JavaScript

Strings

1.length

Ex: var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

document.getElementById("demo").innerHTML = txt.length;

2.Special Characters

The backslash escape character turns special characters into string characters:

Ex: var x = 'It\'s alright';

var y = "We are the so-called \"Vikings\" from the north.";

document.getElementById("demo").innerHTML = x + "<br>" + y;

Output:It's alright  
We are the so-called "Vikings" from the north.

## 3.Strings Can be Objects

## Normally :**varx = "John" // type is string**

## **Using new keyword:vary = new String("John") // type is object**

// (x == y) is true because x and y have equal values

// (x === y) is false because x and y have different types (string and object)

Ex2: var x = new String("John");               
var y = new String("John");

document.getElementById("demo").innerHTML = (x==y);

// (x == y) is false because x and y are different objects

// (x === y) is false because x and y are different objects

NOTE:

1. Comparing two JavaScript objects will **always** return false.
2. Don't create strings as objects. It slows down execution speed.  
   The **new** keyword complicates the code. This can produce some unexpected results:

4.String methods and properties

1.indexOf()

Syn: *string*.indexOf(searchvalue,start)

The indexOf() method returns the position of the first occurrence of a specified value in a string.

This method returns -1 if the value to search for never occurs.

**Note:** The indexOf() method is case sensitive.

Ex1: var str = "Hello world, welcome to the universe.";

var n = str.indexOf("welcome");

document.getElementById("demo").innerHTML = n; //13

Ex2: Find the first occurrence of the letter "e" in a string, starting the search at position 5:

var str = "Hello world, welcome to the universe.";  
 var n = str.indexOf("e", 5);//14

2.lastIndexOf()

The lastIndexOf() method returns the position of the last occurrence of a specified value in a string.

Case sensitive

This method returns -1 if the value to search for never occurs.

*string*.lastIndexOf(searchvalue,start)

EX1: var str = "Hello planet earth, you are a great planet.";

var n = str.lastIndexOf("planet");//36  
var n = str.lastIndexOf("planet", 20); //6

3.search()

*string*.search(searchvalue)

The two methods, indexOf() and search(), are equal.

They accept the same arguments (parameters), and they return the same value.

The two methods are equal, but the search() method can take much more powerful search values.

The search value can be string or a regular expression.

This method returns -1 if no match is found.

Ex: var str = "Mr. Blue has a blue house";  
 var n = str.search("blue");//15

Case-sensitive

var n = str.search(/blue/i);//4

Extracting String Parts

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

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

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

* The method takes 2 parameters: the starting index (position), and the ending index (position).

Ex1: var str = "Apple, Banana, Kiwi";

var res = str.slice(7,13);

document.getElementById("demo").innerHTML = res;//Banana

note: here 13 means it will take 12 because

ex: var str = "Hello world!";  
var res = str.slice(0, 1);//H

var str = "Hello world!";  
var res = str.slice(-1);//!

Ex2: If a parameter is negative, the position is counted from the end of the string.

var str = "Apple, Banana, Kiwi";  
var res = str.slice(-12, -6);//Banana

NOTE: here -6 means -7

Ex3: If you omit the second parameter, the method will slice out the rest of the string:

var str = "Apple, Banana, Kiwi";

var res = str.slice(7); or var res = str.slice(-12)//counting from end

document.getElementById("demo").innerHTML = res;// Banana, Kiwi

2. **substring()** is similar to slice().

The difference is that substring() cannot accept negative indexes.

EX: var str = "Apple, Banana, Kiwi";  
var res = str.substring(7, 13);//Banana

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

Ex: If "start" is greater than "end", this method will swap the two arguments, meaning str.substring(1, 4) == str.substring(4, 1).

var str = "Hello world!";

var res = str.substring(1, 4); == var res = str.substring(4, 1);//ell

ex3: If "start" is less than 0, it will start extraction from index position 0:

var str = "Hello world!";  
var res = str.substring(-3);// Hello world!

1. **substr()** is similar to slice().

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

Ex: var str = "Apple, Banana, Kiwi";  
var res = str.substr(7, 7);// Banana,

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

The second parameter can not be negative, because it defines the length.

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