MIND HUB.

Strings Methods

Index

1 Introduction

2 Integrated to JS

3 Methods

Introduction

To be able to operate with Strings, it is essential **to know the methods** we have.

Using the specific methods we will obtain better results, simpler and more elegant code.

The strings as well as the arrays also have a length property that allow us to obtain the amount of characters contained in the string.

Integrated to JS

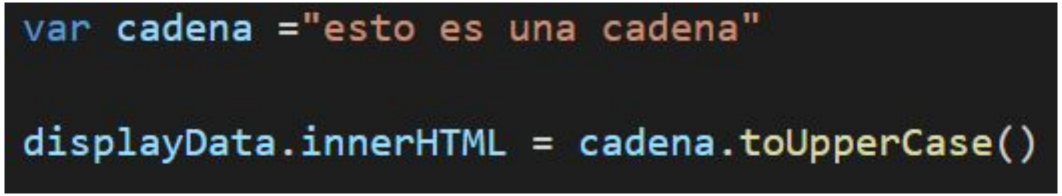
JavaScript comes with built-in methods for operating on Strings. You may already be using them.
Let's take a look at some of them to understand how they work.

- .toUpperCase()
- .toLowerCase()
- .trim()
- .charAt()
- .slice()
- .substring()
- .replace()
- .indexOf()
- .split()
- .startsWith()

There are many more, we will incorporate more of them as the modules run, for now let's focus on these and analyze their advantages.

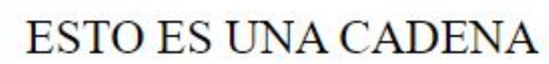
.toUpperCase()

The toUpperCase method converts all characters in the string to **uppercase** and returns them. It does not change the original string.



```
var cadena ="esto es una cadena"

displayData.innerHTML = cadena.toUpperCase()
```

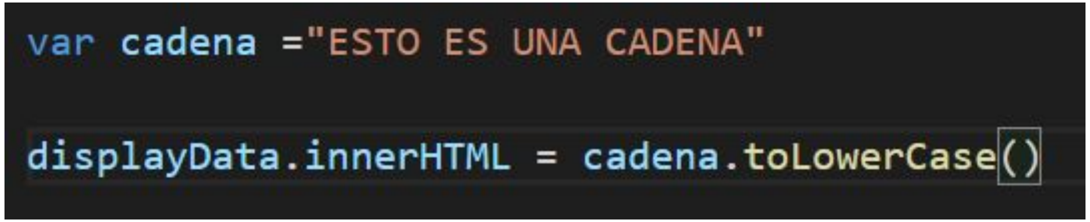
Result



```
ESTO ES UNA CADENA
```

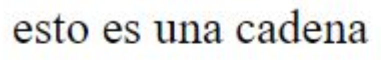
.toLowerCase()

The toLowerCase method converts all characters in the string to **lowercase** and returns them. It does not change the original string.



```
var cadena ="ESTO ES UNA CADENA"

displayData.innerHTML = cadena.toLowerCase()
```

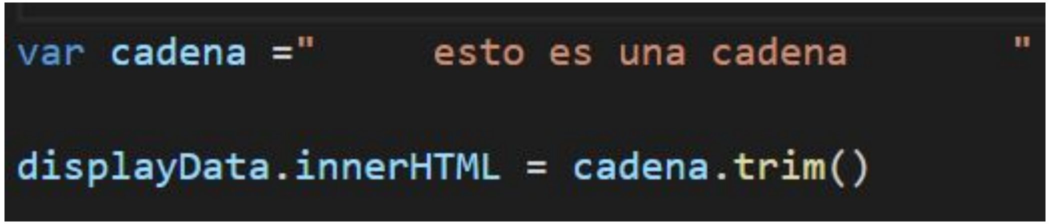
Result



```
esto es una cadena
```

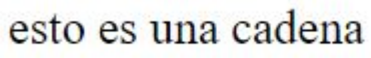
.trim()

The trim method **removes leading and trailing blanks** from the string. It is an in-place operation, that is, it updates the original string.



```
var cadena ="      esto es una cadena        "

displayData.innerHTML = cadena.trim()
```

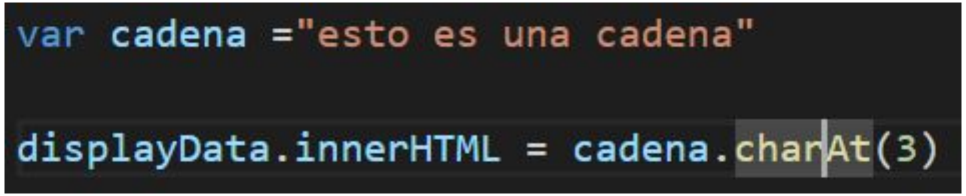
Result



```
esto es una cadena
```

.charAt(index)

The charAt method **returns the character at the given index**. It returns an empty string if the index is invalid.



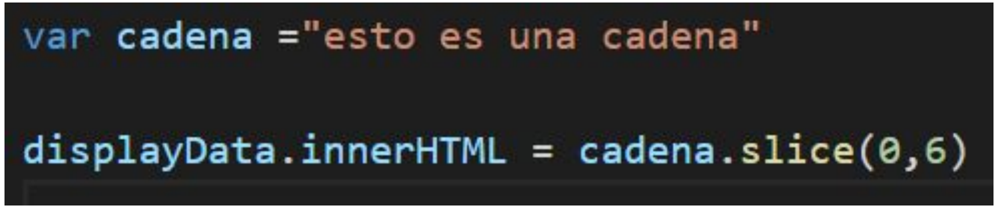
```
var cadena ="esto es una cadena"

displayData.innerHTML = cadena.charAt(3)
```

Result

```
o
```

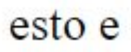
.slice(startIndex, endIndex)

The slice method **returns the substring of the string from startIndex to endIndex (not included)**. The string.slice(0, 6) returns the substring from the 0th index to the 5th index.
The slice method will accept **negative indexes** as well. Negative indices are counted from the end of the string.



```
var cadena ="esto es una cadena"

displayData.innerHTML = cadena.slice(0,6)
```

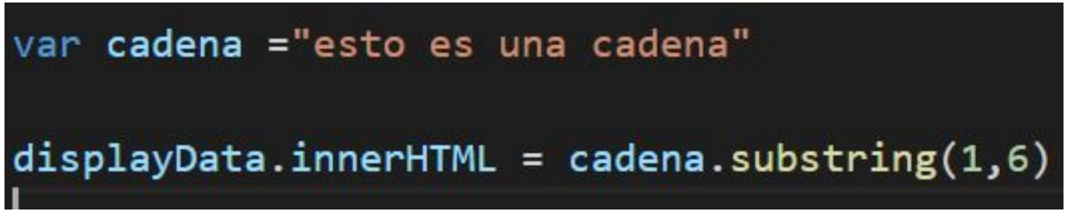
Result



```
esto e
```

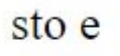
.substring(startIndex, length)

The substr method is similar to the slice method. The only difference is that the substr method **accepts the length of the substring to be extracted from the original string**.



```
var cadena ="esto es una cadena"

displayData.innerHTML = cadena.substring(1,6)
```

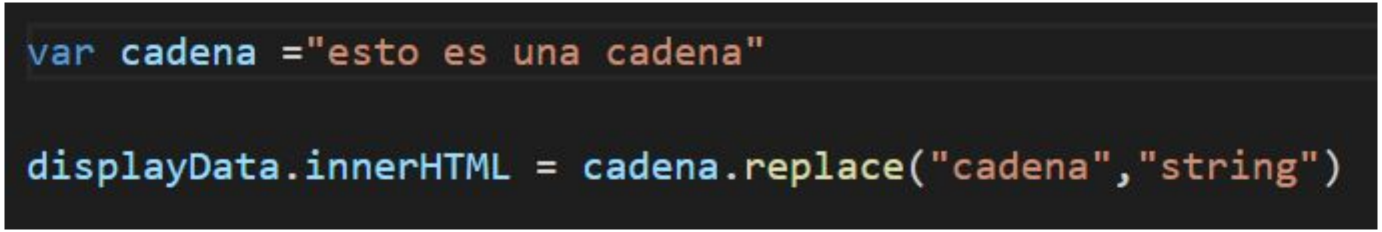
Result



```
sto e
```

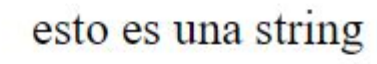
.replace(substring, newSubstring)

The replace method replaces the first instance of the **substring** with **newSubstring**.



```
var cadena ="esto es una cadena"

displayData.innerHTML = cadena.replace("cadena","string")
```

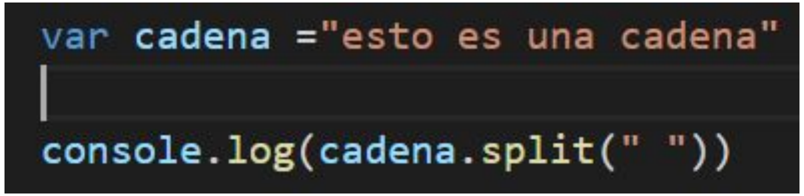
Result



```
esto es una string
```

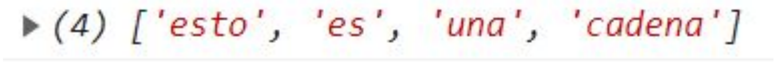
.split(substring)

The split method splits the given string into the substring and **returns the parts as an array**.



```
var cadena ="esto es una cadena"

console.log(cadena.split(" "))
```

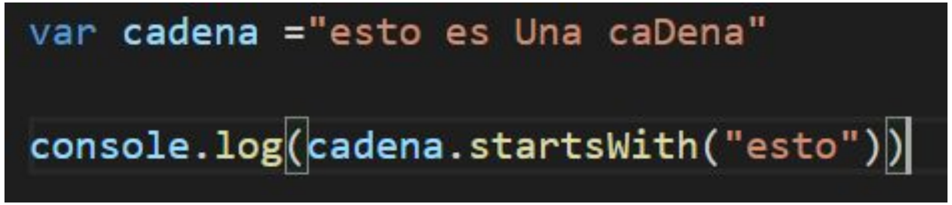
Result



```
▶ (4) ['esto', 'es', 'una', 'cadena']
```

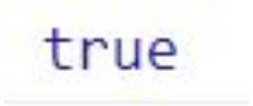
.startsWith(value)

The startsWith() method indicates **whether a string begins with the characters of a particular string**, returning true or false accordingly.



```
var cadena ="esto es Una caDena"

console.log(cadena.startsWith("esto"))
```

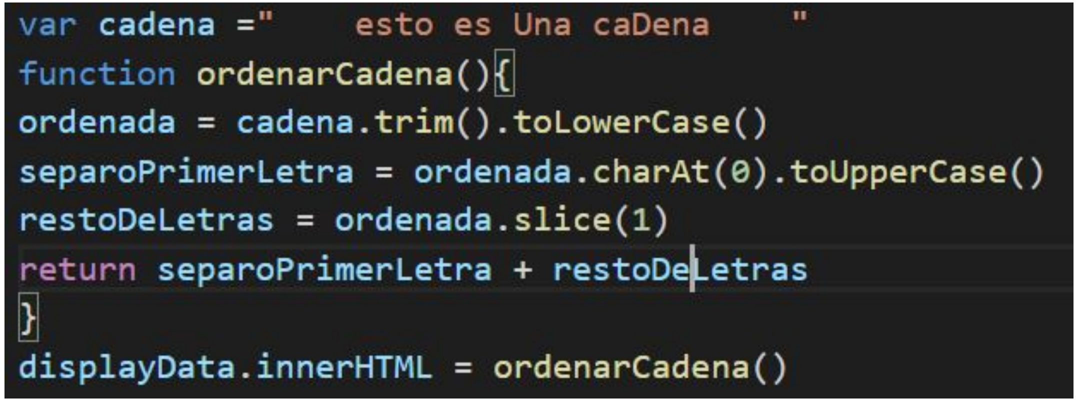
Result



```
true
```

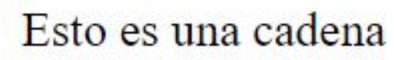
Combination of Methods

All methods can be combined with each other to obtain the expected result, you must always take into account the **order in which each method is called**.



```
var cadena ="     esto es Una caDena     "
function ordenarCadena(){
ordenada = cadena.trim().toLowerCase()
separoPrimerLetra = ordenada.charAt(0).toUpperCase()
restoDeLetras = ordenada.slice(1)
return separoPrimerLetra + restoDeLetras
}
displayData.innerHTML = ordenarCadena()
```

Result



Esto es una cadena

As you can see, each step has a logical order, so that the following step obtains the result you are looking for, e.g.: if initially I do not use trim, the charAt(0), it would no longer be the letter e but a space.

¡Thank you a lot!

MIND
HUB.