Formatação de Strings

> *"O tipo String do JavaScript é usado para representar informações de texto. É um conjunto de "elementos" composto por valores inteiros de 16-bits sem sinal. Cada elemento dentro da String ocupa uma posição dentro dessa String. O primeiro elemento está no índice 0, o próximo no índice 1, e assim sucessivamente. O tamanho de uma String é a quantidade de elementos que ela possui."* - MDN Web Docs

---

Podemos criar strings usando aspas simples ou aspas duplas:

```
'string qualquer';
"string qualquer";
`string qualquer`;
```

As strings possuem métodos diversos, nos quais podemos usá-los em vários momentos do desenvolvimento.

- Métodos:
  - Length → Comprimento da string:

```
//Constante com string:
const texto = 'Clóvis';

console.log(texto.length);
//Resultado: 6
```

  - indexOf → Posição de um caractere dentro da string (caso use uma palavra para pesquisar, é retornado a posição do primeiro caractere):

```
//Constante com string:
const txt = 'Ele foi ao supermercado XYZ';

console.log(txt.indexOf('X'));
//Resultado: 24
```

  - Slice → Corta uma parte de uma string e a coloca em uma nova string:

```
//Constante com string:
const txt = "Apple, Orange, Banana";
//Corte da posição 7 à posição 13:
const res = txt.slice(7, 13);

console.log(res);
//Resultado: Orange


//Caso seja passado um parâmetro negativo, a contagem começará de trás para frente:


//Constante com string:
const ano = '2021';
//Método que recorta as duas últimas posições
const final = ano.slice(-2);

console.log(final);
//Resultado: 21
```

  - Substring → funciona da mesma forma que o Slice, porém não aceita parâmetro negativo:

```
//Constante com string:
const txt = "Apple, Orange, Banana";
//Corte da posição 7 à posição 13:
const res = txt.substring(7, 13);

console.log(res);
//Resultado: Orange
```

Ps.: Caso ambos métodos (Slice e Substring) recebam apenas um parâmetro, o "corte" será feito a partir da posição indicada:

```
//Constante com string:
const txt = "Apple, Orange, Banana";
//Corte a partir da posição 7:
const res = txt.substring(7);
const res1 = txt.slice(7);

console.log(res);
//Resultado: Orange, Banana
console.log(res1);
//Resultado: Orange, Banana
```

- Replace → Substitui um valor por outro ('valorASerSubstituido', 'novoValor'):

```
//Constante com string:
const txt1 = 'Clóvis tem 20 anos';
//Busca o 1º valor do parâmetro e substitui pelo 2º valor:
const txt2 = txt1.replace('20', '21');

console.log(txt2);
//Resultado: Clóvis tem 21 anos
```

Ps.: Caso queira substituir todas as palavras passadas como 1º parâmetro, basta usar uma expressão regular **(/texto/g)**:

- Exemplo:

```
//Constante com string:
const txt1 = 'Clóvis tem 20 anos; Estamos em 1920';
//Busca o 1º valor do parâmetro e substitui pelo 2º valor:
const txt2 = txt1.replace(/20/g, '21');

console.log(txt2);
//Resultado: Clóvis tem 21 anos; Estamos em 1921
```

- toUpperCase → transforma os caracteres passados por parâmetro em letras maiúsculas:

```
//Constante com string:
const txt = 'Clóvis Fernando';
const txt2 = txt.toUpperCase()

console.log(txt2);
//Resultado: CLÓVIS FERNANDO
```

- toLowerCase → transforma os caracteres passados por parâmetro em letras minúsculas:

```
//Constante com string:
const txt = 'Clóvis Fernando';
const txt2 = txt.toLowerCase()

console.log(txt2);
//Resultado: clóvis fernando
```

- chartAt → retorna o caractere especificado a partir de uma string. Caso nenhum valor seja passado por parâmetor, seu valor será 0:

```
let stringQualquer = 'Clóvis';

stringQualquer.charAt();   // "C"
stringQualquer.charAt(0);  // "C"
```

```
stringQualquer.charAt(1);  // "l"
stringQualquer.charAt(2);  // "ó"
stringQualquer.charAt(3);  // "v"
stringQualquer.charAt(4);  // "i"
stringQualquer.charAt(5);  // "s"
stringQualquer.charAt(-1); // ""
```

- charCodeAt → retorna um número inteiro entre 0 e 65535, responsável por representar a unidade de código UTF-16 no índice fornecido. O valor a ser recebido por parâmetro deve ser um inteiro maior ou igual a 0 e menor que o comprimento da string em questão. Caso nenhum valor seja recebido, será considerado como 0:

```
let stringQualquer = 'Clóvis';

stringQualquer.charCodeAt();   // 67
stringQualquer.charCodeAt(1);  // 108
stringQualquer.charCodeAt(2);  // 243
stringQualquer.charCodeAt(3);  // 118
stringQualquer.charCodeAt(4);  // 105
```

- codePointAt → Retorna um número inteiro maior que 0, sendo o valor do ponto de código Unicode:

```
let string = '🕺';
string.codePointAt();  // 127939
let string = '👌';
string.codePointAt();  // 128077
```

- lastIndexOf → Retorna o último índice que um determinado elemento pode ser encontrado num array. Caso não esteja no array, nos trará -1:

```
let arrayQualquer = [11, 5, 9, 7];

arrayQualquer.lastIndexOf();    // -1
arrayQualquer.lastIndexOf(1);   // -1
arrayQualquer.lastIndexOf(2);   // -1
arrayQualquer.lastIndexOf(11);  // 0
arrayQualquer.lastIndexOf(5);   // 1
arrayQualquer.lastIndexOf(9);   // 2
```

- startsWith → Retorna um valor booleano, verifica o valor do início da string, ou seja, se a string começar com o valor passado por parâmetro, retorna *true*, caso não comece, retorna *false*:

```
let string = 'Clóvis tem 20 anos';

string.startsWith('Clóvis');   // true
string.startsWith('Clóvis ');  // true
string.startsWith('20 anos');  // false
string.startsWith('tem');      // false
```

- endsWith → Retorna um valor booleano, verifica se uma string termina com determinados caracteres, ou seja, caso termine com o valor passado entre parâmetro, retorna *true*, caso não, retorna *false*:

```
let string = 'Clóvis tem 20 anos';

string.endsWith('Clóvis');   // false
string.endsWith('tem');      // false
string.endsWith('20 anos');  // true
string.endsWith('20');       // false
```

- includes → Retorna um valor booleano, determina se um conjunto de caracteres está incluído na string em questão, caso esteja, retorna *true*, caso não, retorna *false*:

```
let string = 'Só sei que nada sei';

string.includes('clóvis');   // false
string.includes('nada');     // true
string.includes('só');       // false
```

```
string.includes('Só');       // true
string.includes('sei');      // true
```

- concat → Retorna um array contendo um novo array que contém todos os arrays ou valores passados como parâmetro do método em questão, ou seja, ele concatena os arrays e nos retorna um novo como se fosse a "união" dos outros:

```
let nome1 = 'Clóvis, ';
let nome2 = 'João, ';
let nome3 = 'Francisca.';

let novaString = nome1.concat(nome2, nome3);
// "Clóvis, João, Francisca."
```

- fromCharCode → Retorna uma string contendo os caracteres que correspondem a valores Unicode. Como esse método é considerado um método estático de String, usaremos sempre como String.fromCharCode():

```
String.fromCharCode(65, 66, 67);              // "ABC"
String.fromCharCode(67, 76 , 79, 86, 73, 83); // "CLOVIS"
```

- fromCodePoint → Retorna caracteres criados usando a sequência especificada de pontos de código (por exemplo: numero1, numero2, ..., numeroX). Como esse método é considerado um método estático de String, usaremos sempre como String.fromCodePoint():

```
String.fromCodePoint(40, 41);    // "()"
String.fromCodePoint(10);        // "\n"
String.fromCodePoint();          // ""
String.fromCodePoint(42);        // "*"
String.fromCodePoint(65, 90);    // "AZ"
```

- split → Divide uma determinada string em uma lista de "substrings", colocando-as num array e retornando o mesmo.

```
let string = 'Clóvis tem 20 anos';

string.split();
// ["Clóvis tem 20 anos"]

string.split('t');
// ["Clóvis ", "em 20 anos"]

string.split('20');
// ["Clóvis tem ", " anos"]

string.split('s');
// ["Clóvi", " tem 20 ano", ""]

string.split("");
// ['B', 'o', 'm',' ', 'd', 'i','a']
```

- substr → Retorna uma parte da string, iniciando na posição do índice indicado por parâmetro e seguindo até o fim. Caso queira uma palavra ou região do texto em específico, basta colocar 2 parâmetros (início, fim):

```
let string = 'Clóvis tem 20 anos';

string.substr();       // "Clóvis tem 20 anos"
string.substr(12);     // "0 anos"
string.substr(0, 6);   // "Clóvis"
string.substr(7, 15);  // "tem 20 anos"
```

- match → retorna uma correspondência entre uma string com uma expressão regular. Caso não insira nenhum valor como parâmetro, o método retornará um array com uma string vazia:

```
var string = "Caso tenha dúvidas, reveja o level 99.9";

var expressao = "level 99.9";
var resultado = string.match(expressao);

console.log(resultado);
```

```
/*
[
  'level 99.9',
  index: 29,
  input: 'Caso tenha dúvidas, veja o level 99.9',
  groups: undefined
]
*/
```

- search → Busca uma ocorrência entre uma String e uma expressão regular. Caso encontre determinado padrão ou valor dentro da String, nos retornará um valor maior que 0, sendo esse o índice do elemento procurado. Se não achar, retornará -1:

```
let string = "faculdadE iv2";
let letraMaiuscula = /[A-Z]/g;
let virgula = /[,]/g;

console.log(string.search(letraMaiuscula));
// 8 => índice da primeira letra Maiúscula encontrada
console.log(string.search(virgula));
// -1 => Não foi encontrado nenhuma vírgula na string
```

- repeat → retorna uma nova string com o número de cópias igual ao valor passado entre parâmetros:

```
let string = 'Clóvis '
string.repeat(1);   // Clóvis
string.repeat(5);   // Clóvis Clóvis Clóvis Clóvis Clóvis
string.repeat(2);   // Clóvis Clóvis
```

- trim → é responsável por remover os espaços em branco do início e/ou fim da string:

```
let string = '    Clóvis ';

string.trim();   // Clóvis
```

Template String

É uma string que permite a inserção de expressões em seu escopo, além de utilizar uma string que contenha várias linhas.

- Sintaxe:

```
`conteúdo da string`
```

- Exemplo de uso de expressões:

```
`${expressao}`

`A soma é igual a ${2 + 2}`
// Retorna: A soma é igual a 4
```

Podemos também fazer algumas renderizações, sejam expressões, dados ou até mesmo variáveis. Se quisermos renderizar uma variável, colocamos dentro de `` `${variavel}` ``:

```
let soma = 1 + 4;
let string = `A soma é igual a ${soma}`

console.log(string);
// A soma é igual a 5
```

- Interpolação de expressões:

Para fazermos interpolações, usávamos o seguinte meio:

```
let nome = 'Clóvis';
console.log('Ele se chama ' + nome + ' e tem ' + (2021 - 2001) + ' anos.');
// Retorna: Ele se chama Clóvis e tem 20 anos.
```

Com a template String fica mais simples, por exemplo:

```
let nome = 'Clóvis';
console.log(`Ele se chama ${nome} e tem ${2021 - 2001} anos.`);
// Retorna: Ele se chama Clóvis e tem 20 anos.
```