Programação Estruturada

Aula 09 - Comandos de Seleção

Videoaula 02: Valores Booleanos e Condições

Videoaula 02: Valores Booleanos e Condições

Normalmente, em JavaScript, os booleanos são os valores primitivos `true` e `false`. No entanto, eles também podem ser definidos como objetos com a palavra-chave `new`. Não recomendo que você utilize esse recurso, pois isso diminui a velocidade de execução e complica o código, na maioria das vezes, de maneira desnecessária. Além disso, a comparação entre objetos têm um comportamento particular que não veremos nesta disciplina. É por esse motivo que, nesta disciplina, focaremos apenas no tratamento de valores booleanos como os valores primitivos `true` e `false`.

Você pode acessar o link do QR code no slide (Figura 1) para saber mais sobre objetos booleanos em JavaScript.

Figura 1 - Valores Booleanos

- var a = true;
- var b = new Boolean(false);

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:

- e_1 == e_2, que retorna `true` apenas se o valor da expressão e_1 for igual ao valor da expressão e_2

- e_1 === e_2, também chamada de igualdade estrita, que retorna `true` apenas se o valor e o tipo da expressão e_1 e e_2 forem iguais

- e_1 != e_2, que retorna `true` apenas se o valor da expressão e_1 for diferente do valor da expressão e_2

- e_1 !== e_2, também chamada de desigualdade estrita, que retorna `true` apenas se o valor da expressão e_1 for diferente do valor da expressão e_2 ou se o tipo da expressão e_1 for diferente do tipo da expressão e_2

- e_1 > e_2, que retorna `true` apenas se o valor da expressão e_1 for maior que valor da expressão e_2

- e_1 < e_2, que retorna `true` apenas se o valor da expressão e_1 for menor que valor da expressão e_2

- e_1 >= e_2, que retorna `true` apenas se o valor da expressão e_1 for maior ou igual ao valor da expressão e_2

- e_1 <= e_2, que retorna `true` apenas se o valor da expressão e_1 for menor ou igual ao valor da expressão e_2

Além disso, podemos combinar expressões que usam esses operadores básicos por meio dos operadores lógicos. São eles `e`, `ou` e `não`. No slide (Figura 2), podemos ver uma tabela que chamamos de tabela-verdade, a qual apresenta o resultado desses operadores. Dadas as duas condições c_1 e c_2:

- Usamos c_1 `&&` c_2 para obter o valor de c_1 e c_2

- Usamos c_1 `||` c_2 para obter o valor de c_1 ou c_2

- Usamos `!`c_1 para obter o valor de não c_1

No e-lógico temos que c_1 `&&` c_2 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 c_1 `||` c_2 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 2 - Combinando Condições

c_1	c_2	c_1 && c_2	c_1 || c_2	!c_1
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

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 (ver Código 1), 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`".

Código 1 - 09_7 Valores Verdadeiros e Falsos.html

```html
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 09</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Valores Verdadeiros e Falsos</h1>

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

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

```

```javascript
// 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";
```

```
25 }
26
27 // Procurando elementos na página
28 var nome_elemento = "resultado";
29 if (document.getElementById(nome_elemento)) {
30    document.getElementById(nome_elemento).innerHTML = texto;
31 } else {
32    alert("Não encontrei o elemento "+ nome_elemento)
33 }
34
35 // Note que o código abaixo não faz a mesma coisa
36 // pois document.getElementById(nome_elemento) retorna um objeto
37 /*
38 if (document.getElementById(nome_elemento) == true) {
39    document.getElementById(nome_elemento).innerHTML = texto;
40 } else {
41    alert("Não encontrei o elemento "+ nome_elemento)
42 }
43 */
44
```

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 3 - Exibindo valores booleanos

Agora, nesse exemplo, você testará o código utilizando valores de outros tipos. Comece atribuindo o valor 0 à variável `b` localizada na linha 3 e veja que, ao recarregar a página HTML, será exibido o texto: "0 transformou em false". Dessa forma, o zero será

`false` e qualquer outro valor numérico será `true`. Você pode confirmar fazendo um novo teste. Veja que, ao atribuir 1 à variável `b`, será exibido o texto "1 transformou em true", como exposto na Figura 4. Certo?

Figura 4 - Convertendo números em valores booleanos

E se você colocar a *string* vazia, o que será exibido? A *string* vazia é transformada em `true` ou `false`? Faça o teste e veja que ao recarregar a página a *string* vazia não é exibida, afinal de contas, é uma *string* vazia, mas ela é transformada em `false`. Dessa forma, o valor booleano da *string* vazia é `false`. Veja a Figura 5.

Figura 5 - Convertendo strings em valores booleanos

E se você pegar 10 e dividir pelo texto `'score'`? O JavaScript vai tentar transformar o texto `'score'` em um número e não vai conseguir e o que teremos é o resultado "NaN transformou em false", que corresponde ao booleano `false`.

Figura 6 - Convertendo `NaN` em valores booleanos

E se você simplesmente declarar a variável `b` terá um valor `undefined`. E o resultado na página HTML será "undefined transforma em false" como exposto na Figura 7.

Figura 7 - Convertendo `undefined` em valores booleanos

Veja alguns outros valores verdadeiros, continue na linha 3 e atribua `true` à variável `b`, obviamente `true` é valor `true`. O valor numérico 1, como já visto, também é o valor `true`. Se você colocar 1 ou qualquer valor numérico, com exceção do zero, como, por exemplo 100, também será `true`.

Na divisão de 10 por 5, como sabemos, dará 2, ou seja, um valor numérico, então ele também retorna `true`. E quanto ao texto `'carrot'`, qual será o valor booleano? Fazendo o teste e carregando a página HTML, você verá que o resultado será "carrot transformou em true", porque `'carrot'` é uma *string* que tem um valor dentro dela. Veja a Figura 8.

Figura 8 - Convertendo *strings* em valores booleanos

O que acontece se você tiver o texto `'true'`? O texto `'true'` tem um valor dentro dele que também será transformado em `true`. E se a variável `b` receber o texto `'0'`? Sabe-se que o número 0 é convertido em `false`, mas o texto `'0'` é transformado em `true` porque ele é simplesmente um texto que tem um valor lá dentro. Veja Figura 9.

Figura 9 - Convertendo *strings* em valores booleanos

Então não interessa o que está escrito dentro do texto, desde que tenha algum conteúdo ele será transformado em `true`. Outro exemplo é a variável `b` sendo `'false'`, mas é o texto `'false'` e não o booleano `false`. Novamente, temos um texto com conteúdo dentro dele. Dessa forma, o texto `'false'` tem valor booleano `true`.

Por fim, o que acontece ao atribuir o valor `null` para a variável `b`? Tem-se como resultado "null transforma em false", como exposto na Figura 10, porque o `null` corresponde a um objeto que não tem valor dentro dele, não estamos na orientação a objetos, mas o `null` é transformado em `false`.

Figura 10 - Convertendo `null` em valores booleanos

Na segunda parte desse exemplo, veja como é que você pode verificar se um elemento está na página ou não. Para isso, você precisa criar uma variável chamada `nome_elemento` e dizer que ela é `"resultado"`. E, antes de escrever na página, verifique se o `document.getElementById` passando esse nome é `true`.

Se ele encontrar esse elemento, retorna esse objeto diferente de `null` e esse valor vai ser `true`. Só então você transforma o `innerHTML` desse objeto, passando o texto desejado conforme a Figura 11.

Figura 11 - 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)
}
```

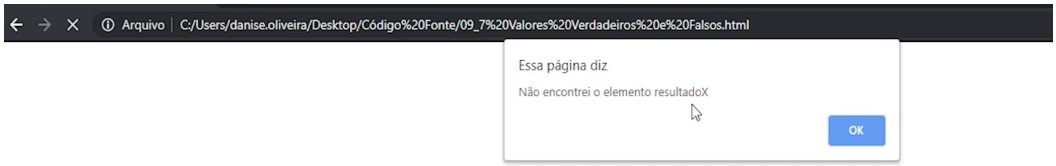
O valor do texto foi apagado, mas foi criada uma variável `texto` contendo `"Deu certo"`. Caso a condição do `if` seja falsa, entraremos no `else` e, neste caso, será exibido um alerta, dizendo "Não encontrei o elemento", passando o nome desse elemento.

Então vamos voltar à página e carregá-la, veja que o texto é exibido porque já sabemos, de antemão, que esse componente resultado existe, e é aquele parágrafo que temos no HTML. Veja a Figura 12.

Figura 12 - Identificando elementos na página HTML

Suponha que eu coloque `"resultadox"` ele não existe na página HTML, ou seja, esse elemento não existe. Se voltarmos à página HTML e tentarmos carregá-la, será exibida a mensagem "Não encontrei o elemento resultadox", que foi o alerta dado. Veja a Figura 13.

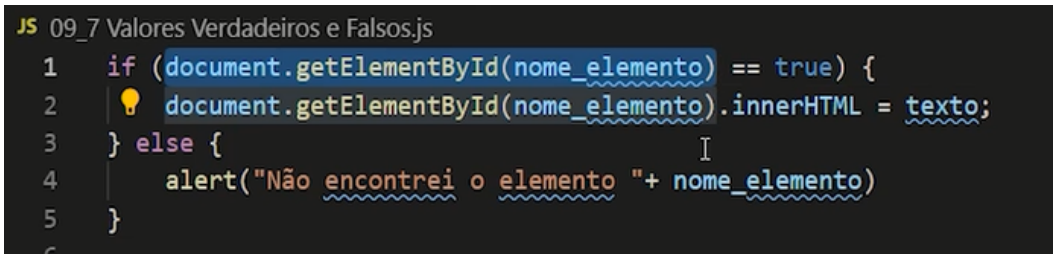
Figura 13 - Identificando ausência de elementos na página HTML



Esse tipo de comparação pode ser usado para verificar se determinados elementos HTML existem antes de tentar fazer alguma coisa com eles. É muito importante, também, saber que esse trecho de código (Figura 14) não tem o mesmo

comportamento daquele que acabamos de ver. Ele tentará pegar o elemento pelo `Id` e comparar esse valor com o valor `true` usando o `==`.

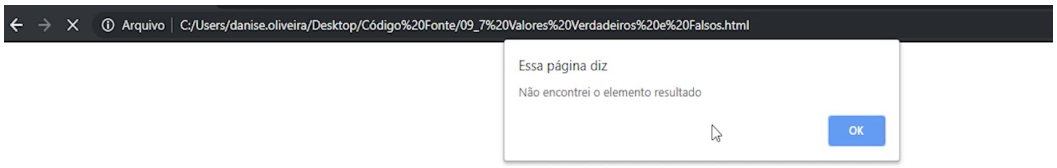
Figura 14 - Erro ao Comparar Usando `==`



Ao invés de simplesmente verificar o valor booleano do objeto, que tentamos recuperar na página HTML, estamos tentando compará-lo com o valor `true`. Se isso acontecer, você coloca o texto na página. Então, o texto continua existindo, caso contrário, daremos o alerta, e veremos o que acontece. Lembre-se de deixar, também, a variável `nome_elemento`, que é a variável `resultado`. Temos o texto, o nome do elemento é `"resultado"`, que é um elemento que existe na página HTML, e tentaremos fazer essa comparação.

Se voltarmos à página HTML e recarregá-la, veremos que o alerta foi exibido (Figura 15). Então essa comparação do `==` com `true` está errada. Não é o esperado porque estamos comparando, na prática, o objeto que recuperamos com o valor `true`.

Figura 15 - Resultado do Erro ao Comparar Usando `==`



E esses valores não são iguais. Então, se quisermos averiguar se um elemento existe na página, basta simplesmente tentar pegar o elemento, colocá-lo na condição do `if`.