{JS}
MASTERY

{JS}
MASTERY

JavaScript CheatSheet

- String Methods
- Array Methods

String Methods

`string[index]`

get a certain character of a string

`string.length`

return the number of characters in a string

`string.split(' ')`

returns an array of words of a string

`string.split('')`

returns an array of characters of a string

`string.toLowerCase()`

returns a lowercased string

`string.toUpperCase()`

returns an uppercased string

String Methods

`string.charAt(index)`

returns a new string consisting of the single character located at the specified offset into the string.

`string.replace(substr, newSubstr)`

returns a new string with a substring (substr) replaced by a new one (newSubstr).

`string.includes(searchString)`

performs a case-sensitive search to determine whether one string may be found within another string, returns true or false.

`string.substr(start, length)`

returns a portion of the string, starting at the specified index and extending for a given number.

String Methods

`string.includes('subtring')`

checks whether a substring exists inside of a string [check the character case]

`string.indexOf(searchValue)`

returns the index of the first occurrence of the specified value, starting the search at fromIndex. Returns -1 if the value is not found.

`string.lastIndexOf(searchValue)`

returns the index of the last occurrence of the specified value, searching backwards from fromIndex. Returns -1 if the value is not found.

`string.slice(beginIndex, endIndex)`

extracts a section of a string and returns it as a new string, without modifying the original string.

Array Methods

`array[index]`

returns a certain value from an array

`push(value)`

adds the value to the end of the array

`pop()`

removes the value from the end of the array

`shift()`

removes the value from the start of the array

`unshift(value)`

adds the value to the start of the array

`splice(fromIndex, no_of_elements)`

removes the number_of_elements, starting from fromIndex from the array

Array Methods

`slice(fromIndex, toIndex)`

copies a certain part of the array

`concat()`

Join several arrays into one

`join('')`

returns a string of array values

`array.length`

returns the number of elements in the array

`reverse()`

reverse the order of the elements in an array

`toString()`

returns a string representing the specified array and its elements.

Array Methods

`toString()`

returns a string representing the specified array and its elements.

`includes(searchElement)`

determines whether an array includes a certain value among its entries, returning true or false as appropriate.

`sort()`

It sorts the elements of an array in place and returns the sorted array. It sorts an array alphabetically.

`indexOf(searchElement)`

returns the index of the first occurance of that value

`lastIndexOf(searchElement)`

returns the index of the last occurance of that value

Looping

For Loop

```
for (st 1; st 2; st 3) {
  // code block to be executed
}
```

st 1 is executed (one time) before the execution of the code block.

st 2 defines the condition for executing the code block.

st 3 is executed (every time) after the code block has been executed.

While Loop

```
while (condition) {
  // code block to be executed
}
```

Looping

Do While Loop

```
do {
  // code block to be executed
}
while (condition);
```

For In Loop

```
for (key in object) {
  // code block to be executed
}
```

For Of Loop

```
for (variable of iterable) {
  // code block to be executed
}
```

Array methods for looping

array.forEach()

It executes a provided function once for each array element.

```
array.forEach((element, index) => {
  // code block to be executed
})
```

---

array.map()

It creates a new array populated with the results of calling a provided function on every element in the calling array.

```
array.map((element, index) => {
  // code block to be executed
})
```

Array method for looping

array.filter()

It creates a new array with all elements that pass the test implemented by the provided function.

```
array.filter((element, index) => {
  // code block to be executed
})
```

---

array.findIndex()

It returns the index of the first element in the array that satisfies the provided testing function

```
array.findIndex((el, idx, arr) => {
  // code block to be executed
})
```

Array method for looping

array.some()

It tests whether at least one element in the array passes the test implemented by the provided function

```
array.some((el, index, array)) => {
  // code block to be executed
})
```

array.every()

It tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

```
array.every((element, index) => {
  // code block to be executed
})
```

Array method for looping

array.reduce()

It runs a function on each array element to produce (reduce it to) a single value. It works from left-to-right.

```
array.reduce((prevValue,currentValue, currentIndex, array)) => {
  // code block to be executed
})
```

array.reduceRight()

It runs a function on each array element to produce (reduce it to) a single value. It works from right-to-left.

```
array.reduceRight((accumulator, currentValue, index, array)) => {
  // code block to be executed
})
```

VALUE VS REFERENCE

Arrays

```
const numbers = [1, 2, 3, 4]
const anotherNumbers = numbers
anotherNumbers.push(5);

console.log(numbers === anotherNumbers); // true
```

Objects

```
const person = { firstName: 'John', lastName: 'Doe' };
const anotherPerson = person;
anotherPerson.lastName = 'Smith';

console.log(person === anotherPerson); // true
```

VALUE VS REFERENCE

Cloning Array

One Level Deep

```
// Shallow cloning - One level Deep
const original = [1, 2, 3, 4];
const newOg = [...original];
newOg.push(5);

console.log(original === newOg) // false
```

VALUE VS REFERENCE

Cloning Array

Two Level Deep

```
// Deep cloning - Two level Deep
const users = [
  { name: 'John', age: 24 },
  { name: 'Victor', age: 31 },
];
const newUser = JSON.parse(JSON.stringify(users))
console.log(users === newUser) // false
```

Want to Become a JavaScript Master ?

If you're serious about learning JavaScript and getting a job as a web developer. I highly encourage you to enroll in my **Complete Path to JavaScript Mastery** Course. Don't waste your time following disconnected, outdated tutorials. My Course has everything you need in one place:

This course has everything you need in one place:

- 10 hours of HD video
- Unlimited access
- Self-paced learning - take your time if you prefer
- Watch it online or download and watch offline
- 30-day money-back guarantee
- Certificate of completion

Special Coupon Code

The price for this course is $129 but the first 200 people who have downloaded this cheat sheet can get more than 50% off using the coupon code:

CHEATSHEET

https://www.completepathtojavascriptmastery.com