Bài: Bài tập về kiểu dữ liệu chuỗi trong JavaScript

Xem bài học trên website để ủng hộ Kteam: Bài tập về kiểu dữ liệu 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!

Trong bài này, Kteam và bạn sẽ thực hiện một số bài tập củng cố kiến thức về Kiểu dữ liệu chuỗi trong JavaScript

Để bài tập đạt hiệu quả tốt nhất bạn nên thực hiện theo các bước sau:

1. Đọc đề các câu hỏi và tự đưa ra đáp án/ lời giải của mình.
2. Tham khảo đáp án tại bài Bài tập về kiểu dữ liệu Boolean và rà sát đáp án/ lời giải của bản thân
3. Xem video giải thích đáp án của bài này và note lại các ý chính để củng cố lại kiến thức.

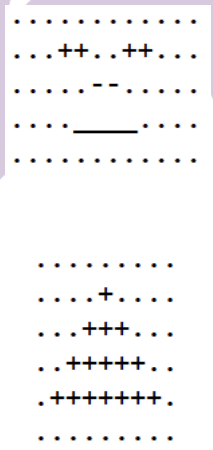
Câu hỏi về chuỗi trong JavaScript

Câu 1: Nêu các dấu hiệu đặc trưng của chuỗi trong Js, cách khởi tạo chuỗi trong Js.

Câu 2: Cho biết kết quả của các phép tính sau:

- '12' + '12'
- 12 + '12'
- '12' * 12
- 'a' * 13
- 'a' + 13
- 'aa' + '12' * 3

Câu 3: Sử dụng phương thức repeat của string, viết chương trình in ra các nhóm kí tự sau (in trực tiếp lên màn hình khi chạy chương trình):



Câu 4: Cho chuỗi a = 'howKteam and kter', cho biết kết quả của các phương thức sau:

a. a.split('t')
b. a.replace(' a', '__')
c. a.padStart('o')
d. a.slice(-13, 8)

Câu 5: Cho biết kết quả của các phép so sánh sau:

- 'cha' < 'con'
- 'toan' < 'TOAN'
- 'ton' == 'tOn'
- 'anh' > 'em'
- 'small' >= 'big'

Câu 6: Hãy chỉ ra các đặc điểm cơ bản của chuỗi khi nó được dùng với cặp `` (backtics)

Đáp án bài kiểu dữ liệu số trong Javascript

Bạn có thể xem chi tiết các câu hỏi ở Bài tập về kiểu dữ liệu Number trong JavaScript

Đáp án câu 1:

Số khi đặt vào trong Js sẽ là number hoặc là bigint. Ngoài ra, trong nhiều trường hợp, một chuỗi gồm các kí tự số có thể xem như là một số.

Đáp án câu 2:

Infinity = Number.POSITIVE_INFINITY

-Infinity = Number.NEGATIVE_INFINITY

Javascript:

```
> typeof(null)
'object'
> typeof(undefined)
'undefined'
> typeof(float)
'undefined'
```

Đáp án câu 3:

Trong số các giá trị trên, chỉ có NaN là number.

- Undefined và null là các kiểu dữ liệu riêng, sẽ được Kteam trình bày trong các bài tiếp theo
- Float không có bất kì ý nghĩa gì trong Js

Các bạn có thể kiểm chứng bằng cách sau:

Javascript:

```
> typeof(null)
'object'
> typeof(undefined)
'undefined'
> typeof(float)
'undefined'
```

Sở dĩ khi ta dùng float, thì đối với console nodejs, nó hiện ra undefined là vì biến float chưa được khởi tạo.

Đáp án câu 4:

Ý tưởng:

- Dùng phương thức toFixed để làm tròn số đó.
- Nhân số đó với chính nó, ta sẽ được kết quả.

Chương trình tham khảo:

Javascript:

```
// n is available
let x = n.toFixed()
console.log(x*x)
```

Đáp án câu 5:

Dưới đây là các ví dụ và chú thích của Kteam:

Javascript:

```
> Math.PI
3.141592653589793
> Math.E
2.718281828459045
> Math.SQRT1_2
0.7071067811865476
> Math.acos(-1) // arccos
3.141592653589793
> Math.asin(1) // arcsin
1.5707963267948966
> Math.max(1, -1, 2, -1, 0, 2) // Lấy số lớn nhất trong một dãy các số.
2
> Math.sqrt(100) // căn bậc 2
10
> Math.cbrt(8) // căn bậc 3
2
```

Đáp án câu 6:

Bigint có thể làm việc với các số vô cùng lớn, mà number không thể làm được. Và đặc điểm cơ bản nhất của nó là có kí tự n sau giá trị số. Bên cạnh đó, các toán tử có thể được sử dụng trên bigint tương tự như với number (ngoại trừ toán tử chia và >>>)

Đáp án câu 7:

Các toán tử giữa các bigint có thể sử dụng như khi chúng được dùng trên number:

Javascript:

```
> 12n + 9n
21n
> 5n * 34n
170n
> 9n - 100n
-91n
> 12n * 12n
144n
> a = 13n
13n
> a++
13n
> a
14n
> --a
13n
> a
13n
```

Tuy vậy, đối với toán tử chia, thì chúng sẽ thực hiện việc "chia lấy phần nguyên".

Javascript:

```
> 100n / 43n
2n
> 12n / 13n
0n
```

Dẫu vậy, các toán tử số học không thể được dùng giữa bigint và number:

Javascript:

```
> a = 1n
1n
> a + 1
Uncaught TypeError: Cannot mix BigInt and other types, use explicitconversions
> a * 3
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
> a / 4
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
```

Đáp án câu 8:

Kteam xin nêu ra một vài ví dụ:

- | (OR): trả về 1 nếu có ít nhất một trong 2 bit bằng 1.
- & (AND): trả về 1 nếu cả 2 bit là 1.
- << (Zero fill left shift): Phép dịch trái.
- ~ (NOT): Đảo ngược các bit trong 1 dãy (1 à 0, 0 à 1).