String Data Type

- String is a primitive data type in JavaScript.
- A string is textual content. It must be enclosed in single or double quotation marks.
- In simple words, a string is a combination of letters/words enclosed in double or Single quotes.
- A string is immutable in JavaScript.
- String can be concatenated using plus (+) operator in JavaScript.
- A string can be treated as character array.
- We can use escape sequences inside strings.
- Two Strings are compared in Lexicographical order.

Example

```
var str1 = "Hello World";
var str2 = 'Hello World';
console.log(str1,str2);
```

Example

```
var str = 'Hello ' + "World " + 'from ' + 'College ';
console.log(str);
```

Example

```
var a = 'a';
var b = 'b';
if (a < b) {
  console.log(a + ' is less than ' + b);
} else if (a > b) {
  console.log(a + ' is greater than ' + b);
} else {
  console.log(a + ' and ' + b + ' are equal.');
}
```

Example

```
var a = 'H';
var b = 'h';
if (a < b) {
  console.log("a is less than b");
}else {
  console.log('b is less than a');
}
```

Example

```
var str = 'Hello World';
str[0] // H
str[1] // e
console.log(str.length); // 11
console.log(str[1]); // e
```

Example

```
var str1 = new String();
str1 = 'Hello World';
console.log(typeof str1); // String
console.log(str1); // Hello World
```

Example

```
var str2 = new String('Hello World');
console.log(typeof str2);
```

Example

```
var str1 = new String('Hello World');
var str2 = new String('Hello World');
var str3 = 'Hello World';
var str4 = str1;                    // reference type
str1 == str2; // false - because str1 and str2 are two different objects
str1 == str3; // true
str1 === str4; // true
typeof(str1); // object
typeof(str3); //string
```

String Properties

a) Length

- The length property of a String object contains the length of the string, in UTF-16 code units. length is a read-only data property of string instances.
- Spaces and special characters also count as a character in string.

Syntax

```
string.length
```

Example

```
var str = "Hello World";
var result = str.length;
console.log(result);
```

Example

```
var str = "Hello users javascript is awesome";
console.log("The length of the string is : ",str.length)
```

String Methods

1) concat()

- The concat() method concatenates the string arguments to the calling string and returns a new string.
- Return type is string

Syntax

```
string.concat(str1, str2, ..., strN)
```

where,
str1,str2,...,strN........Strings to concatenate.

Example

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

Example

```
var str1 = "Hello ";
var str2 = "world!";
var str3 = " Have a nice day!";
var res = str1.concat(str2, str3);
console.log(res)
```

2) charAt()

- The String object's charAt() method returns a new string consisting of the single UTF-16 code unit located at the specified offset into the string.
- This method returns the character of string present at the given index inside the string.
- Return type is string.

Syntax

```
string.charAt(index)
```

where, index is an integer between 0 and str.length - 1. If the index cannot be converted to the integer or no index is provided, the default is 0, so the first character in of str is returned.

Example

```
var anyString = 'Brave new world';
console.log(anyString.charAt());
console.log(anyString.charAt(0));
console.log(anyString.charAt(1));
console.log(anyString.charAt(2));
console.log(anyString.charAt(3));
console.log(anyString.charAt(4));
console.log(anyString.charAt(999));
```

Example

```
var str = "HELLO WORLD";
var res = str.charAt(str.length-1);
console.log(res);
```

3) charCodeAt()

- The charCodeAt() method returns an integer between 0 and 65535 representing the UTF-16 code unit at the given index.
- Return type is number.

Syntax

```
string.charCodeAt(index)
```

where, index is an integer greater than or equal to 0 and less than the length of the string. If index is not a number, it defaults to 0.

Example

```
var str = "HELLO WORLD";
var n = str.charCodeAt(0);
console.log(n);
```

Example

```
var str = "HELLO WORLD";
var n = str.charCodeAt(str.length-1);
console.log(n)
```

4) endsWith()

- The endsWith() method determines whether a string ends with the characters of a specified string, returning true or false as appropriate.
- Return type is boolean.

Syntax

```
string.endsWith(searchvalue, length)
```

where,

- searchvalue is the characters to be searched for at the end of str.
- Length (if provided), is used as the length of str. defaults to str.length.

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.endsWith("universe.");
console.log(n);
```

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.endsWith("world", 11);
console.log(n);
```

5) fromCharCode()

- The fromCharCode() method converts Unicode values into characters.
- The static String.fromCharCode() method returns a string created from the specified sequence of UTF-16 code units.
- Return type is string.

Syntax

```
String.fromCharCode(n1, n2, ..., nX)
```

where, n1,n2, ..., nX are the sequence of numbers that are UTF-16 code units. The range is between 0 and 65535 (0xFFFF). Numbers greater than 0xFFFF are truncated. No validity checks are performed.

Example

```
var res = String.fromCharCode(72, 69, 76, 76, 79);
console.log(res)
```

Example

```
var res = String.fromCharCode(65);
console.log(res);
```

6) includes()

- The includes() method determines whether a string contains the given characters within it or not.
- The includes() method determines whether one string may be found within another string, returning true or false as appropriate.
- Return type is boolean.

Syntax

```
string.includes(searchvalue, position);
```

where,

- searchvalue is a string to be searched for within str.
- position is the position within the string at which to begin searching for searchString. (Defaults to 0.)

Example

```
var str = "Hello World";
var check = str.includes("World");
console.log(check)
```

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.includes("world", 12);
console.log(n);
```

7) indexOf()

- The indexOf() method returns the position of the first choice of a specified value in a string.
- The indexOf() method returns the index within the calling String object of the first occurrence of the specified value, starting the search at fromIndex.
- Returns -1 if the value is not found.

Syntax

```
string.indexOf(searchvalue, start)
```

where,

searchvalue

- The string value to search for.
- If no string is provided, searchValue will be coerced to "undefined", and this value will be searched as a string.
- So, 'undefined'.indexOf() will return 0, as undefined is found at position 0 in the string undefined.
- 'undefine'.indexOf() however will return -1, as undefined is not found in the string 'undefine'.

Start, is an integer representing the index at which to start the search in the string. Defaults to 0.

Example

```
var str = "JavaScript is awesome";
var n = str.indexOf("awesome");
console.log(n)
```

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("e");
console.log(n);
```

Example

```
var str = "undefined";
var n = str.indexOf();
console.log(n)
```

Example

```
var str = "undefine";
var n = str.indexOf();
console.log(n)
```

8) lastIndexOf()

- The lastIndexOf() method returns the position of the last choice of a specified value in a string.
- The lastIndexOf() method returns the index within the calling String object of the last occurrence of the specified value, searching backwards from fromIndex.
- In this method the search starts from the end of the string and returns the index from the starting of the string.
- Returns -1 if the value is not found.

Syntax

```
string.lastIndexOf(searchvalue, start)
```

where,

- searchValue is a string representing the value to search for. If searchValue is an empty string, then fromIndex is returned.
- Start is the index of the last character in the string to be considered as the beginning of a match. The index of the last occurrence of searchValue; -1 if not found.

Example

```
var str = "JavaScript is Awesome";
var n = str.lastIndexOf("e");
console.log(n) // 20
```

Example

```
var str = "Hello planet earth, you are a great planet.";
var n = str.lastIndexOf("planet", 20);
console.log(n);
```

Example

```
'canal'.lastIndexOf('a');     // returns 3
'canal'.lastIndexOf('a', 2);  // returns 1
'canal'.lastIndexOf('a', 0);  // returns -1
'canal'.lastIndexOf('x');     // returns -1
'canal'.lastIndexOf('c', -5); // returns 0
'canal'.lastIndexOf('c', 0);  // returns 0
'canal'.lastIndexOf('');      // returns 5
'canal'.lastIndexOf('', 2);   // returns 2
```

9) localeCompare()

- The localeCompare() method compares two strings.
- Returns a 'negative number' if 'referenceStr' occurs before 'compareString' & a 'positive number' if the 'referenceStr' occurs after 'compareString'; 0 if they are equivalent.

Syntax

```
referenceStr.localeCompare(compareString)
```

where,

- referenceStr is the string that we wants to compare.
- compareString is the string against which the referenceStr is compared.

Example

```
var str1 = "JAVASCRIPT";
var str2 = "JAVASCRIPT";
var n = str1.localeCompare(str2);
console.log(n);
```

Example

```
var str1 = "cd";
var str2 = "ab";
var n = str1.localeCompare(str2);
console.log(n)
```

Example

```
var str = "a";
var n = str.localeCompare('c');
console.log(n)
```

Example

```
var str = "c";
var n = str.localeCompare('a');
console.log(n)
```

Example

```
var str = "c";
var n = str.localeCompare('c');
console.log(n)
```

10) match()

- The match() method retrieves the result of matching a string against a regular expression.
- Return value is an array.

Syntax

```
string.match(regexp)
```

where, regexp is the regular expression object.

Example

```
var string = "Hello JavaScript Hello Students";
var result = string.match(/llo/g);
console.log(result) //[llo,llo]
```

Example

```
var str = "The rain in SPAIN stays mainly in the plain";
var res = str.match(/ain/gi);
console.log(res);
```

11) repeat()

- Make a new string by copying a string.
- The repeat() method constructs and returns a new string which contains the specified number of copies of the string on which it was called, concatenated together.
- Return value is string.

Syntax

```
string.repeat(count)
```

where, count is an integer between 0 and +Infinity, indicating the number of times to repeat the string.

Example

```
var str = "Hello world! ";
var bac = str.repeat(2);
console.log(bac)
```

Example

```
'abc'.repeat(-1)    // RangeError
'abc'.repeat(0)     // ''
'abc'.repeat(1)     // 'abc'
'abc'.repeat(2)     // 'abcabc'
'abc'.repeat(3.5)   // 'abcabcabc' (count will be converted to integer)
'abc'.repeat(1/0)   // RangeError
```

12) replace()

- Replaces a Matched String.
- The replace() method returns a new string with some or all matches of a pattern replaced by a replacement.
- The method have no effect on original string.

Syntax

```
string.replace(regexp/substr, newSubstr)
```

where,

- regexp is the RegExp object or literal.
- Substr is the string that is to be replaced by newSubstr.
- newSubstr is the String that replaces the substring specified by the specified regexp or substr parameter.

Example

```
var str = "Hello World";
var res = str.replace("World", "Students");
console.log(res)
```

Example

```
var str = "Hello World";
var res = str.replace(/World/g, "Students");
console.log(res);
```

13) search()

- The search() method searches a string for a specified value, and returns the position of the match.
- Return type is number.

Syntax

```
string.search(regexp/substr)
```

where,

- regexp is the regular expression object. If a non-RegExp object regexp is passed, it is implicitly converted to a RegExp with new RegExp(regexp).
- Substr is the String which we want to search.

Example

```
var str = "Hello JavaScript";
var n = str.search("JavaScript");
console.log(n);
```

Example

```
var str = "Hello JavaScript Developers";
var n = str.search(/JavaScript/i);
console.log(n);
```

14) slice()

- Extracts parts of a string.
- Cuts the part of a specified string.
- The slice() method extracts a section of a string and returns it as a new string, without modifying the original string.
- Return type is string.

Syntax

```
string.slice(beginIndex, endIndex)
```

where,

beginIndex

- The zero-based index at which to begin extraction. If negative, it is treated as str.length + beginIndex. (For example, if beginIndex is -3 it is treated as str.length - 3.).
- If beginIndex is greater than or equal to str.length, slice() returns an empty string.

endIndex

- If endIndex is omitted, slice() extracts to the end of the string. If negative, it is treated as str.length + endIndex. (For example, if endIndex is -3 it is treated as str.length - 3.)

Example

```
var str = "Hello world!";
var res = str.slice(0, 5);
console.log(res)
```

Example

```
var str = "Hello world!";
var res = str.slice(-1);
console.log(res);
```

Example

```
var str = "Hello";
var res = str.slice(5);
console.log(res); // empty string
```

15) split()

- The split() method divides a String into an ordered set of substrings, puts these substrings into an array, and returns the array.
- Split a string into an array of substrings.
- Returns an array of string.

Syntax

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

where,

- separator is the separator can be a simple string or it can be a regular expression or can be an empty string.
- Limit is a non-negative integer limiting the number of pieces. If limit is 0, no splitting is performed.

Example

```
var str = "Hello JavaScript Developers";
var res = str.split(" ");
console.log(res)
```

Example

```
var str = "How are you doing today?";
var res = str.split();
console.log(res);
```

Example

```
var str = "How are you doing today?";
var res = str.split("");
console.log(res)
```

Example

```
var str = "How are you doing today?";
var res = str.split(" ", 3);
console.log(res)
```

Example

```
var str = "How are you doing today?";
var res = str.split("o");
console.log(res);
```

16) startsWith()

- This method determines whether a string begins with the characters of a specified string.
- Checks whether the string starts with a specified letter.
- The startsWith() method determines whether a string begins with the characters of a specified string, returning true or false as appropriate.
- Returns a boolean value.

Syntax

```
string.startsWith(searchvalue, position)
```

where,

- searchValue is the characters to be searched for at the start of this string.
- Position is the position in this string at which to begin searching for searchString. Defaults to 0.

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.startsWith("Hello");
console.log(n)
```

Example

```
var str = "Hello world, welcome to the universe.";
var n = str.startsWith("world", 6);
console.log(n);
```

17) substr()

- The substr() method returns a portion of the string, starting at the specified index and extending for a given number of characters afterward.
- This method extracts parts of a string.
- Extracts parts of a string.
- Return a string value.

Syntax

```
string.substr(start, length)
```

where,

start

- The index of the first character to include in the returned substring.
- If start is a positive number, the index starts counting at the start of the string. Its value is capped at str.length.
- If start is a negative number, the index starts counting from the end of the string. Its value is capped at -str.length.

length

- Optional. The number of characters to extract.
- If length is omitted, substr() extracts characters to the end of the string.
- If length is undefined, substr() extracts characters to the end of the string.
- If length is a negative number, it is treated as 0.

Example

```
var str = "Hello world!";
var res = str.substr(2);
console.log(res)
```

Example

```
var str = "Hello world!";
var res = str.substr(0, 1);
console.log(res);
```

Example

```
var aString = 'Mozilla';

console.log(aString.substr(0, 1));   // 'M'
console.log(aString.substr(1, 0));   // ''
console.log(aString.substr(-1, 1));  // 'a'
console.log(aString.substr(1, -1));  // ''
console.log(aString.substr(-3));     // 'lla'
console.log(aString.substr(1));      // 'ozilla'
console.log(aString.substr(-20, 2)); // 'Mo'
console.log(aString.substr(20, 2));  // ''
```

18) substring()

- This method extracts the characters from a string, between two specified indices, and returns the new sub string.
- The substring() method returns the part of the string between the start and end indexes, or to the end of the string.
- Return value is string.

Syntax

```
string.substring(indexStart, indexEnd)
```

where,

indexStart

- The index of the first character to include in the returned substring.
- If indexStart is equal to indexEnd, substring() returns an empty string.
- If indexStart is greater than indexEnd, then the effect of substring() is as if the two arguments were swapped.

indexEnd

- The index of the first character to exclude from the returned substring.
- If indexEnd is omitted, substring() extracts characters to the end of the string.

Example

```
var str = "Hello world!";
var res = str.substring(1, 4);
console.log(res);
```

Example

```
var str = "Hello world!";
var res = str.substring(4, 1);
console.log(res);
```