JavaScript Strings 20th July 2020

A JavaScript string stores a series of characters like "BEST Language". A string can be any text inside double or single quotes. The following are examples of JavaScript Strings:

```
var user = 'john Doe Kenya';

var course = "#100Days Of JavaScript With lux";
```

String indexes are zero-based, the first character is in position 0, the second in 1, and so on.

In JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

Strings Object in JavaScript.

JavaScript String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods. Is data that is not an object and has no methods, examples are string, number,, boolean, undefined, symbol ...

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive. Use the following syntax to create a String object −

```
  var val = new String(string);
```

String object Properties.

a) constructor: Returns the string's constructor function.

b) length: Returns the length of a string.

c) prototype: Allows you to add properties and methods to an object.

JavaScript object Methods

a). concat() Method

In JavaScript concat() method adds two or more strings and returns a new single string.

Its syntax is as follows –

```
    string.concat(string2, string3[, ..., stringN]);
```

- string2...stringN − These are the strings to be concatenated and a single concatenated string.

Example:

```
<html lang =”en>
   <head>
      <title>JavaScript String concat() Method</title>
   </head>
     <body>
      <script>

         let str1 = new String( "#100 Days Of JavaScript" );
         let str2 = new String( "Program for Every Developer" );
         let str3 = str1.concat( str2 );
         document.write("Concatenated String :" + str3);

      </script>
   </body>
</html>
```

Output: `Concatenated String : #100 Days Of JavaScript program for Every Developer`

b). **split**() **Method**.

This method splits a String object into an array of strings by separating the string into substrings.

Its syntax is as follows –

```
string.split([separator][, limit]);
```

separator– Specifies the character to use for separating the string. If separator is omitted, the array returned contains one element consisting of the entire string.

Limit– Integer specifying a limit on the number of splits to be found

The split method returns the new array. Also, when the string is empty, split returns an array containing one empty string, rather than an empty array.

Example:

```
<html lang = “en”>
  <head>
    <title>JavaScript String split() Method</title>
  </head>

  <body>
    <script>
      let email = ’gray@example.com’

      let localPart = email.slice(0,email.indexOf(’@’));

      document.write(localPart);
    </script>
  </body>
</html>
```

Output: gray

c). indexOf() Method.

The indexOf() method returns the index within the calling String object of the first occurrence of the specified value, starting the search at fromIndex or -1 if the value is not found.
Use the following syntax to use the indexOf() method.

```
string.indexOf(searchValue[, fromIndex]);
```

searchValue – A string representing the value to search for.

fromIndex – The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is 0.

```
<html lang =”en>
   <head>
      <title>JavaScript String concat() Method</title>
   </head>
     <body>

     <script>
        let str = 'finding substring in string';

        let index = str.indexOf('str');

        document.write(index); // 11

      </script>

   </body>

</html>
```

```
Output: 11
```

d). lastIndexOf() Method.

This method returns the index within the calling String object of the last occurrence of the specified value, starting the search at fromIndex or −1 if the value is not found.

Use the following syntax to use the lastIndexOf() method.

```
string.lastIndexOf(searchValue[, fromIndex])
```

searchValue – A string representing the value to search for.

FromIndex – The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is 0.

The lastIndexOf() method returns the index of the last found occurrence, otherwise -1 if not found.

Example:

```
<html lang =”en>
<head>
     <title>JavaScript String lastIndexOf() Method</title>
  </head>
  <body>
     <script>
       let str = ’Hello, World!’;

       let substr = ’L’;

       let index = str.lastIndexOf(substr);

       document.write(index);

     </script>
      </body>
```

Output: -1

e). substring() Method.

The substrings() method returns a subset of a String object. Use the following syntax to use the substring() method.

```
string.substring(indexA, [indexB])
```

indexA– An integer between 0 and one less than the length of the string.
IndexB– (optional) An integer between 0 and the length of the string.

The substring method returns the new sub-string based on given parameters.

```
<html lang="en">
<head>
    <title>JavaScript String substring() Method</title>
  </head>

  <body>
    <script>
      let str = 'JavaScript Substring';

      let substring = str.substring(0,10);

       console.log(substring);

    </script>
  </body>
</html>
```

Output: JavaScript

The JavaScript substring() returns the substring from a string between the start and end indexes.

f) slice() Method

The slice() Method extracts a section of a string and returns a new string.

The syntax for slice() method is –

```
string.slice( beginslice [, endSlice] );
```

beginSlice – The zero-based index at which to begin extraction.

endSlice – The zero-based index at which to end extraction. If omitted, slice extracts to the end of the string.

If successful, slice returns the index of the regular expression inside the string. Otherwise, it returns -1.

Example:

```
<head lang=”en”>
<head>
     <title>JavaScript String slice() Method</title>
   </head>

   <body>
     <script>
        var str = "Apples are round, and apples are juicy.";
        var sliced = str.slice(3, -2);
        document.write( sliced );
     </script>
   </body>
</html>
```

OutPut: les are round, and apples are juic

g). trim() method.

The trim()method to remove whitespace characters from both ends of a string and doesn't change the original string.

The syntax for trim() method is –

```
let resultString = str.trim();
```

Example:

```
let str = ' JavaScript Course ';

let result = str.trim();

console.log(result);
```

Output: "JavaScript Course"

More String Methods include:

1). charAT() – Returns the character at the specified index.
2). charCodeAt()– Returns a number indicating the Unicode value of the character at the given index.
3). localeCompare) – Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
4). match() – Used to match a regular expression against a string.
5). toUpperCase – Returns the calling string value converted to uppercase.

6). valueOf() – Returns the primitive value of the specified object.

–toLowerCase() – toString()
–substr() – toLocaleUpperCase()
–toLocaleLowerCase()

The following are methods that return a copy of the string wrapped inside an appropriate HTML tag.

– big: Creates a string to be displayed in a big font as if it were in a <big> tag.

– fontcolor: Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.

– anchor: Creates an HTML anchor that is used as a hypertext target.

– blink: Creates a string to blink as if it were in a <blink> tag.

– bold: Creates a string to be displayed as bold as if it were in a <b> tag.

– small: Causes a string to be displayed in a small font, as if it were in a <small> tag.

– links: Creates an HTML hypertext link that requests another URL.

– italics:Causes a string to be italic, as if it were in an <i> tag.

– sup: causes a string to be displayed as a superscript, as if it were in a <sup> tag

– sub: causes a string to be displayed as a subscript, as if it were in a <sub> tag

–strike: causes a string to be displayed as struck–out text, as if it were in a <strike> tag.

Note that these are not standard methods, and may not work as expected in all browsers.

Implementation of bold:

As we said earlier bold method causes a string to be displayed as bold as if it were in a <b> tag and it's syntax is as follows −

```
string.bold( )bold()
```

Method Returns the string with **<bold>** tag.

Let try understanding this with an example

```
<html lang=”en”>
<head>
    <title>JavaScript String bold() Method</title>
  </head>

  <body>
    <script>
      var str = new String("Hello world");
      alert(str.bold());
    </script>
  </body>
</html>
```

Output: `<b>Hello world</b>`

We will later exhaustively discuss string.

“Make it work, make it right, make it fast.”

Best Wishes.

Lux Tech Academy.