Programação Estruturada

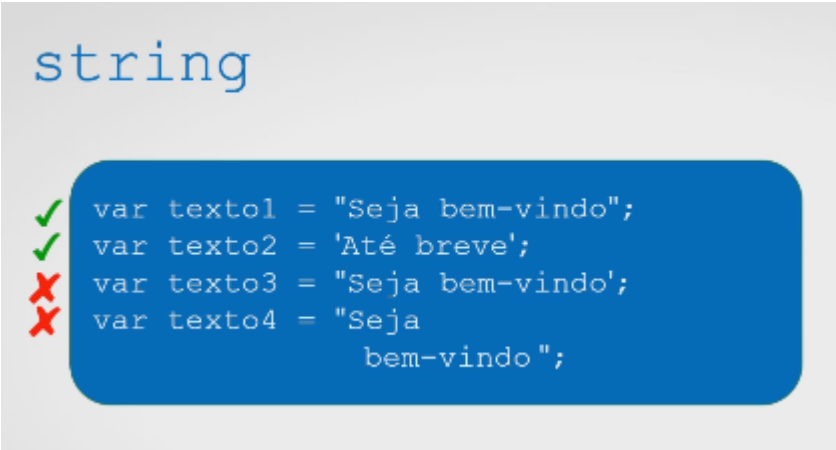
Aula 02 - Tipos de Dados

Videoaula 03 - `string`, `boolean`, `null` e `undefined`

Videoaula 03 - `string`, `boolean`, `null` e `undefined`

Vou começar esta videoaula apresentando um tipo primitivo, que representa um texto e cujo uso é bastante comum: o tipo `string`. Valores do tipo `string` podem ser inseridos no programa usando aspas duplas ou simples, como nos textos "`seja bem-vindo!`" e '`até breve!`'. É importante ressaltar duas restrições na definição de strings (Figura 1): (1) o mesmo tipo de aspas deve ser usado para iniciar e fechar a string. Portanto, a declaração da variável `texto3` está incorreta; (2) a string deve ser definida em UMA ÚNICA linha. Portanto, a declaração da variável `texto4` também está incorreta.

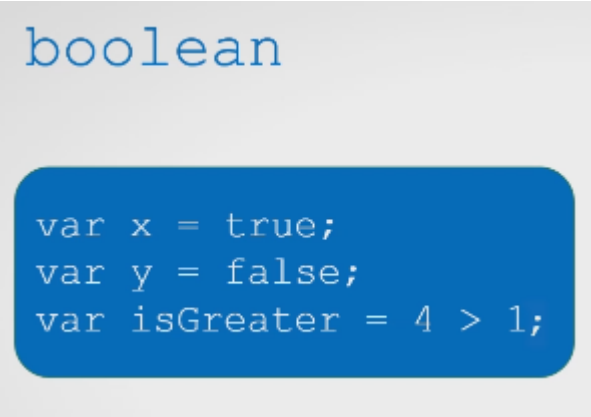
Figura 1 - Tipo de dados JavaScript `string`



Nas aulas seguintes, você verá outras maneiras de digitar valores do tipo `string` e operadores que podem ser usados com esse tipo.

O próximo tipo primitivo (Figura 2), `boolean`, é o tipo lógico de JavaScript. Ele tem apenas dois valores: `true` e `false`, o verdadeiro e o falso.

Figura 2 - Tipo de dados JavaScript `boolean`



Esses valores lógicos podem ser o resultado de uma comparação e são usados nas condições de comandos de seleção como o se-então, e de controle de loops, que veremos ao longo desta disciplina. Além disso, também veremos a aplicação de operadores lógicos como o "e" e o "ou" a valores booleanos.

Vamos ver alguns exemplos?

Nesse exemplo (Código 1), temos as variáveis `x` e `y`, só que agora elas recebem valores booleanos, então `x` recebe o valor `true` e o `y` recebe o valor `false` (linhas 16 e 17).

Código 1 - 02_5 Valores Booleanos.html

```
1   <html>
2
3     <head>
4       <meta charset="UTF-8" />
5       <title>Programação Estruturada - Aula 02</title>
6     </head>
7
8     <body>
9       <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
10
11      <h1>Valores Booleanos</h1>
12
13      <p id="texto"></p>
14
15      <script>
16        var x = true;
17        var y = false;
```

```
18              var isGreater = 4 > 1;
19
20              document.getElementById("texto").innerHTML = isGreater;
21          </script>
22
23      </body>
24
25  </html>
26
```

Então veja que, se colocarmos aqui no campo texto o valor `x` (Linha 20) (Figura 3), por exemplo, temos como resultado o `true` (Figura 4).

Figura 3 - Mudança do valor do campo texto

```
1   document.getElementById("texto").innerHTML = x;
2
```

Figura 4 - Valor exibido do campo texto

Se colocarmos o valor `y` (Linha 20) (Figura 5), que tem o valor `false`, o valor exibido vai ser `false` (Figura 6).

Figura 5 - Mudança do valor do campo texto

```
1        document.getElementById("texto").innerHTML = y;
2
```

Figura 6 - Valor exibido do campo texto

Podemos também fazer operações booleanas sobre esses operadores. Então, `x` e `y`, por exemplo, `true` e `false` (Figura 7), como sabemos é `false` (Figura 8), então `false` permaneceu.

Figura 7 - Usando o "e" lógico

```
1        document.getElementById("texto").innerHTML = x && y;
2
```

Figura 8 - Exibindo o "e" lógico

Se nós colocarmos o valor "ou" (||) (linha 20) (Figura 9), por exemplo, sabemos que `true` ou `false` vai dar `true`, então nosso valor se torna `true` (Figura 10).

Figura 9 - Usando o "ou" lógico

```
1         document.getElementById("texto").innerHTML = x || y;
2
```

Figura 10 - Exibindo o "ou" lógico

Mas valores booleanos também podem ser resultados de expressões de comparação, por exemplo, então vemos aqui uma outra variável `isGreater` (Linha 18), que recebe o valor da expressão `4 > 1`.

Ora, 4 é maior que 1, então o resultado dessa expressão é `true`. Se eu colocar aqui como saída o valor de `isGreater` (Linha 20) (Figura 11) como esperado, teremos, de fato, o valor `true` como saída (Figura 12).

Figura 11 - Mudança do valor do campo texto "`isGreater`"

```
1        document.getElementById("texto").innerHTML = isGreater;
2
```

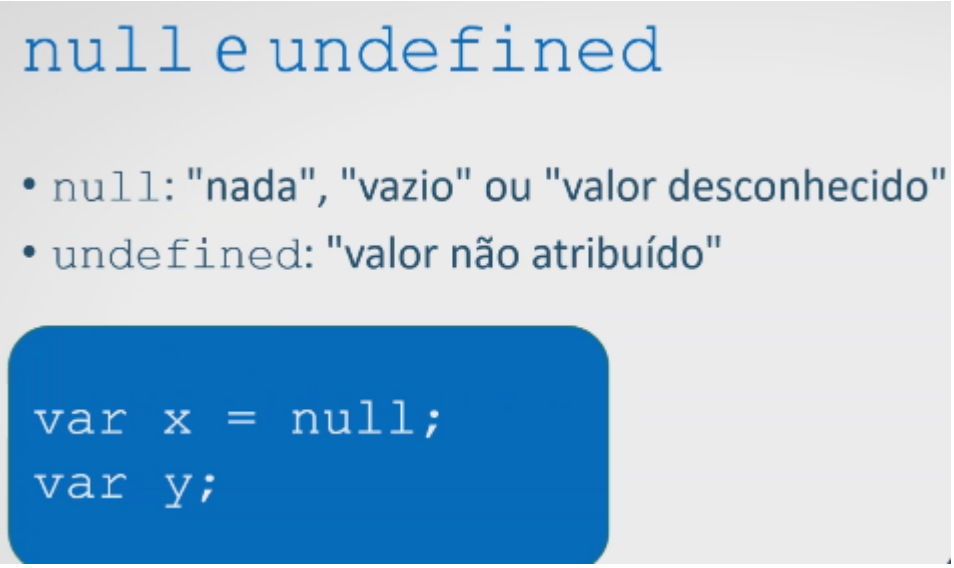
Figura 12 - Valor exibido do campo texto "`isGreater`"

Então note que expressões booleanas retornam valores booleanos, e esses valores podem ser atribuídos às nossas variáveis.

Ao longo desta disciplina, você verá mais detalhes sobre o uso dos valores booleanos.

Os últimos dois tipos básicos são o `null` e o `undefined`, que possuem apenas um valor cada, respectivamente, `null` e `undefined` (Figura 13). Em JavaScript, `null` não é uma "referência a um objeto não existente" ou um "ponteiro nulo", como em algumas outras linguagens. É apenas um valor especial que representa "nada", "vazio" ou "valor desconhecido". Por exemplo, no slide, declaramos a variável `x` com um valor `null`, ou seja, desconhecido ou vazio por algum motivo.

Figura 13 - Valores `null` e `undefined`



Por outro lado, o significado do `undefined` é "valor não atribuído". Se uma variável é declarada, mas não atribuída, seu valor é indefinido. Por exemplo, no slide, a variável `y` tem valor `undefined`.

Na verdade, `undefined` e `null` têm valores iguais, mas têm tipos diferentes. Antes de você ver um exemplo, precisa aprender sobre a função `typeof`, que retorna uma `string` com o nome do tipo do seu parâmetro.

A função `typeof` (Código 2) recebe um parâmetro, uma expressão, e retorna a `string` com o nome do tipo daquela expressão. Então, nesse exemplo, criamos 10 variáveis, `t1` até `t10`, e vamos atribuir o valor dessas variáveis ao campo texto de saída "`texto1`".

Código 2 - 02_6 TypeOf.html

```
1   <html>
2
3     <head>
4       <meta charset="UTF-8" />
5       <title>Programação Estruturada - Aula 02</title>
6     </head>
7
8     <body>
9       <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
```

```
10
11              <h1>Testando Script</h1>
12
13              <p id="texto1"></p>
14
15              <script>
16                 function f() {}
17
18                 var t1 = typeof 0;              // "number"
19                 var t2 = typeof true;           // "boolean"
20                 var t3 = typeof "legal";        // "string"
21                 var t4 = typeof null;           // "object"
22                 var t5 = typeof undefined;      // "undefined"
23                 var t6 = typeof x;              // "undefined"
24                 var t7 = typeof {x:10};         // "object"
25                 var t8 = typeof [1,2,3,4];      // "object"
26                 var t9 = typeof f;              // "function"
27                 var t10 = typeof Symbol("id");  // "symbol"
28
29                 document.getElementById("texto1").innerHTML = t1;
30
31              </script>
32
33           </body>
34
35        </html>
36
```

Então vemos aqui que a variável `t1` (Linha 29), ela retorna o `typeof 0` (Linha 18). Ora, o `typeof` 0, é um número. Então, nesse caso, a nossa saída vai ser o texto `number`, que é o nome do tipo para números em JavaScript (Figura 14).

Figura 14 - Nome do tipo em JavaScript `number`

A variável `t2` é o `typeof true` (Linha 19) e, como você viu, `true` é um valor booleano e o nome desse tipo é `boolean` (Figura 15).

Figura 15 - Nome do tipo em JavaScript `boolean`

A variável `t3` recebe o valor do `typeof` da `string` `"legal"`, é a saída que temos se colocarmos o valor dessa variável `t3` ao nosso texto (Linha 29), então recebemos aqui a saída `string` (Figura 16).

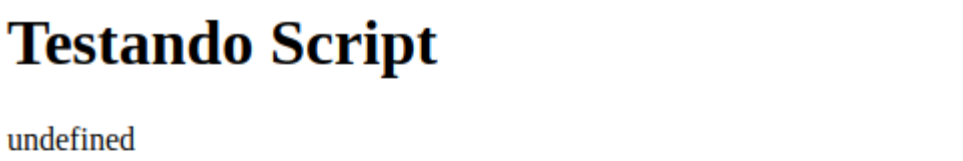
Figura 16 - Nome do tipo em JavaScript `string`

A variável `t4` recebe o `typeof null`, pra gente o valor `null` é um `object`, vou falar isso posteriormente em outras aulas, mas vemos aqui que se atribuirmos o valor `t4` ao nosso texto (Linha 29), temos aqui a saída `object` (Figura 17).

Figura 17 - Nome do tipo em JavaScript `object`

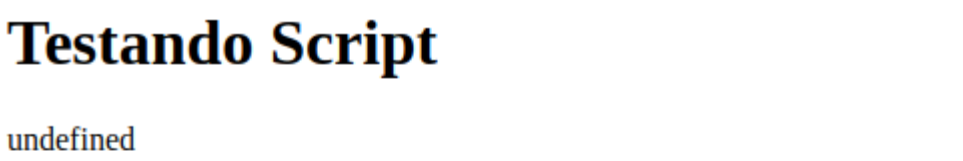
A variável `t5` recebe o `typeof undefined`, tá? Vemos que a saída vai ser o texto `undefined` (Figura 18) se nós colocarmos `t5` no final da linha 29 do código. Então, note que aqui fica claro a diferença entre o valor `null` e o valor `undefined`. Para nós, `null` é do tipo `object` e `undefined` é do tipo `undefined`.

Figura 18 - Nome do tipo em JavaScript `undefined`



A variável `t6` recebe o `typeof x`, note que nesse JavaScript não declaramos a variável `x`. Portanto, se perguntamos qual é o tipo da variável `x`, a resposta deveria ser `undefined`, e é exatamente o que nós temos aqui com essa saída (Figura 19), quando atribuímos o valor da variável `t6` ao nosso texto (Linha 29).

Figura 19 - Nome do tipo em JavaScript `undefined`



A variável `t7` recebe o `typeof` e a expressão `{x:10}`, não iremos entrar em detalhes, mas o que temos é uma declaração de um objeto que ocorre entre chaves, onde o atributo desse objeto `x` tem o valor 10 (Linha 24). Então, se usarmos o `t7` para valorar o nosso texto (linha 29), veremos como saída o texto `object` (Figura 20).

Figura 20 - Nome do tipo em JavaScript `object`

Aqui, também na variável `t8`, temos a declaração de um array, vou falar sobre arrays em aulas posteriores ainda nesta disciplina. Mas, temos o temos o `array [1,2,3,4]` (Linha 25), e array em JavaScript na função `typeof`, ela retorna também o tipo `object`. Então, se nós colocarmos aqui o `t8` (Linha 29), veremos que a saída continua sendo `object` (Figura 21).

Figura 21 - Nome do tipo em JavaScript `object`

O `t9` (Linha 26) pede o retorno do `typeof f`, observe na linha 16 que estamos declarando uma função chamada `f`, que não tem nenhum argumento, e ela não faz nada; ela simplesmente é uma função vazia, mas, mesmo assim, ela é uma função.

Então, se perguntarmos qual o tipo de `f`, vamos receber como resposta o tipo `function`, então vamos aqui colocar o `t9` como valor do nosso texto (Linha 29), e veremos que agora a saída vira `function` (Figura 22).

Figura 22 - Nome do tipo em JavaScript `function`

Por fim, temos o `typeof t10` (Linha 27), estamos usando a declaração de um símbolo. Não vamos entrar em detalhes durante essa disciplina sobre esse tipo `Symbol`, mas, apenas para ilustrar, se você está usando essa expressão e perguntar qual o tipo dessa expressão, de fato, nós vamos ter como saída da função `typeof` (Linha 29), o tipo `symbol` (Figura 23).

Figura 23 - Nome do tipo em JavaScript `symbol`

Bom, então veja que nesse exemplo aplicamos a função `typeof` a várias expressões de tipos diferentes e, dependendo desse tipo de expressão, temos resultados diferentes.

E, só pra lembrar que usando `typeof`, vemos claramente a diferença entre `null` e `undefined` onde `null` é do tipo `object`, o `typeof` aplicado a `null` é `object` e o `typeof` aplicado a `undefined` é `undefined`.

Ok? Antes de concluir a aula, eu gostaria de mostrar um operador especial de JavaScript que também mostra a diferença entre o `undefined` e o `null`. Como você viu, o tipo de `undefined` é diferente do tipo de `null` (Código 03).

Código 3 - 02_7 Undefined e Null.html

```
1   <html>
2
3     <head>
4       <meta charset="UTF-8" />
5       <title>Programação Estruturada - Aula 02</title>
6     </head>
```

```
7
8           <body>
9             <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
10
11            <h1>Testando Script</h1>
12
13            <p id="texto1"></p>
14
15            <script>
16              var mesmoValor = (null == undefined);
17              var mesmoTipoEValor = (null === undefined);
18
19              document.getElementById("texto1").innerHTML = mesmoTipoEValor;
20            </script>
21
22          </body>
23
24        </html>
25
```

Porém, eles têm o mesmo valor. O operador == compara os valores de duas expressões. Neste exemplo (Linha 16), temos a variável `mesmoValor` recebendo o resultado dessa comparação; usando ==, estamos comparando se o valor de `null` é igual ao valor de `undefined`.

Vou colocar essa variável como saída do nosso texto (Linha 19), e veremos aqui que temos a resposta: `true` (Figura 24). Ou seja, de fato, o valor `null` tem o mesmo valor do valor `undefined`, então os valores de ambos são iguais.

Figura 24 - Comparando Valores

Porém, JavaScript tem um operador especial que é === (Linha 17). Esse comparador compara não apenas os valores, como também os tipos de ambas as expressões. Estamos comparando se `null` e `undefined` têm o mesmo valor, e o mesmo tipo, tá? Então, se pegarmos essa variável que está recebendo o valor dessa comparação (Linha 17), o nome da variável aqui é `mesmoTipoEValor`, vamos colocá-la como saída (final da Linha 19), e vemos aqui, que agora temos a resposta `false` (Figura 25), tá?

Figura 25 - Comparando Valores e Tipos

Então, nesse exemplo, conhecendo esse operador aqui === (Linha 17), você vê que, de fato, `null` e `undefined` têm o mesmo valor, porém eles têm tipos diferentes, okay? Agora que você já conhece os tipos básicos, podemos passar para os tipos complexos. Eles serão o conteúdo da próxima videoaula.