



Desenvolvimento Web Completo 2022


Por Jorge Sant Anna, Programador
Jamilton Damasceno, Analista de Sistema e Professor





Seção 4: JavaScript




Aula 11: Casting de Tipos com toString(), parseInt() e parseFloat()

- 
- O que é Casting de tipos?
 - Simples. É converter um tipo de dado em outro tipo de dado. Essa operação de Casting, que em certos contextos, pode interferir no resultado da instrução que estamos criando.
 - A ideia é criar um script que vai receber dois parâmetros e, a partir desses parâmetros, faremos alguns testes que vão ressaltar a importância de ter em mente outro tipo de dado contido dentro da variável.

- 
- Vale lembrar que o JavaScript é uma linguagem fracamente tipada, ou seja, a inferência do tipo da variável é definida no momento de atribuição e, em função disso, ao longo do processamento do script, pode ser que aquela variável receba valores diferentes e, portanto, em determinados momentos, passe a ter tipos diferentes.

- 
- Qual que é a ideia do símbolo de soma (+) no contexto da concatenação?
 - Simples. Juntar a string da esquerda com a string da direita. Porém, valores recuperados de fontes externas dos nossos scripts, como, por exemplo, do prompt que há uma entrada a partir do próprio browser, geralmente traz para gente, informações do tipo string. A mesma coisa aconteceria se, a partir do JavaScript, fosse extraído o valor de algum elemento HTML, ou mesmo, informações extraídas do banco de dados importadas dentro da nossa página.

- 
- Geralmente, com dados externos aos scripts, são capturados em formato string e isso tem um impacto na lógica da aplicação se não for tratado corretamente.


Exemplo


```
//Variaveis  
var variavelPrompt = prompt  
("Digite algum numero");  
var variavel = prompt('Digite outro numero');  
  
document.write(variavelPrompt + variavel);
```


- 
- No caso acima, o que acontece se digitarmos 10 (dez) e 20 (vinte). Vai concatenar ou vai somar?

Casting em JavaScript

1020

- 
- Perceba que a operação é de concatenação. Por mais que digitamos valores numéricos, eles estão sendo recebidos como strings e, naturalmente, não tem como somar strings, não podemos somar texto. E por isso, temos o resultado da concatenação.
 - É nesse momento que entra o casting de dados. Se o nosso objetivo fosse, de fato, somar os valores recebidos, precisaríamos converter esses valores, de fato, do tipo numérico, de tal modo que o interpretador do JavaScript entenda como encaixar o operador da soma e não mais de concatenação.

- 
- Como podemos converter uma string que representa um valor numérico para um valor do tipo number?
 - Simples. Utilizando a função “parseInt”.


Exemplo


```
//Casting de variáveis  
variavelPrompt = parseInt(variavelPrompt);  
variavel = parseInt(variavel);  
  
document.write(variavelPrompt + variavel);
```


- 
- Agora temos o resultado que é a soma dessas duas variáveis.

Casting em JavaScript

30

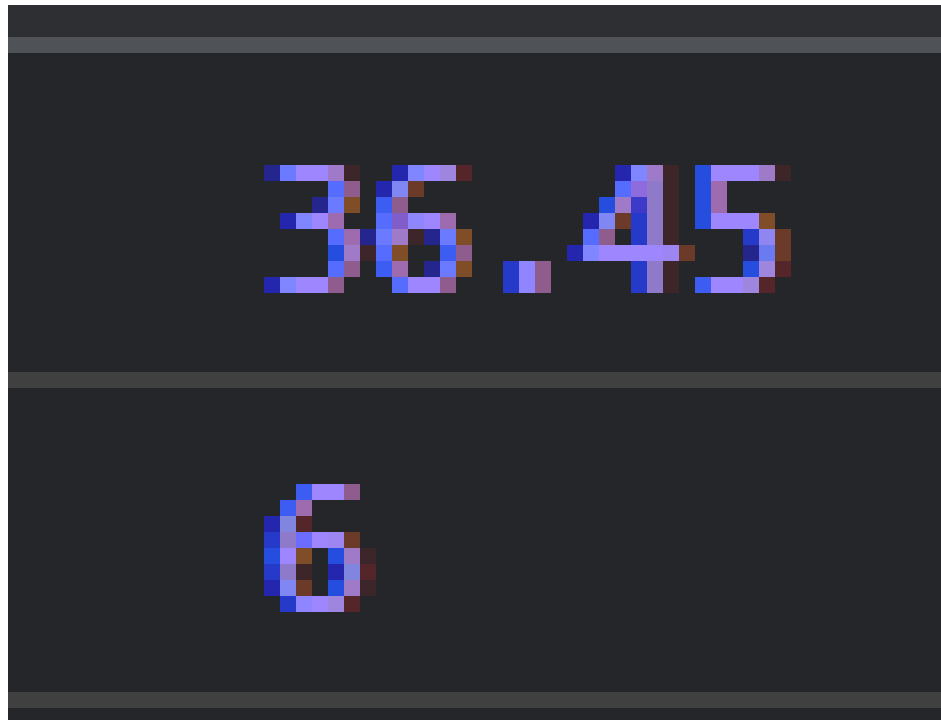
- 
- Então, em vez de concatenar 10 (dez) com o 20 (vinte), estamos, agora, a partir do casting de tipo, de fato somando os valores dez e vinte, resultando em 30 (trinta).
 - Além do `parseInt`, existe, também, um outro método, que é o método `parseFloat`.
 - A diferença entre o `parseInt` e o `parseFloat`, é que o `parseFloat` preserva a fração de um número. Então, podemos receber um valor com uma fração e, através do `parseFloat`, transformamos essa string em um valor numérico que contém fração.


- 
- No `parseInt`, ele extrai apenas o valor inteiro da string, ou seja, apenas a parte inteira do valor numérico contido naquela string. Enquanto o `parseFloat` continua fazendo esse casting de tipo, porém, ele preserva a fração.
 - Então, dependendo da lógica, `parseInt` e `parseFloat` podem impactar de forma diferente no resultado do seu script.


- 
- Por isso, é importante ter essa diferença em mente. Resumindo, o `parseInt` remove a fração, preservando, apenas, a parte inteira do número, enquanto o `parseFloat` preserva essa fração.
 - Além disso, temos a função inversa, ou seja, transformar valores numéricos em valores do tipo `string`.

Exemplo

```
//Casting de variáveis  
variavelPrompt = parseFloat(variavelPrompt);  
variavel = parseInt(variavel);  
  
//document.write(variavelPrompt + variavel);  
console.log(variavelPrompt);  
console.log(variavel);
```




- 
- A forma de escrita da conversão, do casting, da variável do tipo number para a variável do tipo string é um pouco diferente. Ao invés de utilizar um método que retorna a representação numérica de uma string, utilizamos o método “toString()” que, parte da estrutura de um valor numérico. Dessa forma, podemos extrair a representação textual desse valor numérico.

- 
- Então, basta utilizar esse método de “toString()” nas nossas variáveis numéricas para trabalhar com a representação textual e, não mais, com a sua representação numérica.


Exemplo

```
//Casting para o tipo String  
var str = 10;  
var ing = 20;  
  
document.write(str.toString() +  
ing.toString());
```



Casting em JavaScript

1020

- 
- Temos, portanto, a concatenação da representação textual dos valores numéricos 10 (dez) e 20 (vinte).
 - Embora, essas operações sejam, relativamente, muito simples, é importante ter em mente, o conceito de tipos de variáveis e a possibilidade casting desses tipos, para ter certeza que dentro do seu script não vai ter nenhum valor sendo trabalhado de forma errada, até porque, esse tipo de situação pode provocar um bug que, geralmente, é difícil ser encontrado e resolvido.