String.prototype.substring()

La méthode **`substring()`** retourne une sous-chaîne de la chaîne courante, entre un indice de début et un indice de fin.

JavaScript Demo: String.substring()

```
1 var str = 'Mozilla';
2
3 console.log(str.substring(1, 3));
4 // expected output: "oz"
5
6 console.log(str.substring(2));
7 // expected output: "zilla"
8
```

Run ›

Reset

Syntaxe

```
str.substring(indiceA[, indiceB])
```

Paramètres

`indiceA`
Un entier compris entre 0 et la longueur de la chaîne.

`indiceB`
Paramètre optionnel : un entier compris entre 0 et la longueur de la chaine.

Valeur de retour

Une nouvelle chaîne de caractères qui correspond à la section souhaitée de la chaîne appelante.

Description

`substring` extrait des caractères de la chaîne courante à partir de `indiceA` jusqu'à `indiceB` (non compris). On a notamment :

- Si `indiceA` est égal à `indiceB`, `substring` retournera une chaîne vide.
- Si `indiceB` est omis, `substring` effectuera l'extraction des caractères jusqu'à la fin de la chaîne.
- Si l'un des deux arguments est négatif ou vaut `NaN`, il sera traité comme 0.
- Si l'un des deux arguments est plus grand que `str.length`, il sera traité comme `str.length`.

Si `indiceA` est supérieur à `indiceB`, la fonction `substring()` intervertira ces deux valeurs afin de les traiter comme si elles avaient été passées dans le bon ordre. Par exemple : `str.substring(1, 0) == str.substring(0, 1)`.

Exemples

Utiliser `substring()`

Les exemples suivants utilisent la méthode `substring()` pour extraire et afficher des caractères à partir de la chaine `"Mozilla"` :

```
var uneChaîne = "Mozilla";

// Affiche "Moz"
console.log(uneChaîne.substring(0,3));
console.log(uneChaîne.substring(3,0));

// Affiche "lla"
console.log(uneChaîne.substring(4,7));
console.log(uneChaîne.substring(4));
console.log(uneChaîne.substring(7,4));

// Affiche "Mozill"
console.log(uneChaîne.substring(0,6));

// Affiche "Mozilla"
console.log(uneChaîne.substring(0,7));
console.log(uneChaîne.substring(0,10));
```

Remplacer une sous-chaîne dans une chaîne

L'exemple suivant remplace une partie d'une chaine. Elle remplace à la fois les caractères individuels et les sous-chaines. La fonction appelée à la fin de cet exemple transforme la chaine `"Brave New World"` en `"Brave New Web"`.

```
function replaceString(oldS, newS, fullS) {
// On remplace oldS avec newS dans fullS
  for (var i = 0; i < fullS.length; i++) {
    if (fullS.substring(i, i + oldS.length) == oldS) {
     fullS = fullS.substring(0, i) + newS + fullS.substring(i + oldS.len
    }
  }
  return fullS;
}

```

```
11
    replaceString("World", "Web", "Brave New World");
```

Attention : ceci peut résulter en une boucle infinie si `oldS` est elle-même une sous-chaine de `newS` -- par exemple, si on essaie de remplacer "World" par "OtherWorld". Une meilleure solution serait de remplacer les chaines de cette manière :

```
1  function replaceString(oldS, newS,fullS){
2    return fullS.split(oldS).join(newS);
3  }
```

Le code ci-dessus sert d'exemple pour les opérations sur les sous-chaines. S'il est nécessaire de remplacer des sous-chaines, la plupart du temps il faudrait préférer l'utilisation de `String.prototype.replace()`.

Différence entre `substring()` et `substr()`

Il existe une légère différence entre les méthodes `substring()` et `substr()`. Les deux ne doivent pas être confondues.

Les arguments de la méthode `substring()` représentent les indices de début et de fin sur la chaîne. Pour `substr()`, les arguments représentent l'indice de début et le nombre de caractères à utiliser pour la chaîne résultante.

```
1  var texte = "Mozilla";
2  console.log(texte.substring(2,5)); // => "zil"
3  console.log(texte.substr(2,3)); // => "zil"
```

Différences entre `substring()` et `slice()`

Les méthodes `substring()` et `slice()` sont très proches mais certaines différences les distinguent, notamment la façon de gérer les arguments négatifs.

La méthode `substring()` échangera les deux arguments si `indiceA` est supérieur à `indiceB` et renverra donc une chaîne de caractères. La méthode `slice()` n'échange pas les arguments et renvoie donc une chaîne vide si le premier est supérieur au second :

```
var text = 'Mozilla';
console.log(text.substring(5, 2)); // => "zil"
console.log(text.slice(5, 2));     // => ""
```

Si l'un ou l'autre des arguments sont négatifs ou valent `NaN`, la méthode `substring()` les traitera comme s'ils valaient `0`.

```
console.log(text.substring(-5, 2)); // => "Mo"
console.log(text.substring(-5, -2)); // => ""
```

`slice()` traite également `NaN` comme `0`, mais parcourt la chaîne à partir de la fin lorsque des arguments négatifs sont utilisés.

```
console.log(text.slice(-5, 2)); // => ""
console.log(text.slice(-5, -2)); // => "zil"
```

Pour plus d'exemples sur l'utilisation d'arguments négatifs, voir la page `slice()`.

Spécifications

Spécification	État	Commentaires
ECMAScript 1st Edition (ECMA-262)	ST Standard	Implémentée avec JavaScript 1.0.
ECMAScript 5.1 (ECMA-262) La définition de 'String.prototype.substring' dans cette spécification.	ST Standard	
ECMAScript 2015 (6th Edition, ECMA-262) La définition de 'String.prototype.substring' dans cette spécification.	ST Standard	
ECMAScript Latest Draft (ECMA-262) La définition de 'String.prototype.substring' dans cette spécification.	D Projet	

Compatibilité des navigateurs

Update compatibility data on GitHub

`substring`

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

What are we missing?

Support complet

Voir aussi

- `String.prototype.substr()`
- `String.prototype.slice()`

Dernière modification : 9 avr. 2019, by MDN contributors

Sujets associés

Objets standards

`String`

Propriétés

`String.prototype`

`String.length`

Méthodes

`String.fromCharCode()`

`String.fromCodePoint()`

`String.prototype.anchor()`

`String.prototype.big()`

`String.prototype.blink()`

`String.prototype.bold()`

`String.prototype.charAt()`

`String.prototype.charCodeAt()`

`String.prototype.codePointAt()`

`String.prototype.concat()`

`String.prototype.endsWith()`

`String.prototype.fixed()`

`String.prototype.fontcolor()`

`String.prototype.fontsize()`

`String.prototype.includes()`

String.prototype.indexOf()

String.prototype.italics()

String.prototype.lastIndexOf()

String.prototype.link()

String.prototype.localeCompare()

String.prototype.match()

String.prototype.matchAll()

String.prototype.normalize()

String.prototype.padEnd()

String.prototype.padStart()

String.prototype.quote()

String.prototype.repeat()

String.prototype.replace()

String.prototype.search()

String.prototype.slice()

String.prototype.small()

String.prototype.split()

String.prototype.startsWith()

String.prototype.strike()

String.prototype.sub()

String.prototype.substr()

String.prototype.substring()

String.prototype.sup()

String.prototype.toLocaleLowerCase()

String.prototype.toLocaleUpperCase()

String.prototype.toLowerCase()

String.prototype.toSource()

String.prototype.toString()

String.prototype.toUpperCase()

String.prototype.trim()

String.prototype.trimEnd()

String.prototype.trimStart()

String.prototype.valueOf()

String.prototype[@@iterator]()

String.raw()

Héritage :

`Function`

Propriétés

Function.arguments

Function.arity

Function.caller

Function.displayName

Function.length

Function.name

Function.prototype

Méthodes

Function.prototype.apply()

Function.prototype.bind()

Function.prototype.call()

Function.prototype.isGenerator()

Function.prototype.toSource()

Function.prototype.toString()

`Object`

Propriétés

Object.prototype.__count__

Object.prototype.__noSuchMethod__

Object.prototype.__parent__

Object.prototype.__proto__

Object.prototype.constructor

Méthodes

Object.prototype.__defineGetter__()

Object.prototype.__defineSetter__()

Object.prototype.__lookupGetter__()

```
Object.prototype.__lookupSetter__()
Object.prototype.hasOwnProperty()
Object.prototype.isPrototypeOf()
Object.prototype.propertyIsEnumerable()
Object.prototype.toLocaleString()
Object.prototype.toSource()
Object.prototype.toString()
  Object.prototype.unwatch()
Object.prototype.valueOf()
  Object.prototype.watch()
Object.setPrototypeOf()
```