Trabalhando com texto - strings em JavaScript

Samuel Amaro

2 de abril de 2021

Agora vamos dar atenção às strings - isto é como é chamado em programação qualquer parte de texto.

Objetivo: Ganhar familiaridade com o básico de strings em JavaScript.

O poder das palavras

Palavras são muito importante para humanos - elas são uma grande parte de como nos comunicamos. Desde que a Web é largamente baseada em texto, projetada para permitir humanos a comunicar e compartilhar infomação, isto é útil para nós termos controle sobre como apresentar isso. HTML fornece estrutura e significado para nosso texto, CSS nos permite estilizar precisamente ele, e JavaScript contem um número de funcionalidades para manipular strings, criar mensagens de boas-vindas customizadas, mostrando rótulos quando preciso, sorteando termos de acordo como desejado e muito mais.

Muitos dos programas que temos visto até agora no curso está envolvido em alguma parte com manipulação de string.

Strings - O básico

Em um primeiro relance, strings são tratadas de forma parecida como números, mas quando vamos mais a fundo, você começa a ver algumas diferenças importantes. Vamos começar a entrar em linhas básicas no console para nos familiarizar.

Criando uma string

1. Para começar, digite as linhas seguintes:

```
var string = 'The revolution will not be televised.';
string;
```

Listing 1: Criando uma string

   Como fizemos com números, nós declaramos uma variável, inicializando-a com um valor string, e então retornamos o valor. A única diferença aqui é que quando escrevemos uma string, você precisa colocá-la entre aspas.

2. Se você não fez isso, ou esqueceu uma das aspas, você recebeu um erro. Experimente digitar as linhas seguintes:

```
var badString = This is a test;
var badString = 'This is a test;
var badString = This is a test';
```

   Estas linhas não funcionam porque todo texto sem aspas são interpretados como um nome de variável, propriedade do nome, palavra reservada ou algo assim. Se o navegador não puder encontrar, então um é erro é apresentado (ex.: "faltando; declaração anterior"). Se o navegador puder ver onde a string começa, mas não conseguir encontrar o fim, como indicado com as segundas aspas, é retornado um erro (com "string não terminada"). Se seu programa retorna tais erros, então volte e verifique todas suas strings para ter certeza que não faltam aspas.

3. O seguinte funcionará se você definiu previamente a variável `string` - tente isto agora:

```
var badString = string;
badString;
```

   `badString` é agora definido para ter o mesmo valor de `string`.

Aspas simples x aspas duplas

1. Em JavaScript, você pode escolher aspas simples ou duplas para envolver suas strings. Ambas linhas abaixo funcionará bem:

```
1
2           var sgl = 'Single quotes.';
3           var dbl = "Double quotes";
4           sgl;
5           dbl;
6
7
```

Listing 2: Aspas simples x Aspas Duplas

2. Há poucas diferenças entre as duas, e qual você usar é de preferência pessoal. Você deve escolher uma e permanecer nela, entretanto; diferentes aspas no código pode ser confuso, especialmente se você usa diferentes aspas na mesma string! O seguinte retornará erro:

```
1
2           var badQuotes = 'What on earth?";
3
4
```

3. O navegador interpretará como a string não tivesse fechada, porque o outro tipo de aspas pode aparecer dentro da sua string. Por exemplo, ambos exemplos abaixo são okay:

```
1
2           var sglDbl = 'Would you eat a "fish supper"?';
3           var dblSgl = "I'm feeling blue.";
4           sglDbl;
5           dblSgl;
6
7
```

Listing 3: Exemplo de uso das aspas

4. Entretanto, você não pode incluir o mesmo tipo de aspas dentro da sua string, se você usa para conter seu texto. O seguinte será um erro, como é confuso para o navegador onde a string termina:

```
1
2           var bigmouth = 'I've got no right to take my place
    ...';
3
4
```

Caracteres de escape na string

Para corrigir o problema anterior, nós precisamos escapar o problema da aspa. Caracteres de escape significa que nós fazemos algo para ter certeza que eles são reconhecidos como texto, não parte do código. Em JavaScript, nós fazemos isso colocando uma barra invertida logo antes do caracter.Tente isso:

```
1
2  var bigmouth = 'I\'ve got no right to take my place...';
3  bigmouth;
```

Listing 4: Caracteres de escape em string

Isto funciona bem. Você pode escapar outros caracteres do mesmo jeito, ex.: `\"`, e há alguns códigos especiais também.

Concatenando strings

1. Concatenar é uma palavra chique da programação que significa "colocar junto". Para colocar strings juntas em JavaScript, usamos o operador (+), o mesmo usamos para adicionar números, mas neste contexto é algo diferente. Vamos tentar este exemplo no console.

```
1
2         var one = 'Hello, ';
3         var two = 'how are you?';
4         var joined = one + two;
5         joined;
6
7
```

Listing 5: concatenação de strings

O resultado disso é uma variável chamada `joined`, que contém o valor "Hello, how are you?".

2. No último exemplo, nós somente juntamos duas strings, mas você pode fazer quantas quiser, contanto que inclua um + entre cada uma. Experimente isso:

```
1
2         var multiple = one + one + one + one + two;
3         multiple;
4
5
```

Listing 6: concatenação de varias strings

3. Você pode usar um mix de variáveis e strings reais. Tente isso:

```
1
2         var response = one + 'I am fine     ' + two;
3         response;
4
5
```

Nota: Quando você coloca uma string atual no seu código dentro de aspas simples ou duplas, é chamada uma **string literal**.

Concatenação em contexto

Vamos dar uma olhada na concatenação em ação — aqui está um exemplo do curso anterior:

```
1
2     <button>Pressione-me</button>
```

```
1
2     var button = document.querySelector('button');
3
4     button.onclick = function() {
5         var nome = prompt('Qual    o seu nome?');
6         alert('Ol   ' + nome + ', prazer em conhec -lo!');
7     }
```

Aqui estamos usando uma função `Window.prompt()` na linha 4, a qual pergunta ao usuário para responder uma questão via uma caixa de diálogo, então armazena o texto em uma variável — neste caso `nome`. Nós então usamos uma função `Window.alert()` na linha 5 para mostrar outra caixa de diálogo contendo nossa string montada de duas strings literais e a variável `name`,via concatenação.

Números x strings

1. Então o que acontece quando tentamos adicionar (ou concatenar) uma string e um número? Vamos tentar isso no console:

```
1
2         'Front' + 242;
3
4
```

Listing 7: Concatenação de string e um numero

Você pode esperar um erro disto, mas funciona correto. Tentando representar uma string como um número, realmente não faz sentido. Mas representando um número como string, faz. Então o navegador espertamente converte o número em string e concatena as duas.

2. Você pode fazer isto até com dois números — você pode forçar um número a ser string colocando ele entre aspas. Experimente o seguinte (nós estamos usando o operador `typeof` para checar o que a variável é, se um número ou string):

```
1
2       var myDate = '19' + '67';
3       typeof myDate;
4
5
```

3. Se você tem uma variável numérica que precisa converter em string, mas não mudar completamente, ou uma string e quer converter em número, você pode usar a construção seguinte:

   - O objeto `Number` converterá qualquer coisa passada em um número, se for possível. Tente o seguinte:

```
1
2           var myString = '123';
3           var myNum = Number(myString);
4           typeof myNum;
5
6
```

Listing 8: Convertendo texto em numero

   - Por outro lado, todo número tem um método chamado `toString()` que converterá para a `string` equivalente. Tente isto:

```
1
2           var myNum = 123;
3           var myString = myNum.toString();
4           typeof myString;
5
6
```

Listing 9: converte numero em string

Estas construções podem ser bem úteis em algumas situações. Por exemplo, se o usuário colocar um número em um campo de texto, isso será uma string. Entretanto, se você quiser adicionar este número a algo, você precisa que seja um número, então você pode passar isto através do `Number()` para poder manipular.