String Object

String object

Introduction to the String object

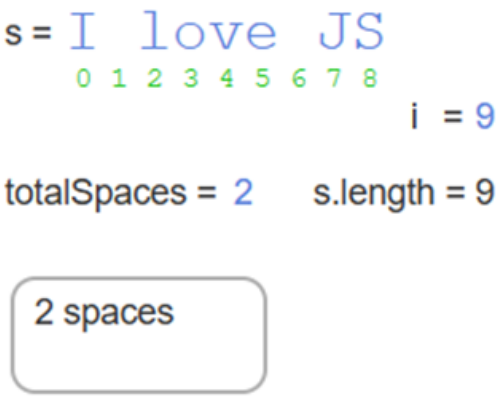
The ***String*** object defines methods to manipulate strings, extract substrings, test for string inclusion, etc. A string literal (a string in "quotes") is automatically converted into a String object when a String method is invoked.

The String method ***charAt()*** returns the character at the specified index as a string.
Ex: "test".charAt(1) returns the character "e" at index 1. The String property ***length*** returns the number of characters in a string. Ex: "test".length returns 4. Calling charAt() with an index ≥ the string's length returns an empty string.

```
let s = "I love JS";
let totalSpaces = 0;

for (let i = 0; i < s.length; i++) {
  if (s.charAt(i) === " ") {
     totalSpaces++;
  }
}

console.log(totalSpaces + " spaces");
```



1. The s variable is initialized to a string literal.
2. The totalSpaces variable is used to count how many spaces are in the string s.
3. Use variable i to iterate through the string s.
4. The loop continues until i is s.length, the number of characters in the string s.
5. charAt(0) is "I", not a space, so totalSpaces is not affected.
6. charAt(1) is a space, so totalSpaces is incremented to 1.
7. The for loop continues to check each character in the string. totalSpaces is 2 when the loop terminates.

Searching and replacing

The String object provides methods to search and replace strings:

- The ***indexOf()*** method returns the index of the search string's first occurrence inside

the String object or -1 if the search string is not found.
- The ***lastIndexOf()*** method returns the index of the search string's last occurrence inside the String object or -1 if the search string is not found.
- The ***replace()*** method replaces one string with another and returns the string with the replacement string inside.

Searching for a string with indexOf() and lastIndexOf().

```
let s = "Seek and you will find.";

s.indexOf("and");    // 5

s.indexOf("e");      // 1 (first occurrence)

s.lastIndexOf("e");  // 2 (last occurrence)

s.indexOf("SEEK");   // -1 (case-sensitive search)
```

Replacing a string with replace().

```
let s = "Seek and you will find.";

s = s.replace("find", "discover");  // "Seek and you will discover"

s = s.replace("Seek", "Search");    // "Search and you will discover"

s = s.replace("SEARCH", "search");  // No change (case-sensitive search)
```

Other String methods

A variety of other String methods exist. Some of the common methods are summarized in the table below.

Common String methods.

Method	Description	Example
substr()	Returns the substring that begins at a given index and has an optional given length.	s = "As you wish."; s.substr(3, 3); *// "you"* s.substr(3); *// "you wish." (length optional)*

substring()	Returns the substring between two indices, not including the second index.	s = "As you wish."; s.substring(3, 6); *// "you"* s.substring(3); *// "you wish." (2nd index optional)*
split()	Returns an array of strings formed by splitting the string into substrings. The given delimiter separates substrings.	s = "As you wish."; s.split(" "); *// ["As", "you", "wish."]*
toLowerCase()	Returns the string converted to lowercase characters.	s = "What?"; s.toLowerCase(); *// "what?"*
toUpperCase()	Returns the string converted to uppercase characters.	s = "What?"; s.toUpperCase(); *// "WHAT?"*
trim()	Returns the string with leading and trailing whitespace removed.	s = " test "; s.trim(); *// "test"*

Template literals

A ***template literal*** is a string literal enclosed by the back-tick (`) that allows embedding expressions with a dollar sign and braces (${expression}). Ex: `test ${1 + 2}` evaluates to "test 3". Template literals replace the need to produce a string with string concatenation.

Template literal simplifies syntax

```
x = 2;
y = 3;
result = x + " * " + y + " = " + (x * y);
console.log(result);
            2     3       6
result = `${x} * ${y} = ${x * y}`;
console.log(result);

console.log(`line 1
line 2`);
```

```
2 * 3 = 6
2 * 3 = 6
line 1
line 2
```

Explanation :

1. String concatenation is required to build a string showing the math equation.
2. A template literal simplifies the syntax to build the same string.
3. Newline characters inserted in a template literal create multi-line strings.