**Examples of the JavaScript string functions in action**

**1. charAt(x)** Returns the character at the “x” position within the string.

//charAt(x)

var myString = 'jQuery FTW!!!';

console.log(myString.charAt(7));

//output: F

**2. charCodeAt(x)** Returns the Unicode value of the character at position “x” within the string.

//charAt(position)

var message="jquery4u"

//alerts "q"

alert(message.charAt(1))

**3. concat(v1, v2,…)**Combines one or more strings (arguments v1, v2 etc) into the existing one and returns the combined string. Original string is not modified.

//concat(v1, v2,..)

var message="Sam"

var final=message.concat(" is a"," hopeless.")

alert(final)

**4. fromCharCode(c1, c2,…)**Returns a string created by using the specified sequence of Unicode values (arguments c1, c2 etc). Method of String object, not String instance. For example: String.fromCharCode().

//fromCharCode(c1, c2,...)

console.log(String.fromCharCode(97,98,99,120,121,122))

//output: abcxyz

console.log(String.fromCharCode(72,69,76,76,79))

//output: HELLO

**5. indexOf(substr, [start])**Searches and (if found) returns the index number of the searched character or substring within the string. If not found, -1 is returned. “Start” is an optional argument specifying the position within string to begin the search. Default is 0.

//indexOf(char/substring)

var sentence="Hi, my name is Sam!"

if (sentence.indexOf("Sam")!=-1)

alert("Sam is in there!")

**6. lastIndexOf(substr, [start])**Searches and (if found) returns the index number of the searched character or substring within the string. Searches the string from end to beginning. If not found, -1 is returned. “Start” is an optional argument specifying the position within string to begin the search. Default is string.length-1.

//lastIndexOf(substr, [start])

var myString = 'javascript rox';

console.log(myString.lastIndexOf('r'));

//output: 11

**7. match(regexp)**Executes a search for a match within a string based on a regular expression. It returns an array of information or null if no match is found.

//match(regexp) //select integers only

var intRegex = /[0-9 -()+]+$/;

var myNumber = '999';

var myInt = myNumber.match(intRegex);

console.log(isInt);

//output: 999

var myString = '999 JS Coders';

var myInt = myString.match(intRegex);

console.log(isInt);

//output: null

**8. replace(regexp/substr, replacetext)** Searches and replaces the regular expression (or sub string) portion (match) with the replaced text instead.

//replace(substr, replacetext)

var myString = '999 JavaScript Coders';

console.log(myString.replace(/JavaScript/i, "jQuery"));

//output: 999 jQuery Coders

//replace(regexp, replacetext)

var myString = '999 JavaScript Coders';

console.log(myString.replace(new RegExp( "999", "gi" ), "The"));

//output: The JavaScript Coders

**9. search(regexp)**Tests for a match in a string. It returns the index of the match, or -1 if not found.

//search(regexp)

var intRegex = /[0-9 -()+]+$/;

var myNumber = '999';

var isInt = myNumber.search(intRegex);

console.log(isInt);

//output: 0

var myString = '999 JS Coders';

var isInt = myString.search(intRegex);

console.log(isInt);

//output: -1

**10. slice(start, [end])**Returns a substring of the string based on the “start” and “end” index arguments, NOT including the “end” index itself. “End” is optional, and if none is specified, the slice includes all characters from “start” to end of string.

//slice(start, end)

var text="excellent"

text.slice(0,4) //returns "exce"

text.slice(2,4) //returns "ce"

**11. split(delimiter, [limit])**Splits a string into many according to the specified delimiter, and returns an array containing each element. The optional “limit” is an integer that lets you specify the maximum number of elements to return.

//split(delimiter)

var message="Welcome to jQuery4u"

//word[0] contains "We"

//word[1] contains "lcome to jQuery4u"

var word=message.split("l")

**12. substr(start, [length])**Returns the characters in a string beginning at “start” and through the specified number of characters, “length”. “Length” is optional, and if omitted, up to the end of the string is assumed.

//substring(from, to)

var text="excellent"

text.substring(0,4) //returns "exce"

text.substring(2,4) //returns "ce"

**13. substring(from, [to])**Returns the characters in a string between “from” and “to” indexes, NOT including “to” inself. “To” is optional, and if omitted, up to the end of the string is assumed.

//substring(from, [to])

var myString = 'javascript rox';

myString = myString.substring(0,10);

console.log(myString)

//output: javascript

**14. toLowerCase()**Returns the string with all of its characters converted to lowercase.

//toLowerCase()

var myString = 'JAVASCRIPT ROX';

myString = myString.toLowerCase();

console.log(myString)

//output: javascript rox

**15. toUpperCase()**Returns the string with all of its characters converted to uppercase.

//toUpperCase()

var myString = 'javascript rox';

myString = myString.toUpperCase();

console.log(myString)

//output: JAVASCRIPT ROX