

Create an Array

By literal way

```
[element0, element1, ..., elementM]
new Array(element0, element1[, ...,
elementM])
new Array(arrayLength)
```

By using the result of a math

an array is returned by

- RegExp.exec
- String.match
- String.replace

By using Methods

```
Array.from(arrayLike[, mapFn[, thisArg]])
```

an array-like object (object with a `length` property and indexed elements, such as `arguments`) or iterable object (object where you can get its elements, such as `Map` and `Set`).

```
Array.of(element0[, element1[, ...,
elementM]])
```

Every argument is considered as an element in the array.

`Array.from` and `Array.of` work like `Array` constructor to create a new array.

Array instance mutator methods

```
copyWithin(target[, start[, end = this.length]])
```

target Target start index

start Source start index, if it is negative, treated as `length + start`

```
fill(value[, start = 0[, end = this.length]])
```

value Value to fill an array

```
sort([compareFunction])
```

Array instance mutator methods (cont)

```
function compare(a, b) {
  if (a less than b, or a should be in front of
  b) {return -1;}
  if (a is greater than b or be behind of b)
  {return 1;}
  return 0; // a must be equal to b
}
```

```
splice(start, deleteCount[, item1[, item2[,
...]])
```

start Index at which to start changing the array. If greater than the length, will set to length; if negative, will begin from the end.

delete If 0, will not delete any element. If
Count Omitted, will be equal to `length - start`

item1, The elements to add to the array
item2

returns an array containing the deleted elements

These methods modify the array

Array instance Accessor methods

concat `var new_array = old_array.concat(value1, value2, ... valueN))`

slice `var shallow_copy = arr.slice([begin[, end]])`

begin Zero-based. If negative, indicate an offset from the end

join `str = arr.join([separator = ','])`

indexOf `index = arr.indexOf(search-Element[, fromIndex = 0])`

lastIndexOf `index = arr.indexOf(search-Element[, fromIndex = arr.length - 1])`

Array instance Iteration methods

forEach

map return new array

filter return new array

every return `true` if every element satisfies testing function

some return `true` if at least one element satisfies testing function

find return the found value or `undefined`

findIndex return the found Index or `-1`

`callback[, thisArg]`



By LeiQ (amethystlei)

cheatography.com/amethystlei/

Published 30th September, 2020.

Last updated 19th October, 2020.

Page 1 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Array instance Iteration methods (cont)

callback	currentValue
	index
	array
reduce	accumulate
reduce-Right	accumulate from end
	<code>callback[, initialValue]</code>
callback	previousValue
	currentValue
	currentIndex
	array
entries	key/value pairs
keys	keys
values	values
	return an new <code>Array Iterator</code> object

Some of them can also use for array-like objects by `Array.prototype.fun.call(array-like object, args)`

All the Array instance methods

copyWithin	fill	pop	push
reverse	shift	sort	splice
unshift	concat	join	slice
toString	toLocaleString	indexOf	lastIndexOf
forEach	entries	every	some
filter	find	findIndex	keys
map	reduce	reduce-Right	values

Create a string

String literals

'string text'

"string text"

`String(text)`

``string text ${variable}`` template strings

Create by Char codes

`String.fromCharCode(num1, ..., numM)`

String instance methods unrelated to HTML

concat `str.concat(string2, string3, ..., stringN)`

repeat `str.repeat(count)`
count will convert to integer
'abc'.repeat(2) -> 'abcabc'

includes `str.funcName(searchString[, position])`
endsWith
startsWith

search-String A string to be searched for within this string

position search from, optional

case-sensitivity, return `true` or `false`

String instance methods related with RegExp

search `str.search(regex)`
return the index of first match or -1

match `str.match(regex)`
same as `regexObj.exec(str)`
return an array containing the entire match result or `null`
result: input, index, 0, 1-n

replace `str.replace(regex|substr, newSubStr|function)`

regex pattern

substr replacement

newSubStr **function**
newSubStr can include some special replacement patterns

`$$` inserts a '\$'

`$&` inserts the matched substring

`$'` inserts the portion of the string that precedes the matched string

`$'` inserts the portion of the string that follows the matched substring



By LeiQ (amethystlei)

cheatography.com/amethystlei/

Published 30th September, 2020.

Last updated 19th October, 2020.

Page 2 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

String instance methods related with RegExp (cont)

`$n` or `$nn` `n` and `nn` are decimal digits, inserts the *n*th parenthesized submatch string

function's result will be used as the replacement, and the arguments is:

`match` like `$&`

`p1,`
`p2, ...` like `$n`

`offset` The offset of the matched substring within the whole string

`string` the whole string

String instance methods related HTML

`anchor` `str.anchor(name)`
create and display an anchor (name property) in a document
e.g. `<a name="name" >str </a>`

`link` `str.link(url)`
create an HTML snippet for a hypertext link
e.g. `<a href="url"> str </a>`

All the String instance methods

<code>charAt</code>	<code>charCodeAt</code>	<code>concat</code>	<code>includes</code>
<code>endsWith</code>	<code>indexOf</code>	<code>lastIndexOf</code>	<code>localeCompare</code>
<code>match</code>	<code>repeat</code>	<code>replace</code>	<code>search</code>
<code>slice</code>	<code>split</code>	<code>startsWith</code>	<code>substr</code>
<code>substring</code>	<code>toLocaleLowerCase</code>	<code>toLocaleUpperCase</code>	<code>toLowerCase</code>

All the String instance methods (cont)

<code>toString</code>	<code>toUpperCase</code>	<code>trim</code>	<code>valueOf</code>
<code>anchor</code>	<code>link</code>		

Escape notation

<code>\0</code>	the NULL character
<code>'</code>	single quote
<code>"</code>	double quote
<code>\\</code>	backslash
<code>\n</code>	new line
<code>\r</code>	carriage return
<code>\v</code>	vertical tab
<code>\t</code>	tab
<code>\b</code>	backspace
<code>\f</code>	form feed
<code>\uXXXX</code>	unicode codepoint
<code>\xxx</code>	the Latin-1 character

Create a RegExp

literal notation

`/pattern/flags`

constructor

`new RegExp(pattern[, flags])`

pattern	The text
flags	

g	global match
i	ignore case
m	multiline

use `test` to test for a match in its string parameter.

Static property you may use

RegExp.lastIndex

The index at which to start the next match

RegExp Quick Reference

Character classes

<code>.</code>	any character except newline
<code>\w \d \s</code>	word, digit, whitespace
<code>\W \D \S</code>	not word, digit, whitespace
<code>[abc]</code>	any of a, b, or c
<code>[^abc]</code>	not a, b, or c
<code>[a-g]</code>	character between a & g

Anchors

<code>^abc\$</code>	start / end of the string
<code>\b \B</code>	word, not-word boundary

Escaped characters

<code>\. * \</code>	escaped special characters
<code>\t \n \r</code>	tab, linefeed, carriage return

Groups & Lookaround

<code>(abc)</code>	capture group
<code>\1</code>	backreference to group #1
<code>(?:abc)</code>	non-capturing group
<code>(?=abc)</code>	positive lookahead
<code>(?!abc)</code>	negative lookahead

Quantifiers & Alternation

<code>a* a+ a?</code>	0 or more, 1 or more, 0 or 1
<code>a{5} a{2,}</code>	exactly five, two or more
<code>a{1, 3}</code>	between one & three
<code>a+? a{2}?</code>	match as few as possible
<code>ab cd</code>	match ab or cd

Reference from regexr.com



By LeiQ (amethystlei)

cheatography.com/amethystlei/

Published 30th September, 2020.
Last updated 19th October, 2020.
Page 3 of 3.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)
Learn to solve cryptic crosswords!
[http://crosswordcheats.com](https://crosswordcheats.com)