MÓDULO III

Lógica de Programação utilizando a Linguagem JavaScript

A Educação é o primeiro passo para um futuro melhor...

MÓDULO III

O que é uma String?

Por definição, **strings** são sequências de caracteres **alfanuméricos** (*letras, números e/ou símbolos*) amplamente usadas em programação. Em **Javascript**, uma **string** sempre estará entre **aspas**.

```
const frase = "Mergulhando em tecnologia com Alura";
```

ou

```
const frase = ‘Mergulhando em tecnologia com Alura’;
```

ou ainda

```
console.log('Mergulhando em tecnologia com Alura')
```

MÓDULO III

Espere! Mas eu declaro minhas **strings** com **aspas duplas** ou **simples**?

Podemos colocar nossas **strings** entre aspas duplas ou simples. Para o **JavaScript**, não há diferença, já que ele considera as duas formas de declaração válidas. Mas, atenção, essa regra pode não se aplicar a outras linguagens. No **Java** ou **C#**, por exemplo, aspas simples são usadas para definir um caractere.

Em alguns momentos, a **string** poderá ser um texto que contém aspas. Nesses casos, é preciso combinar a utilização das aspas simples com aspas duplas e vice-versa, porque um texto como:
“Ela disse: “Adeus””, não funciona corretamente.

MÓDULO III

Vamos ao exemplo:

```
console.log('Ela disse: “Adeus!”')
```

ou

```
console.log(“Ela disse: 'Adeus!' ”)
```

MÓDULO III

É importante ressaltar que, depois que a sequência de caracteres for definida, a **string** é imutável, ou seja, não poderá ter seu valor alterado. Então, como manipular a **string**?

Sempre que manipulamos uma **string**, é criada uma nova instância dela por baixo dos panos, o que significa que será gerado um novo espaço na memória com uma cópia do valor da **string**.
Por isso, temos que utilizar uma variável para armazená-la.

MÓDULO III

Concatenando strings

Quando falamos em concatenar **strings**, quer dizer que vamos juntar duas ou mais **strings** e formar uma nova. Observe o exemplo abaixo:

```
let nome = "André"
let sobreNome ="Silva"
let nomeCompleto = "Meu nome completo é : " + nome + sobreNome
```

Para concatenar as **strings nome** e **sobreNome** com a **string** de texto que é o valor de **nomeCompleto**, usamos o operador de adição (+). Podemos usar também +=, como no próximo exemplo:

MÓDULO III

```
let nome = "André"
let saudacoes = "Seja bem-vindo "
saudacoes += nome
```

Dessa forma, temos a saída: **Seja bem-vindo André**

MÓDULO III

Interpolando strings (template strings)

A **interpolação de strings** é um recurso bem interessante, presente em diversas linguagens. No **JavaScript**, é uma alternativa mais prática para manipular **string** sem a necessidade de fazer concatenação, porque para textos maiores, concatenar pode ser um pouco trabalhoso.

Usando as chamadas *template strings* ou *templates literais*, a pessoa desenvolvedora consegue ter uma flexibilidade maior no trabalho com **strings**, além de facilitar a escrita e leitura do código.

Retomando o exemplo da mensagem de boas vindas, veja abaixo a utilização de *template strings*:

```
let nome = "André"
let saudacoes =`Seja bem-vindo ${nome}`
```

MÓDULO III

```
let nome = "André"
let poema = `
    E agora, ${nome}?
    A festa acabou,
    a luz apagou,
    o povo sumiu,
    a noite esfriou,
    e agora, ${nome}?
    e agora, você?
    você que é sem nome,
    que zomba dos outros,
    você que faz versos,
    que ama, protesta?
    e agora, ${nome}?
`
```

Veja como exemplo o poema “E agora, José?” de Carlos Drummond de Andrade:

Observe que, para a utilização da *template* ***string***, ela deve estar entre acentos graves (`) e, para fazer a interpolação, o valor ou variável deve estar dentro da estrutura **${valor}**. Vale ressaltar que usando *\*template strings\** temos a opção de utilizar a quebra de linha normalmente, sem caracteres de escape para isso, como **\n`**.

MÓDULO III

4 Formas de Concatenar Strings JavaScript

Concatenação de **strings** é basicamente uma forma de juntar dois elementos do tipo **texto** dentro da programação.

Na linguagem **JavaScript** podemos fazer isso de várias formas diferentes. E vou te mostrar através de exemplos práticos como você pode fazer isso.

Vale lembrar que não existe uma forma que seja muito mais performática que outra, no geral qualquer uma delas vão ter uma boa velocidade de execução.

Porém, você precisa estar atento a semântica de cada forma de concatenar **strings Javascript**, isso pode determinar a qualidade do seu código dependendo do projeto que esteja atuando.

MÓDULO III

Plus (+)

Primeiramente vamos utilizar a forma mais clássica de concatenar strings Javascript utilizando o sinal de + entre cada elemento do tipo string.
No exemplo abaixo estamos declarando uma constante chamada nome onde o valor dela é “Viking” logo em seguida usamos o sinal de + para juntar as duas strings:

```
const nome = "Viking";

alert( 'Programador ' + nome );
```

O resultado do script acima deve apresentar um alerta com o conteúdo: **‘Programador Viking’**.

MÓDULO III

Método Join()

Em seguida vamos aprender uma outra forma de concatenar strings javascript utilizando um método chamado **.join()**.

Para isso precisamos informar os elementos dentro de colchetes e em seguida utilizar o método **join()** para juntar esses dois elementos.

O código abaixo mostra como usar o método **join()** para concatenar strings javascript.

```
const nome = "Viking";

alert( ['Programador', nome].join(' ') );
```

Repare que dentro do método join tem um espaço em branco, é esse espaço que irá separar os dois elementos. Caso você informe outro caracter ao invés do espaço ele será separado pelo caracter informado

MÓDULO III

Método concat()

Agora vamos aprender uma outra forma usando um método chamado **concat()**.
Como o próprio nome sugere ele irá concatenar elementos do tipo string.

Para utilizar esse método é necessário chamá-lo a partir de uma string, portanto no exemplo abaixo iremos informar uma string vazia apenas para conseguir usar esse método.

Em seguida, basta chamar o método **concat()** informando os elementos do tipo string que será concatenado.

```
const nome = "Viking";

alert( ''.concat('Programador ', nome) );
```

MÓDULO III

Template String

Por fim, vamos aprender a utilizar uma técnica chamada template string que funciona da seguinte forma.

Vamos usar o **acento grave** ao invés de aspas para definir a nossa string que será:

`Programador nome`

Se executarmos o programa dessa forma o Javascript não vai entender que ele deve pegar o conteúdo da constante nome.

Para isso vamos cercar a palavra nome com o caracter de chaves e antes da primeira chave vamos colocar o sinal de **dólar($)**.

Dessa forma o Javascript vai entender que o que deve ser considerado é o conteúdo que está dentro da nossa constante conforme exemplo abaixo:

```
const nome = "Viking";

alert( `Programador ${nome}` );
```

MÓDULO III

🎯 Atividade 1: Meu Cartão de Apresentação

Objetivo: Criar uma frase apresentando **nome**, **idade** e **hobby** usando concatenação.

Instruções:

```
// Altere os valores abaixo
let nome = "Carlos";
let idade = 14;
let hobby = "tocar violão";

// Crie uma frase de apresentação
let apresentacao = "Oi! Meu nome é " + nome + ", tenho " + idade + " anos e gosto de " + hobby + ".";
console.log(apresentacao);
```

MÓDULO III

🕹️ Atividade 2: Frases de Ação

Objetivo: Montar frases engraçadas combinando ações com personagens e lugares.

Instruções:

```
let personagem = "Mario";
let acao = "correu";
let lugar = "no castelo";

// Monte a frase
let frase = `${personagem} ${acao} ${lugar}!`;
console.log(frase);
```

MÓDULO III

🍕 Atividade 3: Pedido na Lanchonete

Objetivo: Simular um pedido de lanche com nome do cliente e o que ele pediu.
Instruções:

```
let cliente = "Beatriz";
let pedido = "uma pizza de calabresa e um suco de laranja";

// Crie a mensagem
let mensagem = "Pedido de " + cliente + ": " + pedido + ".";
console.log(mensagem);
```

📌 *Desafio extra:*
Peça aos alunos que adicionem o número da mesa ou tempo de entrega.

MÓDULO III

🎬 Atividade 4: Título de Filme Inventado

Objetivo: Criar títulos engraçados de filmes com palavras aleatórias.

Instruções:

```
let palavra1 = "O mistério";
let palavra2 = "do gato";
let palavra3 = "voador";

let titulo = palavra1 + " " + palavra2 + " " + palavra3;
console.log("Título do filme: " + titulo);
```

MÓDULO III

💬 Atividade 5: Mensagem para um amigo

Objetivo: Escrever uma mensagem personalizada.

Instruções:

```
let amigo = "Léo";
let emoji = "😄";
let mensagem = `Oi, ${amigo}! Tudo bem? Espero que sim! ${emoji}`;


console.log(mensagem);
```

MÓDULO III

Métodos para strings

Antes de começarmos, é importante ressaltar que o **JavaScript** diferencia strings como tipos primitivos (*com aspas duplas ou simples*) de objetos Strings (*quando usamos a palavra reservada new*). Mas, por baixo dos panos toda string, mesmo as que criamos com a chamada "**forma literal**", por exemplo const texto = "**Alura**", acaba convertida para um objeto do tipo **String**.
Por isso, temos acesso a uma série de métodos e propriedades deste objeto.

Agora que entendemos isso, vamos ver algumas propriedades e métodos úteis e bem práticos para trabalhar com strings em nossas aplicações.

MÓDULO III

.length

A propriedade **length** serve para nos informar o tamanho de uma *string*. E por que isso é útil? Caso sua aplicação tenha como uma das regras para criação de senhas (*que em geral são alfanuméricas*) o tamanho de **8** caracteres, usar **length** será uma boa opção, pois ajudará a contar a quantidade de caracteres da **string**.

Para testar a propriedade **length**, vamos usar a **string** alura, que retornará o tamanho **5**.

```
const palavra="alura";
console.log(palavra.length) //5
```

Veja que **length** é exatamente a mesma propriedade que acessamos quando queremos descobrir o comprimento (*ou seja, a quantidade de elementos*) em um **array**.

MÓDULO III

charAt()

Com o método **charAt()** conseguimos acessar um caractere de uma **string**. Lembre-se que, por baixo dos panos, **strings** são **arrays** de caracteres, e em cada posição temos o caractere que compõe a **string**.

Veja o exemplo abaixo:

```
console.log("alura".charAt(3)) //r
```

Após a execução do método **charAt()**, ela retornará o caractere **r**, que é o valor que consta na posição **3** da **string** - lembrando que **arrays** em **JavaScript** começam na posição 0 (**zero**).

MÓDULO III

Outra alternativa será usar a notação de **colchetes** para encontrar um caractere da **string**, da seguinte forma:

```
const palavra="Alura"

console.log(palavra[0]) //A
```

Será exibido o caractere **A**, ou seja, o que está na primeira posição da **string**. O resultado da execução do **charAt()** é uma **string**.

Mas e se quisermos saber qual a posição de um caractere dentro da **string**?

MÓDULO III

indexOf()

Respondendo a pergunta anterior, existe a função **indexOf()**, que retorna a posição de um caractere dentro da **string**.

Por exemplo:

```
const palavra="Alura"

console.log(palavra.indexOf("a")) //4
```

O resultado é a posição **4**.

MÓDULO III

Porém, na utilização do **indexOf()**, fique atento caso o caractere que se busca na **string** seja encontrado em mais de uma posição, pois será retornada somente a primeira ocorrência.

Veja no código:

```
const palavra="Divertidamente"

console.log(palavra.indexOf("e")) //3
```

O resultado da execução do **indexOf()** é um valor numérico.

MÓDULO III

toUpperCase() e toLowerCase()

São duas funções bastante utilizadas quando estamos trabalhando com string e precisamos deixar o texto todo em letras **minúsculas** (***lower case***) ou todo em **maiúsculas** (***upper case***).

Vamos ver o código abaixo:

```
const palavra="alura";
console.log(palavra.toUpperCase()) //ALURA
console.log(palavra.toLowerCase()) //alura
```

Após a execução do código, o console irá exibir **ALURA** e **alura** respectivamente.
O resultado da execução dos métodos **toUpperCase()** e **toLowerCase()** é uma nova **string** .

MÓDULO III

substr()

Outra função muito interessante é a **substr()** (***substring***), que permite que façamos a extração de parte de uma **string**, conforme o código abaixo:

```
let frase= "Mergulhando em tecnologia."
console.log(frase.substr(0,11)) // Mergulhando
```

A função recebe como parâmetro o início e o fim da nova **string** a ser retirada da **string** principal.
Na execução do código acima, temos como resultado a palavra **Mergulhando**.
Bem útil, né?
O resultado da execução do método **substr()** é uma nova **string** .

MÓDULO III

slice()

Podemos utilizar também o método **slice()**, que usamos com **arrays**. Ele é similar ao **substring()** e retornará parte de uma **string**, desde que passemos nos parâmetros o índice de **início** e de **fim**. Veja abaixo:

```
let frase= "Mergulhando em tecnologia."
console.log(frase.slice(0,11)) // Mergulhando
```

O resultado da execução do método **slice()** é uma nova **string** .

MÓDULO III

replace()

Com a função **replace()** temos a possibilidade de substituir parte de uma **string** por outra. Essa função recebe como parâmetros duas informações: a **string** que você quer substituir e a **string** que será colocada no lugar. Olhe o exemplo abaixo, em que precisamos substituir a **string** *nomeusuario* no texto padrão de *comunicacao*.

```
let nome = "André";
let comunicacao = " Olá, sr. nomeusurario, o senhor tem 50% de desconto.";
console.log(comunicacao.replace("nomeusurario",nome));
```

Na execução deste exemplo, a **string** *nomeusuario* será substituída pelo conteúdo da variável nome. Como resultado da execução do método **replace()** teremos uma nova **string**.

MÓDULO III

concat()

O método **concat()** é uma opção para concatenar **strings** sem a utilização do operador de adição (+). Ele concatena duas **strings**, adicionando a nova **string** ao fim da anterior.

Observe uma utilização do **concat()**:

```
let novaString = "Programe com linguagens. Explore linguagens como ";
console.log(novaString.concat("JavaScript,").concat(" Python,").concat(" e C#."))
```

O resultado obtido será: Programe com linguagens. Explore linguagens como JavaScript, Python, e C#.

Para a execução do método **replace()** teremos como resultado uma nova **string**.

MÓDULO III

split()

O método **split()** é bem interessante, pois com ele conseguimos quebrar uma **string** com base em caracteres separadores que vamos informar para o método como parâmetro.

Vamos ver um exemplo:

```
let linguagens = "JavaScript;Java;C#;PHP;Python;Go;Vb;SQL;C;C++";
let arrayLinguagens = linguagens.split(";");
console.log(arrayLinguagens)
```

Quando trabalhamos com o **split()**, devemos nos atentar, pois a execução gerará como resultado um array de **strings** com os elementos que foram separados com base no separador desejado.

MÓDULO III

Portanto a execução do código resulta em um array como mostrado a seguir:

```
[ 'JavaScript',
  'Java',
  'C#',
  'PHP',
  'Python',
  'Go',
  'Vb',
  'SQL',
  'C',
  'C++' ]
```

Lembre-se que o resultado da execução do método **split()** é um array de **strings**.

MÓDULO III

trim()

O **trim()** remove os espaços em branco **no início ou fim** de uma **string**. Se em alguma situação precisarmos fazer uma verificação de que o usuário não digitou o login com espaços, faremos;

```
let login = "    andre@emailteste.com      ";
let loginSemEspaco = login.trim();
console.log(loginSemEspaco); //andre@emailteste.com
```

A variável **loginSemEspaco** conterá uma nova **string**, sem os espaços em branco no início ou fim que podem ter sido digitados. Então, quando executado o método **trim()**, o resultado é uma nova **string**.

No **JavaScript** ainda temos algumas variações desta função como: **trimEnd(),trimStart(),trimLeft()** e **trimRight()**, teste também estas variantes e veja o resultado.

MÓDULO III

Exemplos de **Métodos** de **String** em **JavaScript**

1. **length**

```
let nomeUsuario = "CodigoNinja2024";
let caracteresNome = nomeUsuario.length;
console.log("Seu nome de usuário tem " + caracteresNome + " caracteres.");
```

MÓDULO III

2. toUpperCase() e toLowerCase()

```
let mensagemSecreta = "sEcReTo123";
let mensagemPadronizada = mensagemSecreta.toLowerCase();
console.log("Mensagem padronizada: " + mensagemPadronizada);
```

MÓDULO III

3. **slice**(início, fim)

```
let nomeCompleto = "Harry James Potter";
let primeiroNome = nomeCompleto.slice(0, 5); // "Harry"
let ultimoNome = nomeCompleto.slice(11); // "Potter"
console.log("Primeiro nome: " + primeiroNome);
console.log("Último nome: " + ultimoNome);
```

MÓDULO III

4. **substring**(inicio, fim)

```
let codigoDoJogo = "XYZ-12345-ABC";
let nivel = codigoDoJogo.substring(4, 9); // "12345"
console.log("Nível do jogo: " + nivel);
```

MÓDULO III

5. **substr**(inicio, comprimento)

```
let placaCarro = "ABC-1234";
let letras = placaCarro.substr(0, 3); // "ABC"
let numeros = placaCarro.substr(4, 4); // "1234"
console.log("Letras da placa: " + letras);
console.log("Números da placa: " + numeros);
```

MÓDULO III

6. **replace**(valorAProcurar, novoValor)

```
let fraseChat = "Eu odeio segunda-feira!";
let fraseAmigavel = fraseChat.replace("odeio", "adoro");
console.log("Frase amigável: " + fraseAmigavel);
```

MÓDULO III

7. **replaceAll**(valorAProcurar, novoValor)

```
let textoRepetitivo = "A aula é legal. A aula é divertida. A aula é importante.
let textoMelhorado = textoRepetitivo.replaceAll("A aula é", "JavaScript é");
console.log("Texto melhorado: " + textoMelhorado);
```

MÓDULO III

8. **concat**(string1, string2, ...)

```
let cumprimento = "E aí, ";
let nome = "Galera!";
let mensagem = cumprimento.concat(nome);
console.log(mensagem); // "E aí, Galera!"
```

MÓDULO III

9. **trim(), trimStart(), trimEnd()**

```
let nomeComEspacos = "   Jogador123   ";
let nomeLimpo = nomeComEspacos.trim();
console.log("Nome limpo: " + nomeLimpo);
```

MÓDULO III

10. **padStart**(comprimento, preenchimento) e **padEnd**(comprimento, preenchimento)

```
let numeroFase = "7";
let numeroFormatado = numeroFase.padStart(3, "0"); // "007"
console.log("Fase: " + numeroFormatado);
```

MÓDULO III

EXERCÍCIOS...

MÓDULO III

FIM...

A Educação é o primeiro passo para um futuro melhor...