JAVASCRIPT STRINGS

STRINGS

- Javascript strings are for storing and manipulating text.
- A Javascript string is zero or more characters written inside quotes.

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

- Display length of string using innerHTML

```
let text = "we are the so-called "kings" from the north.";
```

- Display the String

EXAMPLE

The solution to avoid this problem, is to use the **backslash escape character**.
The backslash (\) escape character turns special characters into string characters:
The sequence \" inserts a double quote in a string:

```
let text = "We are the so-called \"kings\" from the north.";
```

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

EXAMPLE

- But strings can also be defined as objects with the keyword new:
  - let y = new string("john");
- Not a good practice to create strings as objects</h2>
- Javascript objects cannot be compared.</p>

```
let x = new String("John");
let y = new String("John");
```

```
<script>
let x = new String("John");  // x is an object
let y = new String("John");  // y is an object
document.getElementById("demo").innerHTML = (x==y);
```

JAVASCRIPT STRING METHODS

1. CHARAT(INDEX)
2. STRING SLICE()
3. STRING SUBSTRING()
4. STRING SUBSTR()
5. STRING REPLACE()
6. STRING TOUPPERCASE()
7. STRING TRIM()
8. STRING SPLIT()

CHARAT(INDEX)

First position is 0.

<script>

var str="javascript";

document.write(str.charat(2));

</script>

Output: v

string slice()

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

let text = "apple, banana, kiwi";

let part = text.slice(7, 13); Banana //slice out a portion of a string from position 7 to position 13:

let part = text.slice(7); Banana, Kiwi //Omiting the second parameter, the method will slice out the rest of the string:

let part = text.slice(-12); Banana, Kiwi //if a parameter is negative, the position is counted from the end of the string:

let part = text.slice(-12, -6); Banana //slices out a portion of a string from position -12 to position -6:

substr()

substr(): The JavaScript string substr() method fetch the part of the given string and return the new string

This method doesn't make any change in the original string.

Syntax: string.substr(start,length) where length represents the number of characters to fetch.

```
var str="Javatscript";

document.writeln(str.substr(0,4));
document.writeln(str.substr(5,5));
document.writeln(str.substr(5));
document.writeln(str.substr());
document.writeln(str.substr(-5,5));
```

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

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace("Microsoft", "MSRIT");
```

Output: Please visit MSRIT and Microsoft!

```
let newText = text.replace("MICROSOFT", "W3Schools"); //the replace() method is case sensitive.
```

Output:?

```
let text = "Please visit Microsoft!";
let newText = text.replace(/MICROSOFT/i, "W3Schools"); //use a regular expression with an /i flag
```

```
let newText = text.replace(/Microsoft/g, "W3Schools");  //use a regular expression with an /g flag
```

replaceAll()

```
<script>
let text = "i love cats. CATS are very easy to love. cats are very popular."
text= text.replaceall("cats","dogs");
document.write(text);
text= text.replaceall(“CATS","dogs");
document.getelementbyid("demo").innerhtml = text;
</script>
```

OTHER STRING METHODS

```
let text1 = "hello world!";

let text2 = text1.toUpperCase()
```



```
LET TEXT1 = "HELLO WORLD!";
LET TEXT2 = TEXT1.toLowerCase();
```

`concat()` joins two or more strings:

```
LET TEXT1 = "HELLO";
LET TEXT2 = "WORLD";
LET TEXT3 = TEXT1.CONCAT(" ", TEXT2);
```

STRING TRIM()

The trim() method removes whitespace from both sides of a string:
The trim() method does not change the original string

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


Display the strings and dislay the length of each string

trimLeft()

The **trimLeft()** method removes the whitespace only from the left of the string.

trimRight()

On the other hand, the **trimRight()** method removes the whitespace only from the right of the string.

SPLIT()

The split() method splits a string into an array of substrings.

The split() method returns the new array.

The split() method does not change the original string.

Syntax : string.split(*separator*, *limit*)

If (" ") is used as separator, the string is split between words, The limit is zero or positive integer.
It specifies the number of substrings. The split() method will stop when the number of substrings equals to the limit

```
let text = "How are you doing today?";
const myArray = text.split(" ");
```

```
let text = "How are you doing today?";
const myArray = text.split(" ");
Document.write(myArray[1]);
```

```
let text = "How are you doing today?";
const myArray = text.split(" ", 3);
Document.write(myArray)
```