String Methods (Except For Search)

JavaScript String Length

The length property returns the length of a string:

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
let length = text.length;
```

Extracting String Parts

There are 3 methods for extracting a part of a string:

slice(start, end)
substring(start, end)
substr(start, length)

JavaScript String slice()

slice() extracts a part of a string and returns the extracted part in a new string.

The method takes 2 parameters: start position, and end position (end not included).

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(7,13);
```

A	p	p	l	e	,		B	a	n	a	n	a	,		K	i	w	i
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Banana

If a parameter to slice() is negative...

If a parameter is negative, the position is counted from the end of the string:

let text = "Apple, Banana, Kiwi";
let part = text.slice(-12);

Extract a part of a string counting from the end:
Banana, Kiwi

let text = "Apple, Banana, Kiwi";
let part = text.slice(-12,-6)

Banana

If a parameter to slice() is negative... (Explanation)

A	p	p	l	e	,		B	a	n	a	n	a	,		K	i	w	i
							-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

let s = "Apple, Banana, Kiwi"

s.slice(-4)
'Kiwi'

s.slice(-12,-6)
'Banana'

JavaScript String substring()

substring() is similar to slice().

The difference is that start and end values less than 0 are treated as 0 in substring().

If you omit the second parameter, substring() will slice out the rest of the string.

```
let str = "Apple, Banana, Kiwi";
console.log(str.substring(7,13));
```

The substring() method extract a part of a string and returns the extracted parts in a new string:

Banana

JavaScript String substr()

substr() is similar to slice().

The difference is that the second parameter specifies the length of the extracted part.

If you omit the second parameter, substr() will slice out the rest of the string.

```
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.substr(7,6);
```

Banana

If the first parameter to substr() is negative...

If the first parameter is negative, the position counts from the end of the string.

```
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.substr(-4);
```

Kiwi

Problem

Let s = “In publishing and graphic design, Lorem ipsum is a placeholder text commonly used to demonstrate the visual form of a document or a typeface without relying on meaningful content. Lorem ipsum may be used as a placeholder before final copy is available.”

Console.log(s.substr(34, 11));

Lorem ipsum

Console.log(s.substr(3, 10));

publishing

Replacing String Content

The replace() method replaces a specified value with another value in a string:

Note: replace() by default replaces only the first occurence unless specified otherwise.

```
let text = 'Please visit Microsoft!'
let new_text = text.replace("Microsoft", "W3Schools");

new_text
'Please visit W3Schools!'

let text = 'please visit microsoft and microsoft.'

let new_text = text.replace('microsoft', 'w3schools');
new_text
'please visit w3schools and microsoft.'
```

Note About replace()

The replace() method does not change the string it is called on.

The replace() method returns a new string.

The replace() method replaces only the first match

If you want to replace all matches, use a regular expression with the /g flag set.

As an example:

```
let text = “Please visit Microsoft and Microsoft!”;
new_text = text.replace(/Microsoft/g,"W3Schools");
```

A Glimpse of Regular Expressions

```
let text = “Please visit Microsoft and Microsoft!”;
new_text = text.replace(/Microsoft/g,"W3Schools");
```

What do we learn about Regular Expressions from the above code:

1. Regular Expressions are used usually in search operations on a text / string data.

As in: if we want to do a replacement of word ‘Microsoft’ in a text, would we not first have to look for ‘Microsoft’ in the text? Answer is: yes, we would.

2. A regular expression is written between two ‘/’ foreslashes.

As in: /Microsoft/

3. What is ‘g’ in ‘/Microsoft/g’?
It stands for ‘global’.

JavaScript String ReplaceAll()

In 2021, JavaScript introduced the string method replaceAll():

```
let text = "I love cats. Cats are very easy to love. Cats are very popular."
text = text.replaceAll("Cats","Dogs");
text = text.replaceAll("cats","dogs");
```

The replaceAll() method allows you to specify a regular expression instead of a string to be replaced.

If the parameter is a regular expression, the global flag (g) must be set, otherwise a TypeError is thrown.

```
text = text.replaceAll(/Cats/g,"Dogs");
text = text.replaceAll(/cats/g,"dogs");
```

Note About replaceAll()

replaceAll() is an ES2021 feature.

replaceAll() does not work in Internet Explorer.

A Simple Use Case of RegEx

let text = "I love cats. Cats are very easy to love. Cats are very popular."

text = text.replaceAll("Cats","Dogs");
text = text.replaceAll("cats","dogs");

Can we do this replacement in a single line of code?
No. Because we are replacing ‘cats’ with small ‘c’ with ‘dogs’ with small ‘d’.
And we are replacing ‘Cats’ with big ‘D’ with ‘Dogs’ with capital ‘D’.

What about this replacement: cats -> **** and Cats -> ****

The answer now is: Yes

let text = "I love cats. Cats are very easy to love. Cats are very popular."

text.replaceAll(/cats|Cats/g, "****")
'I love ****. **** are very easy to love. **** are very popular.'

A Simple Use Case of RegEx

A regular expression can capture ‘cats’ and ‘Cats’ at the same time but not replace them with two different values.

This is because a regular expression is like a concept.
Rather than a ‘string’ you want to look for using exact match.

Converting to Upper and Lower Case

A string is converted to upper case with toUpperCase().

A string is converted to lower case with toLowerCase().

JavaScript String toUpperCase()

```
let text1 = "Hello World!";
let text2 = text1.toUpperCase();
```

JavaScript String toLowerCase()

```
let text1 = "Hello World!";       // String
let text2 = text1.toLowerCase();  // text2 is text1 converted to lower
```

JavaScript String concat()

concat() joins two or more strings:

```
let text1 = "Hello";
let text2 = "World!";
let text3 = text1.concat(" ",text2);
```

Hello World!

The concat() method can be used instead of the plus operator. These two lines do the same:

```
text = "Hello" + " " + "World!";
text = "Hello".concat(" ", "World!");
```

Strings are immutable

All string methods return a new string. They don't modify the original string.

Formally said:

Strings are immutable: Strings cannot be changed, only replaced.

JavaScript String trim()

The trim() method removes whitespace from both sides of a string:

```
let text1 = "      Hello World!      ";
let text2 = text1.trim();
```

```
> let text1 = "      Hello World!      ";
  let text2 = text1.trim();
  text2
'Hello World!'
```

trimStart() and trimEnd()

ECMAScript 2019 added the String method trimStart() to JavaScript.

The trimStart() method works like trim(), but removes whitespace only from the start of a string.

```
let text1 = "     Hello World!     ";
let text2 = text1.trimStart();
```

trimEnd()

ECMAScript 2019 added the string method trimEnd() to JavaScript.

The trimEnd() method works like trim(), but removes whitespace only from the end of a string.

JavaScript String Padding

JavaScript String padStart():

The padStart() method pads a string from the start.

It pads a string with another string (multiple times) until it reaches a given length.

```
let text = "5";
text = text.padStart(4,"0");
```

0005

Padding a number

The padStart() method is a string method.

To pad a number, convert the number to a string first.

See the example below.

```
let numb = 5;
let text = numb.toString();
let padded = text.padStart(4,"0");
```

JavaScript String padEnd()

The padEnd() method pads a string from the end.

It pads a string with another string (multiple times) until it reaches a given length.

```
let text = "5";
let padded = text.padEnd(4,"0");
```

```
let text = "5";
let padded = text.padEnd(4,"x");
```

Extracting String Characters

There are 3 methods for extracting string characters:

charAt(position)
charCodeAt(position)
Property access [ ]

JavaScript String charAt()

The charAt() method returns the character at a specified index (position) in a string:

```
let text = "HELLO WORLD";
let char = text.charAt(0);
```

JavaScript String charCodeAt()

The charCodeAt() method returns the unicode of the character at a specified index in a string:

The method returns a UTF-16 code (an integer between 0 and 65535).

```
let text = "HELLO WORLD";
let char = text.charCodeAt(0);
```

Getting Character Using Property Access

ECMAScript 5 (2009) allows property access [ ] on strings:

```
let text = "HELLO WORLD";
let char = text[0];
```

Note:
Property access might be a little unpredictable:

- It makes strings look like arrays (but they are not)
- If no character is found, [ ] returns undefined, while charAt() returns an empty string.
- It is read only. str[0] = "A" gives no error (but does not work!)

Problem

Q: Demonstrate using code:

“If no character is found, [ ] returns undefined, while charAt() returns an empty string.”

```
> let s = "hello world"
< undefined
> console.log(s.charAt(15))

< undefined
> console.log(s[15])
  undefined
< undefined
```

Converting a String to an Array

If you want to work with a string as an array, you can convert it to an array.

JavaScript String split()

A string can be converted to an array with the split() method:

text.split(",") // Split on commas
text.split(" ") // Split on spaces
text.split("|") // Split on pipe

If the separator is omitted, the returned array will contain the whole string in index [0].

If the separator is "", the returned array will be an array of single characters.

```
> let s = 'I love JS'
< undefined
> s.split(" ")
< ▶ (3) ['I', 'love', 'JS']
> let r = "jim,john,jack"
< undefined
> r.split(",")
< ▶ (3) ['jim', 'john', 'jack']
```

Name	Description
charAt()	Returns the character at a specified index (position)
charCodeAt()	Returns the Unicode of the character at a specified index
concat()	Returns two or more joined strings
constructor	Returns the string's constructor function
endsWith()	Returns if a string ends with a specified value
fromCharCode()	Returns Unicode values as characters
includes()	Returns if a string contains a specified value
indexOf()	Returns the index (position) of the first occurrence of a value in a string
lastIndexOf()	Returns the index (position) of the last occurrence of a value in a string
length	Returns the length of a string
localeCompare()	Compares two strings in the current locale
match()	Searches a string for a value, or a regular expression, and returns the matches
prototype	Allows you to add properties and methods to an object
repeat()	Returns a new string with a number of copies of a string
replace()	Searches a string for a value, or a regular expression, and returns a string where the values are replaced
search()	Searches a string for a value, or regular expression, and returns the index (position) of the match