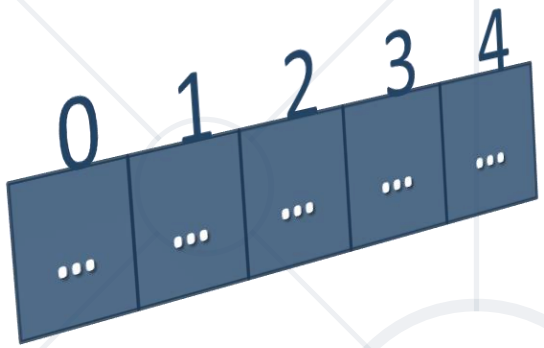
Arrays Advanced

Additional Array Operations



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Questions?

Software University

sli.do

#fund-js

Table of Contents

Basic Operations

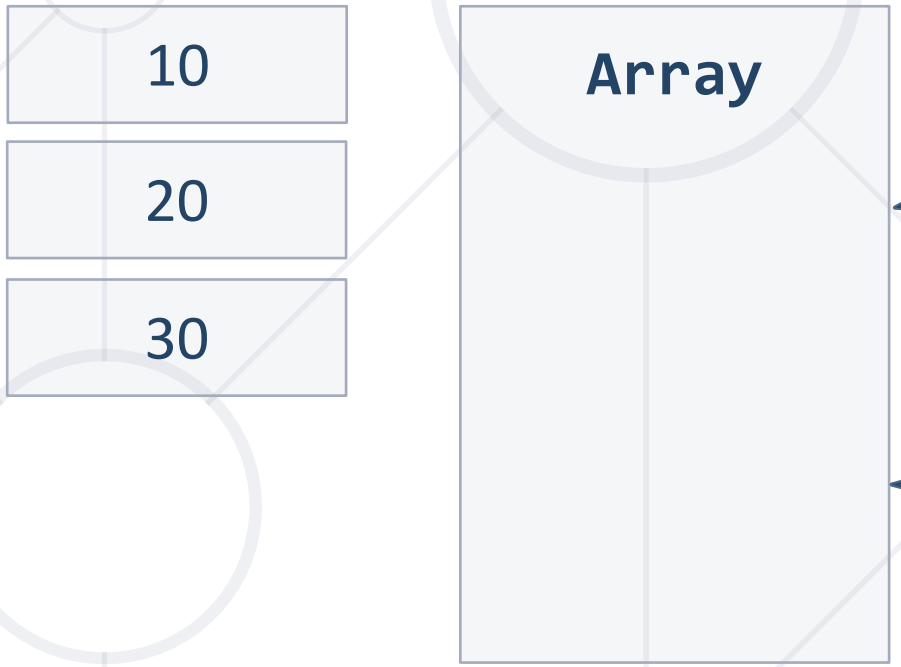
Add, Remove and Find Elements

Advanced Overview

- Advanced functionality of the array consists of the following **functions**:
  - `push()` – add to the end
  - `pop()` – remove from the end
  - `unshift()` – add to the beginning
  - `shift()` – remove from the beginning
  - `includes()` – look for value
  - `indexOf()` – find index of value

Add at the End, Remove from the End

- `push()` adds at the end of the `array`
- `pop()` removes from the end of the `array`



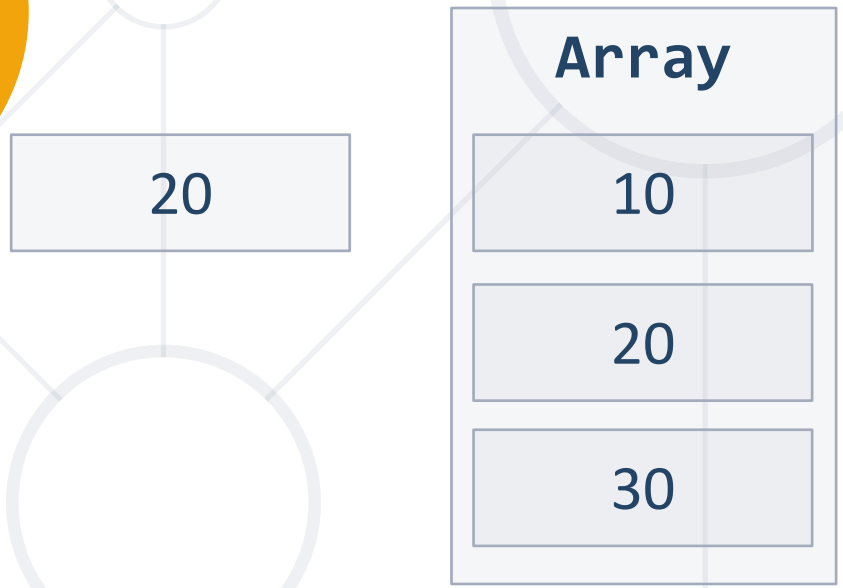
Use `push()` to add at the end.

Use `pop()` to remove from the end.

Add at the Start, Remove from the Start

Software University

- `unshift()` adds at the start of the `array`
- `shift()` removes from the start of the `array`



Use `shift()` to remove from the start.

Use `unshift(20)` to add at the start.

Pop() – Removes the Last Element

- The **pop** method removes the **last** element from an array and **returns** that value to the caller
- If you call **pop()** on an empty array, it returns **undefined**

```
let myArray = ["one","two","three","four","five"];
let popped = myArray.pop();
console.log(myArray); //["one","two","three","four"]
console.log(popped); //"five"
```

Problem: Sum First Last

- Calculate and print the sum of the **first** and the **last** elements in an array
- The input comes as an **array of string** elements holding numbers
- The output is printed on the console

```
['5' , '10']  ➡  15

['20', '30','40']  ➡  60
```

```
function solve(input) {
  input = input.map(Number);
  console.log(input[0]
                + input.pop());
}
```

Pushing into Array

- The **push** method adds **one** or **more** elements to the end of an array and returns the new length of the array

```
let fruits = ["apple","banana","kiwi"];
fruits.push("pineapple");
console.log(fruits);
 // ["apple","banana","kiwi","pineapple"]
```

Element is added at the end

Shifting and Unshifting

- `shift()` - Removes the first element of an array

```
let myArray = ["one","two","three","four","five"];
myArray.shift(); // ["two","three","four","five"]
```

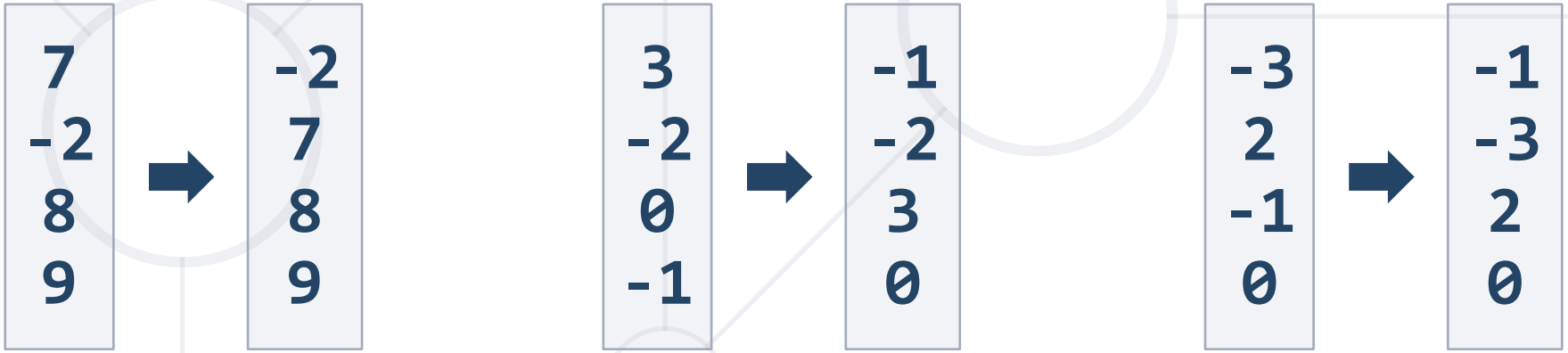
- `unshift()` - Adds elements to the beginning

```
let myArray = ["red","green","blue"];
myArray.unshift("purple");
// ["purple","red","green","blue"]
```

New element added

Problem: Negative / Positive Numbers

- You are given an array of string elements holding numbers
  - Process them one by one and create a new array -> `result`
    - Prepend each negative element at the front of the array
    - Append each positive (or 0) element at the end of the array
    - Print the `result` array, each element at a separate line



Solution: Negative / Positive Numbers


```
function negativePositiveNumbers(arr) {
  let result = [];
  for (let num of arr){
    if (num < 0){
      result.unshift(num); // Insert at the start
    } else {
      result.push(num);    // Append at the end
  }
}
 console.log(result.join('\n'));
}
```

Find Values

- `includes()` – returns `true` if the given value is part of the array

```
let myArray = ["Peter","George","Mary"];
myArray.includes("George"); // true
myArray.includes("John");   // false
```

- `indexOf()` – returns the **index** where the given value is stored
  - Returns **-1** if the value is **not found**

```
myArray.indexOf("Mary"); // 2
myArray.indexOf("Nick"); // -1
```

Manipulating Arrays

Slicing Arrays

- The `slice()` function creates a new array from part of another
- Gets a range of elements from selected **start** to **end** (exclusive)
- Note that the original array will **not be modified**

```
let myArray = ["one","two","three","four","five"];
let sliced = myArray.slice(2);
console.log(myArray);
//["one","two","three","four","five"]
console.log(sliced); // ["three","four","five"]
console.log(myArray.slice(2,4)); // ["three","four"]
```

Splice: Cut and Insert Array Elements

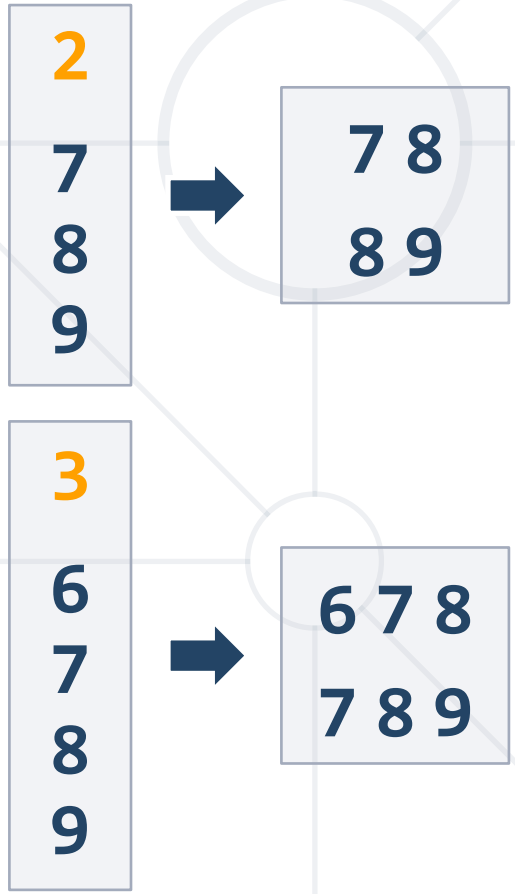
- The `splice()` adds/removes items to/from an array, and **returns** the removed item(s)
- This function **changes the original array**

```
let nums = [5, 10, 15, 20, 25, 30];
let mid = nums.splice(2, 3); // start, delete-count
console.log(mid.join('|'));  // 15|20|25
console.log(nums.join('|')); // 5|10|30
```

```
nums.splice(3, 2, "twenty", "twenty-five");
console.log(nums.join('|')); // 5|10|15|twenty|twenty-five|30
```

Problem: First and Last K Numbers

- You are given an array of numbers
  - The first element holds an integer **k**
  - All other elements are from the array that needs to be processed
  - Print the first **k** and the last **k** elements of the array on a new line (space separated)



Solution: First and Last K Numbers

```
function firstLastKElements(arr) {
  let k = arr.shift();
  console.log(arr.slice(0, k).join(' '));
  console.log(arr.slice(arr.length-k,
    arr.length).join(' '));
}
```

Problem: Sum Last K Numbers Sequence

- Take two integers **n** and **k**
- Generate and print the following sequence:
  - The first element is: **1**
  - All other elements = the **sum** of the previous **k** elements
  - The length of the sequence is **n** elements

```
6
3
```
➡
```
Sequence:
1 1 2 4 7 13
```

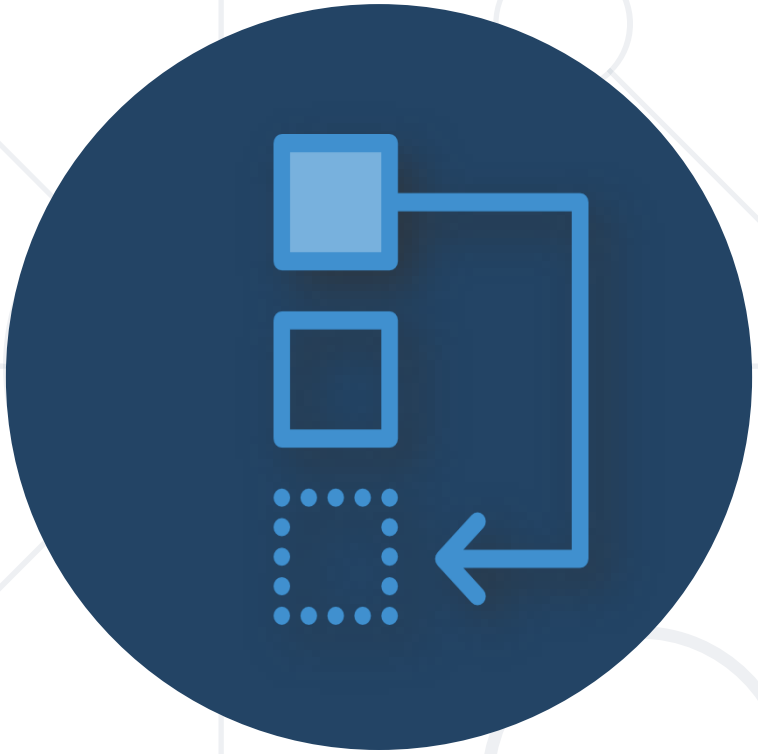
```
8
2
```
➡
```
Sequence:
1 1 2 3 5 8 13 21
```

```
9
5
```
➡
```
Sequence:
1 1 2 4 8 16 31 61 120
```

Solution: Sum Last K Numbers Sequence

Software University

```
function sumLastKNumbersSequence(n, k) {
  let seq = [1];
  for (let current = 1; current < n; current++) {
    let start = Math.max(0, current - k);
    let end = current - 1;
    let sum = // TODO: Sum the values of seq[start … end]
    seq[current] = sum;
  }
  console.log(seq.join(' '));
}
```



Processing Arrays

Transforming, Filtering and Sorting Elements

Transform Elements

- `map()` – creates a **new array** by applying a **function** to every element

```
let myArr = ['one', 'two', 'three', 'four'];
let lengths = myArr.map(x => x.length);
console.log(lengths); // [3,3,5,4]
```

```
let numsAsStrings = ["5","3","14","-2","8"];
let nums = numsAsStrings.map(Number);
console.log(nums);    // [5, 3, 14, -2, 8]
let incr = nums.map(x => x+1);
console.log(incr); // [6, 4, 15, -1, 9]
```

Filter Elements

- `filter()` – creates a **new array** from elements matching **predicate**
  - Predicate is a **function** returning a Boolean value (`true` or `false`)

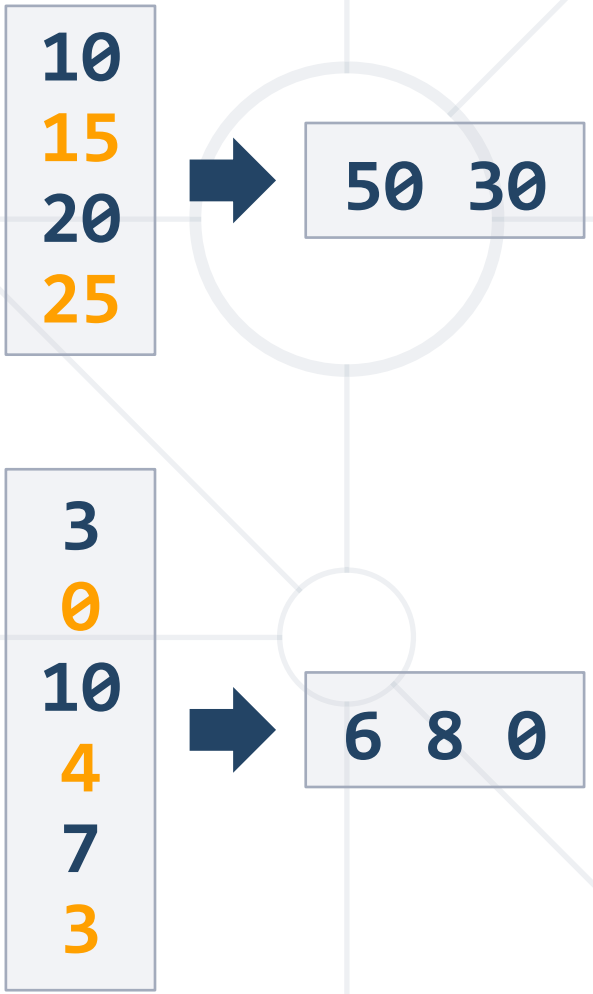
```
let myArr = ['one', 'two', 'three', 'four'];
let longWords = myArr.filter(x => x.length > 3);
console.log(longWords); // ['three','four']
```

```
let nums = [5, -11, 3, -2, 8]
let positiveNums = nums.filter(x => x > 0);
console.log(positiveNums); // [5, 3, 8]
```

Problem: Process Odd Numbers

- You are given an **array of numbers**
  - Print the elements at **odd** positions, **doubled** and **reversed**

```
function solve(arr) {
  let result = arr
    .filter((num, i) => i % 2 == 1)
    .map(x => 2*x)
    .reverse();
  return result.join(' ');
}
```



Sorting Arrays

- The `sort()` function sorts the items of an array
- Depending on the provided **compare function**, sorting can be **alphabetic** or **numeric**, and either **ascending (up)** or **descending (down)**
- By default, the `sort()` function sorts the values as strings in **alphabetical** and **ascending** order
- If you want to sort numbers or other values, you need to provide the correct **compare function**!

Sorting Arrays – Example

```
let names = ["Peter","George","Mary"];
names.sort(); // Default behaviour – alphabetical order
console.log(names); // ["George","Mary","Peter"]
```

```
let numbers = [20, 40, 10, 30, 100, 5];
numbers.sort(); // Unexpected result on arrays of numbers!
console.log(numbers); // [10,100,20,30,40,5]
```

Compare Functions

- A **function** receiving **two parameters**, e.g. **a** and **b**
  - **Returns** either a **positive** number, a **negative** number, or **zero**
  - If **result < 0**, a is sorted **before** b
  - If **result > 0**, a is sorted **after** b
  - If **result = 0**, a and b are **equal** (no change)

```
let nums = [20, 40, 10, 30, 100, 5];
nums.sort((a, b) => a-b); // Compare elements as numbers
console.log(nums.join('|')); // 5|10|20|30|40|100
```

Sorting String Arrays

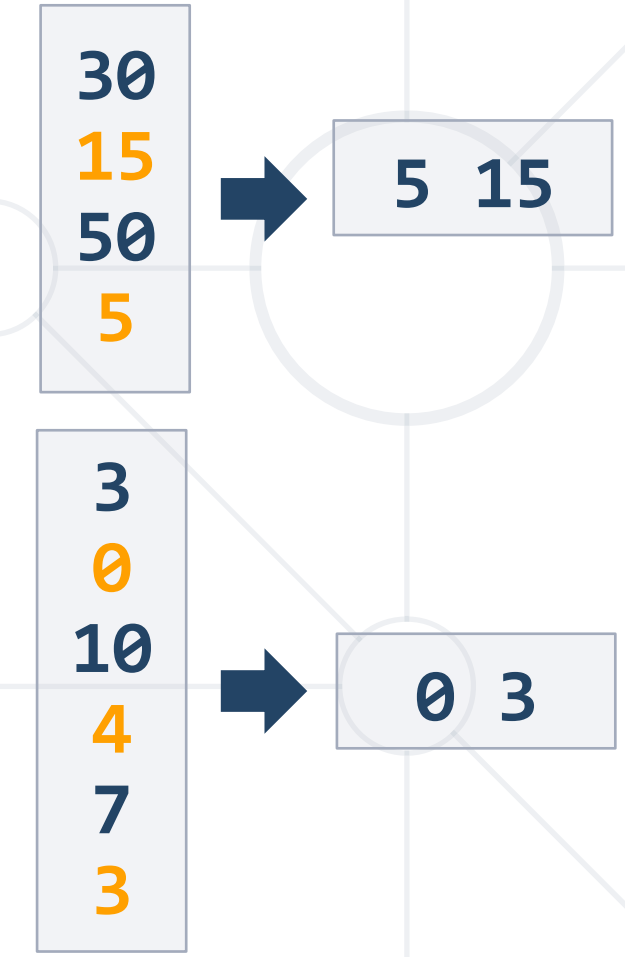
- The **localeCompare()** method is used to compare any two characters without regard for the case used
  - It's a string method so it can't be used directly on an array
  - Pass localeCompare() as the comparison function:

```
let words = ['nest', 'Eggs', 'bite', 'Grip', 'jAw'];
words.sort((a, b) => a.localeCompare(b));
console.log(words);
// ['bite', 'Eggs', 'Grip', 'jAw', 'nest']
```

Problem: Smallest 2 Numbers

- You are given an **array of numbers**
  - Print the **smallest** two numbers

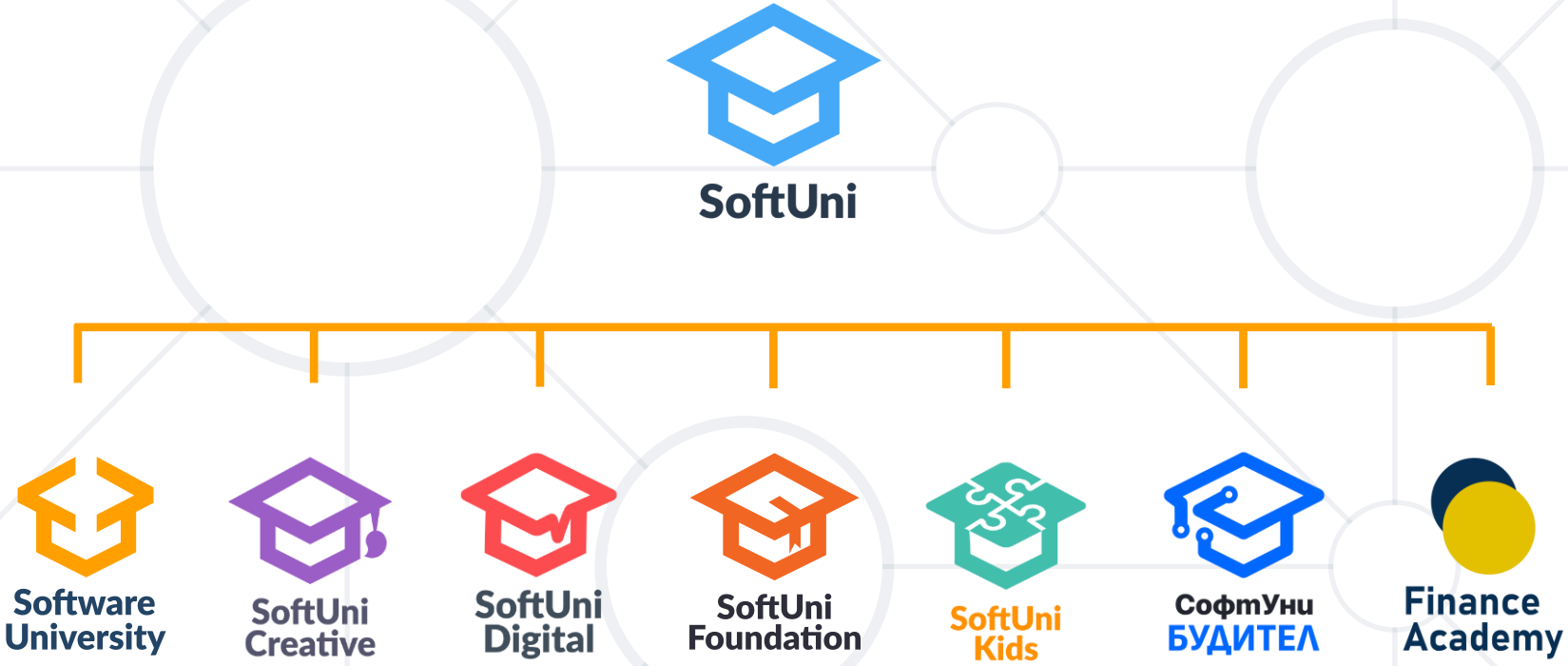
```
function smallestTwoNumbers(arr) {
  arr.sort((a, b) => a-b);
  let result = arr.slice(0, 2);
  console.log(result.join(' '));
}
```



Summary

- Arrays in JavaScript aren't fixed
- Can add / remove / insert elements at runtime
- Sorting arrays can be done with a compare function

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg