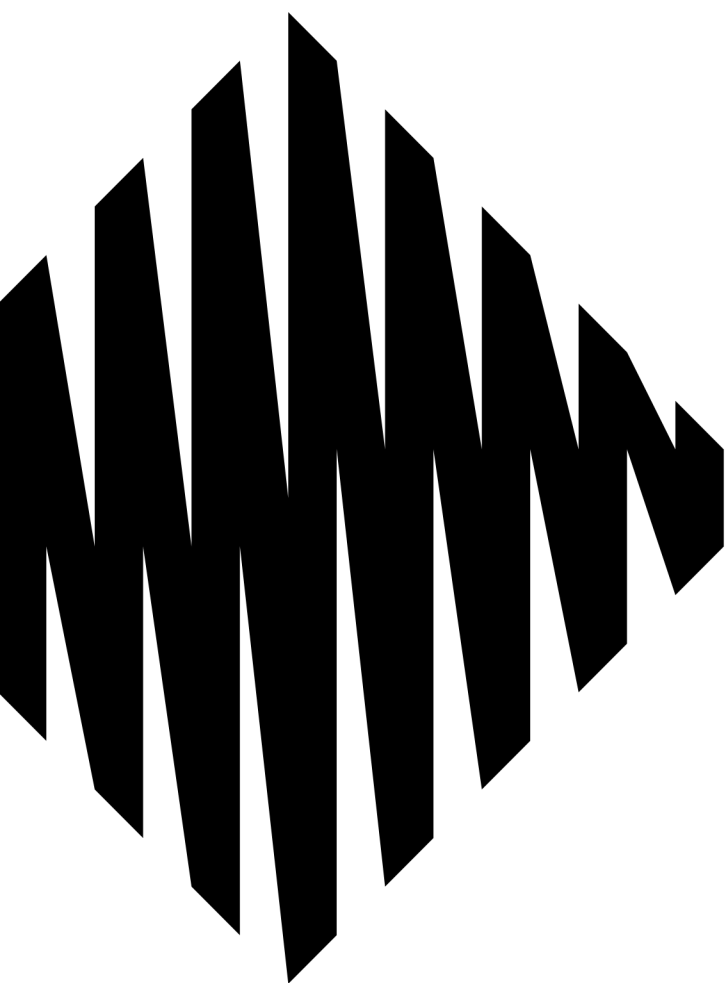


ecx.io Frontend Bootcamp

Array

15.07.2021.



Agenda

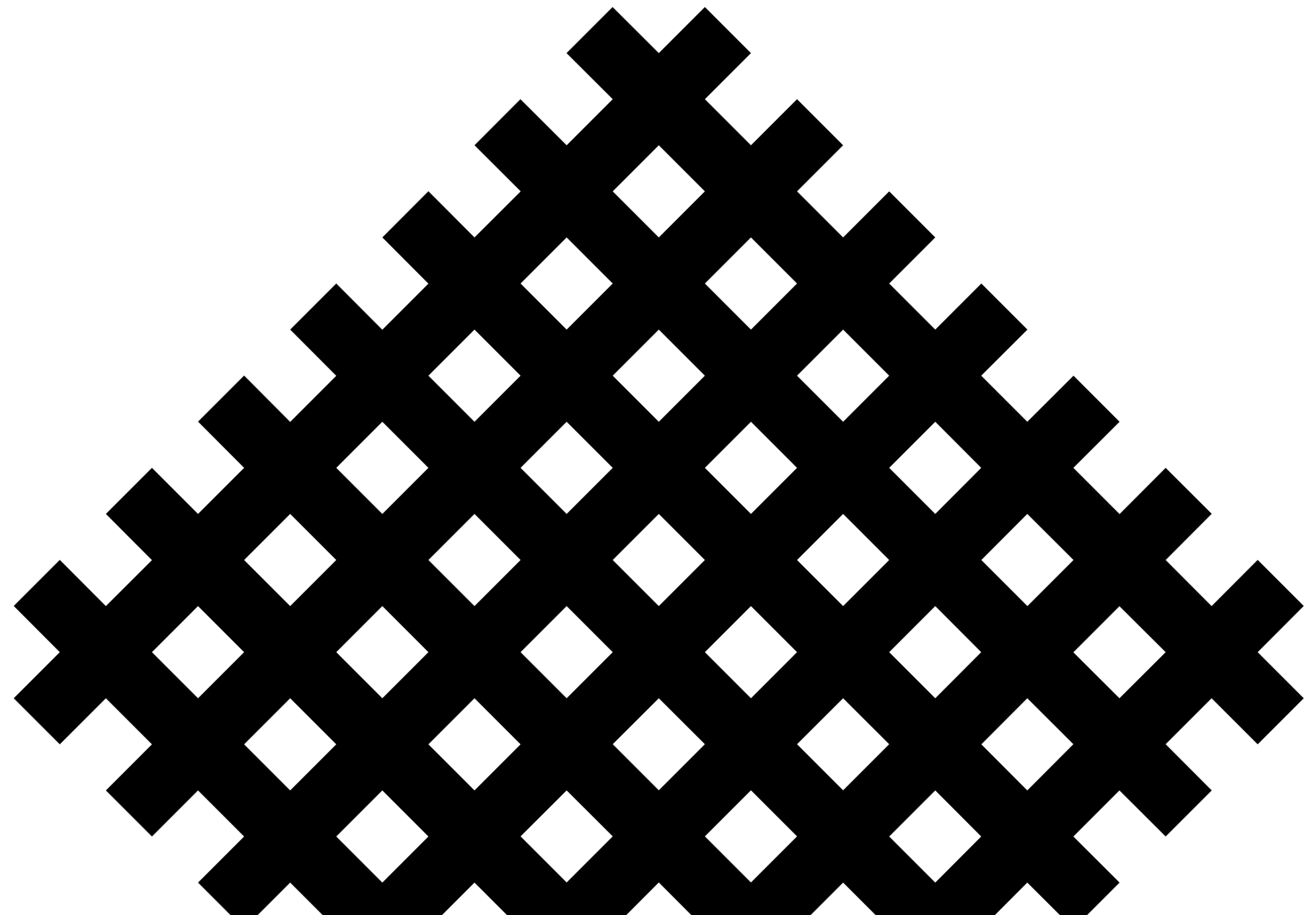
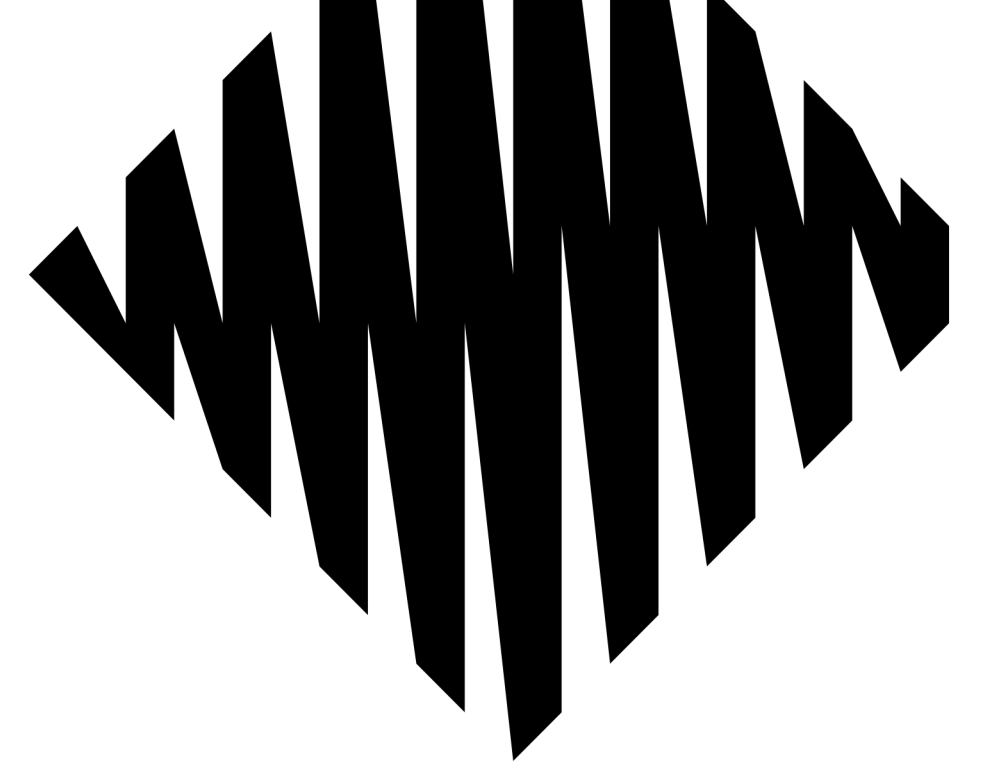
Array intro

Common operations

Array methods



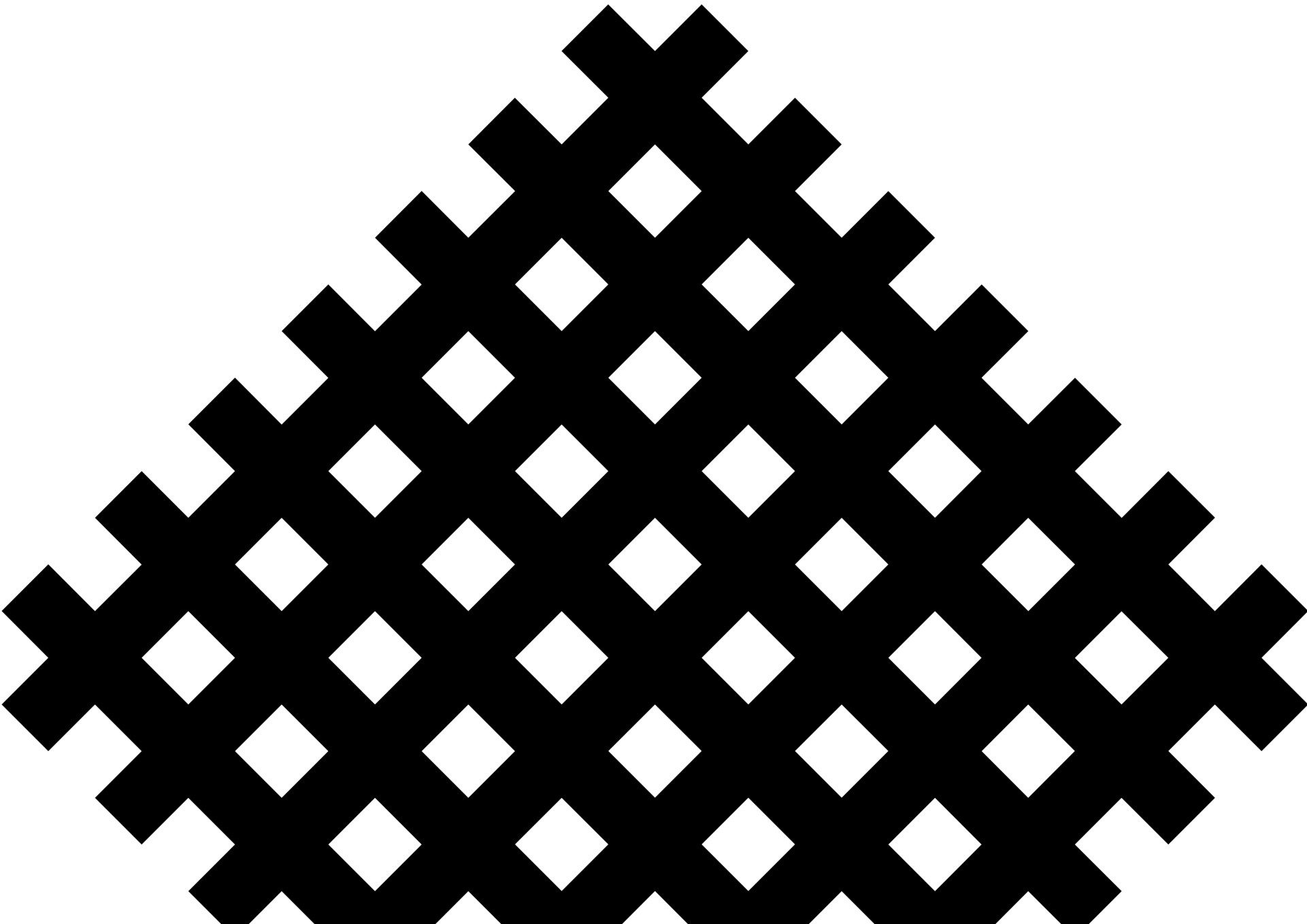
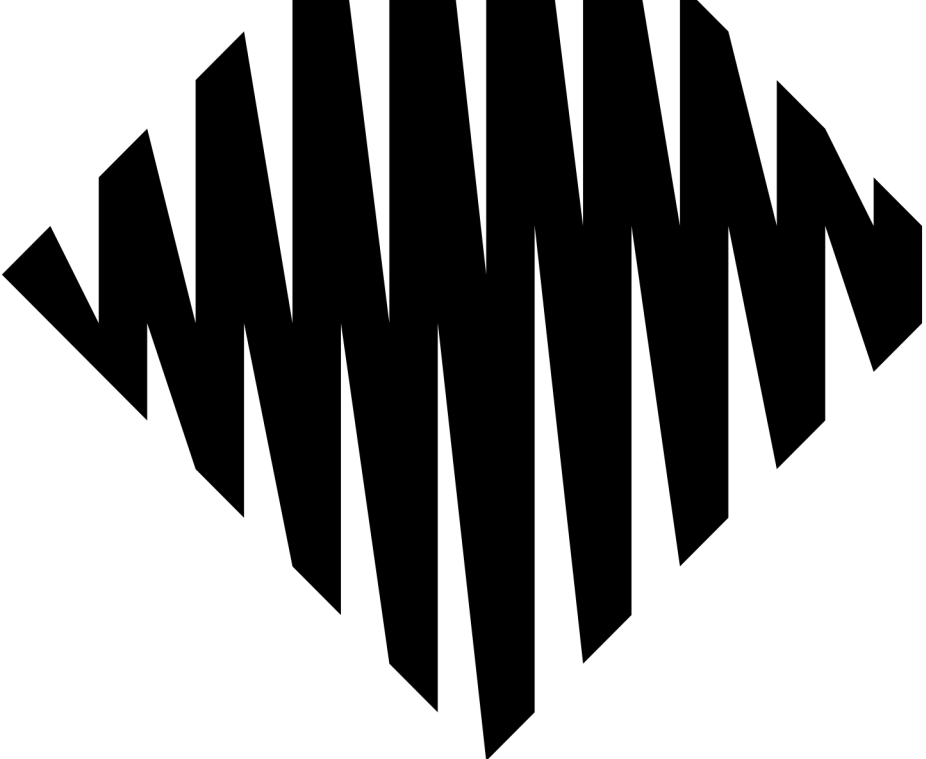
Array intro



Array

The JavaScript Array class is a global object that is used in the construction of arrays; which are high-level, list-like objects.

Common operations



Create an Array

```
1 let fruits = ['Apple', 'Banana']
2
3 console.log(fruits.length)
4 // 2
```

Access an Array item using the index position

```
1 let first = fruits[0]
2 // Apple
3
4 let last = fruits[fruits.length - 1]
5 // Banana
```

Add an item to the end of an Array

```
1 let newLength = fruits.push('Orange')
2 // ["Apple", "Banana", "Orange"]
```

Remove an item from the end of an Array

```
1 let last = fruits.pop() // remove Orange (from the end)
2 // ["Apple", "Banana"]
```

Remove an item from the beginning of an Array

```
1 let first = fruits.shift() // remove Apple from the front
2 // ["Banana"]
```

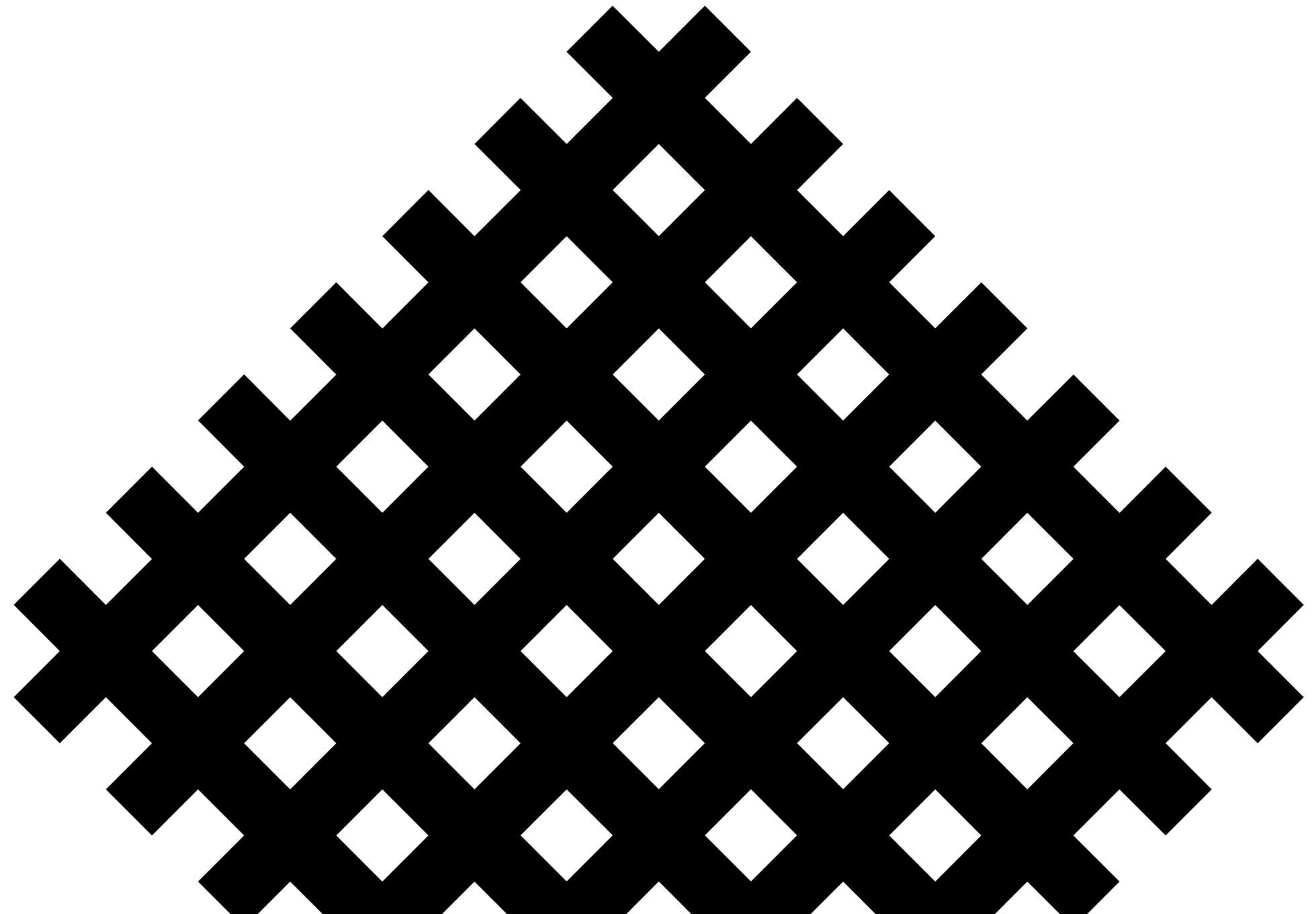
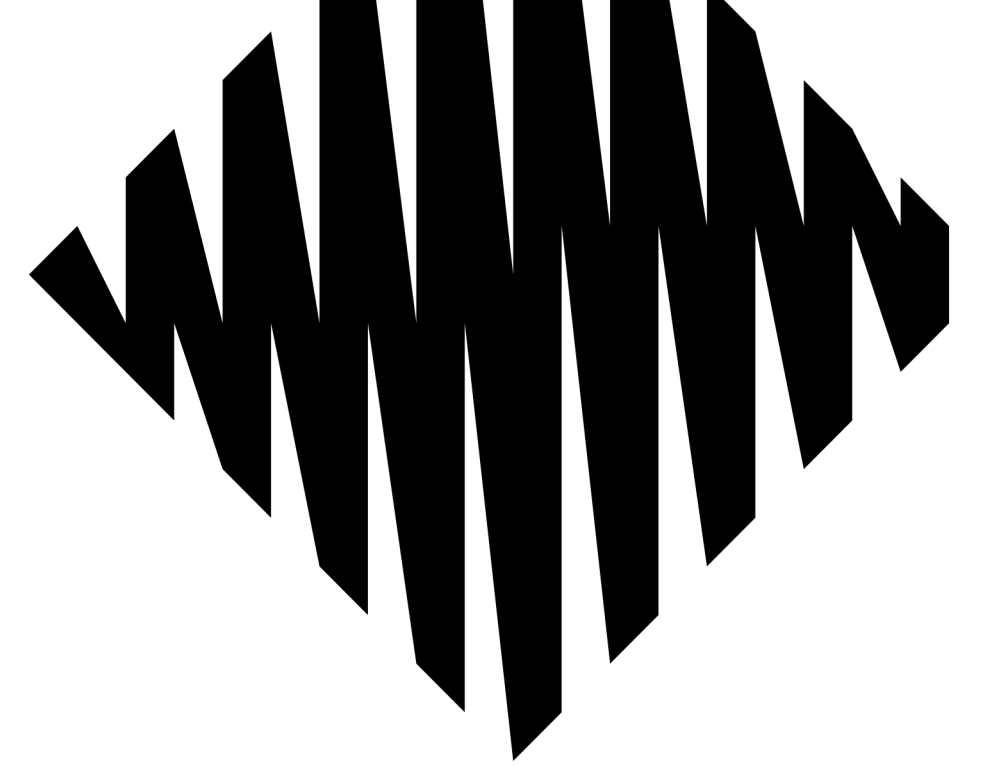
Add an item to the beginning of an Array

```
1 let newLength = fruits.unshift('Strawberry') // add to the front
2 // ["Strawberry", "Banana"]
3
```

Find the index of an item in the Array

```
1 fruits.push('Mango')
2 // ["Strawberry", "Banana", "Mango"]
3
4 let pos = fruits.indexOf('Banana')
5 // 1
```

Array Methods



concat()

Creates a new array based on the elements in the current array. It's important to know that the original array is not altered. The operation is made on the copy of that array.

concat() accepts any number of arguments which will be added at the end of the array. Using it without arguments is a elegant way to clone the initial array.

Return value

The concat() method returns a new array with the elements of the calling array followed by the concatenation of each of the parameters, in their respective order.



```
1 const arr = [true, 2, 3, '4'];  
2 const arr2 = ['con', 'cat'];  
3 const new_array = arr.concat(5, 6, 7, arr2);  
4  
5 console.log(new_array);  
6 // [true, 2, 3, "4", 5, 6, 7, "con", "cat"]
```


Exercise

Wrote an example to get following result from `concat()`. Use at least two arrays.

every()

The `every()` method tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

Return value

true if the callback function returns a truthy value for every array element. Otherwise, false.



```
1 const isBelowThreshold = (currentValue) => currentValue < 40;
2 const array1 = [1, 30, 39, 29, 10, 13];
3
4 console.log(array1.every(isBelowThreshold));
5 // expected output: true
6
7
8 [12, 5, 8, 130, 44].every(x => x >= 10); // false
```

Exercise

Provide a function to test if all elements are even and greater than 20.

filter()

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

NOTE: Avoid chaining filter() methods to have fewer loops over array.

Return value

A new array with the elements that pass the test. If no elements pass the test, an empty array will be returned.



```
1 const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];  
2 const odd_numbers = numbers.filter(item => item % 2 == 1);  
3 console.log(odd_numbers);  
4 // [1, 3, 5, 7, 9]
```

Exercise

1. Find all prime numbers in an array.

2. Filter array items based on search criteria

find()

The find array method returns the value of the first element in the array it is invoked on that matches the expression in the callback function. If there is no matcher to the expression in the callback, the find method will return undefined.

Return value

The value of the first element in the array that satisfies the provided testing function. Otherwise, undefined is returned.

```
1 function condition (human) {
2     return human.age >= 18;
3 }
4
5 const arr = [{
6     name: 'Silvio',
7     age: 19
8 }, {
9     name: 'Zvonimir',
10    age: 17
11 }, {
12    name: 'Mario',
13    age: 20
14 }];
15
16 console.log(arr.find(condition));
17 // Should return { name: "Silvio", age: 19 }
```

Exercise

Find and object in an array by its name.

Note: Use arrow function and destructuring.

findIndex()

The `findIndex()` 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.

Return value

The index of the first element in the array that passes the test. Otherwise, -1.



```
1 const array1 = [5, 12, 8, 130, 44];  
2  
3 const isLargeNumber = (element) => element > 13;  
4  
5 console.log(array1.findIndex(isLargeNumber));  
6 // expected output: 3
```


Exercise

Find index of a fruit "blueberries".

forEach()

The `forEach()` method executes a provided function once for each array element.

Return value

undefined




```
1 const array1 = ['apple', 'banana', 'grapes'];  
2  
3 array1.forEach(element => console.log(element));  
4  
5 // expected output: "apple"  
6 // expected output: "banana"  
7 // expected output: "grapes"
```

forEach()

forEach() does not mutate the array on which it is called, but you could modify an array during forEach() iteration. If elements that are already visited are removed (e.g. using shift()) during the iteration, later elements will be skipped.

The range of the visited items is set before the callback function's first call. Once it starts iterating over an array, any new items you add to the array won't be visited by the callback.



```
1 let words = ['one', 'two', 'three', 'four']
2 words.forEach(function(word) {
3   console.log(word)
4   if (word === 'two') {
5     words.shift() // 'one' will delete from array
6   }
7 }) // one // two // four
8
9 console.log(words); // ['two', 'three', 'four']
```

Exercise

For each string in the array create new string containing current element value and index. Add them to the page.

map()

The `map()` method creates a new array populated with the results of calling a provided function on every element in the calling array.

Return value

A new array with each element being the result of the callback function.



```
1 const array1 = [1, 2, 3, 4];  
2  
3 // pass a function to map  
4 const map1 = array1.map(x => x * 2);  
5  
6 console.log(map1);  
7 // expected output: Array [2, 4, 6, 8]
```

Exercise


Map an array of numbers to an array of power 3 exponents.

reduce()

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

Return value

The single value that results from the reduction.



```
1  const array1 = [1, 2, 3, 4];
2  const reducer = (accumulator, currentValue) => accumulator + currentValue;
3
4  // 1 + 2 + 3 + 4
5  console.log(array1.reduce(reducer));
6  // expected output: 10
7
8  // 5 + 1 + 2 + 3 + 4
9  console.log(array1.reduce(reducer, 5));
10 // expected output: 15
```

Exercise

1. Sum all even numbers.

2. Bond arrays contained in an array of objects using the spread operator and initial value.

slice()

The 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.

Return value

A new array containing the extracted elements.

```
1  const fruits = ['apple', 'banana', 'cherry', 'mango', 'orange'];
2
3  console.log(fruits.slice(2));
4  // expected output: ["cherry", "mango", "orange"]
5
6  console.log(fruits.slice(2, 4));
7  // expected output: ["cherry", "mango"]
8
9  console.log(fruits.slice(1, 5));
10 // expected output: ["banana", "cherry", "mango", "orange"]
```

Exercise

Extract the last two elements in the sequence.

sort()

The `sort()` method sorts the elements of an array in place and returns the sorted array. The default sort order is ascending, built upon converting the elements into strings, then comparing their sequences of UTF-16 code units values.

Return value

The sorted array. Note that the array is sorted in place, and no copy is made.



```
1 const months = ['Mar', 'Jan', 'Feb', 'Dec', 'Oct', 'Sep'];
2 months.sort();
3 console.log(months);
4 // expected output: ["Dec", "Feb", "Jan", "Mar", "Oct", "Sep"]
5
6 const numbers = [4, 20, 5, 10, 3];
7 numbers.sort(function(a, b) {
8     return a - b;
9 });
10 console.log(numbers);
11 // expected output [3, 4, 5, 10, 20]
```

Exercise

1. Sort array of objects by name, ascending.
2. Sort array of objects by postalCode, descending
3. Sort array of objects by type and name, all ascending

splice()

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

Return value

An array containing the deleted elements.

If only one element is removed, an array of one element is returned.

If no elements are removed, an empty array is returned.



```
1 const months = ['Jan', 'March', 'April', 'June'];
2 months.splice(1, 0, 'Feb');
3 // inserts at index 1
4 console.log(months);
5 // expected output: ["Jan", "Feb", "March", "April", "June"]
6
7 months.splice(4, 1, 'May');
8 // replaces 1 element at index 4
9 console.log(months);
10 // expected output: ["Jan", "Feb", "March", "April", "May"]
```

Exercise

1. Remove zero elements from index 1 and insert "banana" and "orange".

2. Remove 2 elements from index 0 and insert "avocado".

Reference links

- Array
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Thank you

© Copyright IBM Corporation 2019.

All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available at [Copyright and trademark information](#).