# Tema 05 – Strings

## Introdução

Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá sobre alguns métodos usados com strings.

## Mais sobre strings

Strings é um tipo especial em toda linguagem de programação e no JavaScript não seria diferente. O tipo de dados String tem propriedades e métodos, que permitem manipulá-las de diversas formas. Vamos aprender alguns desses métodos e dessas propriedades.

Strings é um conjunto de caracteres muito utilizado em diferentes linguagens de programação de diversas formas possíveis. Uma string pode conter letras, números e caracteres especiais e, dessa forma, possibilitar enviar bloco de dados e não somente palavras ou frases. No contexto de aplicações reais, strings são a forma de enviar informações de uma aplicação para outra. Podemos citar o formato JSON (JavaScript Object Notation), que é um padrão de formatação de dados, que é facilmente “parseado” (dividido) em informações úteis. Outro padrão muito utilizado em aplicações é o XML (Extensible Markup Language). Como foi dito anteriormente, no JS, uma string é criada colocando as informações entre aspas simples ou duplas:

|  |
| --- |
| const string1 = "Uma string";  const string2 = 'Também 123 [] é uma strings 231 -1'; |

Ela também pode ser colocada entre crases:

const string3 = `Essa também é uma string entre crases`;

Atenção: Esse modo de utilizar strings entre crases é muito útil quando queremos concatenar (juntar) strings como outras strings ou dados de variáveis etc.

## Concatenando strings

A concatenação de strings pode ser realizada com o operador + de forma semelhante a muito linguagem de programação, tais como: Java, C++, C#, etc.

|  |
| --- |
| **Concatenando strings** |
| "Instituto" + ' ' + 'da' + " " + "Oportunidade" + ' ' + 'Social' |
| **Resultado** |
| Instituto da Oportunidade Social |

Mas quando estamos trabalhando com dados dinâmicos (variáveis, resultados de funções, etc), é mais eficiente utilizar o formato de concatenar utilizando crases. Vamos criar um projeto para ver isso na prática.

### Vamos praticar

Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 04\_JS\_Strings.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

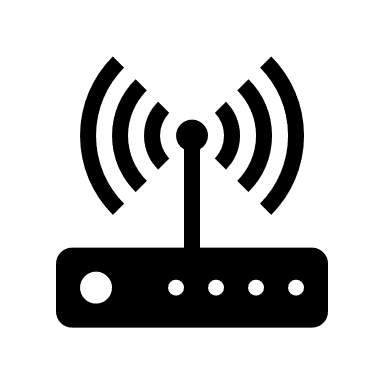
|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>Strings</title>      </head>      <body>  <script src="./js/main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Ao criar o seu arquivo a hierarquia do diretório do projeto deverá ficar assim:

Text

Description automatically generated with medium confidence

Observe que mudamos a organização dos arquivos no projeto, agora o arquivo **main.js** está em uma pasta chamada **js**, enquanto o arquivo **index.html** está no diretório raiz do projeto. Isso é importante para você ser uma pessoa organizada com os seus projetos. Além disso, o caminho do script agora está com apontando para o novo diretório (<script src="**./js/main.js**"></script>).

1. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas vamos completando a implementação do código JavaScript.
2. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**).

Agora, vamos implementar todo o nosso código desse tema no arquivo main.js e ver as mensagens impressas no console.

1. Vamos ver as diferenças de utilizar strings entre aspas e entre crases. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 1 | // Concatenar strings  let valor01 = 100;  const valor02 = 5;  console.log('A multiplicação de ' + valor01 +' por ' + valor02 + ' é ' + valor01 \* valor02 );  console.log(`A multiplicação de ${valor01} por ${valor02} é ${valor01 \* valor02}`); |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

1. Antes de explicar o que foi programado vamos ver o resultado mostrado no console.

Graphical user interface, text, application, email

Description automatically generated

1. As duas instruções produzem o mesmo resultado. A instrução mostra na linha 5 é a forma tradicional para concatenar strings. Isso, pode ser uma vantagem para pessoas, que vêm de outras linguagens de programação, se sentirem mais habituadas com esse formato de concatenação. A desvantagem principal é a quantidade de aberturas e fechamentos de aspas, além dos muitos operadores de soma, que podem fazer a pessoa programadora se perder e acabar esquecendo de abrir ou fechar um par de aspas corretamente ou de colocar um sinal de +.
2. Desse modo, a maneira de concatenar mostrada na linha 7 é conhecida como *template strings*, pois elas permitem embutir expressões de JavaScript dentro da string através do símbolo de $ e a abertura e fechamento das chaves { }

${expressão\_do\_JavaScript}

Essa expressão pode ser qualquer instrução válida no JavaScript. Utilizando a segunda forma de concatenar strings, a programação fica mais prática e rápida. Lembre-se disso, quando estivermos criando uma string enorme com muitas linhas.

## Métodos de strings

Vamos aprender sobre os métodos de strings continuando a nossa programação no arquivo main.js.

### Acesso a caracteres

Vamos começar pelo acesso à caracteres em uma string. Podemos fazer o acesso de duas formas, a primeiro é pelo método chatAt(n), onde n é o valor da posição do caractere na string, e a segunda é através de colchetes [n], onde n é o valor da posição do caractere na string. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 9 | // Métodos de strings  console.clear();  const string01 = 'Instituto da Oportunidade Social';  // Acessando o primeiro caractere  console.log(string01.charAt(0));  console.log(string01[0]);  // Acessando o décimo primeiro caractere  console.log(string01.charAt(10));  console.log(string01[10]); |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Observe que a primeira posição do caractere na string é a zero ( 0 ), isso porque string são vetores (arrays) e em linguagens de programação a primeira posição sempre começa do índice zero. Por isso, nas linhas 14 e 15, as instruções retornaram **I**, pois o caractere na primeira posição (índice 0) é o **I** e o caractere na décima primeira posição (índice 10) é o **d**. Falaremos mais sobre isso em arrays.

Observe também que para acessar um método em JavaScript colocamos o nome\_do\_objeto depois ponto final e depois o nome o método:

|  |  |  |
| --- | --- | --- |
| string01 | . | charAt(10) |
| nome\_do\_objeto | ponto final | nome do método |

### Tamanho da string

A propriedade **length** retorna tamanho da string, ou seja, o número de caracteres que a string possui. Esse método será muito útil em laços de repetição, quando iremos implementar um código para percorrer os caracteres de uma string.

Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 20 | // Tamanho de strings  console.clear();  let texto = 'abcdefehijklmnopqrstuvxwyz';  console.log(texto.length); |
| 21 |
| 22 |
| 23 |
| 24 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Observe que o comando **texto.length** retorna a quantidade de caracteres da string texto, que são 26 caracteres.

Esse método conta todos os caracteres da string, seja letras ou números, espaços em branco. Vejamos o exemplo:

1. Insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 26 | console.clear();  const texto2 = 'Também 123 [] é uma strings 231 -1\n';  console.log(texto2.length); |
| 27 |
| 28 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Olha que nesse exemplo ele contou todos os caracteres da string separadamente, mas ele conta o caractere de escape \n como apenas um caractere, que é o certo. Pode conferir 😊.

### Maiúsculas e minúsculas

Você pode transformar todos os caracteres de uma string para maiúsculo ou para minúsculo, isso facilitam, por exemplo, comparar strings para verificar se são iguais. O método **toUpperCase** transforma todos os caracteres de uma string para maiúsculos e o método **toLowerCase** transforma para minúsculo. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 30 | // Maiusculo e minúsculo  console.clear();  const s = 'Hello World!';  console.log(s.toUpperCase());  console.log(s.toLowerCase()); |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Observe que o comando s.toUpperCase() na linha 34 transformou todos os caracteres da string para maiúsculo, enquanto o comando s.toLowerCase() na linha 35 transformou todos os caracteres da string para minúsculo.

### substring e slipt

O método **substring** corta a string de acordo com os índices indicados entre os parênteses, por exemplo:

|  |
| --- |
| const str = 'Mozilla';  console.log(str.substring(1, 3));  // Saída esperada: "oz"  console.log(str.substring(2));  // Saída esperada: "zilla" |

O comando str.substring(1, 3) corta a string a partir do índice 1 até o índice 3 e o comando str.substring(2) corta a string a partir do índice 2 até o seu final.

O método **split** divide a string em substrings de acordo com a regra colocada entre parênteses. O objeto retornado por esse método é um array. Por exemplo:

|  |
| --- |
| const str1 = 'A raposa é um animal esperto';  const palavras = str1.split(' ');  console.log(palavras[3]);  // Saída esperada: "um"  const chars = str1.split('');  console.log(chars[7]);  // Saída esperada: "a"  const strCopy = str1.split();  console.log(strCopy);  // Saída esperada: Array ["A raposa é um animal esperto"] |

O comando **str1.split(' ')** [atente-se que existe um espaço em branco dentro das aspas simples] separou a string em **substring** de acordo com o espaço entre elas. Desse modo, o comando retorna um array com seis palavras separadas. Então, o comando **palavras[3]** acessa a palavra está no índice 3 que é **um**.

O comando **str1.split('')** [atente-se que não existe mais espaço em branco entre as aspas simples] separou a string em **substring** de caracteres, ou seja, colocou cada caractere em um índice do vetor. Desse modo, o comando retorna um array com vinte oito posições, que são os caracteres da string. Então, o comando **char[7]** acessa o caractere está no índice 7 que é **a**.

Por fim, o comando **str1.split()**, sem nenhuma regra faz uma cópia idêntica da string.

Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 37 | // Substring e spit  console.clear();  const str = 'Mozilla';  console.log(str.substring(1, 3));  console.log(str.substring(2));  const str1 = 'A raposa é um animal esperto';  const palavras = str1.split(' ');  console.log(palavras[3]);  // Saída esperada: "um"  const chars = str1.split('');  console.log(chars[7]);  // Saída esperada: "a"  const strCopy = str1.split();  console.log(strCopy);  // Saída esperada: Array ["A raposa é um animal esperto"] |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |
| 46 |
| 47 |
| 48 |
| 49 |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |
| 56 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

### Substituir uma string dentro da string

O método **replace** substitui um pedaço específico da string por outra coisa que você desejar. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 58 | // Substituir string  console.clear();  let mensagem = 'Venha para a Microsoft!';  console.log(mensagem);  let novaMensagem = mensagem.replace('Microsoft', 'IOS');  console.log(novaMensagem); |
| 59 |
| 60 |
| 61 |
| 62 |
| 63 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Podemos ver, nesse exemplo, que a palavra Microsoft foi substituída por IOS com o comando mensagem.replace('Microsoft', 'IOS').

### Remover espaços da string

O método **trim** remove os espaços os espaços em branco do início de o final de uma string. Isso é útil por exemplo, quando estamos lendo um texto digitado pelo usuário em um formulário e esse texto vem com espaços no início e/ou no final. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 65 | //Remover Espaços  console.clear();  let text = '       Olá Alunos do IOS!        ';  console.log(text.trim()); |
| 66 |
| 67 |
| 68 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Podemos ver, nesse exemplo, os espaços no início e no final da string foram retirados com o comando text.trim().

## Métodos de buscar em strings

Existem alguns métodos usados para buscar algo dentro da string, eles são úteis para programas que precisem encontrar algum valor ou padrão na sequência de caracteres enviada. Os métodos são:

* **String.indexOf()**: esse método retorna o índice (posição) da primeira vez que um texto especificado entre os parênteses foi encontrado na string.
* **String.lastIndexOf()**: esse método retorna o índice da última ocorrência de um texto especificado entre os parênteses foi encontrado na string.
* **String.search()**: esse método busca um valor específico e retorna a posição inicial desse valor na primeira vez que ele é encontrado. Semelhante ao indexOf().
* **String.startsWith()**: esse método verifica se a string inicia com um valor específico.
* **String.endsWith()**: esse método verifica se a string termina com um valor específico.

Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 70 | // Métodos de busca en strings  console.clear();  let frase = 'Sou um aluno do IOS e o IOS é muito bom!';  console.log(frase.indexOf('IOS')); // Returns 16  console.log(frase.lastIndexOf('IOS')); // Returns 24  console.log(frase.search('IOS')); // Returns 16  console.log(frase.startsWith('S')); // Verdadeiro - retorna true  console.log(frase.startsWith('Sou')); // Verdadeiro - retorna true  console.log(frase.startsWith('a')); // Falso - retorna false  console.log(frase.endsWith('!')); // Verdadeiro - retorna true  console.log(frase.endsWith('bom!')); // Verdadeiro - retorna true  console.log(frase.endsWith('x')); // Falso - retorna false |
| 71 |
| 72 |
| 73 |
| 74 |
| 75 |
| 76 |
| 77 |
| 78 |
| 79 |
| 80 |
| 81 |
| 82 |
| 83 |
| 84 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text

Description automatically generated

Podemos ver, o comando **frase.indexOf('IOS')** na linha 73 retorna o índice do início da string, onde ela aparece pela primeira vez. o comando **frase.lastIndexOf('IOS')** na linha 78 retorna o índice do início da string, onde ela aparece pela última vez. O comando **frase.search('IOS')** na linha 76 é semelhante ao indexOf e retorna 16 também. O comando **frase.startsWith('S')** na linha 78 verifica se a string começa com **S** e retorna verdadeiro. O comando **frase.startsWith('Sou')** na linha 79 verifica se a string começa com **Sou** e retorna verdadeiro. O comando **frase.startsWith('a')** na linha 80 verifica se a string começa com **a** e retorna falso. O comando **frase.endsWith('!')** na linha 82 verifica se a string termina com **!** e retorna verdadeiro. O comando **frase.endsWith('bom!')** na linha 83 verifica se a string termina com bom**!** e retorna verdadeiro. O comando **frase.endsWith('x')** na linha 84 verifica se a string termina com **x** e retorna falso.

## Conclusão

Existem muitos outros métodos para manipular strings e gastaríamos muito tempo se quisessem abordar todos esses métodos. À medida que outros métodos forem aparecendo, vamos explicando cada uma desses. O importante é que você sempre deve usar a internet para procurar soluções em programação, mas não apenas copie e cole uma solução, um bom programador procura entender a solução proposta e adaptá-la no seu projeto.

Para saber mais: Você pode consultar a lista de métodos de strings nos links: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String>

<https://www.w3schools.com/js/js_string_methods.asp>

<https://www.w3schools.com/js/js_string_search.asp>

Como desafio, tente entender e implementar alguns desses métodos.