Strings

Overview

The String object is used to represent and manipulate a sequence of characters.
It could be written with both single and double quotes.

```
Example:    let str= “Coding Ninjas” ;
            let str = ‘Coding Ninjas’ ;
```

Length

The length of the string could be found using built-in property.

```
Example:    let text = "Coding Ninjas";
            console.log( text.length) ;    // Output 13
```

Quotes & backslash within Strings

Using double quotes and backslash within a string gives unexpected results.

Double quotes :

```
let str= “My “name” is Joe” ;
console.log( str ) ;
```

- The above line will give an error as an *Unexpected identifier.*

Solution: Use **\” name \”** instead.

Backslash :

```
let str = “My \name\ is Joe” ;
console.log( str ) ;
```

- The above line prints My name is Joe

Solution: Use **\\ name ** instead

Creating Strings

Strings are primitive data types created from literals.

```
let str = "Coding Ninjas" ;
typeOf( str ) ; //String
```

Strings can also be defined as objects with the keyword new as we did in arrays.

```
let str = new String("Coding Ninjas")  ;
typeOf( str ) ; // Object
```

NOTE: Don't create strings as objects. It slows down execution speed.
The new keyword complicates the code. This can produce some unexpected results too

Comparing Strings

Using == operator :

```
let str1 = "Coding Ninjas" ;
let str2 = new String("Coding Ninjas") ;

console.log(str1 == str2) ; //  true as only the values are checked and not the data type
```

Using === operator :

```
let str1 = "Coding Ninjas" ;
let str2 = new String("Coding Ninjas") ;

console.log(str1 ===  str2) ; //  false because of strict equality operator , data types are also checked
```

Objects can not be compared :

```
let str1 = new String("Coding Ninjas") ;
let str2 = new String("Coding Ninjas") ;

console.log(str1 ===  str2) ; //  false
```

String Methods

1. **substring( ) :** It is used to find a contiguous sequence of characters within a string using indxes

```
Example :   let str = “Coding Ninjas” ;
            str.substring( 2 ) ;  // ding Ninjas
            str.substring(3 , 8 ) ; // ing N
```

2. **substr( ) :** It is same as substring method , the difference is that the second parameter specifies the length of the extracted part.

```
Example :   let str = “Coding Ninjas” ;
            str.substr( 2 , 4 ) ;  // ding
```

3. **replace( ) :** It replaces a specified value with another value in a string . But it does not affect the original string rather it makes another string the return the result

```
Example :   let str = “Hello World” ;
            let str2 = str.replace("Hello", "Bye") ;   // Bye world
```

4. **toUpperCase( ) & toLowerCase( ) :** toUpperCase( ) Converts the characters of string to uppercase characters and vice-versa for toLowerCase( )

```
Example :   let str = “Coding Ninjas” ;
            let str2 = str.toUpperCase( ) ;    // CODING NINJAS
            let str3 = str.toLowerCase( ) ;    // coding ninjas
```

5. **trim( ) :** This method removes whitespace from both sides of a string

```
Example :   let str = “    Coding Ninjas     ” ;
            let str2 = str.trim( ) ; // Coding Ninjas
```

Searching Methods in Strings

1. **indexOf( ) :** This method returns the index of (the position of) the ***first*** occurrence of a specified character/text in a string

```
Example :   let str = “Coding Ninjas” ;
            str.indexOf(“Ninjas” ) ; // 7
            str.indexOf(“N” ) ; // 7
            str.indexOf(“n” ) ; // 4
            str.indexOf(“ninjas” ) ; // -1
```

2. **lastIndexOf( ) :** This method returns the index of the last occurrence of a specified character/text in a string

```
Example :   let str = “Coding Ninjas” ;
            str.lastIndexOf(“N” ) ; // 7
            str.lastIndexOf(“n” ) ; // 9
```

3. **includes( ) :** This method returns true if a string contains a specified text

```
Example :   let str = “Coding Ninjas” ;
            str.includes(“Ninjas” ) ; // true
```

NOTE: These methods are case sensitive.