JavaScript arrays come with a variety of built-in methods that make it easier to manipulate and interact with data. This document provides a concise overview of several important array methods, including their and examples.

1. length
The length property returns the number of elements in an array.

```
array.length
Example
let fruits = ['Apple', 'Banana', 'Cherry'];
console.log(fruits.length); // Output: 3
```

2. at()
The at() method returns the item at a given index. It accepts both positive and negative integers.

```
array.at(index)
Example
let fruits = ['Apple', 'Banana', 'Cherry'];
console.log(fruits.at(1)); // Output: Banana
console.log(fruits.at(-1)); // Output: Cherry
```

3. slice()
The slice() method returns a shallow copy of a portion of an array into a new array selected from start to end (end not included).

```
array.slice(start, end)
Example
let fruits = ['Apple', 'Banana', 'Cherry', 'Date'];
let citrus = fruits.slice(1, 3);
console.log(citrus); // Output: ['Banana', 'Cherry']
```

4. splice()
The splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements.

```
array.splice(start, deleteCount, item1, item2, ...)
Example
let fruits = ['Apple', 'Banana', 'Cherry'];
fruits.splice(1, 1, 'Mango');
console.log(fruits); // Output: ['Apple', 'Mango', 'Cherry']
```

5. pop()
The pop() method removes the last element from an array and returns that element. This method changes the length of the array.

```
array.pop()
Example
let fruits = ['Apple', 'Banana', 'Cherry'];
let last = fruits.pop();
console.log(last); // Output: Cherry
console.log(fruits); // Output: ['Apple', 'Banana']
```

6. push()
The push() method adds one or more elements to the end of an array and returns

the new length of the array.
array.push(element1, ..., elementN)
Example
```
let fruits = ['Apple', 'Banana'];
let newLength = fruits.push('Cherry');
console.log(fruits); // Output: ['Apple', 'Banana', 'Cherry']
console.log(newLength); // Output: 3
```

7. shift()
The shift() method removes the first element from an array and returns that element. This method changes the length of the array.

array.shift()
Example
```
let fruits = ['Apple', 'Banana', 'Cherry'];
let first = fruits.shift();
console.log(first); // Output: Apple
console.log(fruits); // Output: ['Banana', 'Cherry']
```

8. unshift()
The unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.
array.unshift(element1, ..., elementN)
Example
```
let fruits = ['Banana', 'Cherry'];
let newLength = fruits.unshift('Apple');
console.log(fruits); // Output: ['Apple', 'Banana', 'Cherry']
console.log(newLength); // Output: 3
```

9. join()
The join() method creates and returns a new string by concatenating all of the elements in an array, separated by commas or a specified separator string.
array.join(separator)
Example
```
let fruits = ['Apple', 'Banana', 'Cherry'];
let fruitString = fruits.join(', ');
console.log(fruitString); // Output: Apple, Banana, Cherry
```

10. concat()
The concat() method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.
array1.concat(array2, ..., arrayN)
Example

```
let fruits = ['Apple', 'Banana'];
let moreFruits = ['Cherry', 'Date'];
let allFruits = fruits.concat(moreFruits);
console.log(allFruits); // Output: ['Apple', 'Banana', 'Cherry', 'Date']
```

11. toString()
The toString() method returns a string representing the specified array and its elements.

array.toString()
Example

```
let fruits = ['Apple', 'Banana', 'Cherry'];
let fruitString = fruits.toString();
console.log(fruitString); // Output: Apple,Banana,Cherry
```

12. forEach()
The forEach() method executes a provided function once for each array element. It does not return a new array.

array.forEach(callback(currentValue, index, array), thisArg)
Example

```
let fruits = ['Apple', 'Banana', 'Cherry'];
fruits.forEach((fruit, index) => {
  console.log(`${index}: ${fruit}`);
});
// Output:
// 0: Apple
// 1: Banana
// 2: Cherry
```

13. map()
The map() method creates a new array populated with the results of calling a provided function on every element in the calling array.
array.map(callback(currentValue, index, array), thisArg)
Example

```
let numbers = [1, 2, 3, 4];
let doubled = numbers.map(number => number * 2);
console.log(doubled); // Output: [2, 4, 6, 8]
```

14. filter()
The filter() method creates a new array with all elements that pass the test implemented by the provided function.
array.filter(callback(element, index, array), thisArg)
Example

```
let numbers = [1, 2, 3, 4, 5];
let evenNumbers = numbers.filter(number => number % 2 === 0);
console.log(evenNumbers); // Output: [2, 4]
```

15. reduce()
The reduce() method executes a reducer function (that you provide) on each element of the array, resulting in a single output value.

array.reduce(callback(accumulator, currentValue, index, array), initialValue)
Example

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce((total, number) => total + number, 0);
console.log(sum); // Output: 10
```