Manipulação de Strings em Javascript

Durante o processo de coleta de dados dentro de um site, existe a necessidade de tratar os dados fornecidos pelos usuários, e nesse pequeno módulo iremos ver um pouco sobre como tratar os dados digitados pelos usuários antes de, por exemplo, cadastrar esses dados em um banco de dados. Vamos começar com o tratamento de textos.

Para podermos exemplificar, vamos considerar a seguinte variável contendo a seguinte string:

```
let mensagem = 'Eu gosto de estudar Javascript'
```

1. Replace() => O método replace() recebe dois parâmetros (x,y) , a função dele é de realizar uma troca, ele procura dentro da string original a ocorrência da substring 'x' e quando encontra substitui pela substring 'y'.

A sua implementação pode ser feita da seguinte forma

```
mensagem.replace('Javascript', 'PHP')
"Eu gosto de estudar PHP"
```

Neste caso acabamos de substituir a substring 'Javascript' pela substring 'PHP'

Mas vale lembrar que dessa forma, a string original ainda permanece como antes, se formos a imprimir o valor da variável 'mensagem' ela ainda continua sem a alteração do 'Javascript' para 'PHP', veja abaixo :

```
mensagem
"Eu gosto de estudar Javascript"
```

Isso se deve ao facto de não termos atribuído a nova alteração da string para a variável 'mensagem', para que a mudança seja definitiva devemos reescrever a variável 'mensagem' passando para ele o novo valor alterado, da seguinte forma:

```
mensagem = mensagem.replace('Javascript', 'PHP')
```

Dessa forma o conteúdo da variável mensagem passa a ter a alteração da substring 'Javascript' para a nova substring 'PHP'

Podemos agora imprimir o novo valor da variável mensagem e veremos que a alteração foi feita, veja:

```
mensagem
"Eu gosto de estudar PHP"
```

Antes de continuar deixa-me só explicar um pouco o que é uma subtring nesse caso.

Imagina por exemplo que você tem um conjunto chamado 'frutas' , os subconjuntos desse conjunto 'frutas' são todas as frutas e todos os outros pequenos e grandes agrupamentos de frutas que está dentro dele. Por exemplo [Manga] , [Banana] , [Manga, Laranja] e por ai vai.

Dessa forma uma substring nesse contexto representa uma pequena parte que faz parte da string mãe. Fazendo uma analogia, O conjunto seria a string e o subconjunto seriam todas as substrings.

Quando a substring não é encontrada dentro da string principal o conteúdo da string permanece inalterado, por exemplo, vamos tentar substituir a substring 'Ruby' para a nova substring 'C#' sendo que 'Ruby' não está presente dentro da nossa string principal.

```
mensagem.replace('Ruby', 'C#')
"Eu gosto de estudar PHP"
```

A string permaneceu do jeito que nós deixamos, isso por que a substring 'Ruby' não foi encontrada para poder se realizara a substituição.

Se tivermos, por exemplo, uma string onde uma substring ocorre duas ou mais vezes, o **replace()** irá substituir apenas na primeira ocorrência.

Para que seja realizada a substituição em todas as ocorrências da substring encontrada devemos usar a função **replaceAll()**.

2. length => O length serve para poder obter o tamanho da string, ou seja, a quantidade de caracteres presentes dentro da string.

Vamos obter o tamanho da nossa string dentro da variável mensagem

```
mensagem.length
23
```

Nesse caso temos 23 caracteres dentro da nossa string, vale lembrar que os 'espaços em brancos' também contam como string.

Devemos prestar atenção em um detalhe, o .length não é uma função, por isso ele não pode receber parâmetros e por esse motivo também é dispensado o uso do '()' no fim da instrução.

3. toLowerCase() => A função toLowerCase() serve para poder colocar toda a string em minúsculas. A sua implementação é bastante simples, veja:

```
mensagem = mensagem.toLowerCase()
```

Dessa forma a string dentro da variável mensagem passa a ser toda ela em minúscula, vamos agora ver o resultado quando imprimimos a variável 'mensagem' :

```
mensagem
"eu gosto de estudar php"
```

Podemos ver agora que a palavra 'php' antes estava toda ela em Maiúscula, mas após o toLowerCase() ela também se tornou minúscula.

4. toUpperCase() => essa função tem um efeito contrário da função toLowerCase(), ela serve para poder colocar toda a string em maiúscula.

```
mensagem = mensagem.toUpperCase()
```

Dessa forma a nossa string passa a ser toda ela em maiúscula, podemos ver o seu resultado:

```
mensagem
"EU GOSTO DE ESTUDAR PHP"
```

5. match() => a função match() serve de forma mais básica para procurar a ocorrência de uma determinada substring dentro de uma string mãe, caso ela ache a ocorrência ela irá retornar um array contendo os dados que ela encontrou, caso ela não encontre então ela irá retornar null. Antes de continuarmos, vamos colocar o conteúdo da nossa mensagem toda ela em minúscula.

```
mensagem = mensagem.toLowerCase()
"eu gosto de estudar php"
```

Pronto, agora vamos achar a substring ‘estudar’ dentro da string

```
mensagem.match(/estudar/)
Array [ "estudar" ]
```

A string dentro do match() deve ser passada da mesma forma que passamos parâmetros, mas com a diferença de colocarmos ele dentro de //, e vale lembrar que antes de colocar a string dentro de // tudo o que estiver na mesma linha depois do // é considerado um comentário.

Agora vamos tentar procurar a palavra ‘LISP’ dentro da nossa string

```
mensagem.match(/LISP/)
null
```

Podemos ver que tivemos um ***null*** como retorno, isso por que a substring ‘LISP’ não foi encontrada dentro da nossa string principal.

6. substr() => Essa função retorna a string principal mas com algumas alterações de acordo com os parâmetros que passamos para a função substr().
A função precisa receber no mínimo um parâmetro, esses parâmetros são índices, esses índices começam de zero. Por exemplo, dentro da palavra ‘JAVASCRIPT’ cada um dos caracteres tem o seu respectivo índice, da seguinte forma:

[0] = J
[1] = A
[2] = v
[3] = A
[4] = S
[5] = C
[6] = R
[7] = I
[8] = P
[9] = T

Dessa forma dentro da função substr() podemos passar um parâmetro que ira indicar a partir de onde que ele vai pegar o conteúdo da string, veja essa implementação abaixo.

Lembrado que o conteúdo da variável que estamos trabalhando agora é essa:

```
mensagem
"eu gosto de estudar php"
```

Vamos agora alterar ela através do substr() :

```
mensagem = mensagem.substr(6)
```

Dessa forma estamos começando a pegar o conteúdo da string apartir da posição 6, vamos agora chamar a mensagem para ver o seu novo conteúdo

```
mensagem
"to de estudar php"
```

Podemos ver que agora o conteúdo da variável começa apartir da posição 6 até o fim da string.

Mas também podemos determinar qual deve ser o final da nova string passando um segundo parâmetro para o substr(), ou seja, passando dois parâmetros (x,y), o 'x' indica o inicio da nova string e o 'y' indica o fim da nova string. Vamos ver isso na prática, mas antes vamos devolver o conteúdo original da nossa string.

```
mensagem = 'Eu gosto de estudar Javascript'
"Eu gosto de estudar Javascript"
```

Agora sim, vamos alterar o conteúdo na nossa string passando os dois parâmetros (x,y)

```
mensagem = mensagem.substr(9, 16)
```

Dessa forma a nova string vai começar da posição 9 até a posição 16 da string original. Vamos agora ver o seu resultado :

```
mensagem
"de estudar Javas"
```

Vale lembra que o valor do índice passado não deve ser superior à quantidade de caracteres da string original, também não pode ser igual ao tamanho da string total por que a contagem do **.length** começa de 1 enquanto que a contagem dos índices começa de 0.

7. search() => De forma mais básica, essa função tem um Objectivo bastante simples, ele retorna a posição do início da ocorrência de uma determinada substring dentro de uma string mãe.

Antes de usarmos essa função, temos que novamente reescrever o conteúdo da nossa variável mensagem para o original.

```
mensagem = 'Eu gosto de estudar Javascript'
"Eu gosto de estudar Javascript"
```

Pronto, agora vamos descobri a partir de onde começa a ocorrência da substring 'gosto' :

```
mensagem.search('gosto')
3
```

Podemos ver que a substring 'gosto' começa a partir da posição 3.

8. toString() => Essa função tem um objectivo bastante simples, tornar um dado numérico em uma string.

Para isso vamos criar uma nova variável chamada codigo e vamos passar para ela o valor 652 :

```
let codigo = 652
```

Vamos supor que por algum motivo eu preciso que esse dado seja do tipo string.

Mas antes, como vamos saber que o tipo de dados realmente mudou?

Bem, além do console do navegador facilitar isso através da forma como o dado é impresso, podemos usar a função **typeof()** do javascript, essa função recebe um parâmetro, e esse parâmetro é a variável a qual eu quero verificar o tipo, na verdade não é exatamente o tipo, mas sim a família do tipo kkkkk, podemos considerar assim ☺, vamos ver como isso funciona:

```
typeof(codigo)
"number"
```

Podemos ver que a nossa variável 'codigo' é da família dos números, eu prefiro chamar de família justamente por que os números podem ser ***inteiros***, ***decimais*** e por aí vai, mas todos eles em geral são números.

Agora sim, vamos converter a nossa variável 'codigo' para uma string, podemos fazer isso da seguinte forma:

```
codigo = codigo.toString()
"652"
```

Podemos ver que a conversão foi feita, olhando para a impressão da variável no console do navegador o conteúdo está dentro de aspas duplas, deixando claro que se trata de uma string.

Mas para podermos verificar isso de outra forma podemos usar a função **typeof()** que vimos antes, veja:

```
typeof(codigo)
"string"
```

Podemos ver que realmente o conteúdo da nossa variável 'codigo' é da família dos números.

Também podemos fazer essa conversão usando o método String(), o resultado é exatamente o mesmo. Para testar isso vamos converter novamente a nossa string para um número através do método **Number()** :

```
codigo = Number(codigo)
652
```

Podemos ver que agora a variável voltou a ser do tipo número, agora vamos tornar ele numa string através do método String() :

```
codigo = String(codigo)
"652"
```

Podemos ver que funciona da mesma forma, a diferença é que no **String()** a variável é passada como parâmetro enquanto que no **.toString()** já não.

Também deve se tomar cuidado na hora de escrever a função **String()**, a primeira letra deve estar em maiúscula, a mesma coisa com a função **Number()**, isso se deve ao facto do Javascript ser ***"Case sensitive"***, isso significa que ela faz distinção entre maiúsculas e minúsculas.

E chegamos ao fim desse nosso pequeno módulo sobre a manipulação de string, claramente que existem varias outras funções para manipulação de strings que não foram abordados aqui dentro desse módulo, mas aquilo que vimos já é suficiente para ajustar muitos dados escritos de forma errada pelo usuário.

Espero que esse conteúdo tenha ajudado a entender um pouco sobre a manipulação de strings na linguagem de programação Javascript.

Para mais conteúdo sobre programação viste o canal **AprendAki Ciência** no Youtube

Link: https://youtube.com/aprendakiciencia

Obrigado ☺

FIM

O Explicador:
Félix Sumburane
+258 84 039 2740