Isaias Soares
Id: 1178008

Coerções (mudança de tipos)

Na programação, as entradas de dados serão sempre textuais, portanto é importante que saibamos fazer a mudança do tipo *string* (texto) para o tipo *number* (inteiros e decimais) para podermos fazer contas.

Considere o programa abaixo:

```
// Solicita um número
const x = prompt('Digite um número:');
const y = prompt('Digite um número:');

// Calcula a média
const media = (x + y) / 2.0;

// Mostra no console
console.log(media);
```

O programa acima não vai funcionar como o esperado, se digitado 10 e 10 como valores ele não produzirá 10 de média, mas sim 505.

Mas, o que pode ter acontecido?

Como dito, as entradas são sempre *strings*, um tipo textual, e acontece que *strings* têm um operador de soma da mesma forma que os números.

No entanto, o somar nas *strings* concatena *strings*, por exemplo:

```
const textoFinal = 'hello' + ' ' + 'world!';
// textoFinal -> 'hello world';
```

O que acontece no nosso programa é que 10 e 10 são concatenados, eles são na verdade `"10"` e `"10"`, observe as aspas que indicam que isso é texto, não número. Sendo assim, temos `"1010" / 2.0`, *strings* não tem o operador de divisão, então nesse momento o JavaScript converte a *string* para número e faz a divisão.

Para não dependermos das conversões implícitas do JavaScript podemos nós mesmos solicitar a mudança de tipos. Usamos duas funções para fazer a **coerção de tipos** (*cast*) de *string* para numéricos:

- `parseInt()`: Converte inteiros (sem casas decimais);
- `parseFloat()`: Converte decimais;
- `Number()`: Construtor do tipo numérico;

Existe diferença na conversão de inteiro e decimais pois, inteiros são representados de uma forma em binário e decimal de outra. Para a máquina é muito mais fácil representar os inteiros do que os decimais. Veremos implicações disso.

Para usar essas funções basta passar o texto entre os parênteses, o que chamamos de parâmetro.

```
const x = parseInt('10');
const y = parseFloat('10.5');
const z = Number('10');
const t = Number('10.5');
```

Strings podem ser declaradas com aspas `""` ou apóstrofos `''`. Então passamos texto para as funções de parse, se olharmos o resultado no console veremos os mesmos valores sem as aspas, o que indica que agora são numéricos.

```
const x = parseInt('10');
const y = parseFloat('10.5');
const z = Number('10');
const t = Number('10.5');

console.log(x); // -> 10
console.log(y); // -> 10.5
console.log(z); // -> 10
console.log(t); // -> 10.5
```

Sabendo disso podemos corrigir nosso programa agora:

Opção 1:

```
// Solicita um número (como texto)
const xDigitado = prompt('Digite um número:');
const yDigitado = prompt('Digite um número:');

// Converte a string para número
const x = parseFloat(xDigitado);
const y = parseFloat(yDigitado);

// Calcula a média
const media = (x + y) / 2.0;

// Mostra no console
console.log(media);
```

Opção 2:

```
// Solicita e converte um número
const x = parseFloat(prompt('Digite um número:'));
const y = parseFloat(prompt('Digite um número:'));

// Calcula a média
const media = (x + y) / 2.0;

// Mostra no console
console.log(media);
```

A única diferença das duas opções é que evitamos adicionar mais constantes na segunda opção, fazendo a coerção direta do resultado de `prompt()`. Ambas funcionam da mesma forma.

Novidades nesses exemplos

- `console.log()`: Uma função que mostra no console o que o programador desejar. Você pode passar entre os parênteses um ou mais parâmetros separados por vírgula " , ", todos serão mostrados no console.
- `prompt()`: Uma função que solicita que o usuário entre com um dado. O texto (entre aspas) passado por parâmetro (entre parênteses) será a mensagem que o usuário verá na tela. Essa função só funciona no navegador.
- Funções podem ser encadeadas. Por exemplo, o resultado de `prompt()` é uma *string*, o parâmetro de `parseFloat()` é uma *string* também, sendo assim podemos encadear `parseFloat()` com `prompt()` como no exemplo 2 acima.

Se isso ainda não estiver claro fique tranquilo, com a prática vai se tornar cada vez mais natural. Também vamos utilizar essa técnica diversas vezes com o passar das aulas.

Referências e Materiais Complementares

- Explicando a coerção de tipos em Javascript

- JavaScript's type system
- JavaScript
- JS++ Documentation

Próximo Tópico