Trabalhando com Strings

As strings são fundamentais no desenvolvimento dinâmico com JavaScript, pois representam sequências de caracteres usadas para armazenar e manipular textos. Nesta aula, vamos entender como declarar e trabalhar com strings, utilizando os principais métodos disponíveis na linguagem. Você aprenderá a realizar operações essenciais, como extração de partes de uma string, substituição de valores, junção de textos, e conversão entre letras maiúsculas e minúsculas. Com isso, será possível manipular de forma eficaz os dados textuais em seus projetos, tornando suas aplicações mais robustas e interativas.

Tipo de Dado String

No JavaScript, a string é um dos tipos de dados primitivos, utilizado para representar uma sequência de caracteres. Esses caracteres podem ser letras, números, símbolos e até mesmo espaços em branco. As strings são utilizadas em quase todas as áreas da programação, desde a exibição de mensagens ao usuário até a manipulação de grandes volumes de dados textuais.

Uma string pode ser definida de várias maneiras, utilizando aspas simples (') ou aspas duplas ("). Por exemplo:

javascript

```
let saudacao = “Olá, mundo!”;

let nome = ‘João’;
```

Aqui, `"Olá, mundo!"` e `'João'` são strings válidas no JavaScript. Ao programar, você pode usar tanto aspas simples quanto duplas, mas é importante manter a consistência no estilo escolhido. Por exemplo, misturar aspas simples e duplas pode resultar em erro:

javascript

let fraseIncorreta = ‘Olá, “mundo”!’; // Correto

let fraseErro = ‘Olá, "mundo!’; // Erro: aspas simples abertas, mas sem fechar.

As strings podem ser usadas para representar textos de todos os tipos: desde palavras simples, como em mensagens de erro ou de boas-vindas, até parágrafos inteiros, como descrições de produtos em e-commerces.

Além disso, strings podem conter caracteres especiais, como números e símbolos. Uma senha, por exemplo, pode ser uma string que mistura letras, números e caracteres especiais:

javascript

let senha = “Abc123$#”;

Strings são essenciais para interações com o usuário, seja para exibir dados ou processar entradas. Ao trabalhar com strings, você pode combinar dados textuais e realizar diversas operações para manipulá-las, como veremos adiante.

Um ponto importante é que uma string é um tipo imutável em JavaScript. Isso significa que, uma vez criada, ela não pode ser alterada. Se você

modificar uma string, o JavaScript criará uma nova string em vez de alterar a original.

Iniciando com Strings

Ao começar a trabalhar com strings, o primeiro passo é compreender como declará-las e usá-las corretamente. Como já mencionado, uma string deve estar entre aspas simples (`'`) ou aspas duplas (`"`). Se omitirmos as aspas, o JavaScript não reconhecerá o conteúdo como uma string e apresentará um erro.

Por exemplo:

javascript

let saudacao = “Olá, mundo!”;

let fraseIncorreta = Olá, mundo!; // Erro: Não há aspas.

O exemplo acima resultará em um erro, pois o JavaScript não entenderá que `Olá, mundo!` é uma string.

Outro conceito importante ao trabalhar com strings é a concatenação. A concatenação permite unir duas ou mais strings em uma única string. Em JavaScript, a concatenação pode ser feita de duas maneiras principais:

1. Usando o operador +:

javascript

let saudacao = “Olá”;

let nome = “Maria”;

```
let frase = saudacao + ", " + nome + "!";

console.log(frase);  // "Olá, Maria!"
```

Aqui, usamos o operador de adição (+) para concatenar as strings "Olá", ", ", "Maria" e "!".

2. Utilizando templates literais com crase (``):

```
javascript

let saudacao = "Olá";

let nome = "Maria";

let frase = ${saudacao}, ${nome}!;

console.log(frase);  // "Olá, Maria!"
```

O uso de templates literais facilita a concatenação, pois permite a inserção de variáveis diretamente no texto, sem a necessidade de usar o operador +.

Além disso, as strings possuem a propriedade `length`, que retorna o número de caracteres da string. Isso inclui letras, números, símbolos e até mesmo espaços em branco:

```
javascript

let nome = "Maria";

console.log(nome.length);  // 5
```

A propriedade `length` é muito útil quando você precisa saber o comprimento de uma string, especialmente ao validar entradas de usuários em formulários, onde há limites de caracteres.

Operações com String: Slice, Trim e Split

O JavaScript oferece uma série de métodos que permitem manipular strings de diferentes formas. Três dos métodos mais comuns são `slice`, `trim` e `split`. Vamos explorar cada um deles com mais detalhes:

Slice: O método `slice` é utilizado para extrair uma parte de uma string, retornando uma nova string com o conteúdo selecionado, sem modificar a string original. Ele recebe dois parâmetros: o índice de início e o índice de fim. Vale lembrar que os índices das strings começam em 0.

Exemplo:

javascript

```
let texto = "JavaScript é incrível";

let parte = texto.slice(0, 10);  // Extrai os primeiros 10 caracteres

console.log(parte);  // "JavaScript"
```

Nesse caso, o `slice(0, 10)` extrai a string que vai do índice 0 até o índice 9, ou seja, os primeiros 10 caracteres de "JavaScript é incrível". Uma observação importante é que, no JavaScript, a string original não é alterada. O método `slice` sempre retorna uma nova string.

Trim: O método `trim` remove os espaços em branco do início e do fim de uma string. Isso é muito útil em aplicações onde os usuários inserem dados, como em formulários, onde os espaços acidentais podem prejudicar a validação dos dados.

Exemplo:

```
javascript

let textoComEspacos = "  Olá, mundo!  ";

let textoLimpo = textoComEspacos.trim();

console.log(textoLimpo);  // "Olá, mundo!"
```

No exemplo acima, os espaços em branco antes e depois da string "Olá, mundo!" são removidos, tornando o dado mais adequado para uso.

Split: O método `split` divide uma string em um array de substrings com base em um delimitador especificado. Esse método é útil quando você precisa quebrar uma string em partes, como quando está processando listas de itens separados por vírgulas ou espaços.

Exemplo:

```
javascript

let frutas = "maçã, banana, laranja";

let listaFrutas = frutas.split(", ");

console.log(listaFrutas);  // ["maçã", "banana", "laranja"]
```

Aqui, a string “maçã, banana, laranja” é dividida em um array, utilizando a vírgula seguida de um espaço como delimitador. O método `split` é particularmente útil ao processar grandes blocos de texto ou ao quebrar dados de entrada em partes utilizáveis.

Operações com String: Substring, Replace e Concat

Agora vamos examinar mais alguns métodos úteis para manipulação de strings: `substring`, `replace` e `concat`.

Substring: O método `substring` é semelhante ao `slice`, mas tem uma diferença: ele não aceita índices negativos e lida de forma mais previsível quando o índice inicial é maior que o final. Ele permite extrair uma parte de uma string.

Exemplo:

```
javascript

let texto = “JavaScript”;

let sub = texto.substring(0, 4);

console.log(sub);  // “Java”
```

Aqui, `substring(0, 4)` retorna os primeiros quatro caracteres da string “JavaScript”.

Replace: O método `replace` substitui um valor específico dentro de uma string por outro valor. Ele é extremamente útil quando você precisa corrigir ou alterar textos.

Exemplo:

javascript

```
let frase = “Olá, mundo!”;

let novaFrase = frase.replace(“mundo”, “JavaScript”);

console.log(novaFrase);  // “Olá, JavaScript!”
```

Nesse exemplo, a palavra “mundo” foi substituída por “JavaScript”. O método `replace` é muito usado para fazer ajustes em textos ou transformar strings antes de exibi-las.

Concat: O método `concat` permite concatenar (juntar) duas ou mais strings em uma única string. Embora seja possível concatenar strings usando o operador +, `concat` pode ser uma alternativa mais clara.

Exemplo:

javascript

```
let saudacao = “Olá”;

let nome = “Maria”;

let frase = saudacao.concat(”, '', nome, “!”);

console.log(frase);  // “Olá, Maria!”
```

No exemplo acima, `concat` junta as strings “Olá”, '', '', “Maria” e “!” em uma única string.

Operações com String: toLowerCase e toUpperCase

Os métodos `toLowerCase` e `toUpperCase` são usados para converter todos os caracteres alfabéticos de uma string para minúsculas ou maiúsculas, respectivamente. Esses métodos são essenciais para padronizar entradas de usuários e para exibir textos de maneira consistente.

toLowerCase: Este método converte todos os caracteres alfabéticos de uma string para minúsculas. É útil, por exemplo, quando queremos garantir que as comparações de strings não sejam sensíveis a maiúsculas ou minúsculas.

Exemplo:

```
javascript

let texto = "JAVASCRIPT";

let minusculo = texto.toLowerCase();

console.log(minusculo);  // "javascript"
```

Ao aplicar o método `toLowerCase`, a string "JAVASCRIPT" é convertida para "javascript".

toUpperCase: Ao contrário do `toLowerCase`, o método `toUpperCase` converte todos os caracteres alfabéticos para maiúsculas.

Exemplo:

```
javascript

let texto = "javascript";

let maiusculo = texto.toUpperCase();
```

```
console.log(maiusculo);  // “JAVASCRIPT”
```

Esse método é muito útil quando se quer destacar um texto, como títulos ou cabeçalhos, ou padronizar a entrada de dados.

Esses métodos são frequentemente usados em formulários de entrada de dados, como quando um site exige que um campo seja preenchido com letras maiúsculas, ou para garantir que um nome ou e-mail seja comparado corretamente, independentemente de como o usuário o digitou.

Conteúdo Bônus

Esse curso gratuito no YouTube, ministrado pelo professor Gustavo Guanabara, é uma excelente introdução à linguagem JavaScript, com aulas que abordam o conceito e a manipulação de strings. O conteúdo é didático e prático.

Título: Curso de HTML5 Completo e GRÁTIS

Canal: Curso em Vídeo

Plataforma: Youtube

Referência Bibliográfica

DEITEL, P. J.; DEITEL, H. M. Ajax, rich internet applications e desenvolvimento web para programadores. Pearson: 2008

FELIX, R. (Org.). Programação orientada a objetos. Pearson: 2016

FERREIRA, R. D. Linguagem de programação. Contentus: 2020.

FLATSCHART, F.; BACHINI, C.; CUSIN, C. Open Web Platform. Brasport: 2013.

NEVES, M. C. B. de A. Sites de Alta Performance. Contentus: 2020

PAGE-JONES, M. Fundamentos do desenho orientado a objeto com UML. Pearson: 2001.

PUGA, S.; RISSETTI, G. Lógica de programação e estruturas de dados, com aplicações em Java. Pearson: 2016.

SEGURADO, V. S. (Org.). Projeto de interface com o usuário. Pearson: 2016.

Atividade Prática 5 – Trabalhando com Strings

Título da Prática: Fixando conhecimento sobre strings

Objetivos: Compreender definições básicas para o uso de strings em JavaScript

Materiais, Métodos e Ferramentas: Computador com uma IDE instalada (recomendado: VS Code) e Javascript instalado via node/npm e o entendimento do contexto abaixo

Atividade Prática

Uma string é um tipo de dado utilizado para representar uma sequência de caracteres. Elas são muito utilizadas no desenvolvimento de aplicações, pois permitem armazenar e manipular textos, podendo conter letras, números, símbolos e até espaços em branco. Para declarar uma string em JavaScript, podemos usar aspas simples (') ou aspas duplas ("). Além disso, strings em JavaScript podem ser manipuladas de várias maneiras.

A concatenação, por exemplo, permite combinar duas ou mais strings em uma única. Para isso, utiliza-se o operador '+'. O comprimento de uma string pode ser obtido pela propriedade 'length', que retorna o número de caracteres da string.

JavaScript também oferece diversos métodos para trabalhar com strings. O método 'slice()' é usado para retornar uma parte da string, conforme os índices fornecidos. O método 'trim()' remove espaços em branco do início e do fim de uma string, enquanto o 'split()' divide a string em um array de substrings, com base em um separador especificado. Já o 'substring()' retorna uma parte da string, com base em dois índices, e o 'replace()' substitui uma substring por outra. Para alterar o caso dos caracteres, temos o 'toLowerCase()' e o 'toUpperCase()', que convertem todos os caracteres alfabéticos da string para minúsculas e maiúsculas, respectivamente.

Esses métodos tornam a manipulação de strings uma tarefa eficiente, permitindo realizar operações como formatação, busca e substituição de texto de forma ágil e eficaz em seus programas.

De acordo com o texto acima e a aula, responda as questões abaixo:

1. Qual é o operador usado para concatenar duas ou mais strings em JavaScript?

a. -

b. *

c. +

d. /

2. Qual método é utilizado para remover os espaços em branco do início e do final de uma string em JavaScript?

a. ‘trim()’

b. ‘slice()’

c. ‘substring()’

d. ‘split()’

3. O que o método ‘split()’ faz em uma string?

a. Divide a string em um array de substrings com base em um separador.

b. Retorna uma parte da string a partir de um índice inicial.

c. Substitui uma substring por outra.

d. Converte todos os caracteres da string para maiúsculas.

4. Qual é a função do método ‘toUpperCase()’ em JavaScript?

a. Converte todos os caracteres alfabéticos da string para minúsculas.

b. Converte todos os caracteres alfabéticos da string para maiúsculas.

c. Retorna uma parte da string a partir de um índice inicial.

d. Substitui uma substring por outra.

Gabarito Atividade Prática

1. c.+

Comentário: O operador + é utilizado para concatenar duas ou mais strings, formando uma única string.

2. a. ‘trim()’

Comentário: O método ‘trim()’ remove os espaços em branco no início e no final de uma string, sem afetar os espaços internos.

3. a. Divide a string em um array de substrings com base em um separador.

Comentário: O método ‘split()’ divide a string em um array de substrings, utilizando um separador especificado.

4. b. Converte todos os caracteres alfabéticos da string para maiúsculas.

Comentário: O método ‘toUpperCase()’ converte todos os caracteres alfabéticos de uma string para maiúsculas, sem afetar os caracteres não alfabéticos.

Ir para exercício