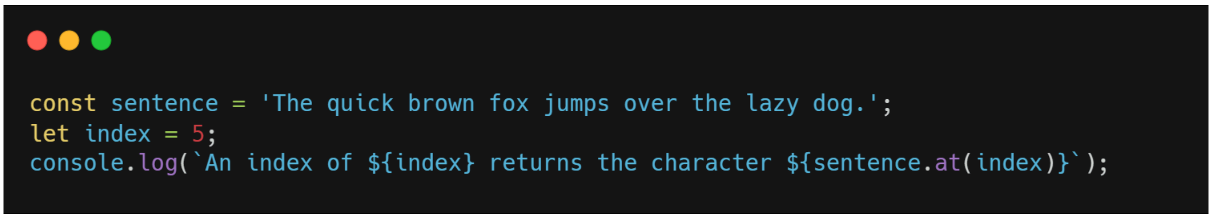
String Methods

Devi kumavath

String methods are built-in functions in JavaScript that operate on strings. These methods provide a wide range of functionalities, allowing you to manipulate, search, and transform strings. Using string methods can improve code readability and efficiency. Here's an overview of some commonly used string methods

At()

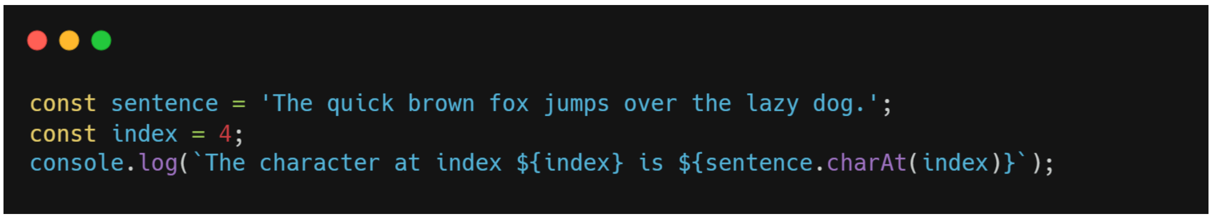
Syntax
at(index)
Description
The at() method of String values takes an integer value and returns a new String consisting of the single UTF-16 code unit located at the specified offset. This method allows for positive and negative integers. Negative integers count back from the last string character.
Example



```
const sentence = 'The quick brown fox jumps over the lazy dog.';
let index = 5;
console.log(`An index of ${index} returns the character ${sentence.at(index)}`);
```

charAt()

Syntax
charAt(index)
Description
The charAt() method of String values returns a new string consisting of the single UTF-16 code unit at the given index.
Example

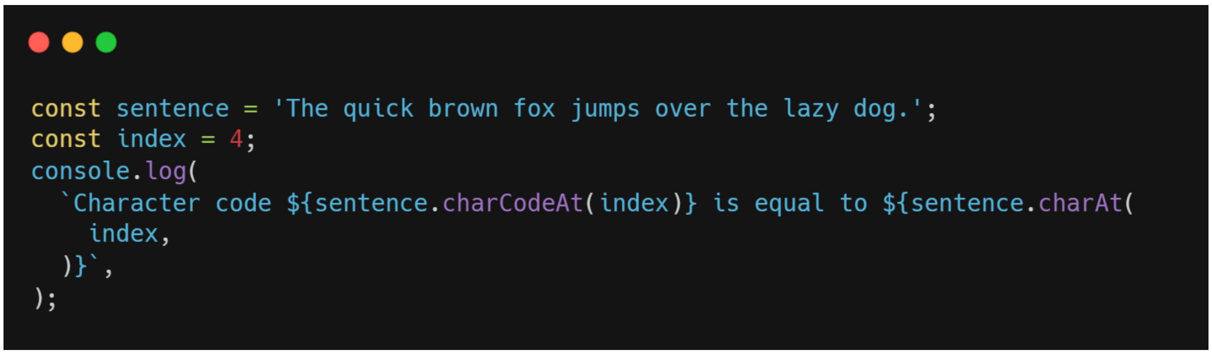


```
const sentence = 'The quick brown fox jumps over the lazy dog.';
const index = 4;
console.log(`The character at index ${index} is ${sentence.charAt(index)}`);
```

charCodeAt()

Syntax
charCodeAt(index)
Description

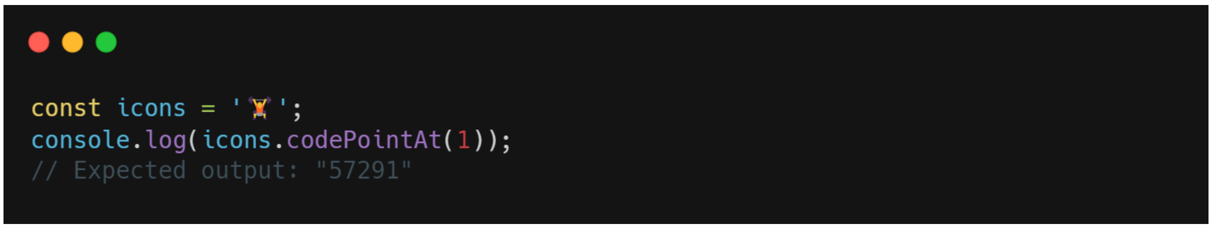
The charCodeAt() method of String values returns an integer between 0 and 65535 representing the UTF-16 code unit at the given index.
Example



```
const sentence = 'The quick brown fox jumps over the lazy dog.';
const index = 4;
console.log(
  `Character code ${sentence.charCodeAt(index)} is equal to ${sentence.charAt(
    index,
  )}`,
);
```

codePointAt()

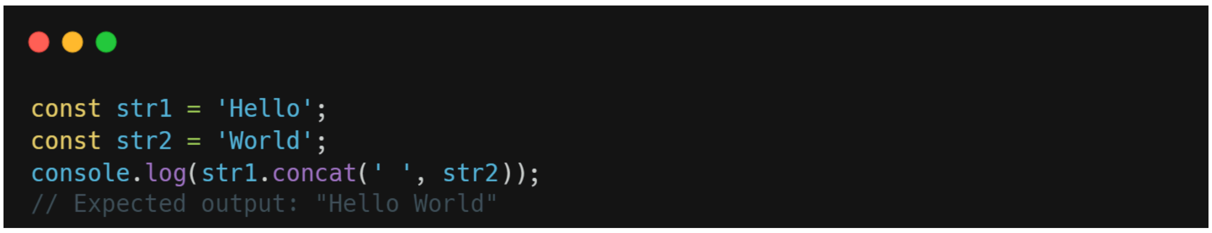
Syntax
codePointAt(index)
Description
The codePointAt() method of String values returns a non-negative integer that is the Unicode code point value of the character starting at the given index. Note that the index is still based on UTF-16 code units, not Unicode code points.
Example



```
const icons = '🏋';
console.log(icons.codePointAt(1));
// Expected output: "57291"
```

concat()

Syntax
concat()
concat(str1)
concat(str1, str2)
concat(str1, str2, /* …, */ strN)
Description
The concat() method of String values concatenates the string arguments to this string and returns a new string.
Example



```
const str1 = 'Hello';
const str2 = 'World';
console.log(str1.concat(' ', str2));
// Expected output: "Hello World"
```

`endsWith()`

Syntax

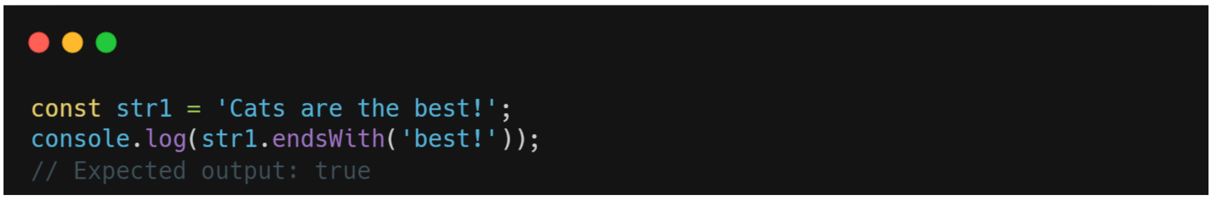
endsWith(searchString)

endsWith(searchString, endPosition)

Description

The endsWith() method of String values determines whether a string ends with the characters of this string, returning true or false as appropriate.

Example



```
const str1 = 'Cats are the best!';
console.log(str1.endsWith('best!'));
// Expected output: true
```

`includes()`

Syntax

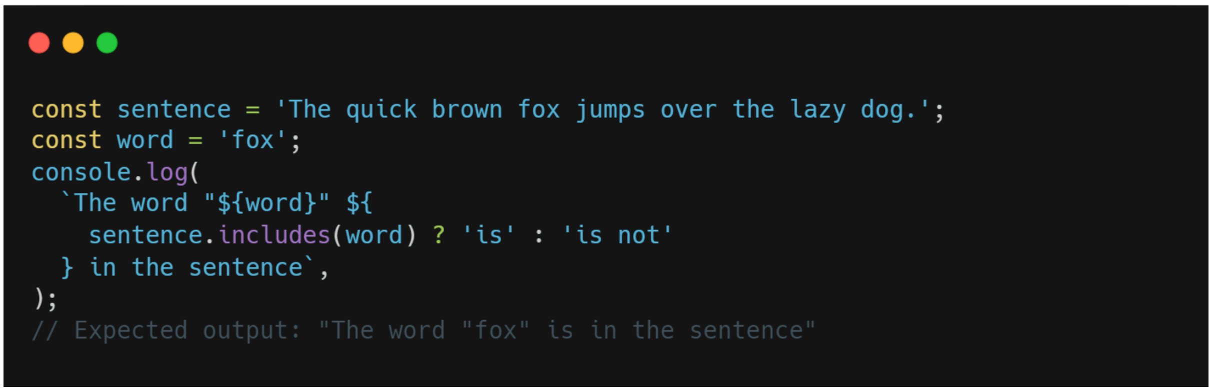
includes(searchString)

includes(searchString, position)

Description

The includes() method of String values performs a case-sensitive search to determine whether a given string may be found within this string, returning true or false as appropriate.

Example



```
const sentence = 'The quick brown fox jumps over the lazy dog.';
const word = 'fox';
console.log(
  `The word "${word}" ${
    sentence.includes(word) ? 'is' : 'is not'
  } in the sentence`,
);
// Expected output: "The word "fox" is in the sentence"
```

`indexOf()`

Syntax

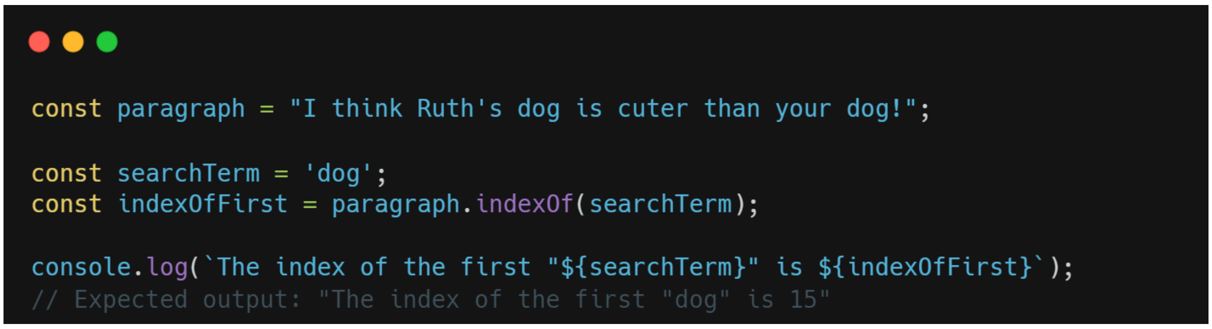
indexOf(searchString)

indexOf(searchString, position)

Description

The indexOf() method of String values searches this string and returns the index of the first occurrence of the specified substring. It takes an optional starting position and returns the first occurrence of the specified substring at an index greater than or equal to the specified number.

Example



```
const paragraph = "I think Ruth's dog is cuter than your dog!";

const searchTerm = 'dog';
const indexOfFirst = paragraph.indexOf(searchTerm);

console.log(`The index of the first "${searchTerm}" is ${indexOfFirst}`);
// Expected output: "The index of the first "dog" is 15"
```

`lastIndexOf()`

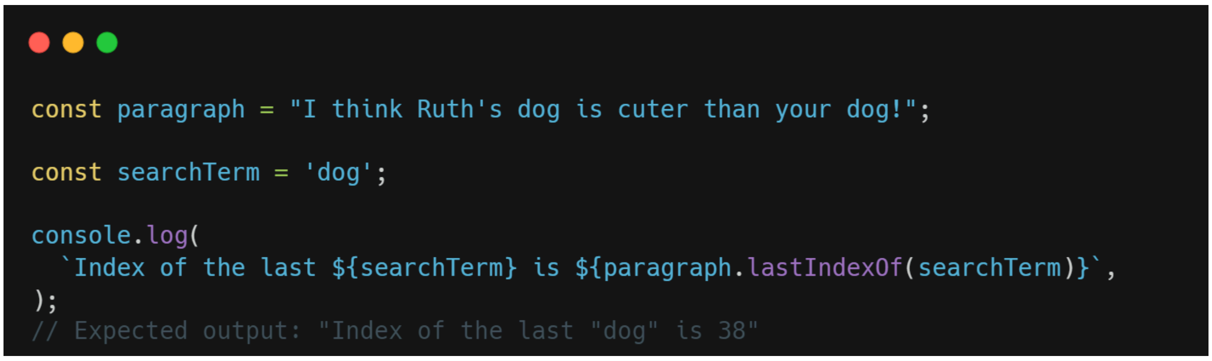
Syntax

lastIndexOf(searchString)
lastIndexOf(searchString, position)

Description

The lastIndexOf() method of String values searches this string and returns the index of the last occurrence of the specified substring. It takes an optional starting position and returns the last occurrence of the specified substring at an index less than or equal to the specified number.

Example



```
const paragraph = "I think Ruth's dog is cuter than your dog!";

const searchTerm = 'dog';

console.log(
  `Index of the last ${searchTerm} is ${paragraph.lastIndexOf(searchTerm)}`,
);
// Expected output: "Index of the last "dog" is 38"
```

`localeCompare()`

Syntax

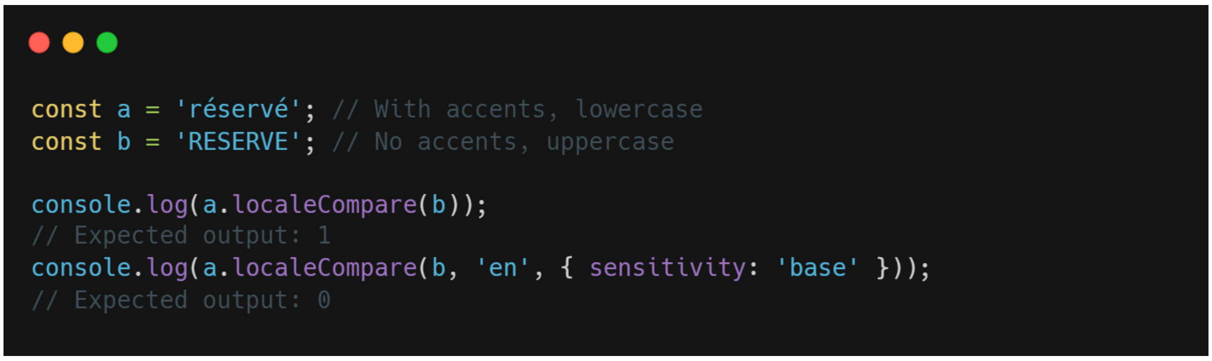
localeCompare(compareString)
localeCompare(compareString, locales)
localeCompare(compareString, locales, options)

Description

The localeCompare() method of String values returns a number indicating whether this string comes before, or after, or is the same as the given string in sort order. In implementations with Intl.Collator API support, this method simply calls Intl.Collator.

When comparing large numbers of strings, such as in sorting large arrays, it is better to create an Intl.Collator object and use the function provided by its compare() method.

Example



```javascript
const a = 'réservé'; // With accents, lowercase
const b = 'RESERVE'; // No accents, uppercase

console.log(a.localeCompare(b));
// Expected output: 1
console.log(a.localeCompare(b, 'en', { sensitivity: 'base' }));
// Expected output: 0
```

match()

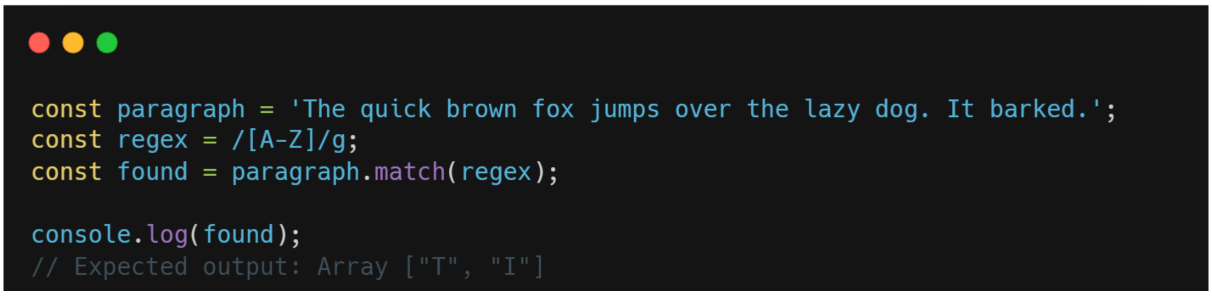
Syntax

match(regexp)

Description

The match() method of String values retrieves the result of matching this string against a regular expression.

Example



```javascript
const paragraph = 'The quick brown fox jumps over the lazy dog. It barked.';
const regex = /[A-Z]/g;
const found = paragraph.match(regex);

console.log(found);
// Expected output: Array ["T", "I"]
```

matchAll()

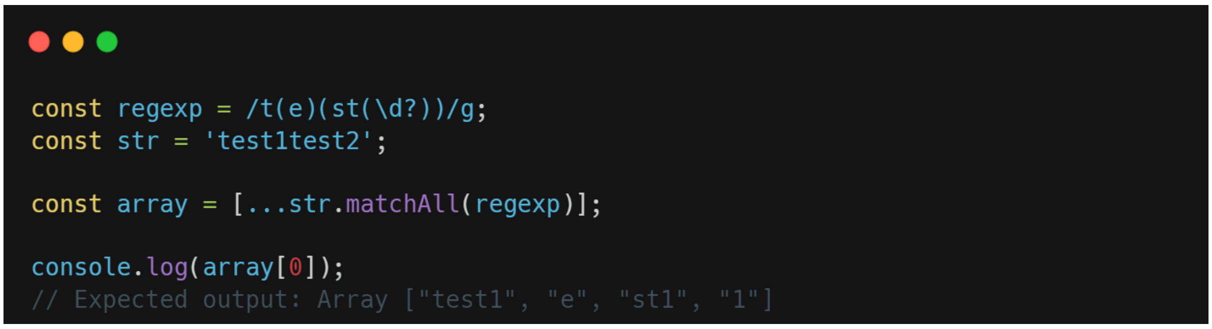
Syntax

matchAll(regexp)

Description

The matchAll() method of String values returns an iterator of all results matching this string against a regular expression, including capturing groups.

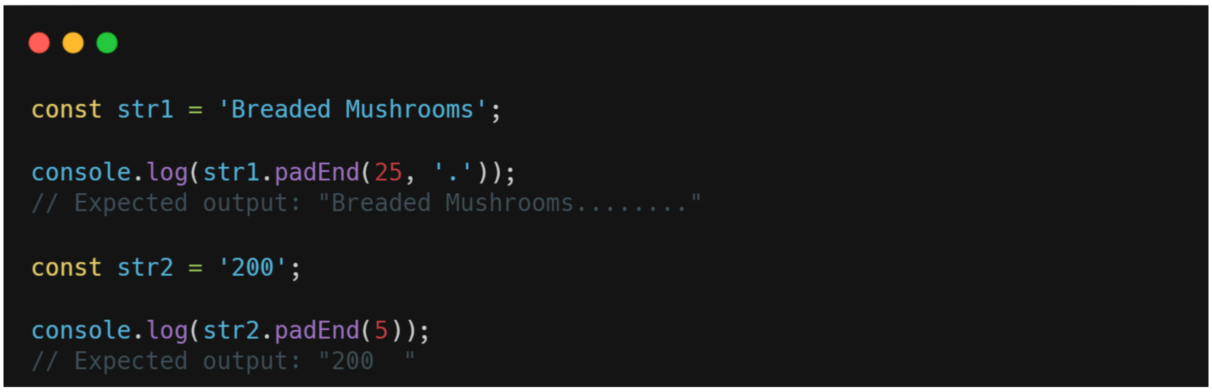
Example



```javascript
const regexp = /t(e)(st(\d?))/g;
const str = 'test1test2';

const array = [...str.matchAll(regexp)];

console.log(array[0]);
// Expected output: Array ["test1", "e", "st1", "1"]
```

`padEnd()`

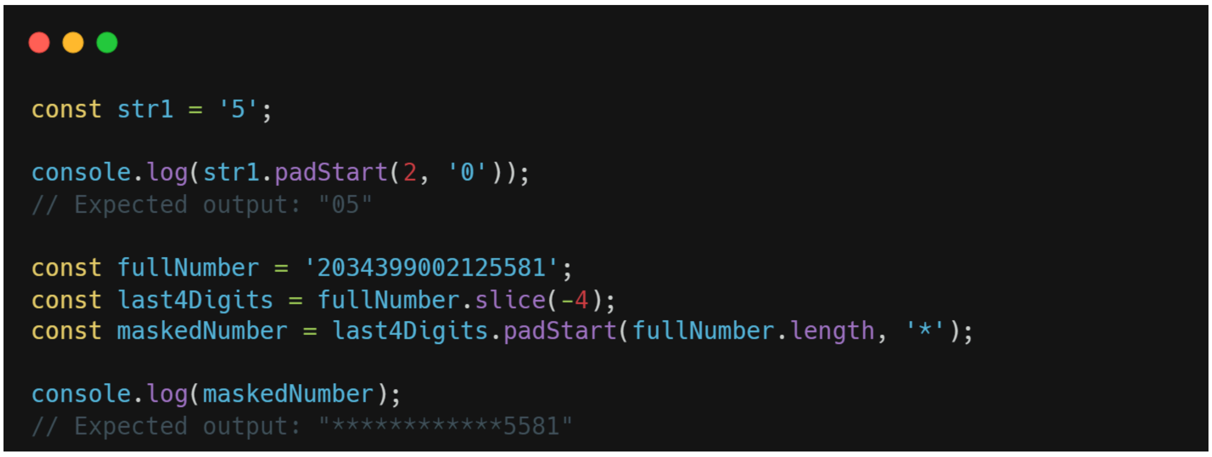
Syntax
padEnd(targetLength)
padEnd(targetLength, padString)
Description
The padEnd() method of String values pads this string with a given string (repeated, if needed) so that the resulting string reaches a given length. The padding is applied from the end of this string.
Example



```javascript
const str1 = 'Breaded Mushrooms';

console.log(str1.padEnd(25, '.'));
// Expected output: "Breaded Mushrooms........"

const str2 = '200';

console.log(str2.padEnd(5));
// Expected output: "200  "
```

`padStart()`

Syntax
padStart(targetLength)
padStart(targetLength, padString)
Description
The padStart() method of String values pads this string with another string (multiple times, if needed) until the resulting string reaches the given length. The padding is applied from the start of this string.
Example



```javascript
const str1 = '5';

console.log(str1.padStart(2, '0'));
// Expected output: "05"

const fullNumber = '2034399002125581';
const last4Digits = fullNumber.slice(-4);
const maskedNumber = last4Digits.padStart(fullNumber.length, '*');

console.log(maskedNumber);
// Expected output: "************5581"
```

`repeat()`

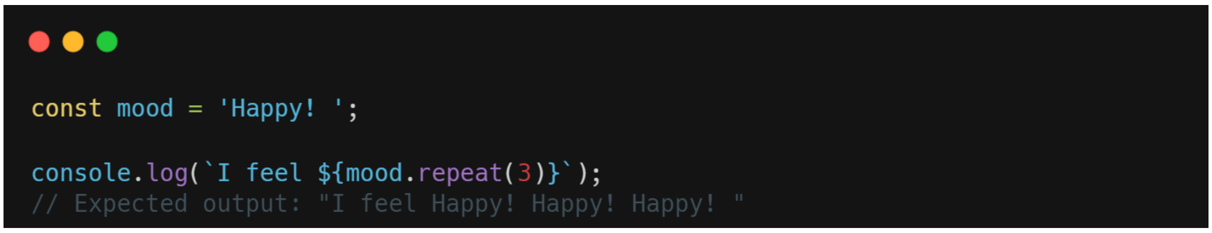
Syntax

repeat(count)

Description

The repeat() method of String values constructs and returns a new string which contains the specified number of copies of this string, concatenated together.

Example



```
const mood = 'Happy! ';

console.log(`I feel ${mood.repeat(3)}`);
// Expected output: "I feel Happy! Happy! Happy! "
```

`replace()`

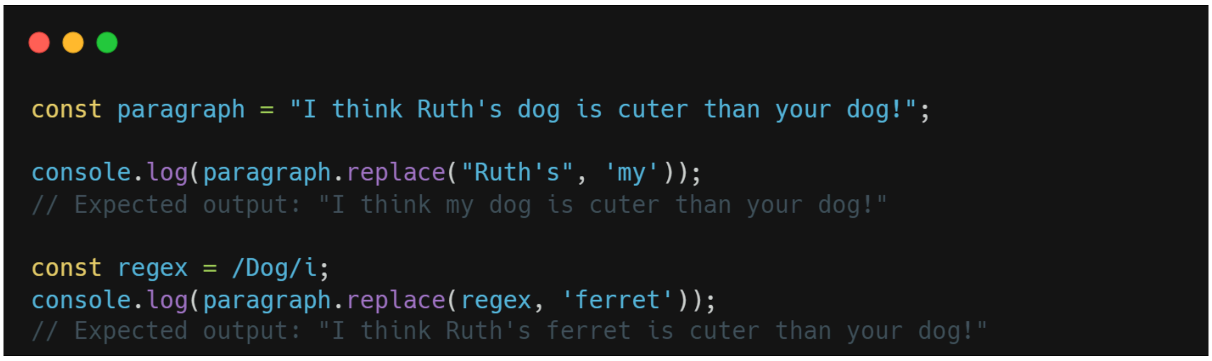
Syntax

replace(pattern, replacement)

Description

The replace() method of String values returns a new string with one, some, or all matches of a pattern replaced by a replacement. The pattern can be a string or a RegExp, and the replacement can be a string or a function called for each match. If pattern is a string, only the first occurrence will be replaced. The original string is left unchanged.

Example



```
const paragraph = "I think Ruth's dog is cuter than your dog!";

console.log(paragraph.replace("Ruth's", 'my'));
// Expected output: "I think my dog is cuter than your dog!"

const regex = /Dog/i;
console.log(paragraph.replace(regex, 'ferret'));
// Expected output: "I think Ruth's ferret is cuter than your dog!"
```

`replaceAll()`

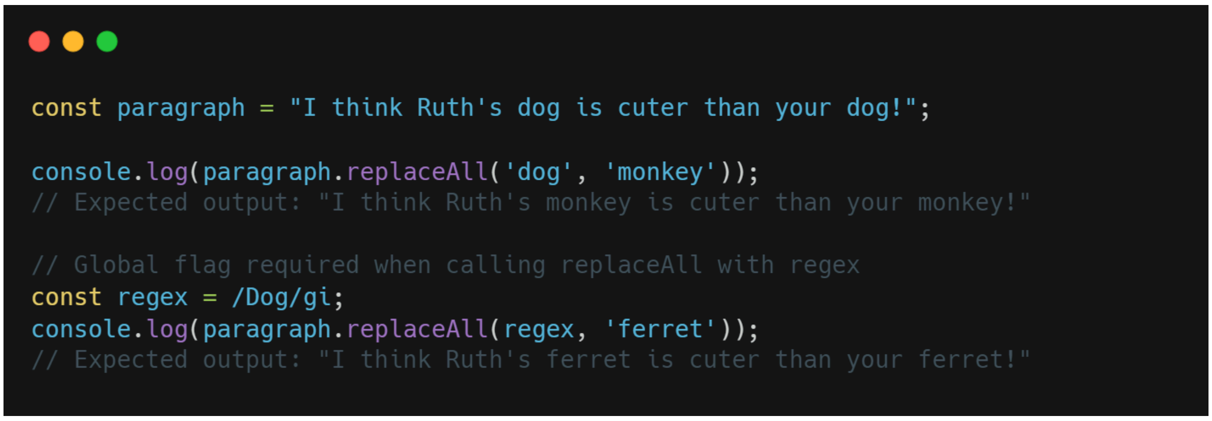
Syntax

replaceAll(pattern, replacement)

Description

The replaceAll() method of String values returns a new string with all matches of a pattern replaced by a replacement. The pattern can be a string or a RegExp, and the replacement can be a string or a function to be called for each match. The original string is left unchanged.

Example



```javascript
const paragraph = "I think Ruth's dog is cuter than your dog!";

console.log(paragraph.replaceAll('dog', 'monkey'));
// Expected output: "I think Ruth's monkey is cuter than your monkey!"

// Global flag required when calling replaceAll with regex
const regex = /Dog/gi;
console.log(paragraph.replaceAll(regex, 'ferret'));
// Expected output: "I think Ruth's ferret is cuter than your ferret!"
```

search()

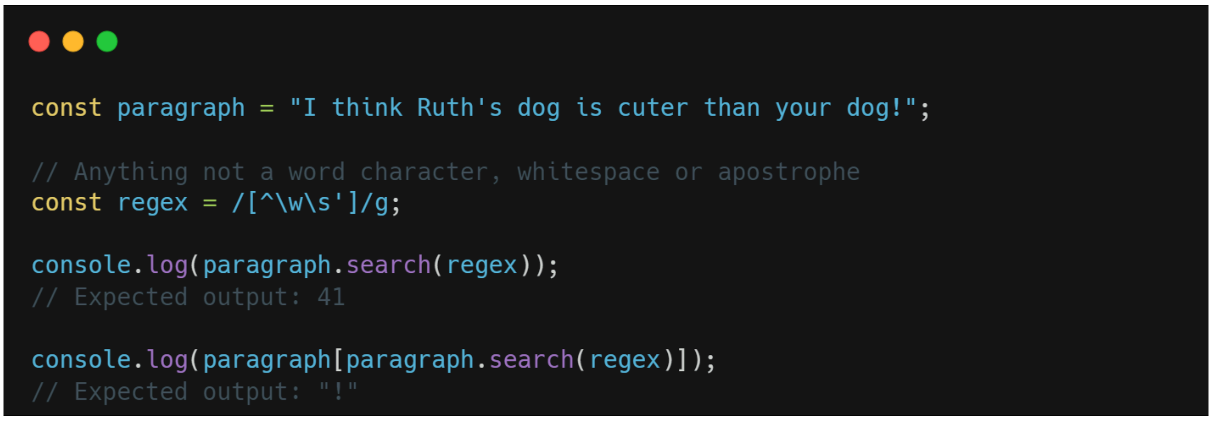
Syntax

search(regexp)

Description

The search() method of String values executes a search for a match between a regular expression and this string, returning the index of the first match in the string.

Example



```javascript
const paragraph = "I think Ruth's dog is cuter than your dog!";

// Anything not a word character, whitespace or apostrophe
const regex = /[^\w\s']/g;

console.log(paragraph.search(regex));
// Expected output: 41

console.log(paragraph[paragraph.search(regex)]);
// Expected output: "!"
```

slice()

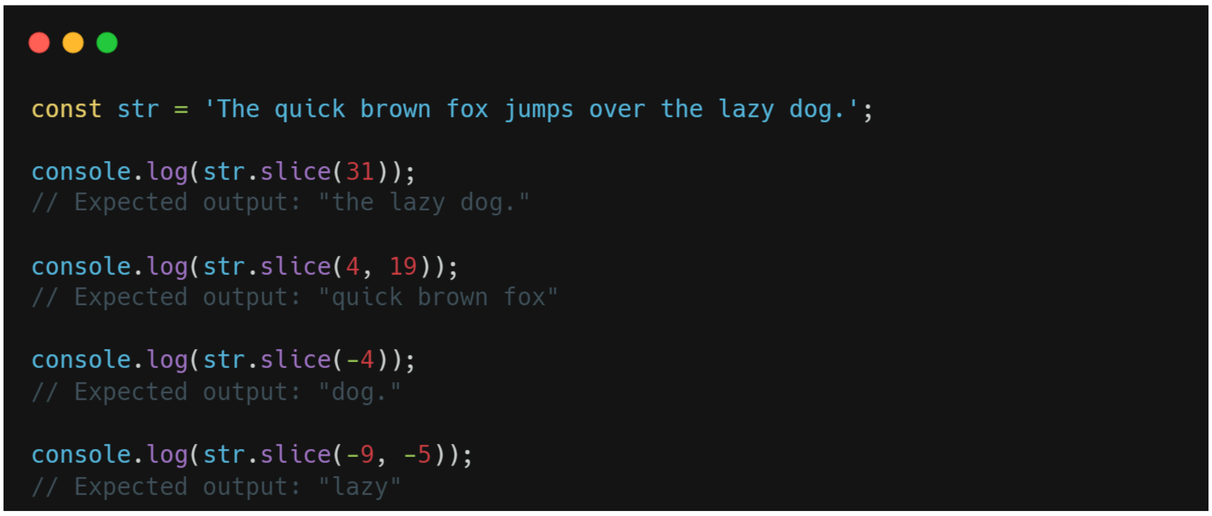
Syntax

slice(indexStart)
slice(indexStart, indexEnd)

Description

The slice() method of String values extracts a section of this string and returns it as a new string, without modifying the original string.

Example



```
const str = 'The quick brown fox jumps over the lazy dog.';

console.log(str.slice(31));
// Expected output: "the lazy dog."

console.log(str.slice(4, 19));
// Expected output: "quick brown fox"

console.log(str.slice(-4));
// Expected output: "dog."

console.log(str.slice(-9, -5));
// Expected output: "lazy"
```

split()

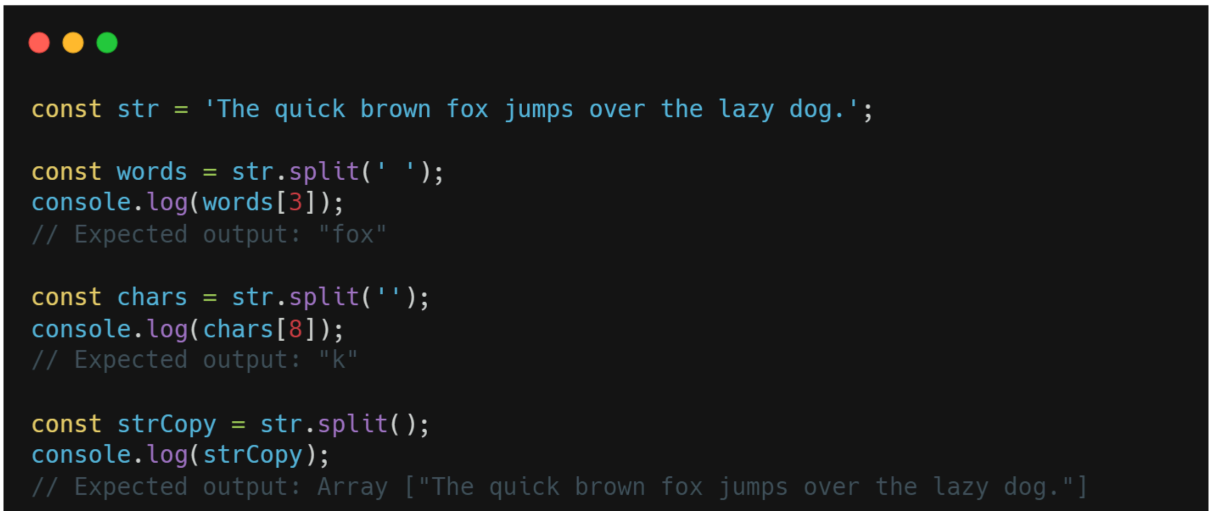
Syntax

split(separator)
split(separator, limit)

Description

The split() method of String values takes a pattern and divides this string into an ordered list of substrings by searching for the pattern, puts these substrings into an array, and returns the array.

Example



```
const str = 'The quick brown fox jumps over the lazy dog.';

const words = str.split(' ');
console.log(words[3]);
// Expected output: "fox"

const chars = str.split('');
console.log(chars[8]);
// Expected output: "k"

const strCopy = str.split();
console.log(strCopy);
// Expected output: Array ["The quick brown fox jumps over the lazy dog."]
```

startsWith()

Syntax

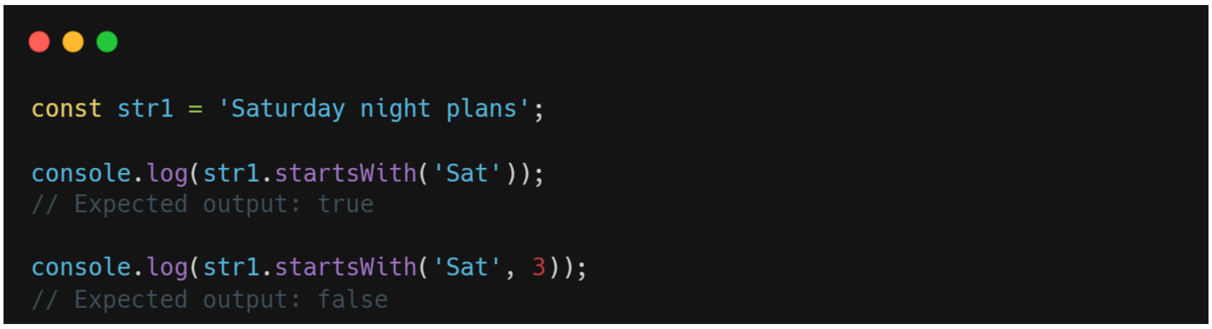
startsWith(searchString)

startsWith(searchString, position)

Description

The startsWith() method of String values determines whether this string begins with the characters of a specified string, returning true or false as appropriate.

Example



```
const str1 = 'Saturday night plans';

console.log(str1.startsWith('Sat'));
// Expected output: true

console.log(str1.startsWith('Sat', 3));
// Expected output: false
```

toLowerCase()

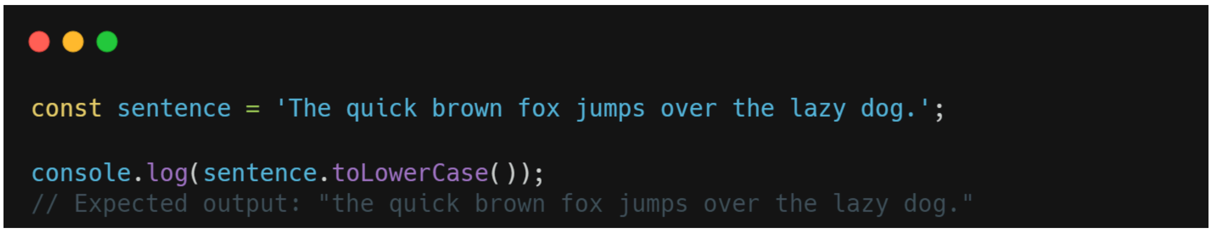
Syntax

toLowerCase()

Description

The toLowerCase() method of String values returns this string converted to lower case.

Example



```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toLowerCase());
// Expected output: "the quick brown fox jumps over the lazy dog."
```

toUpperCase()

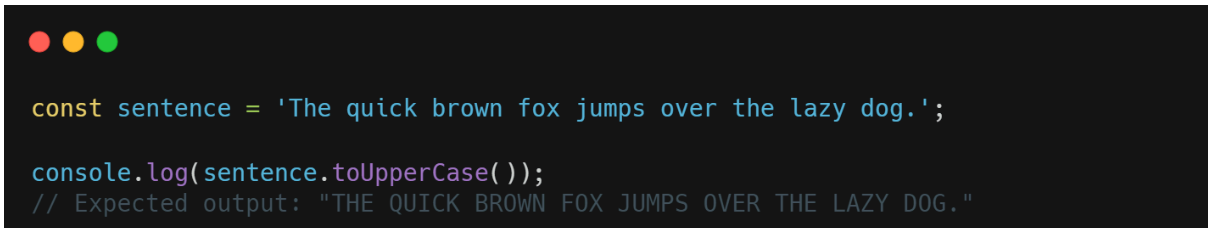
Syntax

toUpperCase()

Description

The toUpperCase() method of String values returns this string converted to uppercase.

Example



```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toUpperCase());
// Expected output: "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."
```

trim()

Syntax

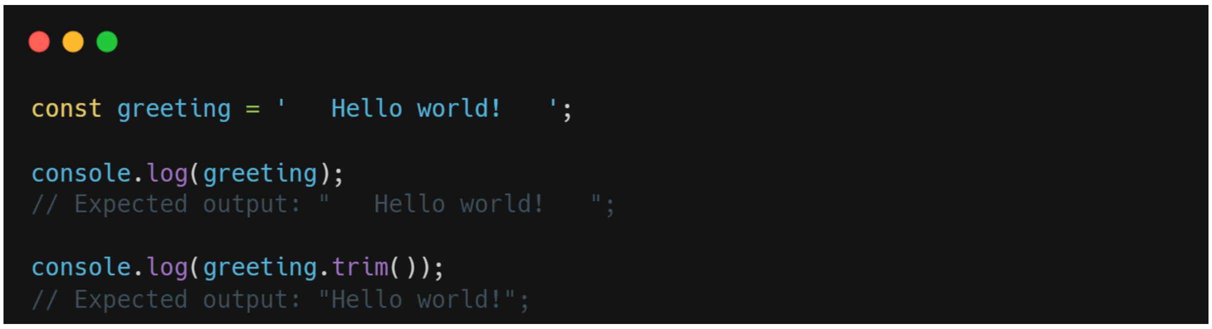
trim()

Description

The trim() method of String values removes whitespace from both ends of this string and returns a new string, without modifying the original string.

To return a new string with whitespace trimmed from just one end, use trimStart() or trimEnd().

Example



```
const greeting = '   Hello world!   ';

console.log(greeting);
// Expected output: "   Hello world!   ";

console.log(greeting.trim());
// Expected output: "Hello world!";
```

trimEnd()

Syntax

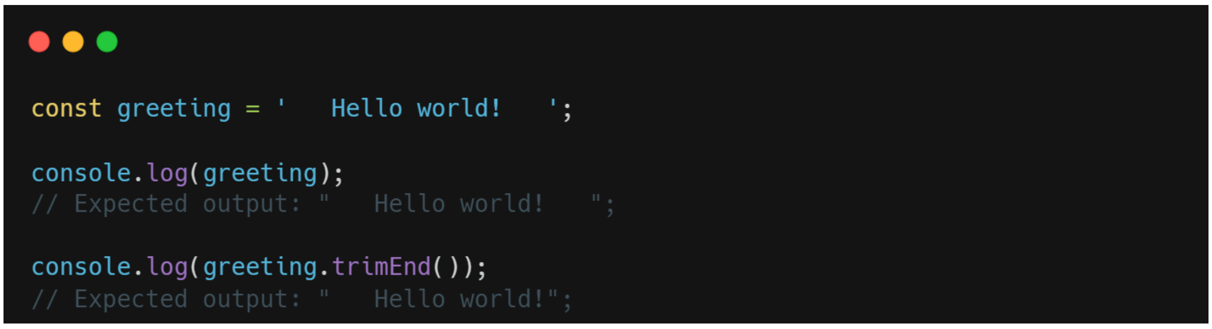
trimEnd()

trimRight()

Description

The trimEnd() method of String values removes whitespace from the end of this string and returns a new string, without modifying the original string. trimRight() is an alias of this method.

Example



```
const greeting = '   Hello world!   ';

console.log(greeting);
// Expected output: "   Hello world!   ";

console.log(greeting.trimEnd());
// Expected output: "   Hello world!";
```

trimStart()

Syntax

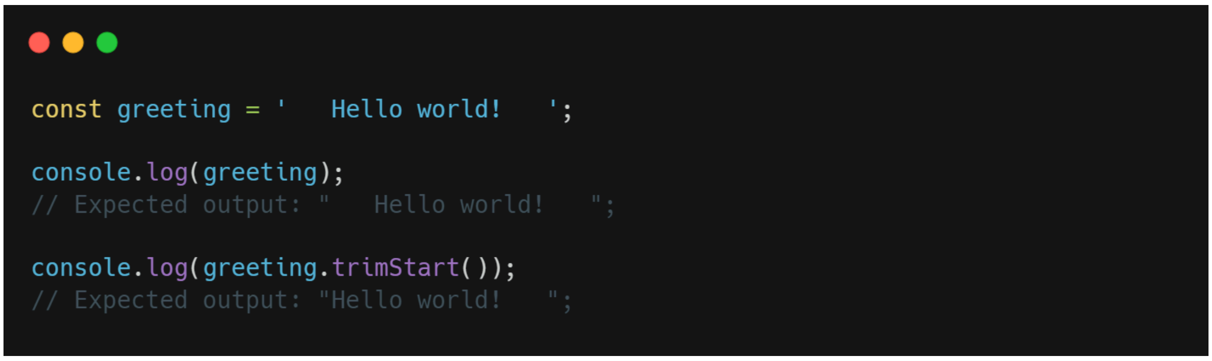
trimStart()

trimLeft()

Description

The trimStart() method of String values removes whitespace from the beginning of this string and returns a new string, without modifying the original string. trimLeft() is an alias of this method.

Example



```
const greeting = '   Hello world!   ';

console.log(greeting);
// Expected output: "   Hello world!   ";

console.log(greeting.trimStart());
// Expected output: "Hello world!   ";
```

`valueOf()`

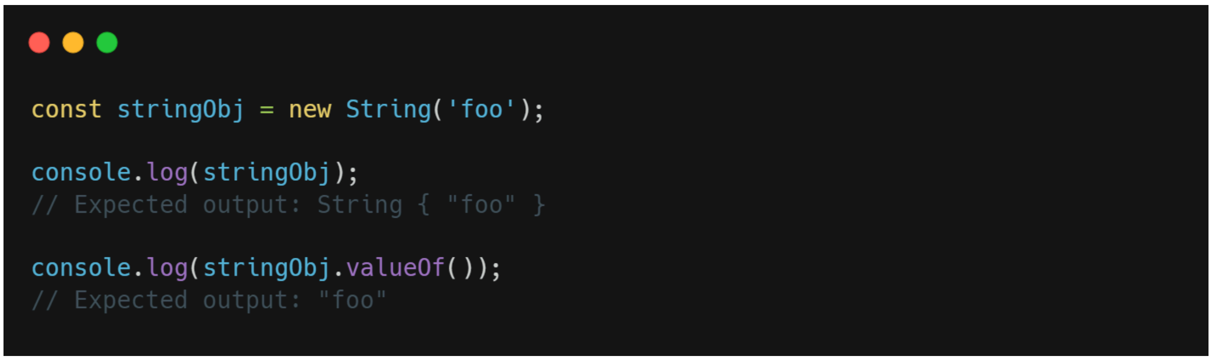
Syntax

valueOf()

Description

The valueOf() method of String values returns this string value.

Example



```
const stringObj = new String('foo');

console.log(stringObj);
// Expected output: String { "foo" }

console.log(stringObj.valueOf());
// Expected output: "foo"
```