Arrays

Sequences of Elements
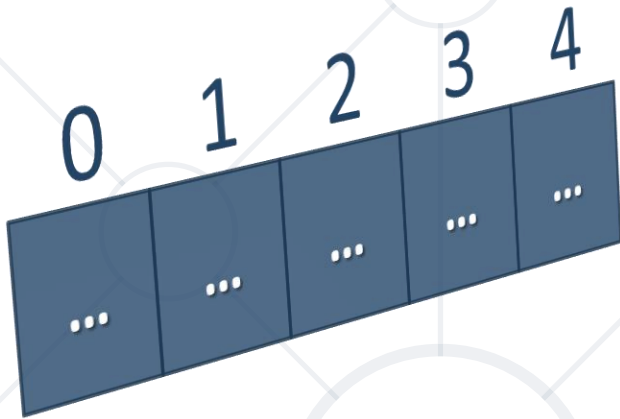
SoftUni Team

Technical Trainers

Software University

SoftUni

Software University
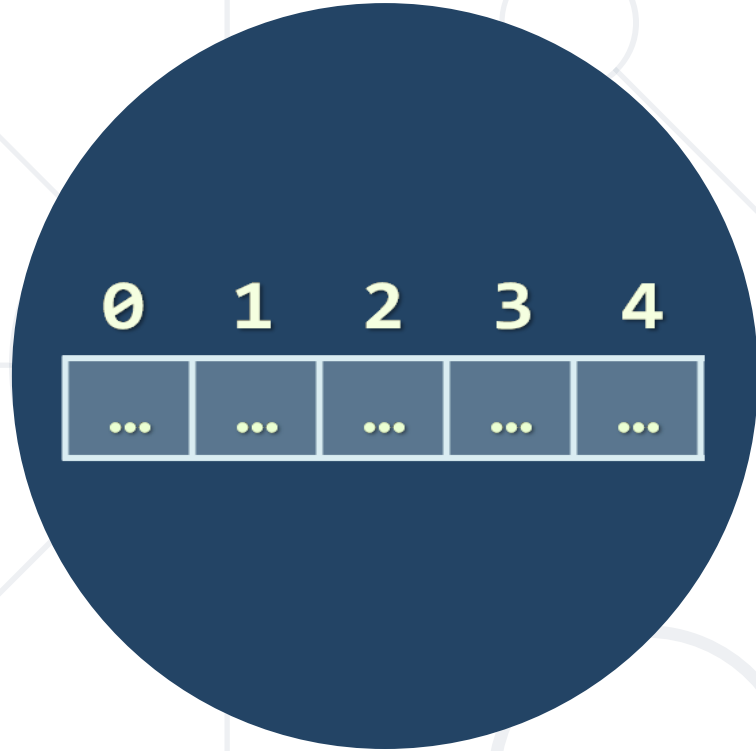
https://softuni.bg

Table of Contents

1. Definition

2. Operations

3. Array Iteration

4. For-of loops

sli.do

#fund-js

Arrays in JS

Definition and Simple Usage

What Are Arrays?

In programming **array** is a **sequence of elements**

0   1   2   3   4   **Element index**

Array of 5 elements

| ... | ... | ... | ... | ... |

Element of an array

- We can store **multiple values** in one variable

- Elements are numbered from **0** to **length-1**

- Arrays have **variable size** (**Array.length**) can be resized (unlike C# / Java)

Creating Arrays

- **Creating** an array of numbers:

```javascript
let numbers = [1, 2, 3, 4, 5];
let names = [];
```

We are creating an Array using the literal [ ]

- **Accessing** array elements by index:

```javascript
console.log(numbers[0]); // 1
```

The [n] operator accesses elements by index
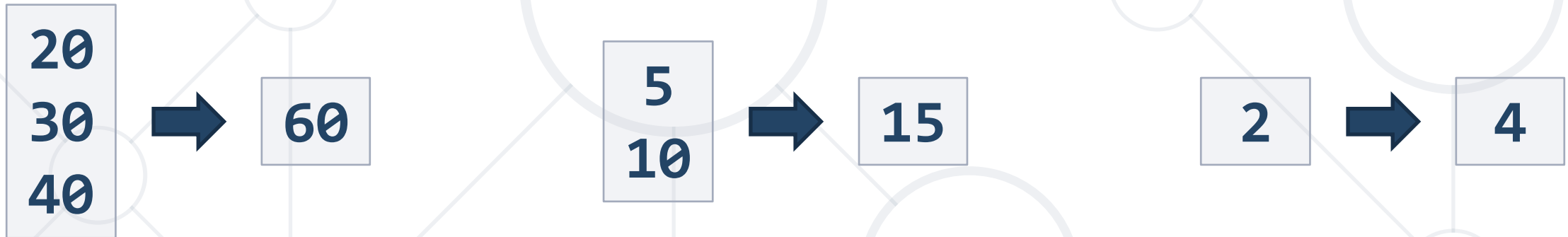
- **Assigning values** to the array elements:

```javascript
numbers[3] = numbers[1] + numbers[2];
console.log(numbers.length); // 5
console.log(numbers); // [1, 2, 3, 5, 5]
```

The length holds the number of array elements

Problem: Sum First and Last Array Elements

You are given an **array of numbers**

- Calculate and print the sum of the **first** and the **last** elements

```
20
30    ➡    60
40
```

```
5
10    ➡    15
```

```
2    ➡    4
```

```
function sumFirstAndLast(arr) {

    console.log(arr[0] + arr[arr.length - 1]);

}
```

Days of Week – Example

The days of week can be stored in array of strings:

```
let days = [
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday",
    "Sunday"
];
```

Index	Value
days[0]	Monday
days[1]	Tuesday
days[2]	Wednesday
days[3]	Thursday
days[4]	Friday
days[5]	Saturday
days[6]	Sunday

Problem: Days of Week

- Write a program which receives a number and prints the corresponding name of the day of week (in English)

- If the number is not a valid day, print "Invalid day!"

3 ➡	**Wednesday**	33 ➡	**Invalid day!**
6 ➡	**Saturday**	-3 ➡	**Invalid day!**

```
function dayOfWeek(day){

    let days = [ "Monday", "Tuesday", "Wednesday", "Thursday",
                 "Friday", "Saturday", "Sunday" ];

    if (day >= 1 && day <= 7)

        console.log(days[day - 1]);

    else

        console.log("Invalid day!");

}
```

> The first day in our array is on index 0, **not** 1.

Arrays of Different Types

```javascript
// Array holding numbers
let numbers = [10, 20, 30, 40, 50];
```

```javascript
// Array holding strings
let weekDays = ['Monday', 'Tuesday', 'Wednesday',
  'Thursday', 'Friday', 'Saturday', 'Sunday'];
```

```javascript
// Array holding mixed data
let mixedArr =
  [20, new Date(), 'hello', {x:5, y:8}];
```

Adding New Elements

You can **add** an element to the end of the array:

```
let arr = [10, 20, 30];

arr[arr.length] = 40;

console.log(arr); // [10, 20, 30, 40]
```

Or you can use the built-in **push** method:

```
arr.push(50);      // Adds an element at the end

console.log(arr); // [10, 20, 30, 40, 50]
```

JS Arrays and Invalid Positions

```js
let nums = [10, 20, 30];
nums[4] = 50; // Will resize the array
console.log(nums); // [10, 20, 30, <empty>, 50]
console.log(nums.length); // 5
console.log(nums[3]); // undefined
```

```js
console.log(nums[-5]); // undefined (invalid index)
nums[-5] = 8; // Will not resize the array
console.log(nums[-5], nums.length); // 8 5
```

Array Methods

Using built-in array functionality

Array Methods

- Arrays are **special objects**

    - They have built-in **properties** and **methods**, like **length**

- Methods are written with a dot after the variable name:

```
let nums = [10, 20, 30];
console.log(nums.length); // 3
```

- Other examples: **push()**, **includes()**, **toString()**, **join()**

- More methods will be examined in the **Arrays Advanced** lesson

Example Usage

■ Check if the array **contains** the specified element:

```javascript
let arr = [10, 20, 30];

console.log(arr.includes(20)); // true

console.log(arr.includes(0)); // false
```

■ Create a string from all elements, **separated** by given string:

```javascript
console.log(arr.join(':')); // 10:20:30

let words = [ "one", "two" ];

console.log(words.join(' - ')); // one - two
```

Array Iteration

Using a for Loop

Printing Arrays On the Console

- To print all array elements, a for-loop can be used
- Print array elements using **toString()**

```
let capitals = ['Sofia', 'Washington', 'London'];

for (let i = 0; i < capitals.length; i++){
  console.log(capitals[i]);
}
```

```
console.log(capitals.toString())
// Sofia,Washington,London
```

Problem: Reverse an Array of Numbers

Receive a number **n** and an **array** of elements, **create** a **new** array with **n** numbers, **reverse** it and print its elements on a single line, space-separated:

| 3 |
| 10 |
| 20 |
| 30 |
| 40 |

➡ `30 20 10`

| 4 |
| -1 |
| 20 |
| 99 |
| 5 |

➡ `5 99 20 -1`

Solution: Reverse an Array of Integers

```
function reverse(n, inputArr) {

  let arr = [];

  for (let i = 0; i < n; i++)

    arr.push(inputArr[i]);

  let output = '';

  for (let i = arr.length - 1; i >= 0; i--)

    output += `${arr[i]} `;

  console.log(output);
}
```

Printing Arrays with for / Join

- Use for-loop:

```
let arr = [ 1, 2, 3, 4, 5 ];
for (let i = 0; i < arr.length; i++)
    console.log(arr[i]);
```
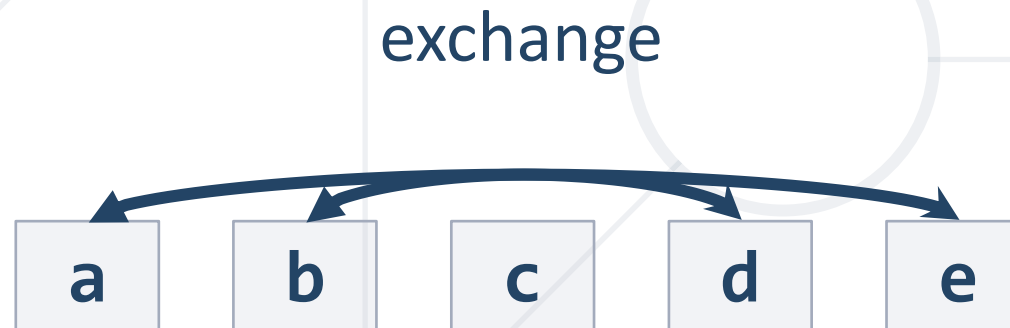
- Use **join(separator):**

```
let nums = [ 1, 2, 3 ];
console.log(nums.join(', ')); // 1, 2, 3
let words = [ "one", "two" ];
console.log(nums.join(' - ')); // one - two
```

Problem: Reverse Array of Strings

- Receive an array of strings (space separated values), **reverse it in place** and print its elements:

| a b c d e | ➡ | e d c b a |

| -1 hi ho w | ➡ | w ho hi -1 |

- Reversing array elements (**without** creating new array):

exchange

| a | b | c | d | e |

```javascript
function reverse(arr) {
  for (let i = 0; i < arr.length / 2; i++) {
    let oldElement = arr[i];
    let previousIndex = arr.length - 1 - i;
    arr[i] = arr[previousIndex];
    arr[previousIndex] = oldElement;
  }
  console.log(arr.join(' '));
}
```
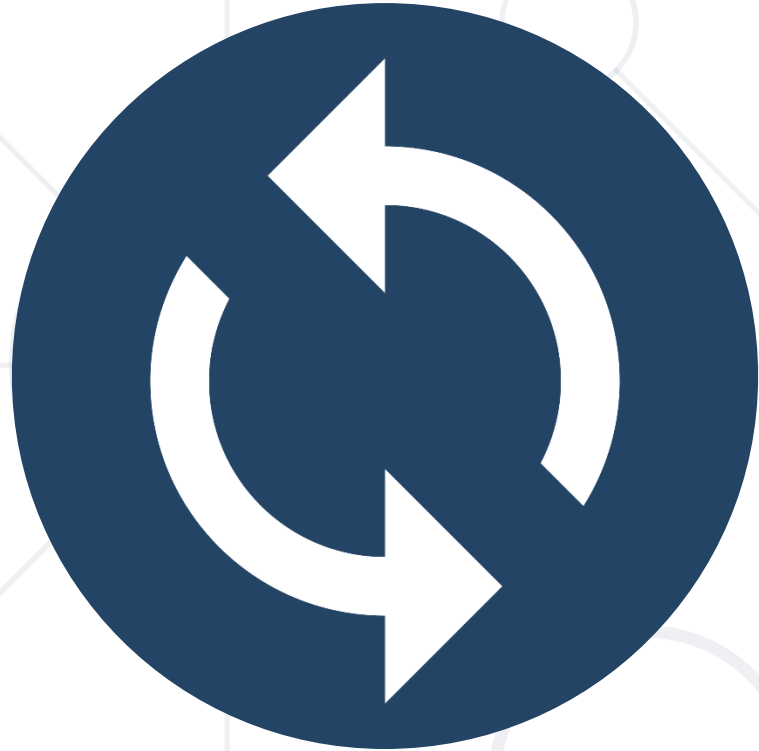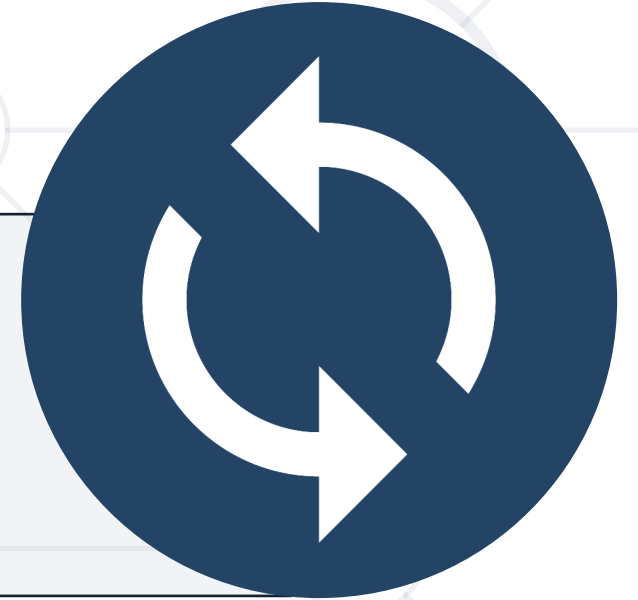
For-of Loops

Alternative Way to Iterate

For-of Loop

- Iterates through all **elements** in a collection

- Cannot access the current index

```
for (let el of collection) {
    // Process the value here

}
```

```
let numbers = [ 1, 2, 3, 4, 5 ];
let output = '';
for (let number of numbers)
    output += `${number} `;
console.log(output);
```
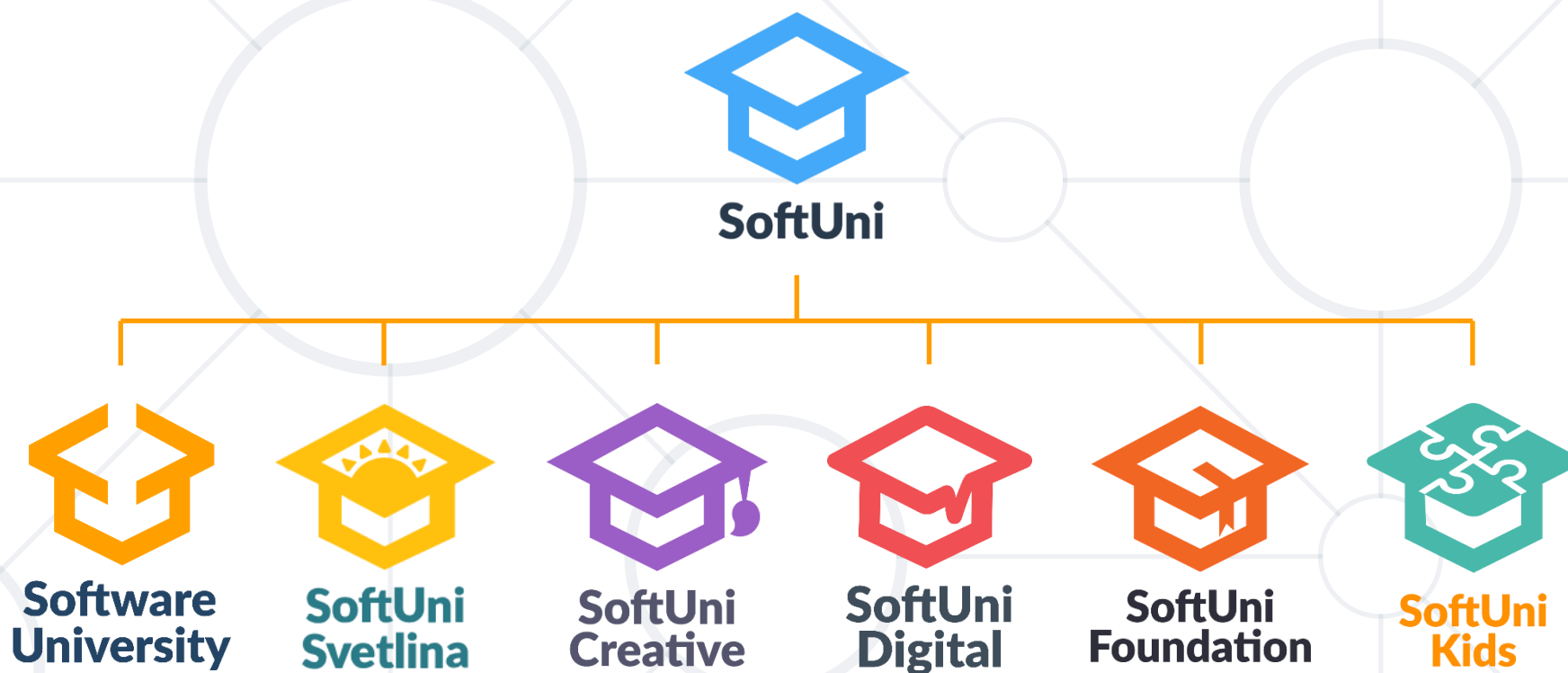
1 2 3 4 5

Live Exercises

Summary

- Arrays are **sequence** of elements
    - Elements are numbered
    from **0** to `length-1`

- Create an array: `let arr = [5,3,7]`

- Access elements: `arr[2] = 4`

- Elements can be iterated with a
standard loop or a **for-of** loop

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Educational Partners

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
- Software University Forums
  - forum.softuni.bg

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