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 + "
" + 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
```

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!
```

3. **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.

5.replace()

The replace() method searches a string for a specified value, or a *regular expression*, and returns a new string where the specified values are replaced.

Note: If you are replacing a value (and not a *regular expression(EX:* (/blue/g*)*), only the first instance of the value will be replaced. To replace all occurrences of a specified value, use the global (g) modifier (see "More Examples" below).

string.replace(*searchvalue, newvalue*)

EX1: var str = document.getElementById("demo").innerHTML;

var res = str.replace("Microsoft", "W3Schools");//W3Schools!

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

EX2: global replacement:

```
var str = "Mr Blue has a blue house and a blue car";
var res = str.replace(/blue/g, "red");//Mr Blue has a red house and a red car
```

EX3: case-*insensitive*

```
var str = "Mr Blue has a blue house and a blue car";
var res = str.replace(/blue/gi, "red");//Mr red has a red house and a red car
```

EX4: function myFunction() {

var str = document.getElementById("demo").innerHTML;

var res = str.replace(/blue|house|car/gi, function myFunction(x){return x.toUpperCase();});

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

}

OUT: Mr BLUE has a BLUE HOUSE and a BLUE CAR.

6.Extracting String Characters

charAt()

string.charAt(*index*)

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

EX1: var str = "HELLO WORLD";

var res = str.charAt(0)

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

E2: Tip: The index of the last character in a string is *string*.length-1, the second last character is *string*.length-2, and so on

```
var str = "HELLO WORLD";

var res = str.charAt(str.length-1);//W
```

7.Converting a String to an Array

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

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

```
txt.split(",");      // Split on commas
txt.split(" ");      // Split on spaces
txt.split("|");      // Split on pipe
```

EX1: var str = "How are you doing today?";

var res = str.split(" ");

document.getElementById("demo").innerHTML = res;//
How,are,you,doing,today?

EX2: **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:

```
var str = "Hello";

var arr = str.split("");

var text = "";

var i;

for (i = 0; i < arr.length; i++) {

  text += arr[i] + "<br>"

}

document.getElementById("demo").innerHTML = text;
```

// H
e
l
l
o

Ex4: var str = "How are you doing today?";

document.getElementById("demo").innerHTML = res;// H,w are y,u d,ing t,day?

Ex5: Omit the separator parameter:

```
var str = "How are you doing today?";

  var res = str.split();

  document.getElementById("demo").innerHTML = res;// How are you doing today?
```

Ex5: var str = "How are you doing today?";
var res = str.split(" ", 3);// H,w are y,u d,ing t,day?

Other methods/properties

toUpperCase(), toLowerCase()

Ex: var str = "Hello World!";

var res = str.toUpperCase();//HELLO WORLD!

```
var res = str.toLowerCase();//hello world!
```

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

valueOf()

Returns the primitive value of a String object

EX:
```
var str = "Hello World!";
var res = str.valueOf();
```

document.getElementById("demo").innerHTML = res;// Hello World!

Note: This method is usually called automatically by JavaScript behind the scenes, and not explicitly in code.

match()

Searches a string for a match against a regular expression, and returns the matches

EX1:
```
var str = "The rain in SPAIN stays mainly in the plain";
var res = str.match(/ain/g);// ain,ain,ain

var res = str.match(/ain/);// ain
```

EX2: Perform a global, case-insensitive search for "ain":

```
var res = str.match(/ain/gi);// ain,AIN,ain,ain
```

Note: If the regular expression does not include the *g* modifier (to perform a *global* search), the match() method will return only the first match in the string.

This method returns *null* if no match is found.

repeat()

Returns a new string with a specified number of copies of an existing string

EX:
```
var str = "Hello world!";
    document.getElementById("demo").innerHTML = str.repeat(2);// Hello
world!Hello world!
```

trim()

Removes whitespace from both ends of a string

```
var str = "       Hello World!        ";
alert(str.trim());//Hello World!
```

concat()

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);

document.getElementById("demo").innerHTML = res;// Hello world!
```