Bài: Kiểu dữ liệu chuỗi trong JavaScript (Phần 4) - Các phương thức với chuỗi trong Javascript

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu chuỗi trong JavaScript (Phần 4) - Các phương thức với chuỗi trong Javascript

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài trước, chúng ta đã cùng tìm hiểu về các phương thức với chuỗi trong Javascript

Ở bài này, chúng ta sẽ tiếp tục tìm hiểu về chủ đề đó.

Nội dung

Những thứ mà bạn cần nắm được trước khi tìm hiểu bài này:

- Kiểu dữ liệu chuỗi trong Javascript

Ở bài này, chúng ta sẽ có 3 nội dung chính

- Các phương thức với chuỗi trong Javascript
  - Các phương thức biến đổi
  - Các phương thức tiện ích
  - Các phương thức phân tách

Các phương thức biến đổi

Phương thức toUpperCase

Cú pháp:

> **<String>.toUpperCase()**

Tác dụng: Dùng để chuyển một chuỗi sang dạng viết hoa

Ví dụ:

Javascript:

```
var a = 'hello world'
// undefined
a.toUpperCase();
// 'HELLO WORLD'
a.charAt(3).toUpperCase();
// 'L'
```

Phương thức toLowerCase

Cú pháp:

> **<String>.toLowerCase()**

Tác dụng: Dùng để chuyển một chuỗi về dạng viết thường.

Ví dụ:

Python:

```
var a = 'HELLO WORLD';
// undefined
a.toLowerCase();
// 'hello world'
a.charAt(7).toLowerCase();
// 'o'
```

Phương thức repeat

Cú pháp:

> **<String>.repeat(<times>)**

Tác dụng: Nó cho phép lặp lại chuỗi **<String>** **<times>** lần. Cho kết quả tương tự khi ta cộng một chuỗi vào chính nó **<times>** lần.

Ví dụ:

Javascript:

```
var a = 'hi ';
// undefined
a.repeat(4);
// 'hi hi hi hi '
a.repeat(2);
// 'hi hi '
a + a + a + a; // nối chuỗi a vào nhau 4 lần
// 'hi hi hi hi '
```

Phương thức padStart

Cú pháp:

> **<String>.padStart(<length>, <padString>);**

Tác dụng: Chèn các chuỗi **<padString>** vào đầu chuỗi **<String>** ban đầu sao cho **<String>** ban đầu có độ dài bằng **<length>**. Trong trường hợp **<length>** nhỏ hơn hoặc bằng độ dài của **<String>**, phương thức này không được thực hiện.

Ví dụ:

Javascript:

```
var a = 'x';
// undefined
a.padStart(5, '.');
// '....x'
a = 'iiiii';
// 'iiiii'
a.padStart(6, '.');
// '.iiiii'
a.padStart(4, '.');
// 'iiiii'
```

Phương thức padEnd

Cú pháp:

> **<String>.padEnd(<length>, <padString>)**

Tác dụng: Chèn các chuỗi **<padString>** vào cuối chuỗi **<String>** ban đầu sao cho **<String>** ban đầu có độ dài bằng **<length>**. Trong trường hợp **<length>** nhỏ hơn hoặc bằng độ dài của **<String>**, phương thức này không được thực hiện.

Ví dụ:

Javascript:

```
var a = 'x';
// undefined
a.padEnd(5, '.');
// 'x....'
a = 'iiiii';
// 'iiiii'
a.padEnd(6, '.');
// 'iiiii.'
a.padEnd(5, '.');
// 'iiiii'
```

Phương thức trim

Cú pháp:

> **<String>.trim()**

Tác dụng: Dùng để xóa các khoảng trắng thừa ở đầu và cuối chuỗi.

> **Lưu ý:** Nếu chỉ muốn xóa ở đầu chuỗi, ta dùng **trimStart**, còn nếu chỉ muốn xóa ở cuối, ta dùng **trimEnd**

Ví dụ:

Javascript:

```
a = '      t      ';
// '      t      '
a.trim()
// 't'
a.trimEnd()
// '      t'
a.trimStart()
// 't
```

Các phương thức tiện ích

Phương thức split

Cú pháp:

> **<String>.split(<seperator>, <limit>)**

Tác dụng: Giúp phân tách một chuỗi thành nhiều phần (các phần phân tách nhau bằng một chuỗi **<seperator>**) và các phần đó được lưu vào một **object** (ta sẽ tìm hiểu kĩ hơn về object trong các bài sau). Object này có thể được hiểu là một mảng.

Bên cạnh đó, **<limit>** quy định số phần tử tối đa của mảng chứa các phần tử. Khi số phần tử trong mảng bằng **<limit>**, phương thức dừng hoạt động.

Ví dụ:

Javascript:

```
var a = 'a s d f g h j';
// 'a s d f g h j'
t = a.split(' '); // tách các ký tự bằng khoảng trắng (ký tự ' ')
// [
  'a', 's', 'd',
  'f', 'g', 'h',
  'j'
]
t
// [
  'a', 's', 'd',
  'f', 'g', 'h',
  'j'
]
var t = 'hello world';
// undefined
t.split('o');
// [ 'hell', ' w', 'rld' ]

str = 'we are kteam'
// 'we are kteam'
str.split(' ', 2)
// [ 'we', 'are' ]
str.split(' ', 1)
// [ 'we' ]
```

Phương thức replace

Cú pháp:

> **<String>.replace(<substr>, <newSubstr>)**

Tác dụng: dùng để thay thế các thành phần trong một chuỗi bằng một thành phần khác. Phương thức này sẽ trả về một chuỗi mới.

Trong cú pháp:

- **<substr>** là chuỗi cần được thay thế.
- **<newSubstr>** là chuỗi được dùng để thay thế cho **<substr>** ở bên trong **<String>**

Ví dụ:

Javascript:

```
a = 'howKteam'
// 'howKteam'
b = a.replace("how", "Kter")
// 'KterKteam'
b
// 'KterKteam'
c = a.replace("Kteam", "Kquizz")
// 'howKquizz'
c
// 'howKquizz'
a
```

Mở rộng: Ngoài cú pháp cơ bản nhất mà Kteam đã đề cập bên trên, phương thức replace còn 3 cú pháp khác (dùng trong 3 trường hợp khác nhau):

1. replace(regexp, newSubstr)
2. replace(regexp, replacerFunction)
3. replace(substr, replacerFunction)

Ở bên trên, ta có thể thấy: 2 cú pháp đầu tiên có liên quan đến việc sử dụng **regexp** (biểu thức chính quy), còn cú pháp thứ 3 liên quan đến một khái niệm mới được gọi là hàm. Trong các bài sau, Kteam sẽ nói rõ hơn đến việc sử dụng regexp, tuy nhiên không đi sâu vào các khái niệm trên.

Các phương thức phân tách

Các phương thức này được dùng để tách các chuỗi thành các phần nhỏ hơn.

> **Lưu ý:** Các phương thức phân tách không thay đổi chuỗi ban đầu, mà trả về một chuỗi mới.

Phương thức slice

Cú pháp:

> **<String>.slice(<start>, <end>)**

Giải thích: chương trình sẽ lấy các ký tự có chỉ số từ **<start>** đến **<end>**-1. Trong trường hợp ta chỉ đặt **<start>**, chương trình sẽ lấy các ký tự từ **<start>** cho đến hết chuỗi. Còn nếu không có cả **<start>** và **<end>**, thì phương thức này trả về một bản sao của chuỗi.

Ví dụ:

Javascript:

```
var str = "abcdef";
// undefined
var slice = str.slice(2, 5);
// undefined
slice
// 'cde'
var unstart = str.slice(1);
// undefined
unstart
// 'bcdef'
var t = str.slice(-4);
// 'cdef'
t
// ‘cdef’
t = str.slice();
// 'howKteam.com'
t
// 'howKteam.com'
```

Phương thức substring

Cũng tương tự như slice, phương thức **substring** giúp ta phân tách một phần của chuỗi. Tuy nhiên, substring không chấp nhận các tham số âm.

Cú pháp:

> **<String>.substring(<start>, <end>)**

Ví dụ:

Javascript:

```
var str = 'howKteam.com';
// undefined
var substring = str.substring(3, 9);
// undefined
substring
// 'Kteam.'
substring = str.substring();
// 'howKteam.com'
substring
// 'howKteam.com'
```

Phương thức substr

Cú pháp:

> **<String>.substr(<start>, <length>)**

Giải thích: Chương trình sẽ lấy chính xác **<length>** ký tự kể từ ký tự thứ **<start>** trong chuỗi. Nếu ta không chỉ định **<length>**, hoặc số ký tự từ **<start>** đến cuối chuỗi nhỏ hơn **<length>**, chương trình sẽ lấy từ **<start>** cho đến hết chuỗi. Còn nếu ta không chỉ định cả **<start>** và **<length>**, phương thức trả về một bản sao của chuỗi.

Ví dụ:

Javascript:

```
var str = 'HowKteam'
// undefined
str.substr(2, 5);
// 'wKtea'
str.substr(2);
// 'wKteam'
str.substr();
// 'HowKteam'
str.substr(-3);
// 'eam'
str.substr(-10, 2);
// 'Ho'
```

Kết luận

Qua bài này, các bạn đã hoàn thiện hơn những kiến thức của bản thân về chuỗi

Ở bài tiếp theo, Kteam sẽ giới thiệu đến các bạn một kiểu dữ liệu mới: boolean

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".