Operadores lógicos, de comparação e conversão de tipos

Comparação entre números e texto

Normalmente operações de comparação servem para verificar se:

- Dois valores são iguais ou idênticos (`==` ou `===`)
- Dois valores são diferentes (`!=`)
- Dois valores são maiores (`>` e `>=`)
- Dois valores são menores (`<` e `<=`)

Comparando números:

```
1 > 2
// false
1 < 2
// true
1 != 2
// true
2 == 2
// true
```

Comparando texto:

Parece estranho, mais podemos utilizar os operadores de comparação para texto.
A regra é o seguinte: As letras maiúsculas sempre serão menores que as minúsculas.

Sendo assim "Z" é menor que "a". Veja:

```
"Z" < "a"
// true
```

Operadores lógicos

Os operadores lógicos trabalham no âmbito booleano comparando dois ou mais valores. Ao todo no JavaScript temos três operadores lógicos: *and*, *or* e *not*.
Estes operadores trabalham retornando um resultado booleano (`true` ou `false`) que podemos utilizar em outras estruturas (comparação e de repetição).

O *and* é um operador binário que retorna verdadeiro (`true`) quando **todos** os valores forem verdadeiros:

```
true && false
// false
```

```
true && true
// true
true && true && false
// false
```

O operador binário *or* produz um retorno verdadeiro se **qualquer** um dos valores forem verdadeiros:

```
true || false
// true
true || true
// true
false || false
// false
true || false || false
// true
```

E o operador *not* que "inverte" o resultado do seu valor de entrada. Tudo que é verdadeiro torna-se falso e vice versa. Vejamos:

```
!true
// false
!false
// true
```

Convertendo tipos

No JavaScript nós precisamos dizer quando queremos converter algo para número ou texto. Isso significa que precisamos utilizar as funções próprias para isso: `String()` e `Number()`.
Este processo chama-se *casting* e é bastante comum em outras linguagens. Veja:

```
String(2) + 2
// "22"
Number("5") + 2
// 7
Number(true)
// 1
```

Identificando o tipo

Ainda no âmbito de tipos, o JavaScript possuí uma função especial chamada `typeof`, que recebe um valor e retorna o tipo. Veja:

```
typeof(3)
// "number"
```

```
typeof(NaN)
// "number"
typeof(Infinity)
// "number"
typeof("Olá")
// "string"
typeof(false)
// "boolean"
```

Esta função especial é muito prática para verificação de valores para funções onde você deve esperar um tipo e evitar que se receba outro.

Desafio

Crie um código de uma possível calculadora que receberá sempre valores como texto e deve retornar os dois valores somados.

Resposta

```
Number("2") + Number("1")
// 3
```

Explicação

Aplicando sempre a conversão de tipos para número você vai garantir que o resultado seja numérico (mesmo se o valor recebido seja do tipo número).