Cheat sheet of Array methods in JavaScript

Array CheatSheet.md

# Array Methods Cheat Sheet

## 1. push()

- **Definition:** Adds one or more elements to the end of an array and returns the new length of the array.
- **Syntax:** `array.push(element1[, ...[, elementN]])`
- **Example:**

```
let arr = [1, 2, 3];
arr.push(4, 5);
console.log(arr); // Output: [1, 2, 3, 4, 5]
```

## 2. pop()

- **Definition:** Removes the last element from an array and returns that element.
- **Syntax:** `array.pop()`
- **Example:**

```
let arr = [1, 2, 3];
let removedElement = arr.pop();
console.log(removedElement); // Output: 3
console.log(arr); // Output: [1, 2]
```

## 3. shift()

- **Definition:** Removes the first element from an array and returns that element.
- **Syntax:** `array.shift()`
- **Example:**

```
let arr = [1, 2, 3];
let shiftedElement = arr.shift();
console.log(shiftedElement); // Output: 1
console.log(arr); // Output: [2, 3]
```

## 4. unshift()

- **Definition:** Adds one or more elements to the beginning of an array and returns the new length of the array.
- **Syntax:** `array.unshift(element1[, ...[, elementN]])`
- **Example:**

```
let arr = [2, 3];
arr.unshift(0, 1);
console.log(arr); // Output: [0, 1, 2, 3]
```

## 5. concat()

- **Definition:** Returns a new array comprised of the array on which it is called joined with the array(s) and/or value(s) provided as arguments.
- **Syntax:** `array.concat(value1[, value2[, ...[, valueN]]])`
- **Example:**

```
let arr1 = [1, 2];
let arr2 = [3, 4];
let newArr = arr1.concat(arr2);
console.log(newArr); // Output: [1, 2, 3, 4]
```

## 6. slice()

- **Definition:** Returns a shallow copy of a portion of an array into a new array object selected from begin to end (end not included) where begin and end represent the index of items in that array.
- **Syntax:** `array.slice(begin[, end])`
- **Example:**

```
let arr = [1, 2, 3, 4, 5];
let newArr = arr.slice(1, 3);
console.log(newArr); // Output: [2, 3]
```

## 7. splice()

- **Definition:** Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.
- **Syntax:** `array.splice(start[, deleteCount[, item1[, item2[, ...]]]])`
- **Example:**

```
let arr = [1, 2, 3, 4, 5];
arr.splice(1, 2, 'a', 'b');
console.log(arr); // Output: [1, 'a', 'b', 4, 5]
```

## 8. forEach()

- **Definition:** Executes a provided function once for each array element.
- **Syntax:** `array.forEach(callback(currentValue [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [1, 2, 3];
arr.forEach(element => console.log(element * 2));
// Output:
// 2
// 4
// 6
```

## 9. map()

- **Definition:** Creates a new array populated with the results of calling a provided function on every element in the calling array.
- **Syntax:** `array.map(callback(currentValue [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [1, 2, 3];
let newArr = arr.map(element => element * 2);
console.log(newArr); // Output: [2, 4, 6]
```

## 10. filter()

```
- **Definition:** Creates a new array with all elements that pass the test
implemented by the provided function.
- **Syntax:** `array.filter(callback(element [, index [, array]])[, thisArg])`
- **Example:**
  ```javascript
  let arr = [1, 2, 3, 4, 5];
```
```

```
let newArr = arr.filter(element => element % 2 === 0);
console.log(newArr); // Output: [2, 4]
```

## 11. find()

- **Definition:** Returns the value of the first element in the array that satisfies the provided testing function. Otherwise, it returns `undefined`.
- **Syntax:** `array.find(callback(element [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [5, 12, 8, 130, 44];
let found = arr.find(element => element > 10);
console.log(found); // Output: 12
```

## 12. findIndex()

- **Definition:** Returns the index of the first element in the array that satisfies the provided testing function. Otherwise, it returns -1.
- **Syntax:** `array.findIndex(callback(element [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [5, 12, 8, 130, 44];
let foundIndex = arr.findIndex(element => element > 10);
console.log(foundIndex); // Output: 1
```

## 13. every()

- **Definition:** Tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.
- **Syntax:** `array.every(callback(element [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [30, 40, 50, 60];
let allOver20 = arr.every(element => element > 20);
console.log(allOver20); // Output: true
```

## 14. some()

- **Definition:** Tests whether at least one element in the array passes the test implemented by the provided function. It returns a Boolean value.

- **Syntax:** `array.some(callback(element [, index [, array]])[, thisArg])`
- **Example:**

```
let arr = [10, 20, 30, 40, 50];
let someOver30 = arr.some(element => element > 30);
console.log(someOver30); // Output: true
```

## 15. includes()

- **Definition:** Determines whether an array includes a certain value among its entries, returning true or false as appropriate.
- **Syntax:** `array.includes(valueToFind [, fromIndex])`
- **Example:**

```
let arr = [1, 2, 3];
let includesTwo = arr.includes(2);
console.log(includesTwo); // Output: true
```

## 16. indexOf()

- **Definition:** Returns the first index at which a given element can be found in the array, or -1 if it is not present.
- **Syntax:** `array.indexOf(searchElement [, fromIndex])`
- **Example:**

```
let arr = [2, 9, 9];
let index = arr.indexOf(9);
console.log(index); // Output: 1
```

## 17. reduce()

- **Definition:** Executes a reducer function on each element of the array, resulting in a single output value.
- **Syntax:** `array.reduce(callback(accumulator, currentValue[, index, array])[, initialValue])`
- **Example:**

```
let arr = [1, 2, 3, 4];
let sum = arr.reduce((acc, curr) => acc + curr);
console.log(sum); // Output: 10
```

## 18. reduceRight()

- **Definition:** Similar to `reduce()`, but applies the function from right to left instead of left to right.
- **Syntax:** `array.reduceRight(callback(accumulator, currentValue[, index, array])[, initialValue])`
- **Example:**

```
let arr = ['a', 'b', 'c', 'd'];
let concat = arr.reduceRight((acc, curr) => acc + curr);
console.log(concat); // Output: 'dcba'
```

## 19. join()

- **Definition:** Creates and returns a new string by concatenating all of the elements in an array, separated by commas or a specified separator string.
- **Syntax:** `array.join(separator)`
- **Example:**

```
let arr = ['Hello', 'World'];
let str = arr.join(' ');
console.log(str); // Output: 'Hello World'
```

## 20. reverse()

- **Definition:** Reverses the elements of an array in place. The first array element becomes the last, and the last array element becomes the first.
- **Syntax:** `array.reverse()`
- **Example:**

```
let arr = [1, 2, 3];
arr.reverse();
console.log(arr); // Output: [3, 2, 1]
```

## 21. sort()

- **Definition:** Sorts the elements of an array in place and returns the sorted array.
- **Syntax:** `array.sort([compareFunction])`
- **Example:**

```
let arr = [5, 2, 1, 4, 3];
arr.sort((a, b) => a - b);
```

```
console.log(arr); // Output: [1, 2, 3, 4, 5]
```

## 22. toString()

- **Definition:** Returns a string representing the specified array and its elements.
- **Syntax:** `array.toString()`
- **Example:**

```
let arr = [1, 2, 3];
let str = arr.toString();
console.log(str); // Output: '1,2,3'
```

## 23. flat()

- **Definition:** Creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.
- **Syntax:** `array.flat([depth])`
- **Example:**

```
let arr = [1, 2, [3, 4, [5, 6]]];
let flatArr = arr.flat(2);
console.log(flatArr); // Output: [1, 2, 3, 4, 5, 6]
```

## 24. flatMap()

- **Definition:** First maps each element using a mapping function, then flattens the result into a new array.
- **Syntax:** `array.flatMap(callback(currentValue[, index[, array]])[, thisArg])`
- **Example:**

```
let arr = [1, 2, 3];
let mappedArr = arr.flatMap(x => [x * 2]);
console.log(mappedArr); // Output: [2, 4, 6]
```

## 25. length

- **Definition:** Returns the number of elements in the array.
- **Syntax:** `array.length`
- **Example:**

```
let arr = [1, 2, 3, 4, 5];
```

```
console.log(arr.length); // Output: 5
```

## 26. isArray()

- **Definition:** Determines whether the passed value is an Array.
- **Syntax:** `Array.isArray(value)`
- **Example:**

```
let arr = [1, 2, 3];
console.log(Array.isArray(arr)); // Output: true
```

## 27. fill()

- **Definition:** Fills all the elements of an array from a start index to an end index with a static value.
- **Syntax:** `array.fill(value[, start[, end]])`
- **Example:**

```
let arr = [1, 2, 3, 4, 5];
arr.fill(0, 2, 4);
console.log(arr); // Output: [1, 2, 0, 0, 5]
```

## 28. keys()

- **Definition:** Returns a new Array Iterator object that contains the keys for each index in the array.
- **Syntax:** `array.keys()`
- **Example:**

```
let arr = ['a', 'b', 'c'];
let iterator = arr.keys();
for (let key of iterator) {
  console.log(key); // Output: 0, 1, 2
}
```

## 29. values()

- **Definition:** Returns a new Array Iterator object that contains the values for each index in the array.
- **Syntax:** `array.values()`
- **Example:**

```
let arr = ['a', 'b', 'c'];
let iterator = arr.values();
for (let value of iterator) {
  console.log(value); // Output: 'a', 'b', 'c'
}
```

## 30. entries()

- **Definition:** Returns a new Array Iterator object that contains the key/value pairs for each index in the array.
- **Syntax:** `array.entries()`
- **Example:**

```
let arr = ['a', 'b', 'c'];
let iterator = arr.entries();
for (let entry of iterator) {
  console.log(entry); // Output: [0, 'a'], [1, 'b'], [2, 'c']
}
```

## 31. from()

- **Definition:** Creates a new, shallow-copied Array instance from an array-like or iterable object.
- **Syntax:** `Array.from(arrayLike[, mapFn[, thisArg]])`
- **Example:**

```
let arrLike = 'hello';
let newArr = Array.from(arrLike);
console.log(newArr); // Output: ['h', 'e', 'l', 'l', 'o']
```

## 32. copyWithin()

- **Definition:** Copies a sequence of array elements within the array.
- **Syntax:** `array.copyWithin(target, start[, end])`
- **Example:**

```
let arr = [1, 2, 3, 4, 5];
arr.copyWithin(0, 3);
console.log(arr); // Output: [4, 5, 3, 4, 5]
```

## 33. at()

- **Definition:** Returns the element at the specified index in the array.
- **Syntax:** `array.at(index)`
- **Example:**

```
let arr = [10, 20, 30, 40, 50];
console.log(arr.at(2)); // Output: 30
```

## 34. lastIndexOf()

- **Definition:** Returns the last index at which a given element can be found in the array, or -1 if it is not present. Searches the array from a specified index if provided.
- **Syntax:** `array.lastIndexOf(searchElement[, fromIndex])`
- **Example:**

```
let arr = [2, 9, 9, 4, 6];
let index = arr.lastIndexOf(9);
console.log(index); // Output: 2
```

## 35. toLocaleString()

- **Definition:** Returns a string representing the elements of the array. The elements are converted to strings using their toLocaleString methods.
- **Syntax:** `array.toLocaleString([locales[, options]])`
- **Example:**

```
let arr = [1, new Date(), 'a', { key: 'value' }];
let str = arr.toLocaleString();
console.log(str); // Output: '1,Sat Feb 27 2024 19:58:46 GMT+0000 (Coordinate
```

## 36. of()

- **Definition:** Creates a new Array instance with a variable number of arguments.
- **Syntax:** `Array.of(element0[, element1[, ...[, elementN]]])`
- **Example:**

```
let arr = Array.of(1, 2, 3, 4, 5);
console.log(arr); // Output: [1, 2, 3, 4, 5]
```