String

1. Tổng quan về string.
2. Giới thiệu object String.
3. Một số thao tác cơ bản với string (props + methods).
4. Làm việc với substring.
5. Tìm kiếm và thay thế.
6. Split and Join.
7. Bài tập thực hành.

1. Tổng quan về string.

- Ngoặc đơn hoặc ngoặc kép.
- Sử dụng `backticks` để tạo chuỗi, trải dài trên nhiều dòng.
- Chuỗi trong JS phân biệt chữ hoa chữ thường.

```
const name = "LongDC";
const desc = 'Should use single quote!'; // recommend
const formatName = `My name is ${name} ${1 + 2}`;
```

- Escape ' " back slash

```
const name = 'I\'m a developer';
```

String is immutable

Không thể thay đổi 1 ký tự trong chuỗi (string), chỉ thay thế loại toàn bộ

```
let name = 'Long Do';
// name[0] = 'l'; not work
// name = 'long Do'
console.log(name);
```

2. Giới thiệu object String.

Trong JavaScript, các chuỗi (strings) là các đối tượng có thể có các thuộc tính (properties) cụ thể của chúng. Dưới đây là các thuộc tính mặc định (instance properties) của các chuỗi trong JavaScript:

Instance Properties

```
const name = 'Long Do';
name.length; // 13 (read-only)
```

Instance Methods

1. charAt(index) Lấy ký tự tại vị trí index
2. concat(str [, ...strN]) Nối chuỗi
3. includes(searchString [, position]) Có chứa chuỗi nào đó hk?
4. startsWith(searchString [, length]) Có bắt đầu với chuỗi searchString?
5. endsWith(searchString [, length]) Có kết thúc bằng chuỗi searchString?
6. indexOf(searchValue [, fromIndex]) Vị trí đầu tiên có searchValue
7. lastIndexOf(searchValue [, fromIndex]) Vị trí cuối cùng có searchValue
8. match(regexp) Liên quan tới regular expresion (tạm bỏ qua)
9. matchAll(regexp) Liên quan tới regular expresion (tạm bỏ qua)
10. padStart(targetLength [, padString]) Thêm vào đầu string
11. padEnd(targetLength [, padString]) Thêm vào cuối string
12. repeat(count) Nhân chuỗi hiện tại lên count lần
13. replace(searchFor, replaceWith) Thay thế chuỗi searchFor đầu tiên bằng replaceWith
14. replaceAll(searchFor, replaceWith) Thay thế tất cả chuỗi searchFor bằng replaceWith
15. slice(beginIndex [, endIndex]) Lấy chuỗi con
16. substring(indexStart [, indexEnd]) Lấy chuỗi con
17. split([sep [, limit]]) Tách chuỗi thành mảng các chuỗi con
18. trim() Bỏ khoảng trắng đầu + cuối string
19. trimStart() Bỏ khoảng trắng đầu string
20. trimEnd() Bỏ khoảng trắng cuối string
21. toLowerCase() Chuyển chuỗi thành chữ viết thường
22. toUpperCase() Chuyển chuỗi thành chữ viết hoa

Ví dụ:

```
'a'.padStart(5, '*'); // ****a
'ab'.padStart(5, '*'); // ***ab
'abc'.padStart(5, '*'); // **abc
'a'.padEnd(5, '*'); // a****
```

```
'ab'.padEnd(5, '*'); // ab***
'abc'.padEnd(5, '*'); // abc**
```

```
'*'.repeat(5); // *****
' Long Do '.trim(); // 'Long Do'
' Long Do '.trimStart(); // 'Long Do '
' Long Do '.trimEnd(); // ' Long Do'
```

Tham khảo

> https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

> https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

3. Một số thao tác cơ bản với string (props + methods).

Lấy ký tự tại vị trị index

```
'Long Do'.charAt(0); // L
'Long Do'.charAt(3); // g
const name = 'Long Do';
name.charAt(name.length - o); // d (the last character)
''.concat('Long'); // 'Long'
''.concat('Long', ' ', 'Do'); // 'Long Do'
```

```
'Long Do'[0]; // L
'Long Do'[3]; // g
const name = 'Long Do';
name[name.length - 1]; // o (the last character)
```

Nối chuỗi

```
const a = 'Long';
const b = 'Do';
const c = a + ' ' + b; // 'Long Do'
const d = a.concat(' ', b); // 'Long Do'
const e = `${a} ${b}` // 'Long Do' (recommended)
```

Duyệt chuỗi

```
const name = 'Long Do';
for (let i = 0; i < name.length; i++) {
```

```
  console.log(name[i])
}
```

Chuyển đổi hoa thường

```
'Long Do'.toLowerCase(); // Long Do
'Long Do'.toUpperCase(); // Long Do
```

Tìm kiếm chuỗi con

```
const name = 'Hello Po';
name.indexOf('o'); // 4
name.lastIndexOf('o'); // 7
```

Kiểm tra chứa chuỗi con

```
const name = 'Long and Do';
name.startsWith('long'); // false
name.startsWith('Long'); // true
name.startsWith('and'); // false

name.includes('Long'); // true
name.includes('and'); // true
name.includes('Do'); // true

name.endsWith('Do'); // true
name.endsWith('and'); // false
name.endsWith('do'); // false
```

4. Làm việc với substring.

- Sử dụng 1 trong 3 hàm đều được. Nhưng mà để đỡ rối, nhớ một hàm để sử dụng là slice thôi.
- substr thì theo MDN, nó được đánh dấu là deprecated.
- slice và substring sử dụng khá tương đồng
- slice thì hỗ trợ số âm. Số âm thì được đếm ngược từ cuối chuỗi.
- substring thì xem số âm là số 0
- substring cho phép start > end, trong khi slice thì không hỗ trợ.

slice(startIndex, endIndex)

Phương thức slice() trong JavaScript được sử dụng để trích xuất một phần của một chuỗi dựa trên các chỉ mục (index) bắt đầu và kết thúc cụ thể. Nó không thay đổi chuỗi gốc, mà trả về một chuỗi mới chứa phần được trích xuất.

```
```

```
string.slice(startIndex, endIndex)
```

startIndex là chỉ mục (index) của ký tự bắt đầu trích xuất. Nếu startIndex âm, nó sẽ đếm từ cuối chuỗi. Nếu không được chỉ định, nó sẽ bắt đầu từ vị trí đầu tiên của chuỗi (index 0).

endIndex là chỉ mục (index) của ký tự kết thúc trích xuất (không bao gồm). Nếu endIndex âm, nó sẽ đếm từ cuối chuỗi. Nếu không được chỉ định, hoặc nếu nó lớn hơn độ dài của chuỗi, nó sẽ kết thúc ở cuối chuỗi.

```
var str = "Hello World";

console.log(str.slice(6));      // Output: "World"
console.log(str.slice(0, 5));   // Output: "Hello"
console.log(str.slice(6, 8));   // Output: "Wo"
console.log(str.slice(-5));     // Output: "World"
console.log(str.slice(0, -6));  // Output: "Hello"
```

Trong ví dụ trên:

str.slice(6) trả về phần của chuỗi từ vị trí 6 đến cuối: "World".

str.slice(0, 5) trả về phần của chuỗi từ vị trí 0 đến 5 (không bao gồm): "Hello".

str.slice(6, 8) trả về phần của chuỗi từ vị trí 6 đến 8 (không bao gồm): "Wo".

str.slice(-5) trả về phần của chuỗi từ vị trí -5 đến cuối: "World".

str.slice(0, -6) trả về phần của chuỗi từ vị trí 0 đến -6 (không bao gồm): "Hello".

substring()

Phương thức substring() trong JavaScript cũng được sử dụng để trích xuất một phần của chuỗi, nhưng khác với slice(), substring() sử dụng chỉ mục bắt đầu và chỉ mục kết thúc để xác định phần được trích xuất.

```
string.substring(startIndex, endIndex)
```

startIndex là chỉ mục (index) của ký tự bắt đầu trích xuất. Nếu startIndex âm, nó sẽ được coi là 0.

endIndex là chỉ mục (index) của ký tự kết thúc trích xuất (không bao gồm). Nếu endIndex âm hoặc không được chỉ định, nó sẽ được coi là độ dài của chuỗi.

```
var str = "Hello World";

console.log(str.substring(6));      // Output: "World"
console.log(str.substring(0, 5));   // Output: "Hello"
console.log(str.substring(6, 8));   // Output: "Wo"
console.log(str.substring(-5));     // Output: "Hello World" (nếu startIndex âm sẽ
```

```
được coi là 0)
console.log(str.substring(0, -6));  // Output: "" (nếu endIndex âm sẽ được coi là
0)
```

str.substring(6) trả về phần của chuỗi từ vị trí 6 đến cuối: "World".

str.substring(0, 5) trả về phần của chuỗi từ vị trí 0 đến 5 (không bao gồm): "Hello".

str.substring(6, 8) trả về phần của chuỗi từ vị trí 6 đến 8 (không bao gồm): "Wo".

str.substring(-5) trả về toàn bộ chuỗi "Hello World", vì startIndex âm sẽ được coi là 0.

str.substring(0, -6) trả về một chuỗi rỗng, vì endIndex âm sẽ được coi là 0.

substr.

> tham khảo https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

5. Tìm kiếm và thay thế.

Tìm kiếm và thay thế chuỗi

- replace(searchFor, replaceWith) Tìm kiếm và thay thế một searchFor bởi replaceWith
- replaceAll(searchFor, replaceWith) Tìm kiếm và thay thế tất cả searchFor bởi replaceWith

> Lưu ý: tạm thời bỏ qua việc sử dụng với Regexp

```
'long do'.replace(' ', '-'); // long-do
'long do'.replace(' ', ''); // longdo
'long do'.replace('long', 'Long'); // Long do
'long do'.replace('long', ''); // ' do'
// replace the first match only
'long long do'.replace('long', ''); // ' long do'
'learn long do'.replaceAll(' ', '-'); // 'learn-long-do'
'learn long do long'.replaceAll('long', ''); // 'learn do '
```

```
// replace with regexp (Regular Expression)
'super 123 cool'.replaceAll(/[0-9]/gi, '')
```

6. Split and Join.

Split:

Phương thức split() được sử dụng để chia một chuỗi thành một mảng các phần tử, dựa trên một điều kiện nhất định. Điều kiện này có thể là một chuỗi hoặc một biểu thức chính quy.

```
string.split(separator, limit)
```

separator (Tùy chọn): Chuỗi hoặc biểu thức chính quy để chia chuỗi. Nếu không được cung cấp, chuỗi sẽ được chia thành một mảng gồm một phần tử duy nhất.

limit (Tùy chọn): Số nguyên dương xác định số lượng phần tử tối đa trong mảng được tạo.

Ví dụ:

```
let str = "apple,banana,orange";
let arr = str.split(","); // Chia chuỗi thành mảng các từ, phân cách bởi dấu phẩy
console.log(arr); // ["apple", "banana", "orange"]
```

Join:

Phương thức join() được sử dụng để tạo một chuỗi mới bằng cách nối các phần tử của một mảng, cách nhau bởi một chuỗi phân tách được chỉ định.

```
array.join(separator)
```

separator (Tùy chọn): Chuỗi sẽ được sử dụng để nối các phần tử trong mảng. Nếu không được cung cấp, các phần tử sẽ được nối với nhau mà không có khoảng trắng hoặc ký tự phân tách nào.

```
let arr = ["apple", "banana", "orange"];
let str = arr.join(", "); // Nối các phần tử trong mảng bằng dấu phẩy và khoảng
trắng
console.log(str); // "apple, banana, orange"
```

Nhớ rằng cả hai phương thức này không thay đổi chuỗi hoặc mảng gốc mà chúng hoạt động trên, mà tạo ra các giá trị mới.

7. Bài tập thực hành: