

*</body>*

*</html>*

***script.js***

*var data = new Date(1969, 06, 20);*

*var dia = data.getDay();*

*document.getElementById("resultado").innerHTML = dia\_da\_semana(dia);*

*function dia\_da\_semana(dia) {*

*var resposta = "";*

*switch (dia) {*

*case 0:*

*resposta = "Domingo";*

*break;*

*case 1:*

*resposta = "Segunda-feira";*

*break;*

*case 2:*

*resposta = "Terça-feira";*

*break;*

*case 3:*

*resposta = "Quarta-feira";*

*break;*

*case 4:*

*resposta = "Quinta-feira";*

*break;*

*case 5:*

*resposta = "Sexta-feira";*

*break;*

*case 6:*

*resposta = "Sábado";*

*break;*

*default:*

*resposta = "Dia Inválido";*

*}*

*/\**

*switch (dia) {*

*case 0:*

*case 6:*

*resposta = resposta + ". Fim de Semana.";*

*break;*

*case 1:*

*case 2:*

*case 3:*

*case 4:*

*case 5:*

*resposta = resposta + ". Dia Útil.";*

*break;*

*}*

*\*/*

*return resposta;*

*}*

Temos a criação de uma variável data com os campos do ano de 1969. O mês de julho é o mês 06, lembre-se que em JavaScript a numeração dos meses começa em 0, então 0 é janeiro, logo julho vai ser o mês 6 e o dia 20. Pegamos o dia dessa data, usando o método getDay, em seguida passamos esse dia, ou melhor, esse número que corresponde ao dia da semana dessa data. Passaremos para a função dia\_da\_semana, que vai retornar o texto em português, com o nome em português daquele dia da semana. E essa função, que é o que nos interessa, foi declarada usando o switch.

Veja a Figura 3.

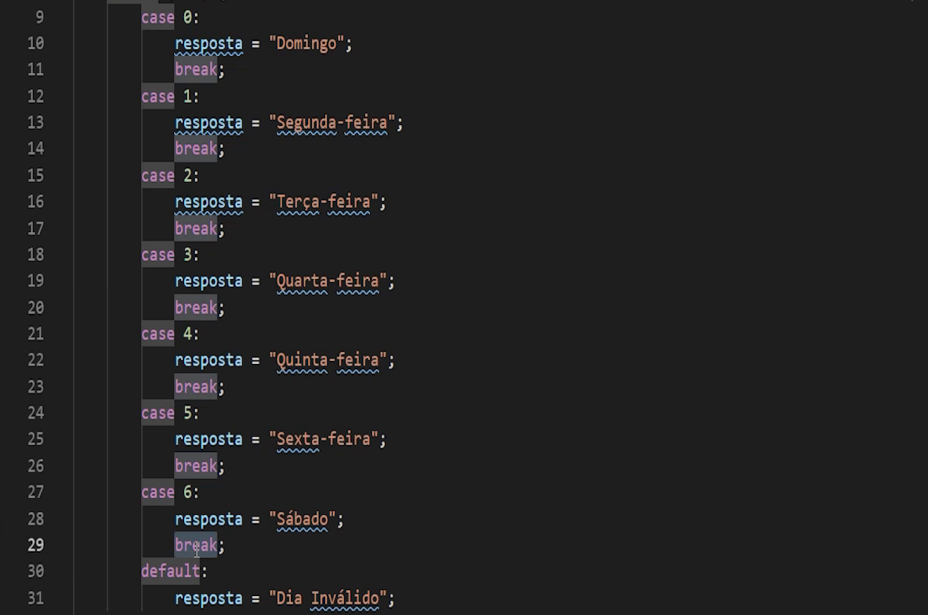
**Figura 03** - Comando switch no Exemplo



Ela recebe o número associado ao argumento dia e faz um switch no valor dessa variável chamada dia. Temos o case se o valor for 0, diz que a resposta é "Domingo" e temos que dar o break para sair do bloco do switch. Caso seja 1, é "Segunda-feira", caso seja 2, é "Terça-feira" e assim por diante até chegar no 6, que é o "Sábado", na linha 28, depois temos o break, e, por fim, nas linhas 30 e 31, um caso default. Então se o valor passado não for 0, 1, 2, 3, 4, 5 ou 6, cairemos no valor default e a resposta que será retornada é "Dia Inválido".

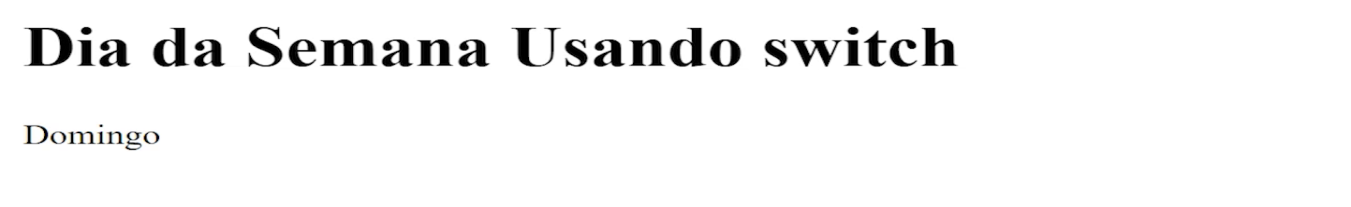
Veja a Figura 4.

**Figura 04** - Caso default



Como já foi dito, esse dia histórico aconteceu no Domingo e, ao carregar a página HTML, de fato, o texto que aparece é "Domingo", conforme esperado, como exposto na Figura 5.

**Figura 05** - Exibição da Página HTML



Agora, iremos à linha 4 e passaremos o valor 9, por exemplo. Mas, no exemplo, o switch só faz case de 0 a 6, então o que é que acontece ao passar esse valor? O resultado será "Dia Inválido", como exposto na Figura 6, porque ele comparou esse valor com 0, 1, 2, 3, 4, 5, 6, e ele não é igual a nenhum desses valores, então ele caiu no valor default.

**Figura 06** - Exibição da Página HTML do Caso default



Por fim, veremos o que acontece se retirarmos o caso default. Apagaremos as linhas 30 e 31 e passaremos o valor 9, como foi feito na linha 4. A função vai comparar esse valor com 0, 1, 2 3, 4, 5, 6, e será diferente em todos os casos. Não teremos um caso default, e o comando switch não fará absolutamente nada.

Como a resposta começou com o valor vazio na linha 7, vai ser retornado à string vazia. Se formos para a página HTML, nada será exibido, porque a função retorna a string vazia, como apresentado na Figura 7.

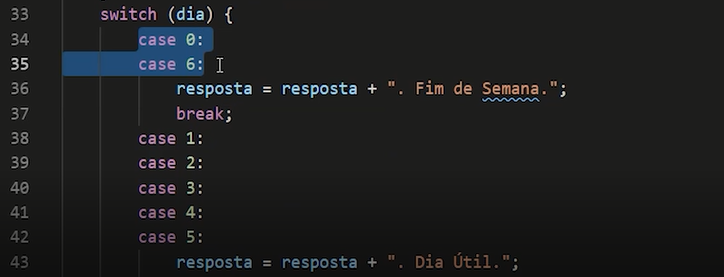
**Figura 07** - Exibição da Página HTML sem um Caso default



Manteremos a versão original colocando o dia na chamada da função dia\_da\_semana e teremos o switch que dirá qual foi o dia da semana em português, se aquele valor corresponde, e criaremos um novo switch que faz o seguinte: para os case 0 e case 6, ele executa a linha de código: resposta = resposta + ". Fim de Semana". Então 0 é o "Domingo" e 6 é o "Sábado". Ele vai colocar ao final do texto concatenado à resposta que nós já começamos a criar no switch anterior o texto "Fim de Semana" Caso contrário, se for case 1, 2, 3, 4, 5, veremos que será acrescentada à resposta o texto "Dia Útil".

Faça o teste e verifique se o Domingo aparece como fim de semana. Você pode também voltar e colocar um dia da semana, coloque o 5, por exemplo, que é uma sexta-feira. O resultado deverá ser: "Sexta-feira. Dia Útil". Nesse exemplo, tivemos um switch normal, mas também temos um switch em que vários cases compartilham do mesmo código, como o que acontece nas linhas 34 e 35, exposto na Figura 8.

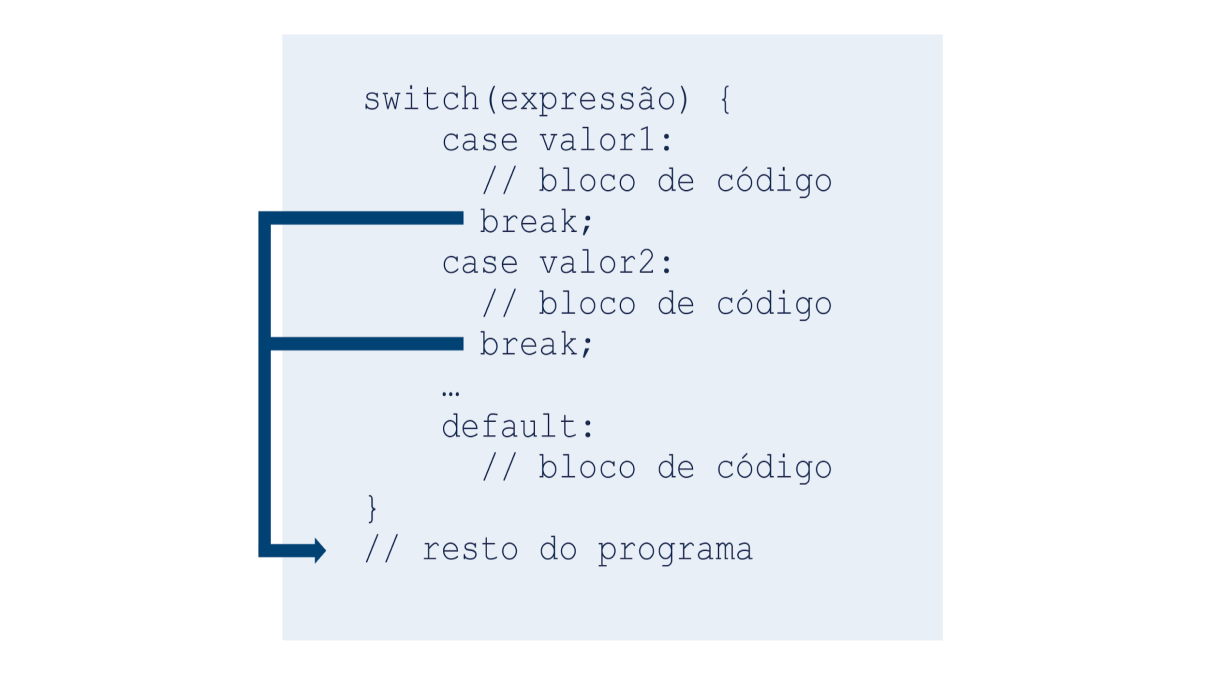
**Figura 08** - Comando switch com casos compartilhados



O comando switch possui algumas peculiaridades que merecem a nossa atenção. Primeiro, quando o JavaScript alcança o comando break, ele sai do bloco do switch. Isso interromperá a execução dentro do bloco. Note que não é necessário usar o break no último caso do bloco, pois ele termina lá de qualquer maneira. Além disso, se você omitir a instrução break, o próximo caso será executado mesmo que a avaliação não seja correspondente.

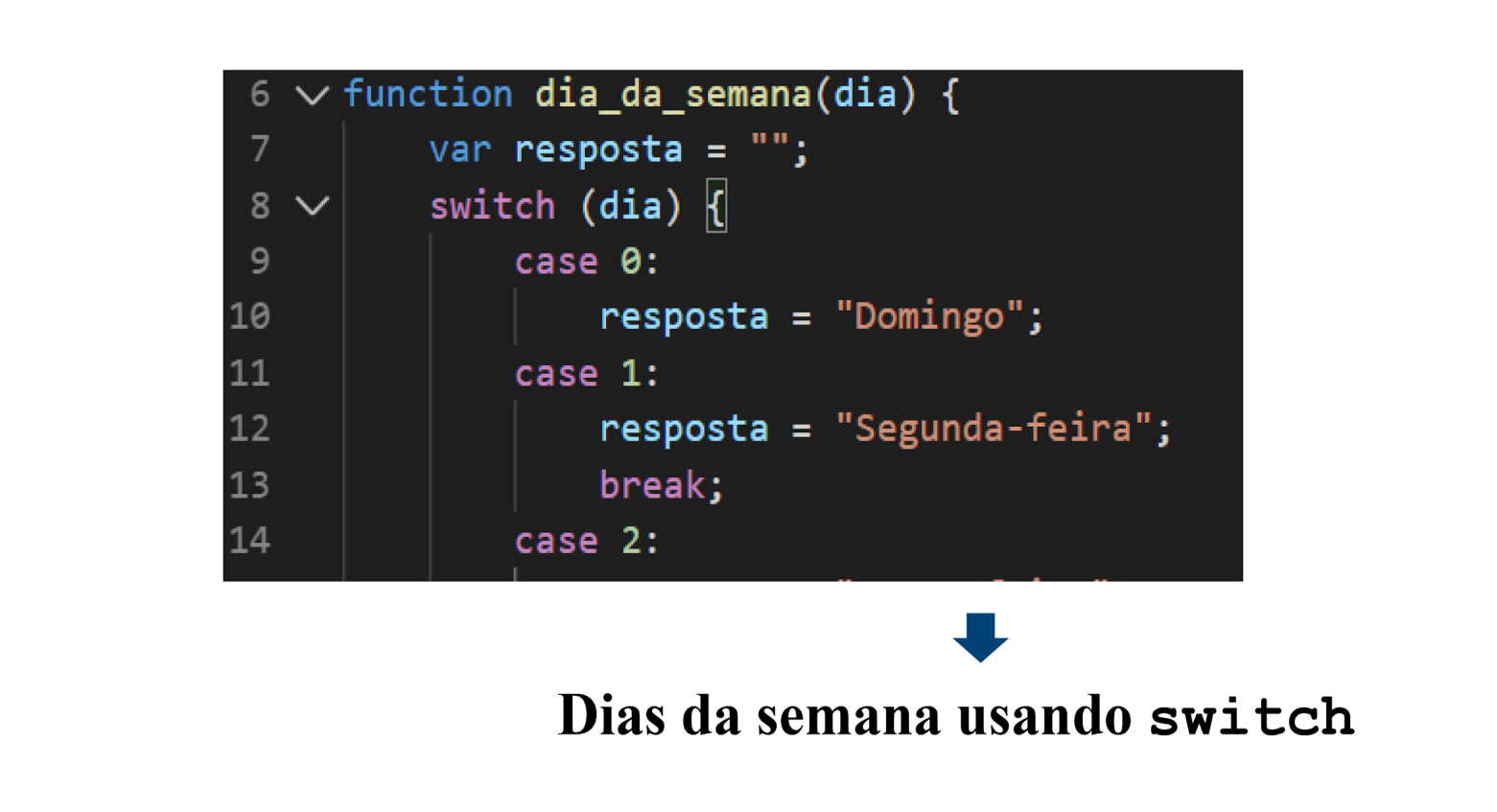
Veja a Figura 9.

**Figura 09** - Comando break



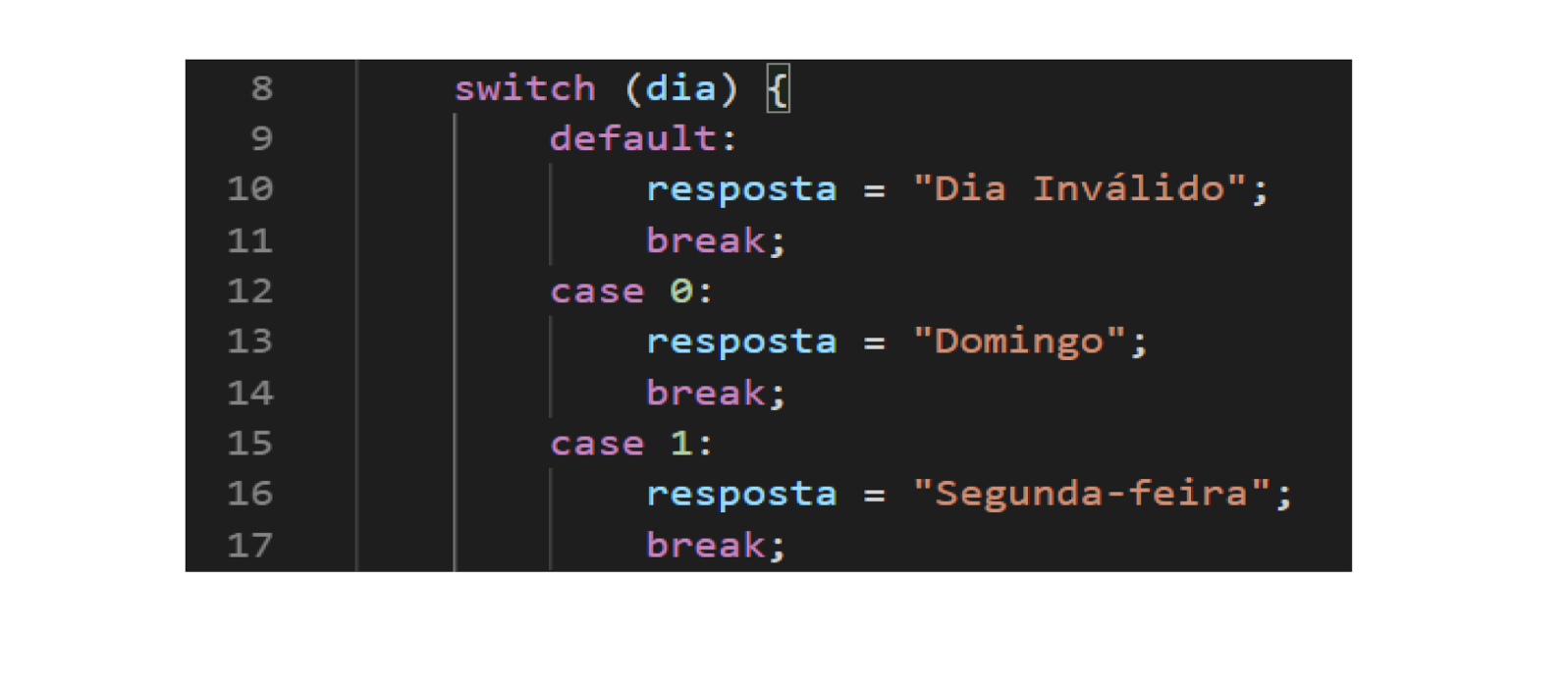
No exemplo, se esquecermos de colocar o break no caso 0, como vemos na (Figura 10), a página exibirá o texto "Segunda-feira". Isso porque, como esquecemos o break, a atribuição da linha 12 será executada e apenas o break da linha 13 fará com que a execução saia do bloco switch.

**Figura 10** - Esquecendo o break



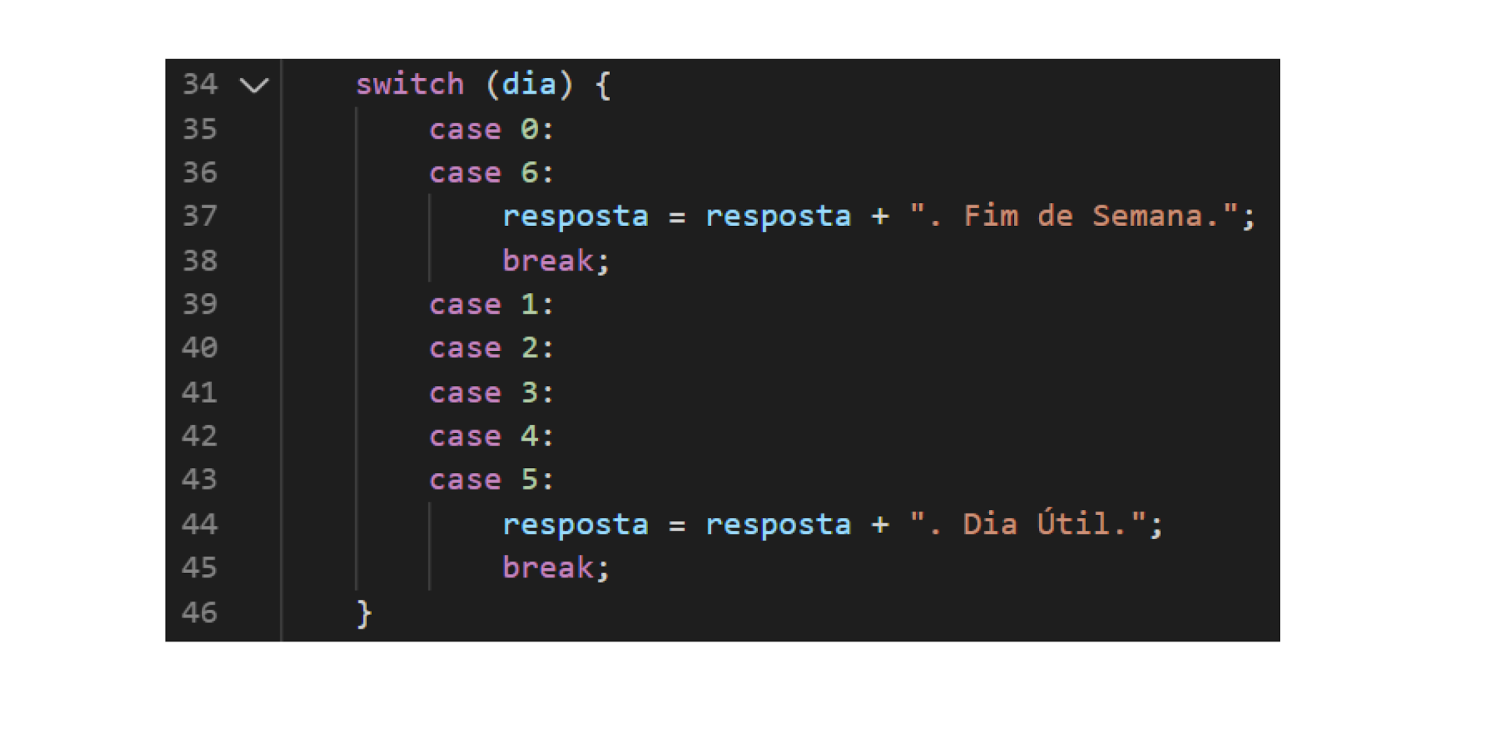
Outro ponto interessante é que o caso default não precisa ser o último. No entanto, caso ele não seja o último, como podemos ver na (Figura 11), você deverá usar o break da mesma maneira que precisa usar nos outros casos.

**Figura 11** - Localização do Comando default



Além disso, casos diferentes podem compartilhar o mesmo trecho de código. No exemplo (Figura 12), os casos 0 e 6 compartilham o mesmo bloco de código, ou seja, a atribuição da linha 37. Por outro lado, os casos 1, 2, 3, 4 e 5 compartilham outro bloco de código, ou seja, a atribuição da linha 44.

**Figura 12** - Compartilhando Comandos



Você deve estar se perguntando o que acontece se o valor for igual em mais de um caso. Nessa situação, apenas o primeiro caso será selecionado. Veja na (Figura 13) que, se tivermos dois casos com o valor 0, apenas o primeiro caso é selecionado. Dessa forma, apenas a atribuição da linha 10 foi executada e o texto "Domingo" foi exibido na página. Note que a atribuição da linha 13 NÃO foi executada. E lembre-se, se nenhum caso tiver o valor igual ao da expressão passada, o caso default será executado. Se ele não existir, o comando switch terminará sem fazer nada.

**Figura 13** - Casos com o mesmo valor



Por fim, é muito importante saber que JavaScript usa a comparação estrita, ou seja ===, entre o valor da expressão e os valores dos casos. Você deve lembrar que essa comparação apenas é verdade quando o valor e o tipo das expressões são iguais. Por exemplo, a comparação do texto "0" com o número 0 usando a comparação normal, ou seja, "0" == 0, retorna true. No entanto, a função apresentada no (Código 3) faz a declaração de uma variável chamada dia, cujo valor é o texto "0" e retorna o texto "Outro dia". Note que esse é o valor atribuído à saída no caso default e não no caso 0, pois a comparação estrita "0" === 0 retorna false porque as expressões têm tipos diferentes, String e Número.

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Switch</title>*

*</head>*

*<body>*

*<h1>Comparação Estrita do Switch</h1>*

*<p id="resultado"></p>*

*<script src="09\_12 Comparação Estrita do Switch.js"></script>*

*</body>*

*</html>*

***script.js***

*document.getElementById("resultado").innerHTML = dia\_da\_semana();*

*function dia\_da\_semana(dia) {*

*var dia = "0";*

*var resposta = "";*

*switch (dia) {*

*case 0:*

*resposta = "Domingo";*

*break;*

*default:*

*resposta = "Outro dia";*

*}*

*return resposta;*

*}*

Você conheceu os comandos de seleção if e switch. Você deve ter notado que eles são ligeiramente diferentes: o switch testa igualdade, enquanto o if testa uma determinada condição lógica, ou seja, uma expressão que pode assumir valor verdadeiro ou falso.

Você deve ter observado também que o comando if pode vir associado ao else, no caso de duas situações complementares (quando uma for verdadeira, a outra é falsa e vice-versa). Além disso, comandos if-else podem ser encadeados para melhorar a legibilidade. No entanto, note que em uma série de comandos if todas as condições serão verificadas mesmo se uma correspondência tiver sido encontrada. Dessa forma, para algumas situações como a que apresentamos no início da aula, onde tínhamos dois blocos if em sequência comparando a nota m com o valor 7, o desempenho desse método é mais lento que o do switch.

**Comandos de repetição(iteração)**

Você aprendeu a utilizar os comandos de seleção. O comando if-else, por exemplo, testa uma expressão booleana. Já o comando switch compara o valor de uma expressão com uma lista de valores. Esses dois comandos são utilizados para decidir qual bloco de código deverá ser executado, certo?

Entretanto, existem situações nas quais você precisará repetir a execução de um mesmo bloco de código. Por exemplo, para calcular a nota de todos os alunos de uma sala, você precisará executar o código de cálculo de média repetidamente, uma vez para cada aluno da sala.

Os comandos utilizados em tais situações, são chamados de comandos de iteração.

São eles:

* for: percorre um bloco de código várias vezes;
* for/in: percorre as propriedades de um objeto;
* for/of: percorre os valores de um objeto iterável;
* while: percorre um bloco de código enquanto uma condição especificada é verdadeira;
* do/while: também percorre um bloco de código enquanto uma condição especificada é verdadeira, mas executa esse bloco de código pelo menos uma vez.

Vamos começar pelo termo iteração. O termo "iteração" significa "repetição".

Esses comandos são utilizados em situações em que o computador precisa executar certos trechos de um programa de forma repetida, ou seja, várias vezes, até que uma determinada condição seja alcançada. Alguns autores chamam essas estruturas de "laços", "laços de repetição" ou, em inglês, loops.

Você perceberá que cada um dos comandos de iteração disponíveis na linguagem JavaScript são, de maneira geral, equivalentes, mas cada um deles pode ser utilizado mais facilmente em situações específicas. Então fique atento para saber fazer a escolha certa quando for utilizar cada um deles.

**for**

O primeiro comando de iteração que você irá aprender é o laço for. Imagine que você precisasse exibir na página HTML os números de 1 a 10. Como você faria?

Uma possibilidade é escrever o código exibido no exemplo. Note que inicializamos o texto com a string vazia na linha 2 e, depois disso, para cada um dos valores de 1 a 10, concatenamos o texto com esse valor.

**Código 1** - Contagem\_For.html

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Comandos de Iteração</title>*

*</head>*

*<body>*

*<h1>Contagem</h1>*

*<button onclick="contagem()">OK</button>*

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*function contagem() {*

*var texto = "";*

*texto = texto + 1 + "<br>";*

*texto = texto + 2 + "<br>";*

*texto = texto + 3 + "<br>";*

*texto = texto + 4 + "<br>";*

*texto = texto + 5 + "<br>";*

*texto = texto + 6 + "<br>";*

*texto = texto + 7 + "<br>";*

*texto = texto + 8 + "<br>";*

*texto = texto + 9 + "<br>";*

*texto = texto + 10 + "<br>";*

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

*}*

Existem, porém, dois problemas nessa solução. O primeiro é que há uma duplicação de código onde comandos similares são usados para concatenar o valor ao texto várias vezes, mudando apenas o valor. O segundo problema é que esse código tem um comportamento fixo, ou seja, não é possível escolher a quantidade de vezes que o comando de concatenação será executado depois que o código estiver em execução. Deve haver uma forma de receber do usuário um número positivo N e imprimir de 1 (um) até esse número N.

Para evitar duplicação de código usamos os chamados comandos de iteração, laços ou loops.

O primeiro comando que veremos aqui é o comando for. Veja como funciona o mesmo código agora implementado com esse comando de iteração.

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>Comandos de Iteração </title>*

*</head>*

*<body>*

*<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>*

***script.js***

*function contagem(entrada1) {*

*var texto = "";*

*var n = Number(document.getElementById(entrada1).value);*

*for (i = 1; i <= n; i++) {*

*texto = texto + i + "<br>";*

*}*

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

*}*

No exemplo, temos uma página HTML bem simples. Há apenas um campo de número e, ao clicar no botão "OK", irá chamar a função contagem, passando o nome desse campo de número, chamado de n1. Logo, será apresentado como é essa função contagem.

Essa função contagem recebe um campo e é preciso buscar o valor numérico que está nele e atribuir esse valor à variável n. E na linha 2, há o texto que começa vazio. Já na linha 5 vem a parte principal desse exemplo, que é o laço for. Em seguida, tem os campos que serão descritos daqui a pouco e, entre as chaves, o comando que vai ser executado cada vez que passar por esse laço.

Esse laço começa na linha 5, após o primeiro parêntese, onde define uma variável i que começa com o valor 1 e, enquanto esse i for menor ou igual a n, que é o valor colocado na página HTML, e foi pego de lá, como descrito na linha 4, será executado o comando da linha 6. Será acrescentado ao texto o valor i, e uma tag de quebra de texto de HTML <br>, ou seja, vai adicionar o i e quebrar a linha (linha 6). No final incrementa o i, vai acrescentar e passar com i igual a 1, 2, 3, 4, até o valor que o usuário colocar na página HTML.

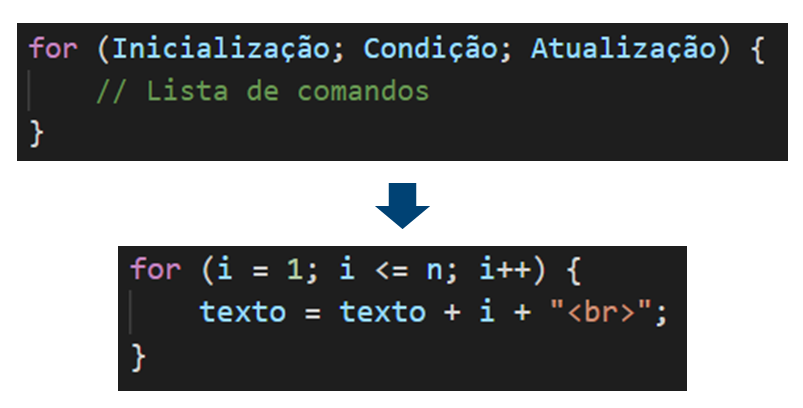
**Figura 01** - Saída da Contagem



Você notou que o programa ficou bem mais enxuto em termos de número de linhas de código. Vamos agora entender como funciona esse código mostrado. Para isso, observemos primeiro, o formato geral do laço de iteração for em JavaScript, o qual é o mesmo encontrado em diversas outras linguagens de programação.

O laço for possui três instruções: uma inicialização, uma condição e uma atualização.

**Figura 02** - Estrutura do Comando For



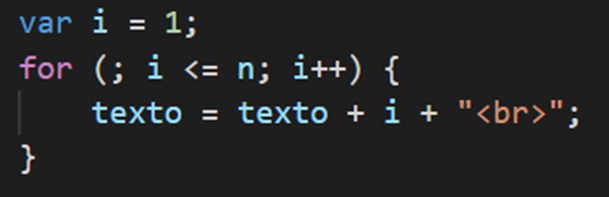
A inicialização é geralmente utilizada para declarar uma variável que será utilizada como variável de controle de execução do laço. No exemplo, temos a inicialização da variável i com o valor 1. Observe que essa variável está sendo declarada e inicializada ao mesmo tempo e antes da execução do laço. Depois da inicialização da variável, temos a condição. Nessa condição, você deve colocar uma expressão que resulte no valor verdadeiro quando for necessário executar o laço, e falso caso esteja na hora de pará-lo. No nosso exemplo temos a condição i <= n, ou seja, o laço será executado caso o valor de i seja menor ou igual a n. Caso contrário, a execução do laço termina.

Vejamos como isso funciona. Ao executar o bloco de código do comando for (lista de comandos entre chaves), essa condição é testada. Se seu valor retornar true, o laço será executado mais uma vez para depois testar novamente a condição. Caso contrário, ele irá parar. Isso quer dizer que o laço mostrado continuará a ser executado até o momento no qual a variável i tiver valor maior que n. Para que essa condição de parada do laço chegue, usualmente você utilizará a seção de atualização do laço for, a qual é a última do comando for.

Essa seção é utilizada para você alterar valores de variáveis, geralmente, o valor da variável de controle. No exemplo, temos o incremento i++, que corresponde ao comando i = i + 1. Dessa forma, a instrução i++ incrementa o valor de i em uma unidade. É um comando bem enxuto e bastante utilizado dentro da seção de atualização dos laços for.

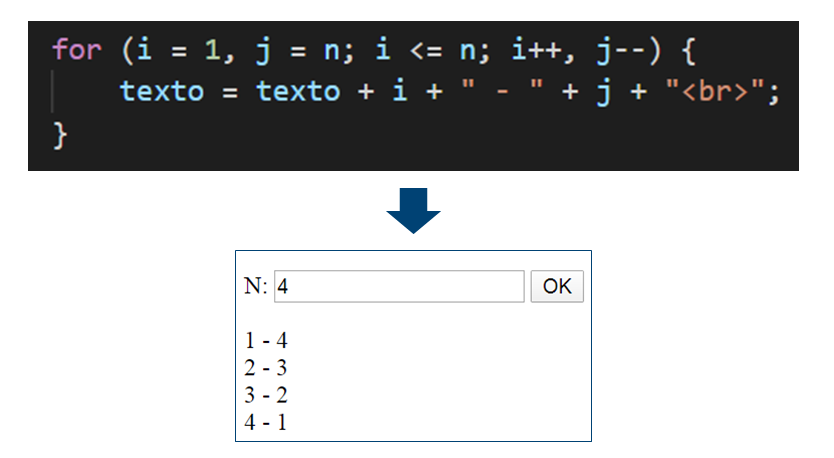
Existem algumas curiosidades sobre o comando for. Primeiramente, a inicialização nem sempre é obrigatória. Na Figura 03 vemos um exemplo de laço que utiliza uma variável i que foi declarada e inicializada antes do laço. Portanto, a inicialização do laço não precisa ser feita.

**Figura 03** - Comando for sem inicialização



Além disso, a inicialização pode ser utilizada para atribuir valores iniciais a variáveis, como vemos no exemplo onde inicializamos i com 1 e j com n, ou seja, o valor dado como entrada. Para isso, basta separar os comandos utilizando uma vírgula. A vírgula também pode ser utilizada para separar os comandos de atualização. Note, no exemplo, que incrementamos o i em 1 e decrementamos o j em 1 usando j--.

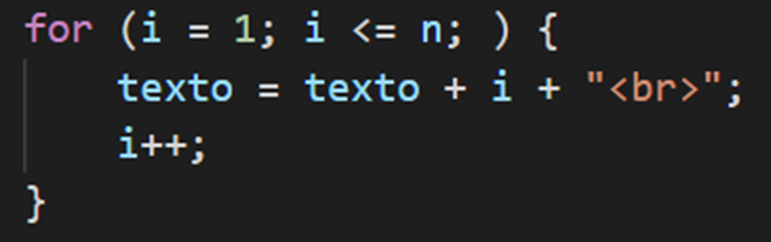
**Figura 04** - Comando for com várias inicializações e atualizações



A seção de atualização também não é obrigatória, pois a atualização pode ser feita no bloco de comando do próprio laço. No exemplo, podemos remover a seção de atualização da declaração do laço e colocar o comando de atualização no final do corpo do laço.

A página HTML resultante exibirá o resultado da mesma forma.

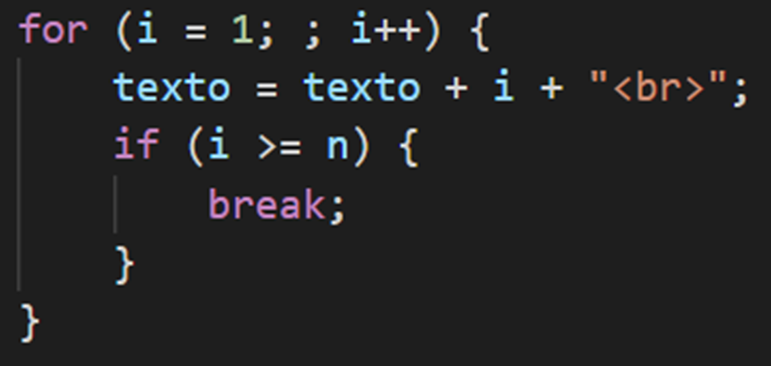
**Figura 05** - Comando for sem atualização



Por fim, a declaração da condição do laço também é opcional. No entanto, se você omitir esta seção, deve utilizar o comando break dentro do laço a fim de parar a sua execução. Caso contrário, o laço nunca terminará. Chamamos isto de Laço Infinito, o que fará com que a página de seu navegador trave. Em geral, laços infinitos são erros no programa, pois raras são as situações em que precisamos utilizá-los.

O comando break interrompe o laço e continua a execução do código após o laço (se houver). Por outro lado, o comando continue interrompe uma iteração (no laço), se uma condição especificada ocorrer, e continua com a próxima iteração no laço. Na Figura 06, podemos ver como utilizar o break no nosso exemplo ao removermos a condição do loop a fim de evitar um laço infinito. Note que, no final do corpo do laço, paramos a sua execução quando o valor de i fica maior ou igual ao valor de n.

**Figura 06** - Comando for sem condição de parada



Apesar de termos diferentes possibilidades de variação na declaração do laço for, a forma mais recomendada é inicializar a variável de controle e incrementá-la (ou decrementá-la) dentro da própria instrução for.

Isso porque geralmente a variável i só é útil para o conteúdo dentro do laço for, por isso é interessante ela ter escopo local ao bloco.

Veja agora um passo a passo do que se deve fazer ao usar o laço for:

1. Primeiro, cria-se uma variável de controle, que poderá servir como uma espécie de contador, ou seja, para contar o número de vezes que o bloco de comandos será executado.
2. Depois, a variável recebe um valor inicial (chamamos esse processo de inicialização). No caso de um contador, geralmente, esse valor inicial é 0 (zero) ou 1 (um).
3. No terceiro passo uma condição determinada (por exemplo, se o valor final do contador foi atingido) é testada e, se for verdadeira, a lista de comandos que pertence ao laço é executada. É através dessa condição que é determinado o número de repetições do laço.
4. No quarto passo a variável de controle recebe a atualização.
5. Por fim, o programa voltará ao terceiro passo do laço para testar novamente a condição. Se a condição for falsa (ou seja, se o valor de término tiver sido atingido), o laço é interrompido. Caso contrário, o laço continuará com os passos 3 a 5.

Ao usar o laço for, você deve ter o cuidado de inicializar corretamente as variáveis utilizadas.

**for/of**

JavaScript também possui duas outras maneiras de utilizar um laço for. A primeira, chamada de for/in percorre as propriedades de um objeto, mas não veremos essa utilização do laço for neste momento.

A outra maneira é o for/of, que permite percorrer estruturas de dados sobre os quais podemos iterar, ou seja, podemos acessar cada um de seus elementos. Alguns exemplos dessas estruturas são strings, arrays, mapeamentos e listas.

Vejamos um exemplo de uso do laço for/of.

**Código 1** - Laço\_For\_Of.html

**Exemplo:**

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>* *Laço\_For\_Of </title>*

*</head>*

*<body>*

*<h1>Laço For/Of</h1>*

*Texto: <input type="texto" id="N1" value="">*

*<button onclick="quebraLinhas('N1')">OK</button>*

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

**script.js**

*function quebraLinhas(entrada1) {*

*var str = document.getElementById(entrada1).value;*

*var texto = "Letras de "+str+":<br>";*

*for (c of str) {*

*texto = texto + c + "<br>";*

*}*

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

*}*

Nesse exemplo a entrada será um pouco modificada, ao invés de ser um campo numérico, será colocado um campo tipo texto, e ao clicar no botão OK, chamará a função quebraLinhas, que está no arquivo JavaScript. Essa função pega o valor do campo texto (linha 2 do Código 1) e declara uma variável chamada texto (linha 4), que vai ser o valor que atualizará o campo de "resultado" na página HTML.

**Código 2** - Laço For Of.js

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> Laço\_For\_Of </title>*

*</head>*

*<body>*

*<h1>Laço For/Of</h1>*

*Texto: <input type="texto" id="N1" value="">*

*<button onclick="quebraLinhas('N1')">OK</button>*

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*function quebraLinhas(entrada1) {*

*var str = document.getElementById(entrada1).value;*

*var texto = "Letras de "+str+":<br>";*

*for (c of str) {*

*texto = texto + c + "<br>";*

*}*

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

*}*

No final dessa função, no elemento "resultado", é atribuído o valor do texto, ou seja, é onde será trabalhado o laço for/of.

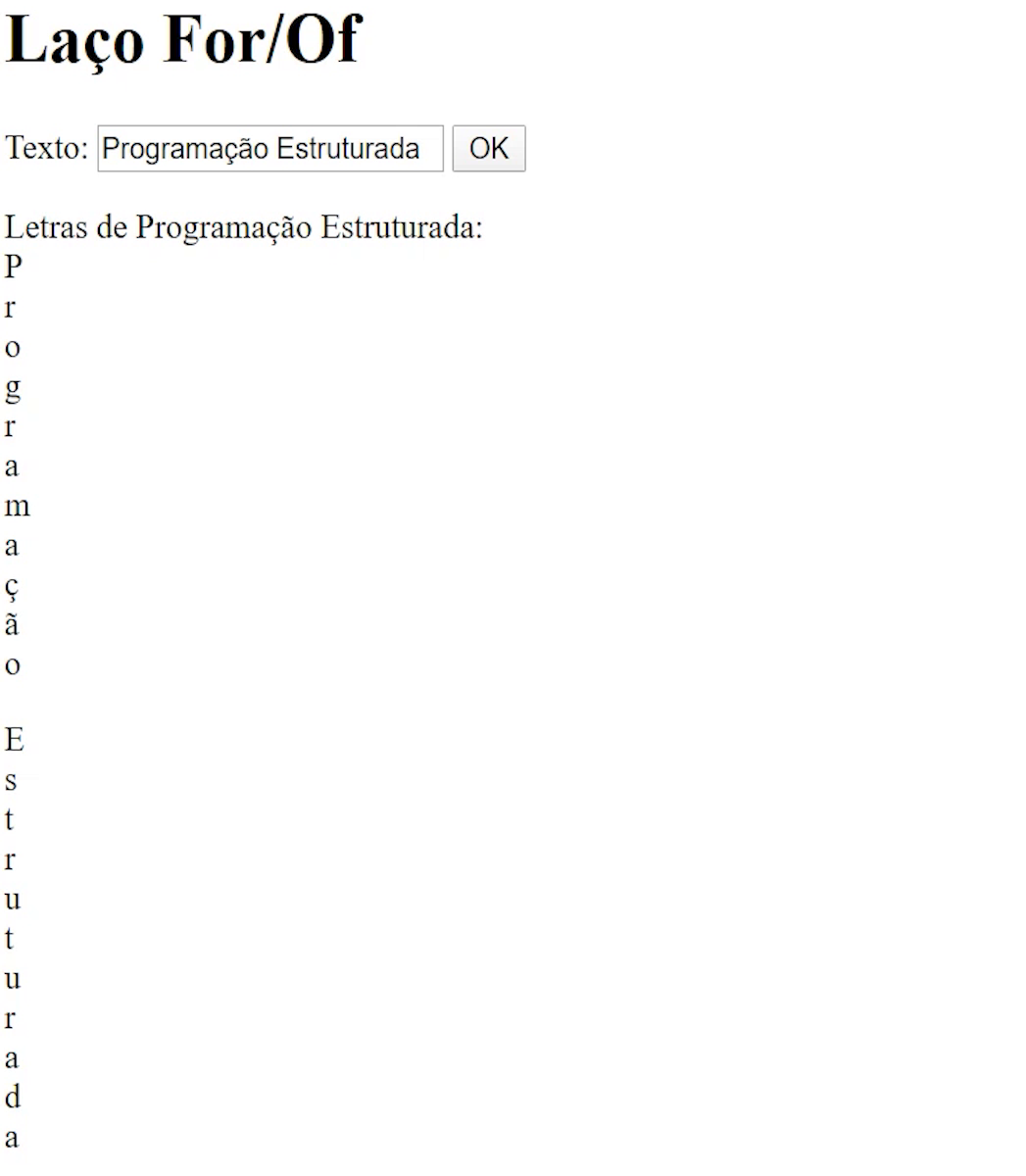
A variável texto começa com "Letras de", informa o texto que recebeu e quebra linha, (linha 4). E, na linha 5, usa-se um laço for diferente, ele vai fazer um for com a variável c, em seguida vem o of e o str, como apresentado na linha 5.

Para cada caractere da string, ele vai executar esse trecho de comando entre chaves que aparece na linha 6. O que está sendo feito nesse exemplo é: concatenar ao texto aquele caractere mais a tag <br> HTML, ou seja, uma quebra de linha. Em outras palavras, está sendo escrito na página HTML, uma letra da string em cada linha.

Então, para cada caractere da string str esse laço for/of vai atribuir esse caractere à variável c, e executa a linha de comando, adiciona a variável e quebra uma linha.

Assim, ao voltar na página HTML e colocar: "Programação Estruturada" e clicar em OK, aparecerá o texto "Letras de Programação Estruturada" e logo abaixo aparecerá um caractere por linha, como mostra a figura 4.

**Figura 04** - Exemplo de Uso do Laço for/of



**while**

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 na Figura 01, 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 01** - 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 [Contagem While.html](https://materiais.imd.ufrn.br/materialV2/assets/downloads/programacao-estruturada/10_6-Contagem-While.zip) para testar o uso do laço while no nosso exemplo.

**Código 1 -** Contagem While.html

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title>While</title>*

*</head>*

*<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>*

***script.js***

*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** - Sorteio While.html

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Sorteio</h1>*

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

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*function sorteio() {*

*var n = Math.floor(Math.random() \* 10) + 1;*

*var num;*

*var contagem = 1;*

*num = prompt("Escolha um número entre 1 e 10?");*

*while (num != n) {*

*num = prompt("Escolha um número entre 1 e 10?");*

*contagem++;*

*}*

*document.getElementById("resultado").innerHTML =*

*"O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";*

*}*

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** - Sorteio While.js

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Sorteio</h1>*

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

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*function sorteio() {*

*var n = Math.floor(Math.random() \* 10) + 1;*

*var num;*

*var contagem = 1;*

*num = prompt("Escolha um número entre 1 e 10?");*

*while (num != n) {*

*num = prompt("Escolha um número entre 1 e 10?");*

*contagem++;*

*}*

*document.getElementById("resultado").innerHTML =*

*"O número era " + n + ". Você acertou em "+contagem+" tentativas!!!";*

*}*

Essa função inicialmente atribui à 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.

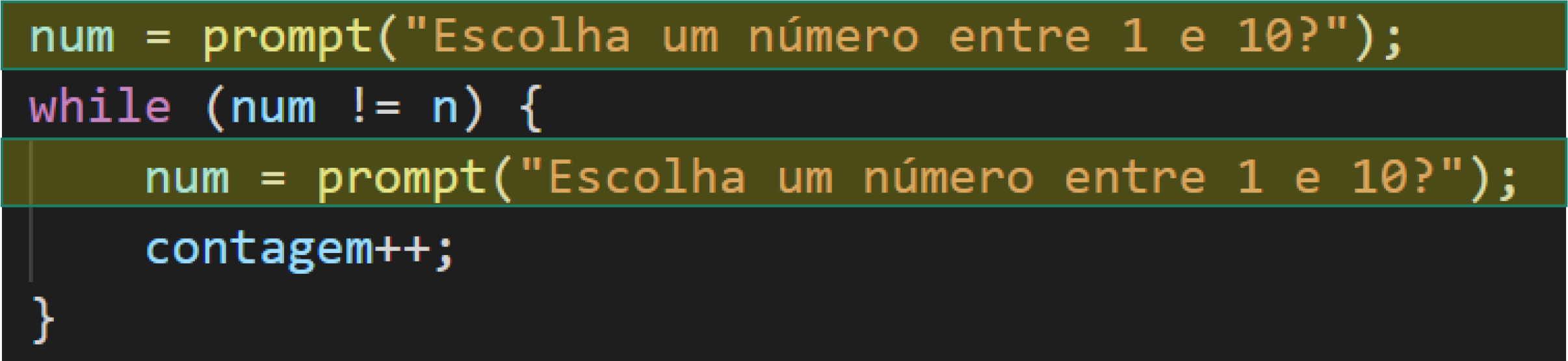
Se não for é porque acertou. Caso contrário, se for diferente vai perguntar novamente, e vai incrementar a contagem. E no final informa: "O número era" e o n que sorteei e "Você acertou em", contagem de "tentativas". Assim, enquanto o número que o usuário inserir no Prompt não for aquele número sorteado,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

No nosso exemplo existe uma repetição do trecho de código marcado em amarelo na imagem. 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 02** - Repetindo trechos de código



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** - Sorteio\_Do-While.html

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Sorteio</h1>*

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

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*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** - Sorteio Do-While.js

*html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Sorteio</h1>*

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

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*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!!!";*

*}*

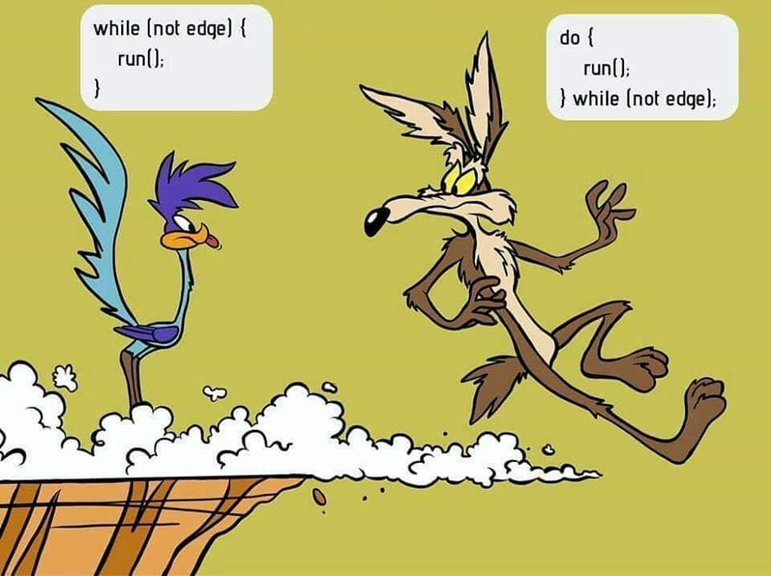
Essa contagem é igual a zero porque agora vai, necessariamente, perguntar o número dentro do laço do/while, como consta nas linhas 6 a 9. Na linha 7, pergunta e na 8, incrementa a contagem. Então, inicialmente pergunta um número ao usuário e já incrementa a contagem, passando a contagem para 1. E isso será feito enquanto o número que o usuário colocou for diferente de n. Perceba que é um pouquinho diferente, pois há a garantia de que vai executar pelo menos uma vez essa solicitação e incremento. 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.

**do/while**

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.

**Figura 03** - Comparando while com do/while



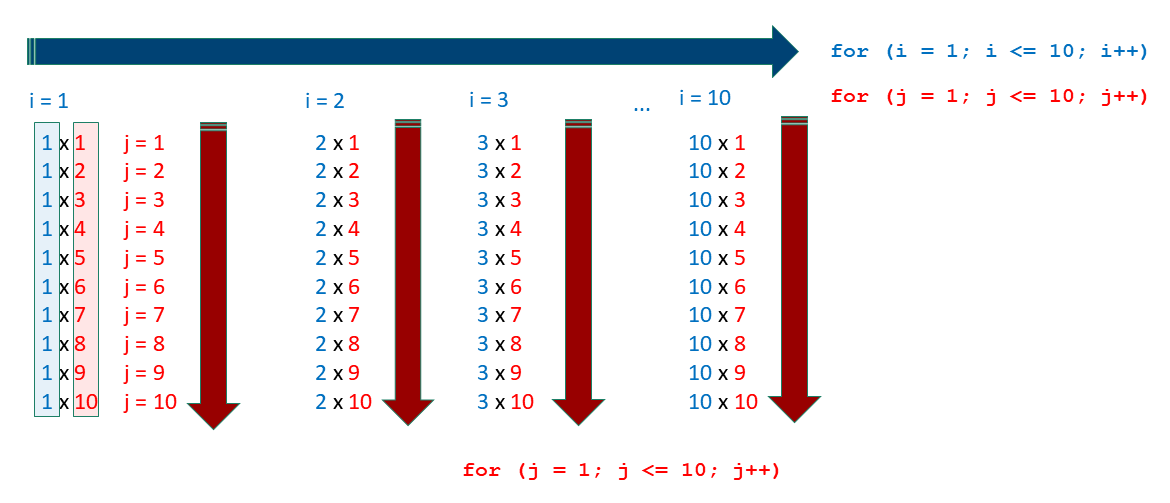
Note a diferença entre os comandos do papa-léguas e do coiote. O papa-léguas executou o while. Por esse motivo, ele primeiro verificou se estava na beira do precipício antes de correr, desse modo, permaneceu parado. Por outro lado, o coiote executou o do/while. Por isso, ele inicialmente correu para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

**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 na Figura 04. 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 04** - Laços Encadeados



Olhe novamente para a imagem! 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** - abuada.html

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Cronômetro</h1>*

*<button onclick="tabuada()">OK</button>*

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*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 idéia 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

*<html>*

*<head>*

*<meta charset="UTF-8" />*

*<title> While </title>*

*</head>*

*<body>*

*<h1>Cronômetro</h1>*

*<button onclick="tabuada()">OK</button>*

*<p id="resultado"></p>*

*<script src='script.js'></script>*

*</body>*

*</html>*

***script.js***

*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;*

*}*

Então, para cada tabuada foi inserida uma linha especial ( linha 6) para marcar, apenas com hífens e a quebra de linha. Na linha 7, está escrito "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. Em seguida, concatenamos 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.

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.

**Saiba mais:**

[*https://www.youtube.com/watch?v=WP0Hl9SC1oI*](https://www.youtube.com/watch?v=WP0Hl9SC1oI)

[*https://www.youtube.com/watch?v=-11qkJczuqo*](https://www.youtube.com/watch?v=-11qkJczuqo)

[*https://www.youtube.com/watch?v=dPl3OTOOJwM*](https://www.youtube.com/watch?v=dPl3OTOOJwM)

[*https://www.youtube.com/watch?v=HFG\_p4K2MAc*](https://www.youtube.com/watch?v=HFG_p4K2MAc)

[*https://www.youtube.com/watch?v=y8otiWoVfRI*](https://www.youtube.com/watch?v=y8otiWoVfRI)

[*https://www.youtube.com/watch?v=y8otiWoVfRI*](https://www.youtube.com/watch?v=y8otiWoVfRI)

[*https://www.w3schools.com/js/js\_if\_else.asp*](https://www.w3schools.com/js/js_if_else.asp)

[*https://www.w3schools.com/js/js\_switch.asp*](https://www.w3schools.com/js/js_switch.asp)

[*https://www.w3schools.com/js/js\_loop\_for.asp*](https://www.w3schools.com/js/js_loop_for.asp)

[*https://www.w3schools.com/js/js\_loop\_forof.asp*](https://www.w3schools.com/js/js_loop_forof.asp)

[*https://www.w3schools.com/js/js\_loop\_while.asp*](https://www.w3schools.com/js/js_loop_while.asp)