Programação Estruturada

Aula 06 - Operadores JavaScript: Aritmética

Videoaula 06 - Métodos de Number

Videoaula 06 - Métodos de Number

Quando você estiver trabalhando com números, poderá usar os métodos e propriedades do objeto `Number` nele. Esses métodos oferecem funcionalidades muito úteis para o dia a dia de um programador e podem ser usados em literais, variáveis e expressões numéricas. Alguns desses métodos são:

1. `toString()`: retorna um número como uma String
2. `Number()`: retorna um número convertido de seu argumento
3. `parseInt()`: analisa seu argumento e retorna um número inteiro
4. `parseFloat()`: analisa seu argumento e retorna um número de ponto flutuante
5. `toExponential()`: retorna uma String, com um número arredondado e escrito usando notação exponencial. Um parâmetro opcional define o número de caracteres atrás do ponto decimal.
6. `toFixed()`: retorna uma string, com o número escrito com um número especificado de casas decimais.
7. `toPrecision()`: retorna uma string, com um número escrito com um comprimento especificado.

Além desses métodos, esse objeto também possui as seguintes propriedades:

- `MAX_VALUE`: representa o maior número possível em JavaScript
- `MIN_VALUE`: representa o menor número possível em JavaScript
- `POSITIVE_INFINITY`: representa o infinito positivo
- `NEGATIVE_INFINITY`: representa o infinito negativo

- `NaN`: representa um valor especificado como "Não é um número"

Que tal conhecer o uso desses métodos e propriedades na prática? Vamos lá!

Veremos agora vários exemplos dos métodos de números JavaScript. No primeiro exemplo, temos a variável `numero` recebendo o valor 10 e estou aplicando o **método `toString`** a essa variável `numero`. Atribuo também isso à variável `texto` e escrevendo o valor desse texto na tela (linha 22), podemos escrever o valor do texto, que é a *string*, escrevendo também o tipo desse texto. Se carregarmos isso na tela, podemos ver que temos o texto valor 10, convertemos o número 10 na *string* "10", e que, de fato, esse texto agora tem o tipo *string*.

Figura 1 - Método `toString`

O outro método que eu gostaria de apresentar para você é o **método `Number`**, que converte uma *string* em um número. Vou declarar a variável `numero` recebendo o valor da aplicação do método `Number` a *string* que tenha o caracter "3" (linha 17), e vou colocar isso na página HTML; se eu carregar agora essa página, de fato, temos o valor 3 sendo colocado na tela, então a *string* "3" foi convertida em um número e eu escrevi este valor na tela. Esse método ignora completamente os espaços em branco, então eu posso colocar vários espaços em branco depois do 3 (linha 17) mas o que é escrito na tela continua sendo o valor 3.

Se eu agora colocar a *string* "3.14159265359", que é o valor da constante *pi (π)*, e converter isso em um número, veremos que, de fato, esse valor é convertido em um número, e esse valor é impresso na tela.

Figura 2 - Método `Number`

E é importante lembrar que em JavaScript não usamos a vírgula para começar as casas decimais, usamos o ponto, então se a gente colocar na *string* uma vírgula, o valor `NaN` é retornado pelo método `Number`, e isso acontece porque nós não temos um número naquela *string*. Se nós também esquecermos o ponto ou a vírgula, colocarmos só o espaço em branco, por exemplo, se deixarmos a *string* "3 14" (linha 17), o resultado dessa tentativa de conversão dessa *string* em um número é também `NaN` e não temos um número impresso na tela.

O mesmo acontece com qualquer texto que não seja numérico, como, por exemplo, temos a *string* "IMD" e, ao tentar converter essa *string* em um número novamente, nós não temos um número, temos o `NaN`.

Vamos agora ver o método `parseInt`. Vou usar o `parseInt` para converter o texto que eu passar para um inteiro, estou convertendo a *string* que tem o caracter "3" (linha 16), usando o método `parseInt`, atribuindo isso a um número e escrevendo-o na tela. Convertendo a minha tela, vou recarregar, e vemos que, de fato, o número 3 é escrito. Certo?

Agora, se nós convertermos aquela *string* que representa o número `pi` que tem casas decimais, o que teremos, de fato, ao recarregar ainda é o número 3, isso porque nós estamos usando o método `parseInt`. Então, ele está convertendo a *string* em um inteiro. Se eu colocar o "3 14" o que ele vai fazer também é pegar o número 3 e ignorar qualquer coisa que venha depois do espaço. Inclusive, se eu colocar "3 14 PI" na minha *string*, como você pode ver aí na linha 16, continua apenas pegando o primeiro inteiro, que é o 3, ok?

Agora, se eu coloco a *string* "PI 3 14" a nossa conversão vai dar o `NaN`, o `Not a Number`, isso porque a primeira *string* que ele pegou, que é o que vem antes do espaço em branco é o texto "PI", que não é um número.

Código 1 - 06_7 Métodos de Number.html

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 06</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Métodos de Number</h1>
    <p id="texto"></p>

    <script>
    // Método parseInt()
    numero = parseInt("3 14 PI");
    //numero = parseInt("PI 3 14");
    document.getElementById("texto").innerHTML = numero;

  </script>
  </body>
</html>

```

Vamos agora fazer alguns exemplos com o método `parseFloat`. Vamos começar usando os mesmos valores que nós usamos para o `parseInt` e ver o que acontece. Então, ao passar a *string* que tem o caracter "3", o que temos é o número 3, agora vamos passar à *string* que tenha casas decimais, que representa o valor da constante *pi*. E aí, ao carregar, temos agora um *float* sendo impresso, então todas as casas decimais foram consideradas.

Código 2 - 06_7 Métodos de Number.html

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 06</title>
  </head>
  <body>

```

```
7       <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9       <h1>Métodos de Number</h1>
10      <p id="texto"></p>
11
12      <script>
13      // Método parseFloat()
14      numero = parseFloat("3.14159265359");
15      //numero = parseFloat("3 14");
16      //numero = parseFloat("3 14 PI");
17      //numero = parseFloat("PI 3 14");
18      document.getElementById("texto").innerHTML = numero;
19
20    </script>
21    </body>
22  </html>
23
```

E o comportamento com relação a espaços em branco e textos que nós vimos no final do `parseInt`, vamos rever agora com o `parseFloat` que é o mesmo, então se eu botar "3 14" na minha *string*, ele vai pegar apenas o 3. Se eu pegar "3 14 PI" novamente, estou recarregando a página, e observaremos que ele vai pegar apenas o 3. E se eu usar o "PI 3 14" teremos o `Not a Number`, o valor `NaN` na nossa tela, porque ele tentou converter essa *string* "PI" em número, o que não vai dar certo, vai dar um `Not a Number`, vai dar um `NaN`.

O outro método, o `toExponential`, vou demonstrar sua aplicação fazendo um `parseFloat` do nosso `pi`, e vamos usar esse número (linha 17) e aplicar o método `toExponential` a ele com 2 como argumento, o que acontecerá com esse número?

Código 3 - 06_7 Métodos de Number.html

```
1   <html>
2     <head>
3       <meta charset="UTF-8" />
4       <title>Programação Estruturada - Aula 06</title>
5     </head>
6     <body>
7       <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9       <h1>Métodos de Number</h1>
10      <p id="texto"></p>
11
```

```
12      <script>
13      // Método toExponential()
14      numero = parseFloat("3.14159265359");
15      numero = numero.toExponential(2); // Altere para 4
16      document.getElementById("texto").innerHTML = numero;
17
18    </script>
19    </body>
20  </html>
21
```

Vemos no HTML que ele ficou um número exponencial, usando a notação exponencial com duas casas decimais. Se eu aumentar o número 2 para 4, o que teremos como resposta agora são 4 casas decimais, ou seja, passa a ser um número exponencial, na notação exponencial, com 4 casas decimais, ok?

Figura 3 - Método `toExponential`

Agora, vamos ao método `toFixed`. Vou novamente fazer um `parseFloat` naquela *string* que tem a constante `pi` dentro dela. Vamos aplicar o método `toFixed` a esse número usando 2 como parâmetro, tá? Então o que teremos é o valor 3.14 com o tipo *string*, porque convertemos isso para uma *string*, ou seja, o número agora é uma *string*.

Figura 4 - Método `toFixed`

Então, se eu pegar o `typeof numero` (linha 17) e uma *string* que tem o número fixo de casas decimais, nesse exemplo eu utilizei o 2, se eu por acaso usar o 4 o que teremos é uma *string* que pegou as quatro primeiras casas decimais do meu número, ok? Importante notar que antes tínhamos a numeração 3.1415, mas o próximo número era 9, então ele arredondou essa última casa decimal de 5 para 6. Ok?

Código 4 - 06_7 Métodos de Number.html

```
1  <html>
2    <head>
3      <meta charset="UTF-8" />
4      <title>Programação Estruturada - Aula 06</title>
5    </head>
6    <body>
7      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9      <h1>Métodos de Number</h1>
10     <p id="texto"></p>
11
12     <script>
13     // Método toFixed()
14     numero = parseFloat("3.14159265359");
15     numero = numero.toFixed(2); // Altere para 4
16     document.getElementById("texto").innerHTML = "Valor " + numero + " tem tipo "+ (typeof numero)
17
18   </script>
19   </body>
20 </html>
21
```

Agora, se eu usar o método `toPrecision`, o exemplo é praticamente o mesmo, ao invés de usar `toFixed`, eu estou usando `toPrecision` com o valor 2, e aí nós teremos dois dígitos, tá certo? Se eu usar o `toPrecision` 4, teremos quatro dígitos, e notem novamente que a terceira casa decimal era o número 1, porém, como o próximo decimal era 5, ele arredondou isso para 2, ficando o número ou a *string* "3.142".

Figura 5 - Método `toPrecision(2)`

Figura 6 - Método `toPrecision(4)`

Por fim, veremos algumas das propriedades, as constantes, que temos no objeto `Number` (linhas 16 a 21). Temos a constante `MAX_VALUE`, que, como você pode ver aí na tela, é um número realmente muito grande, é 1.79 vezes 10 elevado a 308. O valor mínimo também é um número bastante grande, só que agora do ponto de vista negativo.

O infinito positivo é o `INFINITY`. O infinito negativo é o `-INFINITY`. Nós temos o `Not a Number`, que pode ser referenciado como `Number.NaN`, e também pode ser referenciado diretamente, sem ir ao objeto `Number`, é simplesmente `NaN`.

Código 5 - 06_7 Métodos de Number.html

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 06</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Métodos de Number</h1>
    <p id="texto"></p>

    <script>
    // Propriedades de Number
    numero = Number.MAX_VALUE;
    //numero = Number.MIN_VALUE;
    //numero = Number.POSITIVE_INFINITY;
    //numero = Number.NEGATIVE_INFINITY;
    //numero = Number.NaN;
    //numero = NaN;
    document.getElementById("texto").innerHTML = numero;

  </script>
  </body>
</html>

```

Então, são esses os exemplos de alguns dos métodos que podemos usar com os números em JavaScript.