**String properties & methods**

As everything in JS is an object with properties and methods, string data has one property and several methods.

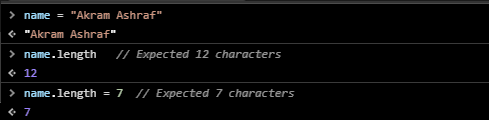
**First; Property:**

**Stringname.length**

This property is used to:

1. **Print** number of characters found in this string.
2. **Set** the string length you want.

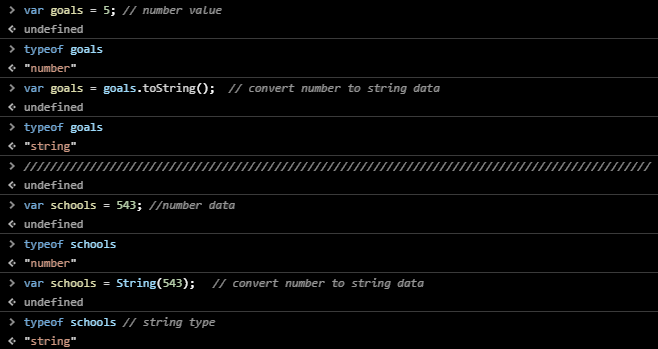
**Example:**



**Second; Methods:**

1. **Stringname.toString(); & String(given data);**

These methods are used to convert any given data to string data.



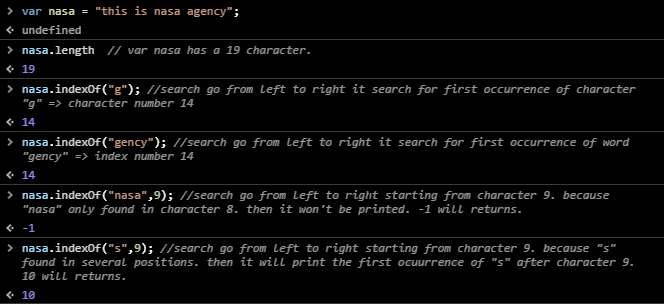
1. **Stringname.indexOf(“n,ST”);**

* This method is used to **return** the index number (n) where n is a **word|character** in this string.

**Note:** if the word | character is not found -1 will returns.

* Starting point is an **optional** setting and it used to start the searching for **n** from a specific index number. If you don’t determine starting point then searching will be from the **beginning** of the string to the **end** of it.
* If the word | character is present more than once, the indexOf method returns the position of the **first** occurrence.

**Example:**

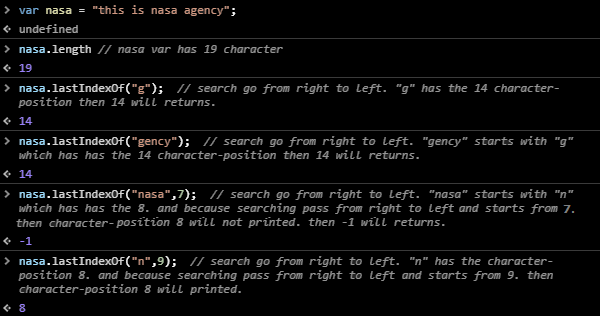


1. **Stringname.lastIndexOf(N, ST);**

* This method is used to **return** the index number (n) where n is a **word|character** in this string.

**Note:** if the word | character is not found -1 will returns.

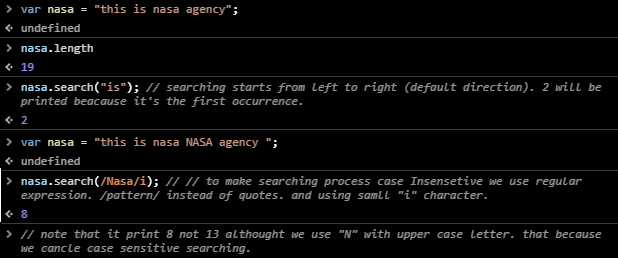
* Starting point is an optional setting and it used to start the searching from a specific point. If you don’t determine starting point then searching will be from the **ending** of the array to the **beginning** of it.
* If the word | character is present more than once, the lastIndexOf method returns the position of the **last** occurrence.



1. **Stringname.search(“word | character”);**

This method is also used to search for a specific word or character.

**Note:** use regular expression to make searching case Insensetive.



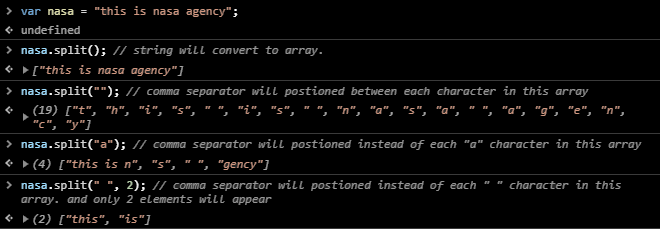
1. **Stringname.split(“Position”, limit);**

This method is used to convert string to **array** then **split** this array to multiply divisions.

**Note:**

1. Position means “where put comma separator”. Optional setting.
   * If you use empty value “” then comma will positioned between each character in this array.
2. Limit means “How long length I want to appear”. Optional setting
3. The cutting part was put in a new string; the main string remains as it’s.

**Example:**

****

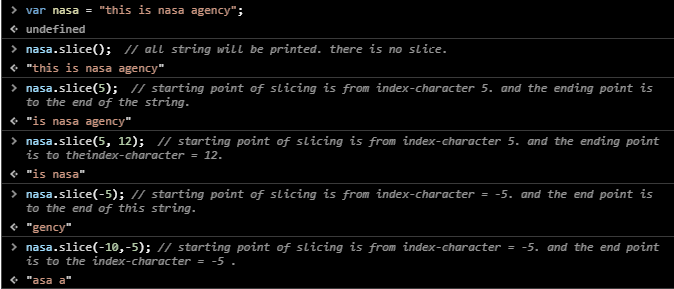
1. **Stringname.slice(S.p, E.p);**

This method is used to cutting (slitting) the strings from a specific starting point to a specific ending point (ending point won’t be printed).

**Note**:

* The cutting part was put in a new string; the main string remains as it’s.
* Slice doesn’t support swiping. Where index-of endpoint must always > index-of starting point

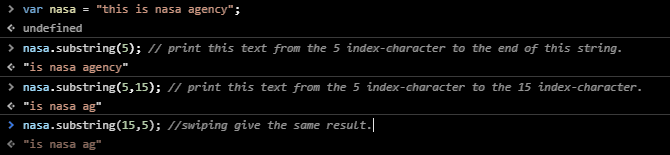
**Example:**



1. **Stringname.subString(S.p, E.p);**

This method is used to cutting (slitting) the strings from a specific starting point to a specific ending point (ending point won’t be printed) Like **slice** method. The difference between this method and method number 7 is that you can make a swipe in the first one.

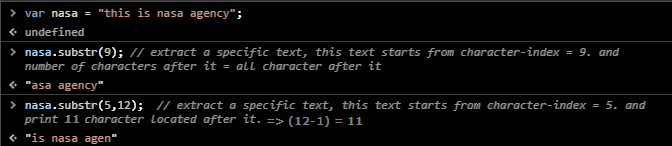
**Example:**

****

1. **Stringname.substr(S.p, length );**

This method is used to extract some text from the string.

**Note:** **length** referee to number of character i want to appear after the starting point minus 1. (number of characters+1). So if you want to print 13 characters after the starting point you should write 14. Because starting point character also calculated in this process. **It may be deprecated,**



1. **Converting string letter cases**

* **String.toLowerCase();**

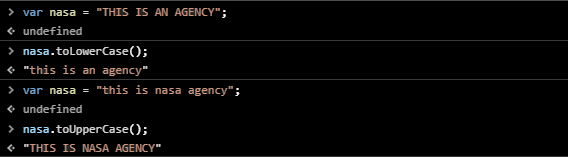
This method is used to convert letter cases of a string to lowercases.

* **String.toUpperCase();**

This method is used to convert letter cases of a string to uppercase.

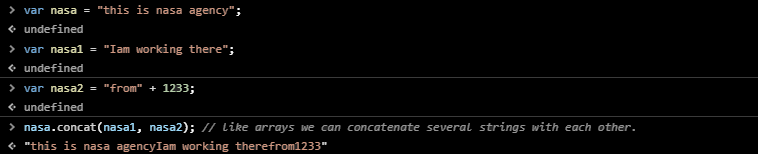
The converting part was put in a new string; the main string remains as it’s.

**Example:**



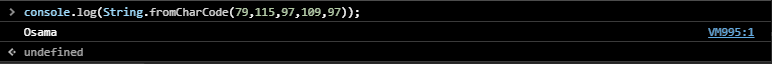
1. **Stingname.concat();**

This method is used to concatenate several strings with each other.



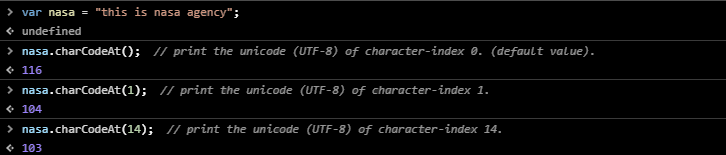
1. **String.fromCharCode(letter code1, letter code2, letter code3,…);**

This method enables you to write strings with Unicode instead of letters.



1. **Stringname.charCodeAt(character-index);**

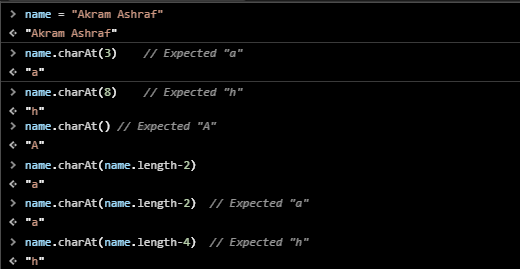
This method is used to print the Unicode of a specific character in the current string.



1. **Stringname.charAt();**

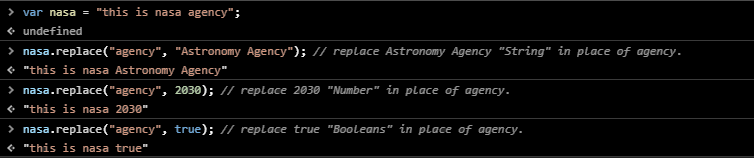
This method enable you to search for a specific **word | character** through character- index number.

This method is an equivalent to => Stringname.charAt(Stringname.length-character index);



1. **Stringname.replace(old value, new value);**

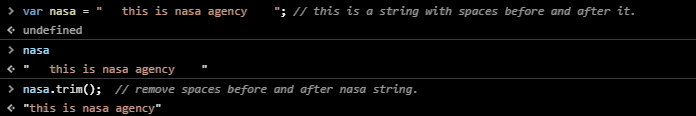
This method is used to replace value with another value in the current string.



1. **Stringname.trim();**

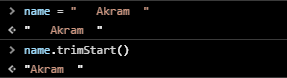
This method is used to remove spaces in the beginning and in the end of the current string.

**Note:** this method doesn’t remove spaces within the string itself.



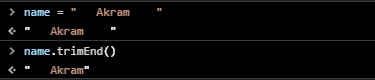
1. **Stringname.trimStart();**

This method is used to remove spaces from the beginning of the string.



1. **Stringname.trimEnd();**

This method is used to remove spaces from the ending of the string.



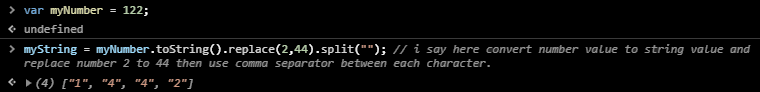
1. **Stringname.link(“URL”);**

This method is used to add links to your document. It’s similar to <a> element.



1. **Stringname.method1.method2.method3….;**

This method is known as a chain method used to specify several methods to only one Stringname to shorten the code.

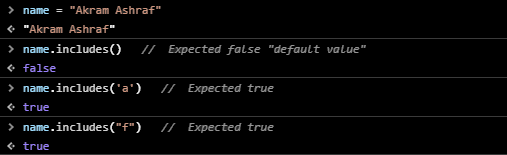


1. **Stringname.includes(“n”);**

This method is used to check the presence of a specific character (“n”) inside the string; if it presence return ‘true’, else return ‘false’.

**Note:** the quotes around the character.

**Example:**

****

1. [**Compare methods**](Compare%20strings.docx)press to see more details.