JavaScript Array, String and RegExp Cheat Sheet

by LeiQ (amethystlei) via cheatography.com/27322/cs/7820/

Create an Array

By literal way

[*element0*, *element1*, ..., *elementN*]

new Array(*element0*, *element1*[, ...[, *elementN*]])

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*[, ...[, *elementN*]]])

Every argument is considered as an element in the array.

`Array.from` and `Array.of` work like `Array` constructor to create an 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.
deleteCount	If 0, will not delete any element. If Omitted, will be equal to `length - start`
item1, item2	The elements to add to the array

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(*searchElement*[, *fromIndex* = 0])
lastIndexOf	*index* = *arr*.indexOf(*searchElement*[, *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.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

JavaScript Array, String and RegExp Cheat Sheet

by LeiQ (amethystlei) via cheatography.com/27322/cs/7820/

Array instance Iteration methods (cont)

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

Create a string

String literals

'string text'

"string text"

String(*text*)

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

Create by Char codes

String.fromCharCode(*num1*[, ...[, *numN*]])

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 endsWith startsWith	*str.funcName*(*searchString*[, *position*])	
	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(*regexp*)	
	return the index of first match or -1	
match	*str*.match(*regexp*) same as *regexObj*.exec(*str*)	
	return an array containing the entire match result or **null**	
	result: input, index, 0, 1-n	
replace	*str*.replace(*regexp*|*substr*, *newSubStr*|*function*)	
	regexp substr	pattern
	newSubStr function	replacement
		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.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Cheatography

JavaScript Array, String and RegExp Cheat Sheet

by LeiQ (amethystlei) via cheatography.com/27322/cs/7820/

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="n ame " >st r </ a>`
link	*str*.link(*url*)
	create an HTML snippet for a hypertext link e.g. `<a href="u rl"> str </ a>`

All the String instance methods

charAt	charCodeAt	concat	includes
endsWith	indexOf	lastIn-dexOf	locale-Compare
match	repeat	replace	search
slice	split	startsWith	substr
substring	toLocaleL-owerCase	toLoca-leUppe-rCase	toLowe-rCase

All the String instance methods (cont)

toString	toUpperCase	trim	valueOf
anchor	link		

Escape notation

\0	the NULL character
\'	single quote
\"	double quote
\\	backslash
\n	new line
\r	carriage return
\v	vertical tab
\t	tab
\b	backspace
\f	form feed
\uXXXX	unicode codepoint
\xXX	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

RegEx Quick Reference

Character classes	
.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g
Anchors	
^abc$	start / end of the string
\b \B	word, not-word boundary
Escaped characters	
\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
Groups & Lookaround	
(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead
Quantifiers & Alternation	
a* a+ a?	0 or more, 1 or more, 0 or 1
a{5} a{2,}	exactly five, two or more
a{1, 3}	between one & three
a+? a{2,}?	match as few as possible
ab|cd	match ab or cd

Reference from regexr.com

By **LeiQ** (amethystlei)

cheatography.com/amethystlei/

Published 30th September, 2020.
Last updated 19th October, 2020.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com