Training day18 Report

(2 July 2024)

JavaScript String Methods

JavaScript strings are the sequence of characters. They are treated as **Primitive data types**. In JavaScript, strings are automatically converted to string objects when using **string methods** on them. This process is called **auto-boxing.**
JavaScript provides a bunch of string methods for representing and manipulating strings. From retrieving specific characters with charAt(), converting strings to uppercase with toUpperCase(), to combining strings with concat(), these methods simplify a wide range of tasks for developers. In this article, we will explore some essential JavaScript string methods that every developer should be familiar with.

Basic JavaScript String Methods

JavaScript provides several **built-in methods** and properties for working with strings. Below is the **list of JavaScript string functions** that you need to know, for mastering JavaScript strings.

JavaScript slice() Method

JavaScript slice method extracts a part of the string based on the given stating-index and ending-index and returns a new string.
Example: This example describes the JavaScript String slice() method.
JavaScript

```
// Define a string variable
let A = 'Geeks for Geeks';

// Use the slice() method to extract a substring
let b = A.slice(0, 5);
let c = A.slice(6, 9);
let d = A.slice(10);

// Output the value of variable
console.log(b);
console.log(c);
console.log(d);
```

Output

```
Geeks
```

```
for

Geeks
```

Explanation:
The code initializes a string A with the value “Geeks for Geeks”. It then uses the slice() method to extract substrings b, c, and d from A based on specified start and end indices and prints them individually.

JavaScript substring() Method

JavaScript substring() method returns the part of the given string from the start index to the end index. Indexing starts from zero (0).

Example: This example shows the implementation of the above-explained approach.

JavaScript

```
// Define a string variable
let str = "Mind, Power, Soul";

// Use the substring() method to extract a substring
let part = str.substring(6, 11);

// Output the value of variable
console.log(part);
```

Output

```
Power
```

Explanation:
The code initializes a string str with the value “Mind, Power, Soul”. It then uses the substring() method to extract a substring starting from index 6 and ending before index 11 (excluding the character at index 11). The extracted substring “Power” is stored in the variable part, and then it is printed to the console.

JavaScript substr() Method

JavaScript substr() method This method returns the specified number of characters from the specified index from the given string. It extracts a part of the original string.

Example: This example shows the implementation of the above-explained approach.

JavaScript

```
// Define a string variable 'str'
let str = "Mind, Power, Soul";

// Use the substr() method to extract a substring f
let part = str.substr(6, 5);

// Output the value of variable
```

```
console.log(part);
```

Output

```
Power
```

Explanation:
The code initializes a string str with the value “Mind, Power, Soul”. It then uses the substr() method to extract a substring starting from index 6 and including the next 5 characters. The extracted substring “Power” is stored in the variable part, and then it is printed to the console.

JavaScript replace()

JavaScript replace() method replaces a part of the given string with another string or a regular expression. The original string will remain unchanged.
Example: This example shows the implementation of the above-explained approach.
JavaScript

```
// Define a string variable 'str'
let str = "Mind, Power, Soul";

// Use the replace() method to replace the substring
let part = str.replace("Power", "Space");

// Output the resulting string after replacement
console.log(part);
```

Output

```
Mind, Space, Soul
```

Explanation:
The code initializes a string str with the value “Mind, Power, Soul”. It uses the replace() method to replace the first occurrence of the substring “Power” with “Space”. The modified string is stored in the variable part, and then it is printed to the console.

JavaScript replaceAll()

JavaScript replaceAll() method returns a new string after replacing all the matches of a string with a specified string or a regular expression. The original string is left unchanged after this operation.
Example: This example shows the implementation of the above-explained approach.
JavaScript

```
// Define a string variable 'str'
let str = "Mind, Power, Power, Soul";
```

```
// Use the replaceAll() method to replace all occurrences
//of "Power" with "Space" in the string 'str'
let part = str.replaceAll("Power", "Space");

// Output the resulting string after replacement
console.log(part);
```

Output

```
Mind, Space, Space, Soul
```

Explanation:
The code initializes a string str with the value “Mind, Power, Power, Soul”. It then uses the replaceAll() method to replace all occurrences of the substring “Power” with “Space”. The modified string is stored in the variable part, and then it is printed to the console.

JavaScript toUpperCase()

JavaScript toUpperCase() method converts all the characters present in the String to upper case and returns a new String with all characters in upper case. This method accepts single parameter **stringVariable** string that you want to convert in upper case.
Example: This example describes the JavaScript String toUpperCase() method.
JavaScript

```
// Define a string variable
let gfg = 'GFG ';

// Define another string variable
let geeks = 'stands-for-GeeksforGeeks';

// Convert the string 'geeks' to uppercase using the toUpperCase() method
console.log(geeks.toUpperCase());
```

Output

```
STANDS-FOR-GEEKSFORGEEKS
```

Explanation:
The code initializes two strings, gfg and geeks. It then uses the toUpperCase() method on the string object geeks. This method converts all characters in the string to uppercase and returns the modified string. Finally, it prints the uppercase version of the geeks string to the console.

JavaScript toLowerCase()

JavaScript toLowerCase() method converts all the characters present in the so lowercase and returns a new string with all the characters in lowercase.
Example: This example describes the JavaScript String toLowerCase() method.
JavaScript

```
// Define a string variable
let gfg = 'GFG ';

// Define a string variable
let geeks = 'stands-for-GeeksforGeeks';

// Convert the string 'geeks' to lowercase using the toLowerCase() method
console.log(geeks.toLowerCase());
```

Output

```
stands-for-geeksforgeeks
```

Explanation:
The code initializes two strings, gfg and geeks. It then uses the toLowerCase() method on the string object geeks.

JavaScript concat() Method

JavaScript concat() method method combines the text of two strings and returns a new combined or joined string. To concatenate two strings, we use the **concat()** method on one object of string and send another object of string as a parameter. This method accepts one argument. The variable contains text in double quotes or single quotes.
Example: This example describes the JavaScript String concat() method.
JavaScript

```
let gfg = 'GFG ';
let geeks = 'stands for GeeksforGeeks';

// Accessing concat method on an object
// of String passing another object
// as a parameter
console.log(gfg.concat(geeks));
```

Output

```
GFG stands for GeeksforGeeks
```

Explanation:
The code initializes two strings, gfg and geeks. It then uses the concat() method on the string object gfg, passing geeks as a parameter. The concat() method concatenates the string geeks to the end of gfg, and the resulting string is printed to the console.

JavaScript trim() Method

JavaScript trim() method is used to remove either white spaces from the given string. This method returns a new string with removed white spaces. This method is called on a String object. This method doesn't accept any parameter.
Example: This example describes the JavaScript String trim() method.

JavaScript

```
let gfg = 'GFG    ';
let geeks = 'stands-for-GeeksforGeeks';

// Storing new object of string
// with removed white spaces
let newGfg = gfg.trim();

// Old length
console.log(gfg.length);

// New length
console.log(newGfg.length)
```

Output

```
7

3
```

Explanation:
The code initializes two strings, gfg and geeks. It then uses the trim() method to remove leading and trailing whitespace from the string gfg, storing the result in newGfg. The code prints the lengths of the original string gfg (including whitespace) and the trimmed string newGfg (excluding whitespace) to the console.

JavaScript trimStart() Method

JavaScript trimStart() method removes whitespace from the beginning of a string. The value of the string is not modified in any manner, including any whitespace present after the string.

Example: This example shows the implementation of the above-explained approach.

JavaScript

```
// Define a string variable
let str = "  Soul";

// Output the original value of the string
console.log(str);

// Use the trimStart() method to remove leading whitespace from the string
'str'
let part = str.trimStart();

// Output the resulting string after removing leading whitespace
console.log(part);
```

Output

```
  Soul
Soul
```

Explanation:
The code initializes a string variable str with the value ” Soul”. It prints the string to the console. Then, it uses the trimStart() method to remove leading whitespace from the string. Finally, it prints the trimmed string part to the console. The first console.log() prints ” Soul”, and the second prints “Soul”.

JavaScript trimEnd() Method

JavaScript trimEnd() method removes white space from the end of a string. The value of the string is not modified in any manner, including any white-space present before the string.

Example: This example shows the implementation of the above-explained approach.

JavaScript

```
// Define a string variable
let str = "Soul  ";

// Output the original value of the string 'str'
console.log(str);

// Use the trimEnd() method to remove trailing whitespace from the string
'str'
let part = str.trimEnd();

// Output the resulting string after removing trailing whitespace
console.log(part);
```

Output

```
Soul
Soul
```

Explanation:
The code initializes a string variable str with the value “Soul “. It prints the string to the console. Then, it uses the trimEnd() method to remove trailing whitespace from the string. Finally, it prints the trimmed string part to the console. The first console.log() prints “Soul “, and the second prints “Soul”.

JavaScript padStart() Method

JavaScript padStart() method pad a string with another string until it reaches the given length. The padding is applied from the left end of the string.

Example: This example shows the implementation of the above-explained approach.

JavaScript

```
// Define a string variable
let stone = "Soul";

// Use the padStart() method to add padding characters "Mind "
//to the beginning of the string 'stone'
stone = stone.padStart(9, "Mind ");

// Output the resulting string after padding
console.log(stone);
```

Output

```
Mind Soul
```

Explanation:
The code initializes a string variable stone with the value “Soul”. It then uses the padStart() method to pad the string with spaces or a specified string (“Mind “) until the resulting string reaches a length of 9 characters. Finally, it prints the modified string to the console, resulting in “Mind Soul”.

JavaScript padEnd() Method

JavaScript padEnd() method pad a string with another string until it reaches the given length. The padding is applied from the right end of the string.
Example: This example shows the implementation of the above-explained approach.
JavaScript

```
// Define a string variable
let stone = "Soul";

// Use the padEnd() method to add padding characters
//" Power" to the end of the string 'stone'
stone = stone.padEnd(10, " Power");

// Output the resulting string after padding
console.log(stone);
```

Output

```
Soul Power
```

Explanation:
The code initializes a string variable stone with the value “Soul”. It then uses the padEnd() method to pad the string with spaces or a specified string (” Power”) until the resulting string reaches a length of 10 characters. Finally, it prints the modified string to the console, resulting in “Soul Power”.

JavaScript charAt() Method

JavaScript charAt() method returns the character at the specified index. String in JavaScript has zero-based indexing.
Example:This example describes the JavaScript string charAt() method.
JavaScript

```
let gfg = 'GeeksforGeeks';
let geeks = 'GfG is the best platform to learn and\n'+
'experience Computer Science.';

// Print the string as it is
console.log(gfg);

console.log(geeks);

// As string index starts from zero
// It will return first character of string
console.log(gfg.charAt(0));

console.log(geeks.charAt(5));
```

Output

```
GeeksforGeeks

GfG is the best platform to learn and

experience Computer Science.

G

s
```

Explanation:
The code initializes two strings gfg and geeks. It prints both strings to the console. Then, it uses the charAt() method to return the character at the specified index within each string. The first console.log() prints the first character of the string gfg (‘G’). The second console.log() prints the character at index 5 in the string geeks (‘i’).

JavaScript charCodeAt() Method

JavaScript charCodeAt() method returns a number that represents the **Unicode** value of the character at the *specified index*. This method accepts one argument.
Example: This example describes the JavaScript String charCodeAt() Method.
JavaScript

```
let gfg = 'GeeksforGeeks';
let geeks = 'GfG is the best platform\n\
to learn and experience\n\
Computer Science.';
```

```
// Return a number indicating Unicode
// value of character at index 0 ('G')
console.log(gfg.charCodeAt(0));
console.log(geeks.charCodeAt(5));
```

Output

```
71

115
```

Explanation:
The code initializes two strings gfg and geeks. It then uses the charCodeAt() method to return the Unicode value of the character at the specified index within each string. The first console.log() prints the Unicode value of the character at index 0 in the string gfg (‘G’). The second console.log() prints the Unicode value of the character at index 5 in the string geeks (‘o’).

JavaScript split() Method

JavaScript split() method splits the string into an array of sub-strings. This method returns an array. This method accepts a single parameter **character** on which you want to split the string.
Example: This example describes the JavaScript String split() method.
JavaScript

```
let gfg = 'GFG '
let geeks = 'stands-for-GeeksforGeeks'

// Split string on '-'.
console.log(geeks.split('-'))
```

Output

```
[ 'stands', 'for', 'GeeksforGeeks' ]
```

Explanation:
The code initializes two strings gfg and geeks. It then uses the split() method to split the string geeks at each occurrence of the ‘-‘ character and returns an array containing the resulting substrings. Finally, it prints the array to the console.