

FUNÇÕES NATIVAS

“

Javascript tem diversas funções nativas, que nos ajudam a realizar expressões matemáticas, manipular arrays e strings.



“

Descobrimos que todo array tem uma propriedade chamada **length** que traz a quantidade de índices, mas o JS oferece muito mais, temos **funções nativas** para trabalhar com **arrays**!



Array.pop()

A função pop é sempre utilizada a partir de um array, e remove sempre o último elemento de um array e por fim retorna o seu valor. Veja esse exemplo:

```
var frutas = [ "Bananas", "Uva", "Maçã", "Laranja" ]  
  
// além de remover a função pop, retorna o valor removido!  
var ultimaFruta = frutas.pop() // "Laranja"  
  
console.log(frutas)  
// [ "Bananas", "Uva", "Maçã"]
```

Array.push()

Você já se perguntou como podemos adicionar novos itens em um array criado? Bom, fazemos isso com a função push. Passamos a informação que queremos inserir dentro do array como parâmetro, e o push irá adicionar o novo item na última posição do array:

```
var frutas = [ "Bananas", "Uva", "Maçã", "Laranja" ]  
frutas.push("Goiaba")
```

```
{ }
```

```
console.log(frutas)  
// ["Bananas", "Uva", "Maçã", "Laranja", "Goiaba" ]
```

Array.indexOf()

Essa função vai te ajudar muito quando você quiser descobrir se um elemento existe em um array. Você deve passar o valor procurado como parâmetro dessa função, e se ela encontrar o valor, irá retornar a posição do item dentro do array. Caso não encontre, irá sempre retornar -1.

```
var frutas = [ "Bananas", "Uva", "Maçã", "Laranja" ]  
console.log(frutas.indexOf('Melancia'))  
// Irá retornar -1, pois não existem nenhum elemento que  
faça match  
console.log(frutas.indexOf('Uva'))  
// Irá retornar 1, pois é a posição que a uva se encontra
```

“

O JS já possui diversas **funções nativas**, que fazem os **cálculos** e retornam os resultados prontos. Nós só precisamos saber qual o nosso objetivo e entender o que cada função faz.



Math.random()

A função Math.random() retorna um número aleatório entre 0 e 1.
Exemplo:

```
{ }
```

```
Math.random() // retorna 0.4031609856267999
```

Digamos que se deseja gerar um número aleatório inteiro entre 0 e 50. Para isso, realizamos dois passos. Multiplicamos o número por 50:

```
{ }
```

```
Math.random() * 50 // retorna 20.158049281
```

Depois utilize a função Math.round() para arredondar e obter o número inteiro:

```
{ }
```

```
Math.round(Math.random() * 50) // retornou 20
```


Math.min()

A função Math.min() recebe 2 ou mais parâmetros e retorna o menor valor. Exemplo:

```
{}
```

```
Math.min(16, 10, 15) // retorna 10
```

O interessante é que podemos utilizar variáveis como argumentos para tornar mais dinâmico. Vejamos em um cenário onde queremos comparar preços de produtos:

```
{}
```

```
var precoTubaina = 3.50  
var precoGuarana = 4.00  
  
Math.min(precoTubaina, precoGuarana)  
// retorna 3.50 (precoTubaina)
```

Math.max()

A função `Math.max()` é muito semelhante a `Math.min()`, mas nesse caso retorna o maior número entre os parâmetros.

```
{}
```

```
Math.max(16, 10, 15) // retorna 16
```

Voltando ao exemplo dos produtos, desta vez quero trazer o maior valor, o mais caro:

```
{}
```

```
var precoTubaina = 3.50
```

```
var precoGuarana = 4.00
```

```
Math.max(precoTubaina, precoGuarana) // retorna 4.00  
(precoGuarana)
```

“

Entre as **funções nativas** da linguagem, temos algumas funções que trabalham exclusivamente com o tipo de dado **String**!



String.repeat()

A função .repeat() é utilizada para repetir uma string. Para utilizar basta chamar a função logo após a string e passar por parâmetro quantas vezes a string deve se repetir:

```
{}
```

```
"Azul".repeat(3) // retorna "AzulAzulAzul"
```

Também pode ser utilizada com uma variável pré definida:

```
{}
```

```
var bomDia = "Bom dia! "
```

```
bomDia.repeat(3) // retorna "Bom dia! Bom dia! Bom dia! "
```

String.toUpperCase()

A função .toUpperCase() modifica a string e converte todas os caracteres para letras maiúsculas:

```
var bomDia = "Bom dia"  
  
bomDia.toUpperCase() // retorna "BOM DIA"
```