**Básico de String**

Em JavaScript, pode-se usar string com aspas simples(‘ ‘) ou duplas(“ ”).

**Ex.:**

**let a = ‘você’**

**let b = “tu”**

**Nota:** se começou usando aspas simples ou dupla, termine com a que você começou.

**Ex.:**

***Let c = ‘teste” -> vai dar erro***

**Formatação de String**

Para juntar strings, existem duas maneiras principais:

* Pode ser usando ***Concatenação***;
* Pode ser **Template Literal** ou **Template String**.

**Concatenação:** uma forma de juntar strings usando **operador +**

Ex.:

***var a = ‘texto’***

***a = ‘Estamos escrevendo ’ + a -> return ‘Estamos escrevendo texto’***

**Nota:** em strings muito longas, recomenda-se o uso de **backslash(\)** no final de cada string para indicar que a string vai continuar em outra linha.

Ex.:

***var texto = ‘testando\***

***para ver\***

***se você\***

***é’***

**Nota 2:** tambémé possível juntar números e strings.

Ex.:

***Var num = 10***

***Var texto = ‘ testando’***

***Var num\_texto = 10 + texto -> return ’10 testando’***

**Template String:** é uma forma de juntar string e números da seguinte forma:

* Toda template string está entre **crase(` `);**
* Caso queria adicionar uma variável de fora dentro da string, ela deve tá entre **${variável},** chamado de **placeholder**;

**Ex.:**

***Var a = ‘texto’***

***Var b = `Eu estou escrevendo um ${a}` -> return ‘Eu estou escrevendo um texto’***

**Sequência de Scape**

Existem alguns códigos que podem ser colocados dentro das strings para fazer uma formatação especial. São eles:

* **\’ (single quote)** -> caso queira adicionar **uma** **aspa simples** sem que faça parte do código;

**Ex.:**



* **\’’ (double quote)** -> caso queira adicionar **uma aspa dupla** sem que faça parte do código;

***Ex.:***



* ***\\(backslash):***caso queira botar \(backslash) sem fazer parte do código;

***Ex.:***



* ***\n(new line):***caso queira pular uma linha;

***Ex.:***

* ***\t(tabulação):*** dá um TAB;

***Ex.:***

* ***\b(backspace):*** dá SPACE;

***Ex.****:*

***Principais Métodos e Atributos das Strings***

***n.lenght =>*** atributo que retorna o **comprimento da string**;

***Ex.:***

***n.toUpperCase() =>*** método que deixa todas as letras maiúsculas;

Ex.:



***n.LowerCase() =>*** método que deixa todas as letras minúsculas;

Ex.:



***n.replace() =>*** método que substitui uma parte da string por outra;

Ex.:



**Tirando caracteres específicos da String**

**Primeira forma:** depois da string, colocar **[n]**, onde n é o index do caractere.

**Nota:** se o index não estiver fora do alcance, retorna **Undefined**

Ex.:



**Segunda forma:** usando o método ***.charAt(n)***, onde n é o index do caractere.

**Nota:** se o index estiver fora do alcance, retorna **“”(string vazia)**

Ex.:



Caso queira retirar um char específico, é recomendável usar o método charAt!

**Achando o index de um determinado caractere ou substring**

Basta usar o método **indexOf(‘string’),** que irá retorna o index da **primeira ocorrência** do caractere ou substring.

**Nota:** se o caractere ou substring não for encontrado, retorna -1.

**Ex.:**





**Nota:** se quiser o index da **última ocorrência** de uma substring ou um caractere, use o método **.lastIndexOf()**, que funciona da mesma forma que o .indexOf(), mas começa a contagem de trás pra frente.

**Ex.:**

**Fatiando uma String**

Para isso, usamos o método **.slice(x, y),** onde:

* X -> index inicial;
* Y -> index final;
* Se a posição y for informada, ele vai até o final da string;
* O sentido da contagem é da esquerda para direita;
* A última posição não é exibida, por isso use um unidade a mais;

Exs.:



* Se os números forem negativos, a contagem começará de trás para frente e contagem vai até o final de string;

**Ex.:** ‘ê’ -> -1 ‘c’ -> -2 ‘o’ -> -3 e assim por diante



**Método para Concatenar Strings**

Para concatenar strings sem o operador +, usamos o método .***concat(x , y)***, onde:

* X -> primeira string;
* Y -> segunda string

Ex.:



**Métodos para adicionar strings no Começo e no Final de um String**

Os dois métodos abaixo servem para ‘acolchoar’ duas strings em um determinando comprimento.

* **.*padStart(x, y)***
* ***.padEnd(x, y)***

**Onde:**

* X -> é o comprimento que a string final vai ocupar;
* Y -> é a string que queremos juntar

Ex.:







**Nota:** se o tamanho da nova string for maio que o necessário, o valor y será duplicado para preencher o valor fornecido em x.

Ex.:



**Nota 2:** se o espaço for insuficiente, o valor y será cortado.



**Método para remover espaços laterais**

Para remover espaços laterais, use o método ***.trim()***

**Ex.:**