1. Definition for AT: the method takes an integer value and returns the item at that index.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>at</title>
    <script>
        var arr1 = ['car' ,'bus' ,'bike']

        function returnLast(arr){
            return arr.at(-1);
        }

        var item1 = returnLast(arr1);
        console.log(item1);
    </script>
</head>
<body>
</body>
</html>
```

2. Definition for contact: method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
        var arr1 = ['j' ,'o' ,'s','h','n','a'];
        var arr2 = ['I' ,'B' ,'M'];
        var arr3 =arr1.concat(arr2);
        console.log(arr3);
    </script>
</head>
<body>
</body>
</html>
```

3. Definition for copy Within: method shallow copies part of an array to another location in the same array and returns it without modifying its length.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
       var arr1 = ['i' , 'b' , 'm' ,'e' ,'r'];
       console.log(arr1.copyWithin(0,1,3,4));
    </script>
</head>
<body>
</body>
</html>
```

4. Definition for entire: method returns a new **Array Iterator** object that contains the key/value pairs for each index in the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>

    <p id="demo"></p>
    <script>
       var house = ['villa','dublux','apartment', 'resort'];
        var h = house.entries();
        for(let x of h) {
            document.getElementById("demo").innerHTML +=x + "<br>";
        }
    </script>
</head>
<body>
</body>
</html>
```

5. Definition for every: method tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <p id="demo"></p>
    <script>
       var ages = ['19','20','21','22'];
       document.getElementById("demo").innerHTML = ages.every(checkAge);
       function checkAge(age) {
           return age > 12;
       }
    </script>
</head>
<body>
</body>
</html>
```

6. Definition for fill: method changes all elements in an array to a static value, from a start index (default `0`) to an end index (default `array.length`). It returns the modified array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
       var arr1 = ['19','20','21','22'];
       console.log(arr1.fill(6));
    </script>
</head>
<body>
</body>
</html>
```

7. Definition for filter: method **creates a new array** with all elements that pass the test implemented by the provided function.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
       var arr1 = ['dry','cry','shye','fry','whyeee','trie'];
       var result = arr1.filter(word => word.length > 5);
       console.log(result);
    </script>
</head>
<body>
</body>
</html>
```

8. Definition for find: method returns the first element in the provided array that satisfies the provided testing function. If no values satisfy the testing function, `undefined` is returned.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
       var arr1 = [19,20,21,22];
       var found = arr1.find(element  => element>  13)
       console.log(found);
    </script>
</head>
<body>
</body>
</html>
```

9. Definition for find index: method returns the **index** of the first element in the array **that satisfies the provided testing function**. Otherwise, it returns `-1`, indicating that no element passed the test.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
```

```
    <p id="demo"></p>
    <script>
       var ages = [3,10,18,20];
       document.getElementById("demo").innerHTML = ages.findIndex(checkAge);
       function checkAge(age){
           return age > 18
       }
    </script>
</head>
<body>
</body>
</html>
```

10. **Definition for flat:** method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
       var arr1 = [1,19,20,[[[[21,22]]]]];
       console.log(arr1.flat());
    </script>
</head>
<body>
</body>
</html>
```

11. **Definition for flat Map:** method returns a new array formed by applying a given callback function to each element of the array, and then flattening the result by one level. It is identical to a `map()` followed by a `flat()` of depth 1, but slightly more efficient than calling those two methods separately.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
```

```
        var arr1 = [1,2,5,22];
        console.log(arr1.flatMap(x => [x*2]));
    </script>
</head>
<body>
</body>
</html>
```

12. **Definition for for each:** method executes a provided function once for each array element.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
        var array1 = ['j', 'o', 's', 'h', 'n', 'a'];
         array1.forEach(element => console.log(element));
    </script>
</head>
<body>
</body>
</html>
```

13. **Definition for includes:** method determines whether an array includes a certain value among its entries, returning `true` or `false` as appropriate.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
      var pets = ['cat', 'dog', 'hen'];

      console.log(pets.includes('dog'));

    </script>
</head>
<body>
</body>
</html>
```

14. **Definition for index of:** method returns the first index at which a given element can be found in the array, or -1 if it is not present

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
      var beasts = ['ant', 'bison', 'camel', 'duck', 'bison'];

       console.log(beasts.indexOf('camel'));

    </script>
</head>
<body>
</body>
</html>
```

15. **Definition for join:** method creates and returns a new string by concatenating all of the elements in an array (or an [array-like object]), separated by commas or a specified separator string. If the array has only one item, then that item will be returned without using the separator.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
      var elements = ['Fire', 'Air', 'Water'];

console.log(elements.join());
// expected output: "Fire,Air,Water"

console.log(elements.join(''));
// expected output: "FireAirWater"

console.log(elements.join('-'));
// expected output: "Fire-Air-Water"

    </script>
</head>
```

```
<body>
</body>
</html>
```

16. Definition for keys: method returns a new **Array Iterator** object that contains the keys for each index in the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
     var array1 = ['ab', 'bc', 'cd'];
     var iterator = array1.keys();

     for (const key of iterator) {
     console.log(key);
    }

    </script>
</head>
<body>
</body>
</html>
```

17. Definition for last index of: method returns the last index at which a given element can be found in the array, or -1 if it is not present. The array is searched backwards, starting at `fromIndex`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
<script>
    var color = ['red', 'white', 'black', 'skyblue'];
    console.log(color.lastIndexOf('black'));
</script>
</head>
<body>
</body>
</html>
```

18. Definition for map: method **creates a new array** populated with the results of calling a provided function on every element in the calling array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
     var array1 = [1, 4, 9, 16];
  var map1 = array1.map(x => x * 2);
  console.log(map1);
    </script>
</head>
<body>
</body>
</html>
```

19. Definition for pop: method removes the **last** element from an array and returns that element. This method changes the length of the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
    var vegitables = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];
    console.log(vegitables.pop());
    </script>
</head>
<body>
</body>
</html>
```

20. Definition for push: method adds one or more elements to the end of an array and returns the new length of the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
   const animals = ['pigs', 'goats', 'sheep'];

var count = animals.push('dogs');
console.log(count);
console.log(animals);
    </script>
</head>
<body>
</body>
</html>
```

21. Definition for reduce: Perhaps the easiest-to-understand case for `reduce()` is to return the sum of all the elements in an array

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
 var array1 = [4, 5, 6, 8];
var initialValue = 0;
var sumWithInitial = array1.reduce(
  (previousValue, currentValue) => previousValue + currentValue,
  initialValue);
console.log(sumWithInitial);
</script>
</head>
<body>
</body>
</html>
```

22. Definition for reduce Right: method applies a function against an accumulator and each value of the array (from right-to-left) to reduce it to a single value.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
  var array1 = [[0, 1], [2, 3], [4, 5]];
var result = array1.reduceRight((accumulator, currentValue) =>
accumulator.concat(currentValue));
console.log(result);
    </script>
</head>
<body>
</body>
</html>
```

23. Definition for reverse: method reverses an array *in place*. The first array element becomes the last, and the last array element becomes the first.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
   var array1 = ['one', 'two', 'three'];
console.log('array1:', array1);
const reversed = array1.reverse();

    </script>
</head>
<body>
</body>
</html>

```

24. Definition for shift: method removes the **first** element from an array and returns that removed element. This method changes the length of the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
```

```
   var array1 = [111, 222, 333];
   var firstElement = array1.shift();
   console.log(array1);
   console.log(firstElement);
   </script>
</head>
<body>
</body>
</html>
```

25. Definition for slice: method returns a shallow copy of a portion of an array into a new array object selected from `start` to `end` (`end` not included) where `start` and `end` represent the index of items in that array. The original array will not be modified.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>
  var animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];
   console.log(animals.slice(-2));
   console.log(animals.slice(2, -1));
   </script>
</head>
<body>
</body>
</html>
```

26. Definition for some: method tests whether at least one element in the array passes the test implemented by the provided function. It returns true if, in the array, it finds an element for which the provided function returns true; otherwise it returns false. It doesn't modify the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>cancat</title>
    <script>
  var array = [1, 2, 3, 4, 5,8,13,22,46];
const even = (element) => element % 2 === 0;
console.log(array.some(even));
</script>
</head>
<body>
</body>
</html>
```

27. **Definition for sort:** The time and space complexity of the sort cannot be guaranteed as it depends on the implementation.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
       // var arr = [99,10,88,54,53,26,6,1,76,66,89,60,55,1,77,44,31];
       /*  var arr = new Array(20);

        console.log(arr.length);
        arr[30] = "hello there";
        console.log(arr); */
        var arr = [99,10,88,54,53,26,6,1,76,66,89,60,55,1,77,44,31,78,88];
         console.log( arr.join(' ~ ') );
        // console.log(arr.sort());
       function sortfun(val1, val2){
            // 1 ascending
            // -1 descending
            // 0 no sorting
            if(val1 > val2){
                return 1
            }else if(val1 < val2){
                return -1
            }else{
                return 0
            }
       };

       console.log(arr.sort(sortfun));
    </script>
</head>
<body>
```

```
</body>
</html>
```

28. Definition for splice: ethod changes the contents of an array by removing or replacing existing elements and/or adding new elements [in place](). To access part of an array without modifying it, see `slice()`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>

var months = ['Jan', 'March', 'April', 'June'];
months.splice(4, 1, 'May');
console.log(months);
</script>
</head>
<body>
</body>
</html>
```

29. Definition for locale string: The elements are converted to Strings using their `to Locale String` methods and these Strings are separated by a locale-specific String (such as a comma ",").

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>

  var array1 = [22, 't', new Date('22 Dec 1999 6:2:00 UTC')];
var localeString = array1.toLocaleString('en', { timeZone: 'UTC' });

console.log(localeString);
</script>
</head>
```

```
<body>
</body>
</html>
```

30. Definition for string: method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place. To access part of an array without modifying it, see `slice()`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>

var array1 = [1, 2, 'cr', 'ibm'];

console.log(array1.toString());
</script>
</head>
<body>
</body>
</html>
```

31. Definition for unshift: method adds one or more elements to the beginning of an array and returns the new length of the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>

  var array1 = [1, 2, 3];

console.log(array1.unshift(4, 5));

console.log(array1);
</script>
</head>
<body>
</body>
```

```
</html>
```

32. Definition for values: method returns a new *array iterator* object that contains the values for each index in the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cancat</title>
    <script>

  var array1 = ['fr', 'c', 'r'];
var iterator = array1.values();

for (var value of iterator) {
  console.log(value);
}
</script>
</head>
<body>
</body>
</html>
```