Bài: Kiểu dữ liệu chuỗi trong JavaScript (Phần 1) - Khái quát về kiểu dữ liệu chuỗi trong Js

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu chuỗi trong JavaScript (Phần 1) - Khái quát về kiểu dữ liệu chuỗi trong Js

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 các bài trước, chúng ta đã cùng tìm hiểu về kiểu dữ liệu số trong Javascript và thực hiện bài tập về kiểu dữ liệu này.

Trong bài này sẽ giới thiệu cho các bạn về một kiểu dữ liệu khác đó là **Kiểu dữ liệu chuỗi trong Javascript - Khái quát về kiểu dữ liệu chuỗi**

Nội dung

Mời các bạn xem qua các yêu cầu về kiến thức trước khi tìm hiểu bài này:

- Biến trong Javascript
- Cài đặt sẵn môi trường nodejs

Nội dung của bài bao gồm:

- Chuỗi là gì ? Đặc điểm của một chuỗi.
- Escape character trong Javascript
- Các toán tử cơ bản đối với chuỗi trong Javascript

Chuỗi là gì ?

Chuỗi, hiểu đơn giản, là tất cả những gì được đặt trong cặp '', "" hoặc ``.

Chuỗi giúp biểu thị các kí tự, văn bản trong Javascript

Ví dụ:

Javascript:

```
var s1 = 'abc'; // Chuỗi s1 có giá trị là 'abc'
var s2 = "123"; // Chuỗi s2 có giá trị là "123"
```

Khởi tạo một chuỗi trong Js

Để khởi tạo một chuỗi, có 3 cách chính:

Cách 1: Sử dụng **constructor String**

Cú pháp:

```
String(<value>)
```

Trong đó, **<value>** là một giá trị hợp lệ bất kì

Giá trị trả về chính là một chuỗi tương ứng với giá trị đầu vào.

Cách 2: Sử dụng phương thức toString cho giá trị tương ứng

Cú pháp:

```
<value>.toString()
```

Trong đó, **<value>** cũng tương tự như trên, chính là giá trị mà chúng ta muốn **convert** sang **string**. Tuy nhiên, đối với việc chuyển từ số sang chuỗi, thì ta buộc phải dùng tên biến thay cho giá trị, cụ thể:

Javascript:

```
5.toString()
// 5.toString()
^^

// Uncaught SyntaxError: Invalid or unexpected token
a = 5
// 5
a.toString()
// '5'
```

Cách 3: Đặt các giá trị trong cặp dấu **nháy** (``, '' hoặc "")

Cách này là cách đơn giản nhất và dễ thực hiện nhất trong 3 cách. Nó sẽ chuyển toàn bộ những gì được đặt trong cặp dấu nháy thành 1 string duy nhất

Dưới đây là ví dụ cho cả 3 cách:

Javascript:

```
a = 5
// 5
b = '12'
// '12'
c = [1, 2]
// [ 1, 2 ]
a.toString()
// '5'
b.toString()
// '12'
String(c)
// '1,2'
String(a)
// '5'
str = 'khanhdeptrai!'
// 'khanhdeptrai!'
```

Vị trí của các kí tự trong chuỗi

Để thuận tiện trong việc thao tác với chuỗi, thì các kí tự trong chuỗi sẽ được đánh số thứ tự, từ trái sang phải, với kí tự đầu tiên có vị trí là 0. Lấy ví dụ với chuỗi "howKteam":

Vị trí	h	o	w	K	t	e	a	m
Kí tự	0	1	2	3	4	5	6	7

Ta có thể lấy ra kí tự trong một chuỗi dựa vào vị trí của nó.

Ví dụ:

Javascript:

```
var a = 'howKteam';
// undefined
a[4]
// 't'
a[3]
// 'K'
```

> **Lưu ý**: ta không thể thay đổi chuỗi bằng cách thực hiện phép gán.

Javascript:

```
a = 'howKteam'
// 'howKteam'
a[2] = 't'
// 't'
a
// 'howKteam'
```

Độ dài của chuỗi

Ta dùng thuộc tính **length** để hiển thị độ dài của một chuỗi.

Javascript:

```
var str = 'Hello world';
// undefined
str.length
// 11
```

Tiếp theo, mời các bạn xem xét ví dụ sau:

Javascript:

```
a = 'I'm Beginner'
// a = 'I'm Beginner'
       ^

// Uncaught SyntaxError: Unexpected identifier
```

Tại sao chương trình báo lỗi?

- Hiểu một cách đơn giản, là trong câu lệnh đó đã có một phần bị thừa ra.
- Sau khi ta dùng dấu ' lần đầu tiên, thì đó là việc xác định một chuỗi.
- Đến dấu ' lần thứ 2, thì đó là việc kết thúc một chuỗi (trong ví dụ, thì chuỗi chỉ bao gồm một chữ "I")
- Và tất nhiên, những nội dung sau đó đều bị tính là phần dư thừa, gây nên lỗi.

Đặt vấn đề: Thế nếu ta muốn dùng nhiều các dấu ' hoặc " trong một chuỗi, thì làm như thế nào ?

- **Cách đầu tiên**: Nếu muốn dùng dấu ' trong chuỗi, thì chuỗi đó phải được đặt trong cặp "" và ngược lại.
- **Một cách khác** là dùng cặp dấu `` (Kteam sẽ đi chi tiết vào nội dung này ở bài sau).
- **Cách cuối cùng**: mời các bạn đọc phần bên dưới.

Escape character trong JavaScript

Khi làm việc với chuỗi trong JavaScript, sẽ có những lúc mà bạn cần đến các **Escape character**. Hiểu một cách đơn giản, escape character là những kí tự có chức năng phụ trợ cho chuỗi khi ta làm việc với nó. Nó sẽ bao gồm một dấu "\" đứng trước một kí tự.

Ví dụ với chương trình sau:

Javascript:

```
var a = 'Welcome\nto HowKteam';
// undefined
console.log(a); // In ra 'Welcome', sau đó xuống dòng và in 'to HowKteam'
// Welcome
// to HowKteam
// undefined
var b = '\tFree education';
// undefined
console.log(b); // In ra 8 kí tự trống, sau đó in 'Free education' (giống như khi ta ấn tab trong lúc gõ văn bản)
//        Free education
// undefined
```

Dưới đây là bảng liệt kê các escape character hay được sử dụng:

Kí tự	Ý nghĩa
\n	Đưa con trỏ xuống dòng tiếp theo
\t	In ra một horizotal tab (giống khi ta tab lúc gõ văn bản)
\b	Xóa kí tự đứng trước con trỏ (nếu có)
\"	In ra kí tự "
\'	In ra kí tự '
****	In ra kí tự "\"

Lưu ý: muốn sử dụng escape character, ta phải dùng lệnh **console.log()** (như trong các ví dụ trên)

Các toán tử với chuỗi trong JavaScript

Toán tử so sánh

Chương trình sẽ so sánh từng cặp kí tự có cùng vị trí trong 2 chuỗi. Nếu độ dài của 2 chuỗi khác nhau, thì khi so sánh đến hết chuỗi ngắn hơn, chương trình dừng so sánh.

Trường hợp	Thì ta có:	Ví dụ
a = b	Hai chuỗi y hệt nhau	'aaa' = 'aaa'
A ≠ b	Hai chuỗi khác độ dài, hoặc có ít nhất một kí tự khác nhau.	'a' ≠ 'ab' 'a' ≠ 'z' '1' ≠ '3'

A < b	Ở kí tự đầu tiên khác nhau (có cùng vị trí), kí tự trong chuỗi a đứng trước kí tự b trong bảng mã Unicode	'a' < 'b' 'abc' < 'z' '12' < '2'
A > b (ngược lại với trường hợp trên)		

Toán tử cộng chuỗi trong JavaScript

Toán tử cộng (+) nối hai chuỗi với nhau và trả về một chuỗi mới.

Ví dụ:

Javascript:

```
var a = "Welcome to";
// undefined
var b = "kquiz";
// undefined
a + ' ' + b;
// 'Welcome to kquiz'
var t = "Hello \n";
// undefined
var f = "Toan";
// undefined
var u = t + f;
// undefined
u
// 'Hello \nToan'
console.log(u);
// Hello
// Toan
// undefined
```

Các toán tử khác

Ngoài toán tử cộng, các toán tử so sánh, mọi toán tử khác trên các chuỗi kí tự có bao gồm chữ cái (hoặc kí tự đặc biệt) đều trả về giá trị **NaN** (đối với các chuỗi Number-String, ta sẽ tìm hiểu sau).

Ví dụ:

Javascript:

```
'abc' / 'abc'
// NaN
'one' * 'two'
// NaN
'per' * 3
// NaN
'miss' - 'mi'
// NaN
'3' * '$'
// NaN
```

Kết luận

Ở bài này, các bạn đã được tìm hiểu về những kiến thức cơ bản của chuỗi trong Javascript

Qua bài sau, các bạn sẽ được tìm hiểu kĩ hơn về string, cụ thể là các thuộc tính và các phương thức.

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**".