Metodos úteis de string

Samuel Amaro

6 de abril de 2021

Metodos úteis de string

Agora que nós vimos o básico de strings, vamos engatar a próxima marcha e começar a pensar sobre as operações úteis que podemos fazer em strings com métodos embutidos, como encontrar o comprimento de um string, unir e dividir sequências de caracteres, substituindo um caracter em uma string por outro, e muito mais.

Objetivo: Entender que strings são objetos e aprender a usar alguns do métodos básicos disponíveis nesses objetos para manipular strings.

Strings como objetos

Como dissemos antes e diremos novamente — tudo é um objeto em JavaScript. Quando você cria um string, usando por exemplo

```
1
2  var string = 'The is my string';
```

Listing 1: Criando uma string

sua variável torna-se uma instância do objeto string e, como resultado, tem um grande número de propriedades e métodos diponíveis para ela.

Agora, antes de seu cérebro começar a derreter, não se preocupe! Você não precisa saber sobre a maioria deles no início da sua jornada de aprendizado. Mas há alguns que você potencialmente usará com bastante frequência que veremos aqui.

Vamos digitar alguns exemplos em um console novo.

Descobrindo o comprimento de uma string

Essa é fácil — você simplesmente usa a propriedade `length`. Tente digitar as linhas a seguir:

```
1
2     var browserType = 'mozilla';
3     browserType.length;
```

Listing 2: metodo de comprimento de uma string

Isso deve retornar o número 7, porque "mozilla"tem 7 caracteres. Isso é útil por vários motivos; por exemplo, você pode querer encontrar os comprimentos de uma série de nomes para que você possa exibi-los em ordem de comprimento, ou deixar um usuário saber que um nome de usuário que ele informou em um campo de formulário é muito longo se este for maior do que um certo comprimento.

Recuperando um caractere de string especifico

Nota complementar: você pode retornar qualquer caractere dentro de uma string usando a **notação colchete** - isso significa que você inclui colchetes (`[]`) no final do nome da variável. Dentro dos colchetes, você inclui o número do caractere que deseja retornar, por exemplo, para recuperar a primeira letra, faça o seguinte:

```
1
2          browserType[0];
```

Listing 3: recuperando caracter da string

Computadores contam a partir de 0, não 1! Para recuperar o último caractere de qualquer string, nós podemos usar a linha a seguir, combinando essa técnica com a propriedade `length` que vimos anteriormente:

```
1
2          browserTyper[browserType.lenght - 1];
```

Listing 4: recuperando ultimo caracter da string

O comprimento de "mozilla"é 7, mas porque a contagem começa de 0, a posição do caractere é 6, daí precisamos usar `length-1`. Você pode usar isso para, por exemplo, encontrar a primeira letra de uma série de strings e ordená-los alfabeticamente.

Encontrando uma substring dentro de uma string e extraindo-a

1. Às vezes você quer sabe se uma string menor está presente dentro de uma maior (geralmente dizemos se uma substring está presente dentro de uma string). Isso pode ser feito usando o método `indexOf()`, que leva um único `parameter` - a substring que deseja procurar. Experimente isso:

```
1
2          browserType.indexOf('zilla');
```

   Isso nos dá o resultado 2, porque a substring "zilla"se inicia na posição 2 (0, 1, 2 — então, 3 caraceteres) dentro de "mozilla". Esse código poderia ser usado para filtrar cadeias de caracteres. Por exemplo, podemos ter uma lista de endereços da web e apenas queremos imprimir aqueles que contenham "mozilla".

2. Isso pode ser feito de outro jeito, que é possivelmente mais eficaz. Experimente isso:

```
1
2  browserType.indexOf('vanilla');
```

   Isso deve lhe dar um resultado -1 — isso é retornado quando a substring, neste caso 'vanilla', não é encontrada na string principal.

Você pode usar isso para encontrar todas as instâncias de strings que **não contém** a substring 'mozilla', ou **contém**, se você usar o operador de negação, conforme mostrado abaixo. Você poderia fazer algo assim:

```
1
2          if(browserType.indexOf('mozilla') !== -1) {
3             // faz coisas com a string
4          }
```

Quando você sabe onde uma substring começa dentro de uma string e você sabe em qual caractere você deseja que ela termine, `slice()` pode ser usado para extrair isto. Tente o seguinte:

```
1
2    browserType.slice(0,3);
```

Isso retorna "moz"— o primeiro parâmetro é a posição do caractere a partir da qual será iniciada a extração, e o segundo parâmetro é a posição seguinte do último caractere a ser extraído. Então, a fatia ocorre da primeira posição, até a última posição, mas não incluindo. Você também pode dizer que o segundo parâmetro é igual ao comprimento da string que está sendo retornada.

Também, se você sabe que você deseja extrair todos os caracteres restantes em uma string após um certo caracter, você não tem que incluir o segundo

parametro! Você apenas precisa incluir a posição do caracter a partir de onde você deseja extrar os caracteres restantes em uma string. Tente o seguinte:

```
1
2     browserType.slice(2);
```

Isso retornará "zilla"— isso é porque a posição de caracter 2 é a letra z, e porque você não incluiu o segundo parametro, a substring retornou todos os caracteres restantes na string.

Note: O segundo parametro do `slice()` é opcional: Se você não incluir ele, o slice finaliza no fim da string original.

Mudando entre maiúsculas e minúsculas

O método string `toLowerCase()` e `toUpperCase()` toma a string e converte todos os caracteres para minusculo - ou maiusculo, respectivamente. Este pode ser util, por exemplo, se você quer normalizar todas as entradas de dados do usuário antes de armazenar em um banco de dados.

Vamos testar inserindo as seguintes linhas para ver o que acontece:

```
1
2     var radData = 'My NaMe Is MuD';
3     radData.toLowerCase();
4     radData.toUpperCase();
```

Atualizando partes de uma string

Você pode substituir uma substring dentro de uma string com uma outra substring usando o método `replace()`. Este funciona muito simples em um nível básico, apesar haver algumas coisas avançadas que você pode fazer com ele, nós não iremos tão longe ainda.

Ele toma dois parametros — A string que você quer substituir e a string que você quer que substitua o primeiro parametro. Tente este exemplo:

```
1
2     browserType.replace('moz','van');
```

Observe que para realmente obter o valor atualizado refletido na variavel `browserType` em um programa real, você teria que setar o valor da variavel para ser o resultado da operação; não apenas atualizar o valor da substring automaticamente. Assim você teria que realmente escrever isso:

```
browserType = browserType.replace('moz','van');
```