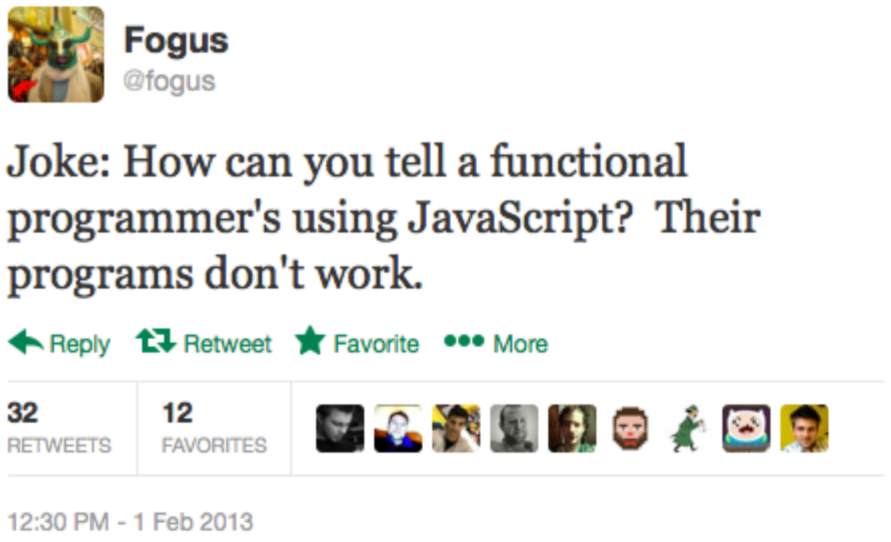
Searching and Sorting **without** Loops

@**ariya**hidayat

Jan 28, 2014

SFJS SAN FRANCISCO JAVASCRIPT MEETUP



Fogus
@fogus

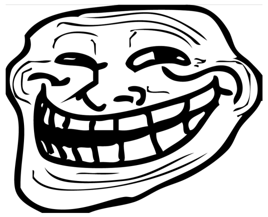
Joke: How can you tell a functional programmer's using JavaScript? Their programs don't work.

12:30 PM - 1 Feb 2013

https://twitter.com/fogus/status/297441838745395201

/usr/bin/whoami

shapesecurity.com



“Software Provocateur”

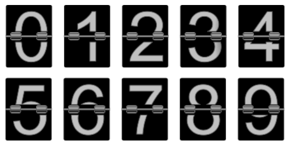
PhantomJS

ES Esprima

Look ma, no loops!

Array methods:

map, filter, reduce, some, every



Sequences:
prime numbers, factorials, Fibonacci series

Searching: every, some, reduce

Sorting algorithm implementation

Caveat Emptor

- Just because you **can** do it,
  doesn’t mean you **should** do it
- Be advised of any **performance** implication
- Don’t optimize prematurely,
  judge wisely between **readability** and **speed**

Array Methods

	map **filter**	**reduce**	**every** **some**
Return value	a new array	depends	Boolean
Visit every element?	**Yes**	**Yes**	No

Array.prototype.map

Section 15.4.4.19

```
[ ... ].map(callbackfn)
```

map calls *callbackfn* **once for each** element in the array, in ascending order, and constructs a new Array from the results.

callbackfn is called with **three** arguments:

- the value of the element
- the index of the element, and
- the object being traversed.

Examples of Array.prototype.map

x = element

```
[1, 2, 3].map(function (x) {
  return x * x;
});
```

```
[1, 4, 9]
```

y = index

```
[7, 7, 7].map(function (x, y) {
  return y;
});
```

```
[0, 1, 2]
```

With Arrow Function

ECMAScript 6

```
[1, 2, 3].map((x) => x * x);            [1, 4, 9]

[7, 7, 7].map((x, y) => y);             [0, 1, 2]
```

http://ariya.ofilabs.com/2013/02/es6-and-arrow-function.html

Array.prototype.filter

Section 15.4.4.20

```
[ ... ].filter(callbackfn)
```

filter calls *callbackfn* **once for each** element in the array, in ascending order, and constructs a new array of all the values **for which** *callbackfn* returns true.

callbackfn is called with **three** arguments:

- the value of the element
- the index of the element, and
- the object being traversed.

Examples of Array.prototype.filter

```
[-2, -1, 0, 1, 2].filter(function (x) {
  return x >= 0;
});
```

```
[0, 1, 2]
```

```
[2, 3, 4, 5].filter(function (x) {
  return x & 1;
});
```

```
[3, 5]
```

Array.prototype.reduce

Section 15.4.4.21

```
[ ... ].reduce(callbackfn, initial)
```

callbackfn is called with four arguments:

- the *previousValue* (or value from the previous call to callbackfn),
- the *currentValue* (value of the current element)
- the *currentIndex*, and
- the *object* being traversed.

Examples of Array.prototype.reduce

```
[1, 2, 3, 4, 5].reduce(function (sum, i) {
  return sum + i;
});
```

```
15
```

```
[1, 2, 3, 4, 5].reduce(function (sum, i) {
  return sum + i;
}, 100);
```

```
115
```

```
[1, 2, 3].reduce(function(result, x) {
  return result.concat(x + 2);
}, []);
```

```
[3, 4, 5]
```

Array.prototype.every

Section 15.4.4.16

```
[ ... ].every(callbackfn)
```

every calls *callbackfn* once for each element present in the array, in ascending order, **until** it finds one where *callbackfn* returns **false**.

If such an element is found, *every* **immediately** returns false. Otherwise, if *callbackfn* returned true for all elements, *every* will return true.

Examples of Array.prototype.every

```
[24, 17, 45].every(function (age) {
  return age >= 18;
});
```

`false`

```
[7, 8, 9].every(function (x) {
  return x > 5;
});
```

`true`

Array.prototype.some

Section 15.4.4.17

```
[ ... ].some(callbackfn)
```

some calls *callbackfn* once for each element present in the array, in ascending order, **until** it finds one where *callbackfn* returns **true**.

If such an element is found, *some* **immediately** returns true. Otherwise, *some* returns false.

Examples of Array.prototype.some

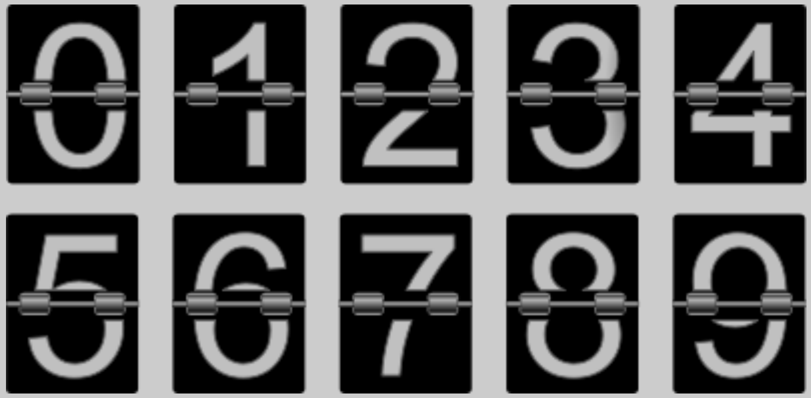
```
[3.14159, 3.2, 3.14].some(function(x) {
  return x.toFixed(2) == '3.14';
});
```

`true`

```
[60, 62, 65].some(function (fps) {
  return fps < 60;
});
```

`false`

Creating Sequences



Numbers

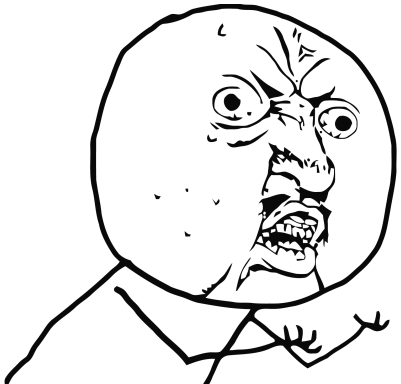
```
var result = [];
for (var i = 1; i < 4; ++i) result.push(i)
console.log(result);  // [1, 2, 3]
```

Characters

```
var list = '';
for (var i = 0; i < 26; ++i)
  list += String.fromCharCode(i + 65);

console.log(list); // 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

Creating an Array

```
var x = Array(3);

x.length;              3


console.log(x);      []
```



The array is “*empty*”

Operator in

Section 11.8.7

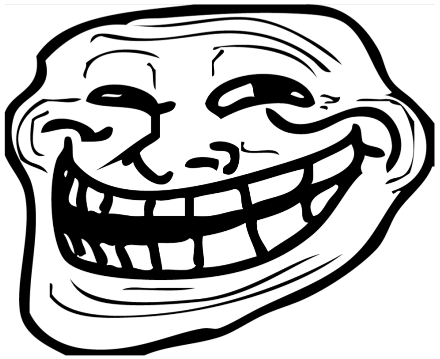
```
0 in Array(3);    // false
1 in Array(3);    // false
2 in Array(3);    // false

2 in [,,9];       // true
```

toString relies on **join** (Section 15.4.4.5)
join convertes *undefined* or *null* to an empty string

Fill the Array

```
var x = Array.apply(0, Array(3));

console.log(x);
```

```
[undefined, undefined, undefined]
```



The array is filled with ***undefined***

Function.prototype.apply

Section 15.3.4.3

```
Math.max.apply(Math, [14, 3, 77]);
```

Array

```
Math.max(14, 3, 77);
```

Parameters

http://www.2ality.com/2011/08/spreading.html

Demystifying Array.apply

```
Array.apply(0, Array(3));

Array.apply(0, [,,]);

Array(undefined, undefined, undefined);
```

“ghost elements” got converted into ***undefined***

Series of Numbers

```
Array.apply(0, Array(3))
```

```
[undefined, undefined, undefined]
```

```
Array.apply(0, Array(3)).map(function (x, y) {
  return y + 1;
});
```

```
[1, 2, 3]
```

```
Array.apply(0, Array(3)).map(function (x, y) {
  return (y + 1) * (y + 1);
});
```

```
[1, 4, 9]
```

Strings

```
Array.apply(0, Array(26)).map(function(x,y) {
  return String.fromCharCode(y + 65);
}).join('');
```

Array Comprehension

ECMAScript 6

for .. of

```
[for (i of Array.apply(0, Array(26)).map((x, y) => y))
String.fromCharCode(65 + i)].join('');
```

Arrow function

http://ariya.ofilabs.com/2013/01/es6-and-array-comprehension.html

More Info

“Sequences using JavaScript Array”

http://ariya.ofilabs.com/2013/07/sequences-using-javascript-array.html

Prime Number or Not?

```
function isPrime(i) {
  for (var c = 2; c <= Math.sqrt(i); ++c)
    if (i % c === 0) return false;
  return true;
}
```

Can we divide ***i*** by ***c***?

http://en.wikipedia.org/wiki/Primality_test

23 vs 27

isPrime(23)

Math.sqrt(23) = 4.79583

23 % 2 = 1

23 % 3 = 2

23 % 4 = 3

true

isPrime(27)

Math.sqrt(27) = 5.1961

27 % 2 = 1

27 % 3 = 0 ←

27 % 4 = 3

27 % 5 = 2

false

List of Prime Numbers

```
function primeList(N) {
  var list = [];
   for (var i = 2; i < N; ++i)
     if (isPrime(i)) list.push(i);
  return list;
}
```

I DONT' ALWAYS SEARCH FOR PRIME NUMBERS
BUT WHEN I DO, I DON'T USE FOR LOOPS
memegenerator.net

Because loops are overrated!

Scan using Array.prototype.every

```
function isPrime(i) {
  return (i > 1) &&
    Array.apply(0, Array(1 + ~~Math.sqrt(i))).
    every(function (x, y) {
      return (y < 2) || (i % y !== 0);
    });
}
```

~~ is Math.floor

Can we divide ***i*** by ***y***?

Sequence + Filter

0..*N*-1

```
function primeList(N) {
  return Array.apply(0, Array(N)).map(function (x, y) { return y }).
    filter(function (i) {
      return (i > 1) && Array.apply(0, Array(1 + ~~Math.sqrt(i))).
        every(function (x, y) { return (y < 2) || (i % y !== 0) });
    });
}
```

Primality test

Comprehension Flavor

ECMAScript 6

```
function primeList(N) {
  return [for (i of Array.apply(0, Array(N)).map((x, y) => y))
    if ((i > 1) && Array.apply(0, Array(1 + ~~Math.sqrt(i))).
      every((x, y) => (y < 2) || (i % y !== 0) ))
    i];
}
```

http://ariya.ofilabs.com/2013/01/es6-and-array-comprehension.html

Factorial

```
function factorial(n) {
  var result = 1;
  for (var i = 1; i <= n; ++i) result *= i;
  return result;
}
```

```
factorial(5)                            120
```

```
1 * 2 * 3 * 4 * 5
```

With Array.prototype.reduce

```
function factorial(n) {
  return Array.apply(0, Array(n))
  .reduce(function(x, y, z) {
    return x + x * z;
  }, 1);
}
```

0..N-1

Accumulate

Factorial of 5

x + x * z

x	z
1	
1	0
2	1
6	2
24	3
120	4

Leonardo Fibonacci

“..the growth of an idealized (biologically unrealistic) rabbit population..”

http://en.wikipedia.org/wiki/Fibonacci_number

Fibonacci Series

0, 1, 1, 2, 3, $\underline{5}$, $\underline{8}$, 13, 21,...

5 + 8 = 13

The First *N* Fibonacci Numbers

```
function fibo(n) {
  var f = [];
  for (var c = 0; c < n; ++c) {
    f.push((c < 2) ? c : f[c-1] + f[c-2]);
  }
  return f;
}
```

Two previous numbers

```
fibo(5)
```

```
[ 0, 1, 1, 2, 3 ]
```

Rabbits and Reduce

```
function fibo(n) {
  return Array.apply(0, Array(n)).reduce(function(x, y, z){
    return x.concat((z < 2) ? z : x[z-1] + x[z-2]);
  }, []);
}
```

Rabbit Population

```
x.concat((z < 2) ? z : x[z-1] + x[z-2])
```

x	z
[ ]	
[0]	**0**
[0, **1**]	**1**
[$\underline{0}$, $\underline{1}$, **1**]	2
[0, $\underline{1}$, $\underline{1}$, **2**]	3
[0, 1, $\underline{1}$, $\underline{2}$, **3**]	4

More Info

“Prime Numbers, Factorial, and Fibonacci Series with JavaScript Array”

http://ariya.ofilabs.com/2013/07/prime-numbers-factorial-and-fibonacci-series-with-javascript-array.html

Searching

Locate an Employee

```
function findEmployee(id) {
  for (var i in employees)
    if (employees[i].id === id)
      return employees[i];
}
```

Locate an Employee v2

```
function findEmployee(id) {
    var employee;
    employees.forEach(function (e) {
        if (e.id === id) employee = e;
    });
    return employee;
}
```

Always check every employee

With Array.prototype.some

```
function findEmployee(id) {
    var employee;
    employees.some(function (e) {
        if (e.id === id) {
            employee = e;
            return true;
        }
    });
    return employee;
}
```

More Info

“Searching with Array.prototype.some”

http://ariya.ofilabs.com/2013/08/searching-with-array-prototype-some.html

Find the Longest String

```
function findLongest(array) {
  for (var i = 0, longest = ''; i < array.length; ++i)
    if (array[i].length > longest.length)
      longest = array[i];
  return longest;
}
```

```
    findLongest('ab', 'abc', 'a')                'abc'
```

With Array.prototype.reduce

```
function findLongest(array) {
  return array.reduce(function (longest, entry) {
    return entry.length > longest.length ? entry : longest;
  }, '' });
}
```

```
findLongest('ab', 'abc', 'a')          'abc'
```

Step-by-step of reduce

entry	entry.length	longest	longest.length
		`''`	0
`'ab'`	2	`'ab'`	2
`'abc'`	3	`'abc'`	3
`'a'`	1	`'abc'`	3

Also Get the Index

```
function findLongest(array) {
  return array.reduce(function (longest, entry, index) {
    return entry.length > longest.value.length ?
      { index: index, value: entry } : longest;
  }, { index: -1, value: '' });
}
```

```
findLongest('ab', 'abc', 'a')     { index: 1, value: 'abc' }
```

Step-by-step of reduce

entry	longest.index	longest.value
	-1	`''`
`'ab'`	0	`'ab'`
`'abc'`	1	`'abc'`
`'a'`	1	`'abc'`

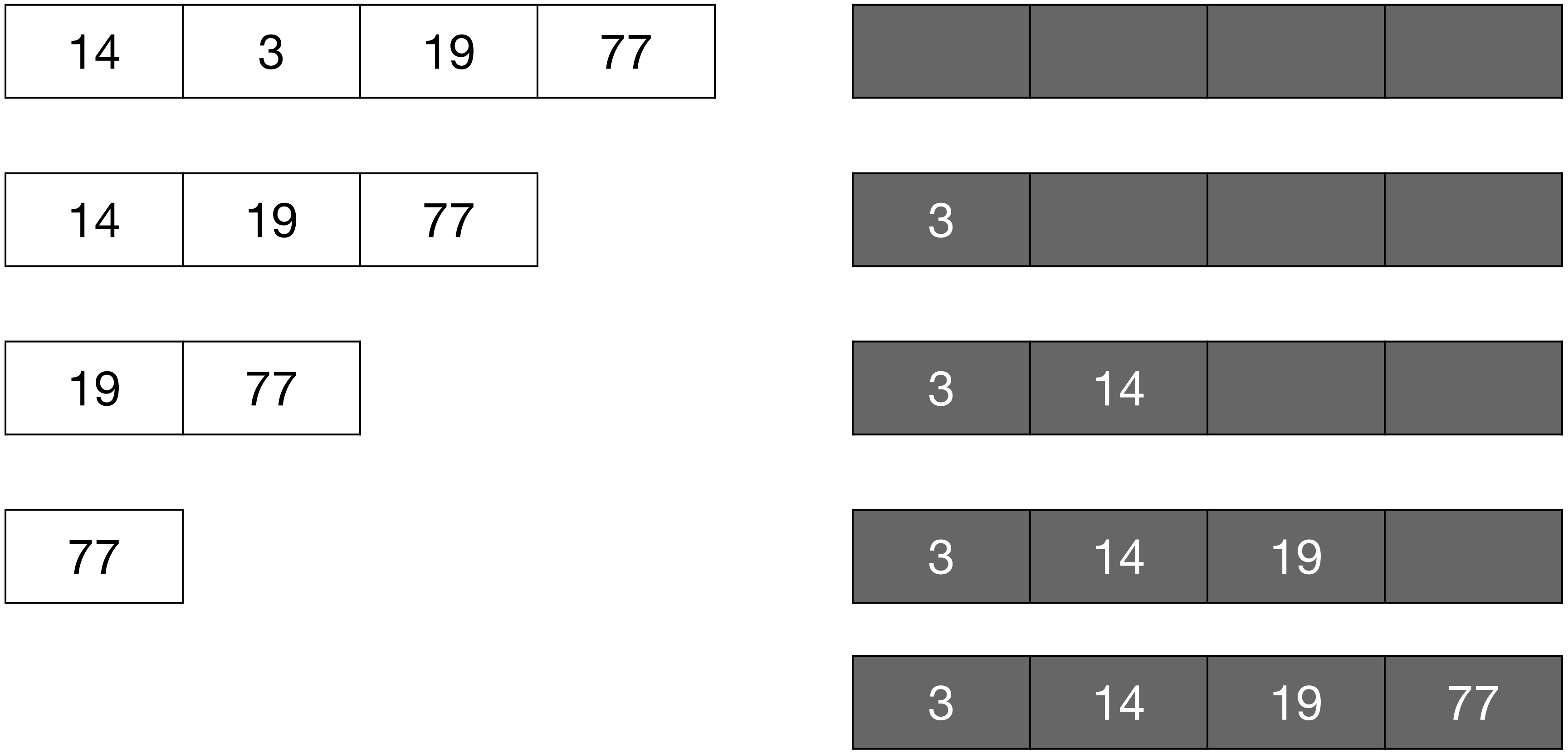
More Info

“Searching using Array.prototype.reduce”

http://ariya.ofilabs.com/2013/10/searching-using-array-prototype-reduce.html

Sorting

Step-by-Step Sorting



N-element Array = N steps

0..*N*-1

```
Array.apply(0, Array(array.length)).map(function () {
  // Do something
});
```

Inner loop

Search for the Smallest

```
function findSmallest(array) {
  return array.reduce(function (min, entry, index) {
    return min.value < entry ?
      { index: index, value: entry } : min;
  }, { value: null });
}
```

```
findSmallest([14, 3, 19, 77])    { index: 1, value: 3 }
```

Repeat the Search

```
function sort(input) {
  var array = input.slice(0);
  return Array.apply(0, Array(array.length)).map(function () {
    return array.splice(findSmallest(array).index, 1).pop();
  });
}
```

Before splice

```
[14, 3, 19, 77]
```

```
{ index: 1, value: 3 }
```

After splice

```
[14, 19, 77]
```

Complete Code for Sorting

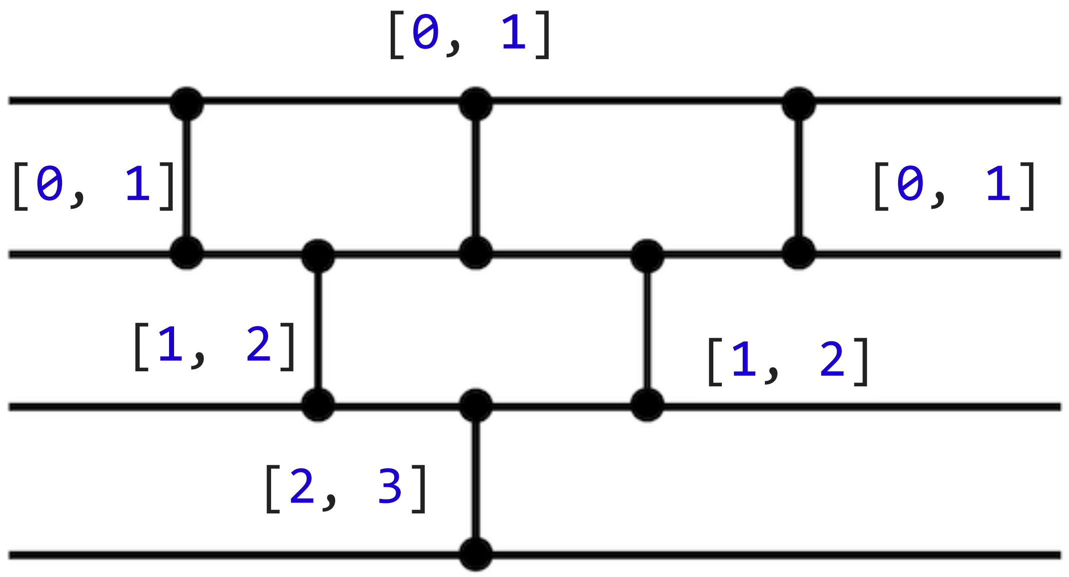
```
function sort(input) {
  var array = input.slice(0);
  return Array.apply(0, Array(array.length)).map(function () {
    return array.splice(array.reduce(function (min, entry, index) {
      return min.value < entry ? min : index: index, value: entry
};
    }).index, 1).pop();
  });
}
```

More Info

"Searching using Array.prototype.reduce"

http://ariya.ofilabs.com/2013/10/searching-using-array-prototype-reduce.html

Sorting Network



4-element Array Sorting

```
function compareswap(array, p, q) {
  if (array[p] < array[q]) {
    var temp = array[q];
    array[q] = array[p];
    array[p] = temp;
  }
}
```

“Comparator”

Sorting sequences

```
compareswap(entries, 0, 1);
compareswap(entries, 1, 2);
compareswap(entries, 2, 3);
compareswap(entries, 0, 1);
compareswap(entries, 1, 2);
compareswap(entries, 0, 1);
```

3-element Array Sorting

```
function compareswap(array, p, q) {
  if (array[p] < array[q]) {
    var temp = array[q];
    array[q] = array[p];
    array[p] = temp;
  }
}
```

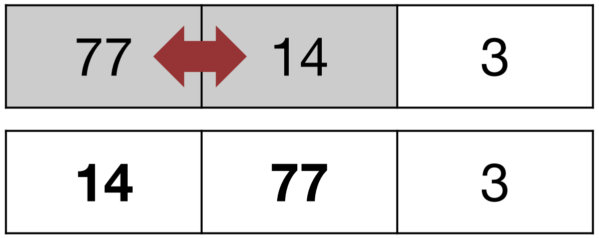
“Comparator”

Sorting sequences

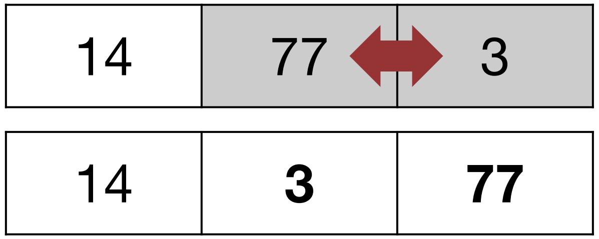
```
compareswap(entries, 0, 1);
compareswap(entries, 1, 2);
compareswap(entries, 0, 1);
```

Step-by-Step Sorting

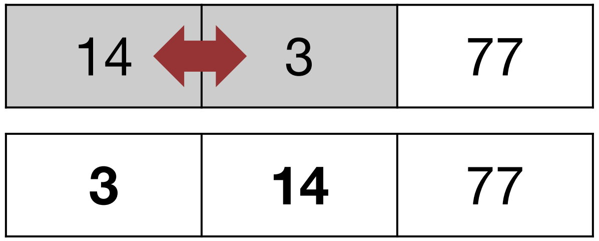
```
compareswap(entries, 0, 1);
```



```
compareswap(entries, 1, 2);
```



```
compareswap(entries, 0, 1);
```



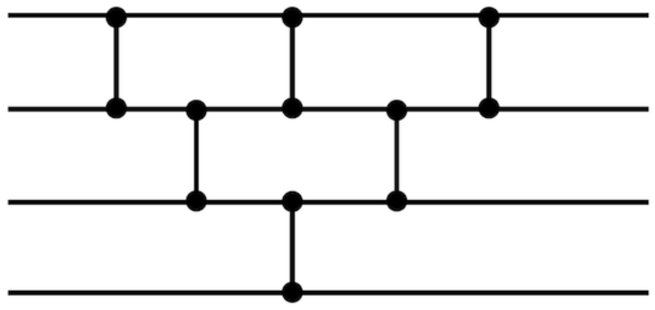
Generalized Form

```
function sort(network, entries) {
  for (var i = 0; i < network.length; ++i)
    compareswap(entries, network[i], network[i] + 1)
}
```

Build the Network (for N)

```
function createNetwork(N) {
  return Array.apply(0, Array(N)).reduce(function (network, _, y) {
    return network.concat(Array.apply(0, Array(N - y - 1))
      .map(function(_, x) {
        return x;
      }));
  }, []);
}
```

```
[0, 1, 2, 0, 1, 0]
```



More Info

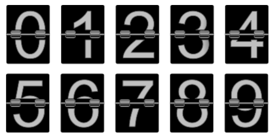
“Sorting Networks using Higher-Order Functions of JavaScript Array”

http://ariya.ofilabs.com/2013/10/sorting-networks-using-higher-order-functions-of-javascript-array.html

Summary

Array methods:

map, filter, reduce, some, every



Sequences:
prime numbers, factorials, Fibonacci series

Searching: every, some, reduce

Sorting algorithm implementation

Final Words

Higher-order functions are **cool**

Thank You

@ariyahidayat

ariya.ofilabs.com

speakerdeck.com/ariya

Some artworks are from http://openclipart.org.