Arrays in JS

IT TALENTS
Training Camp

Overview

- What is an Array and how it helps us?
- Declaration and initialization of Arrays
- Access of elements
- Array size
- More about Arrays
- How to check if a variable is an Array?
- Iterating an Array
- Comparing arrays
- Copying arrays
- Find an element in an Array
- Built in operations for Arrays - the easy way

What is an Array and how it helps us?

- What do we do when we have a lot of data from the same context?

  example: Grades of a student group

- What is the problem with that?

  Is it rational to define multiple variables for every item?

- **Arrays to the rescue!**

IT TALENTS
Training Camp

What is an Array and how it helps us?

- The Array is the most common **data structure** that you will use!

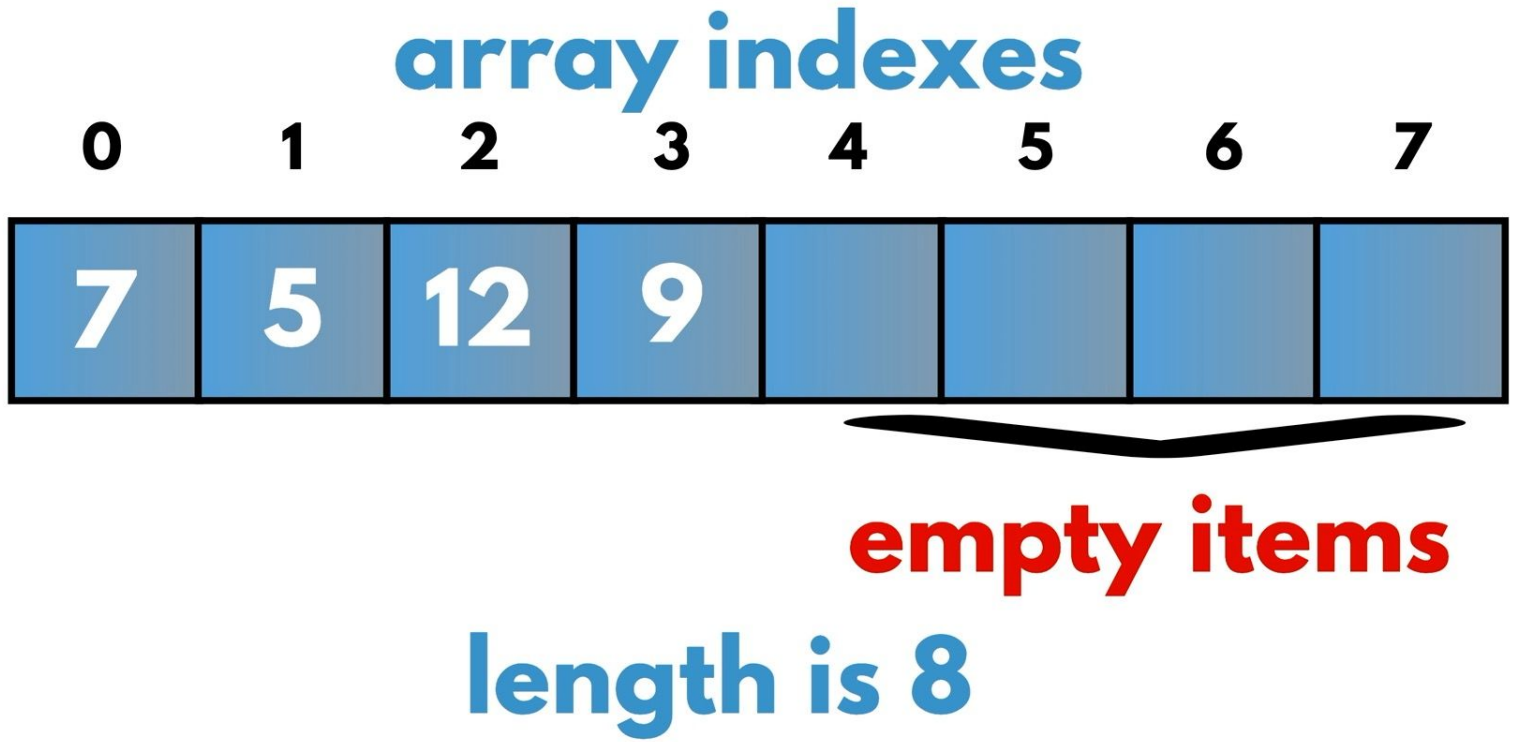
  - **Sequence** of multiple elements
  - Can store data **from any type** simultaneously
  - **Order** of the items stays the same
  - **Dynamic length** - can be expanded or shrunk
  - **Direct access** to the items via **index**

```
1   let gradeList = [3,3,4,3,6,5,2]; // single variable - multiple values
2
3   let firstGrade = gradeList[0]; // direct access via 0 based index
```

- When we talk about data structures…

IT TALENTS
Training Camp

What is an Array and how it helps us?

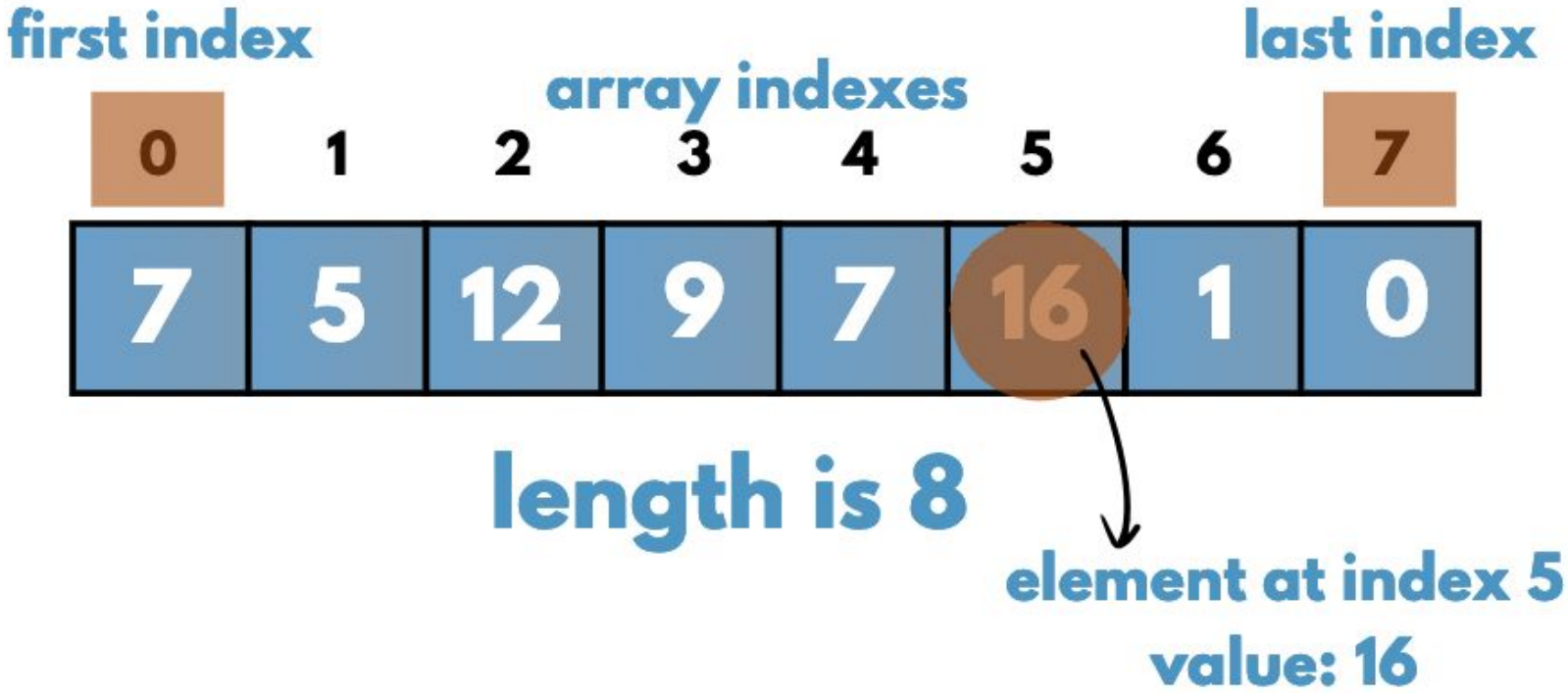


Declaration and initialization of Arrays

```
1   // declare and initialize an empty array
2   let array = [];
3   // -> []
4
5   // declare and initialize an empty array - !!!bad practice!!!
6   let array1 = new Array();
7   // -> []
8
9   // multiple data types in one array
10  let array2 = [1,3,5,8,'Ivan',false, {name: 'Commodeus', reign: 12 }];
11  // -> [1,3,5,8,'Ivan',false, {name: 'Commodeus', reign: 12 }]
12
13  // multiple data types in one array - same as array2
14  let array3 = new Array(1,3,5,8,'Ivan',false, {name: 'Commodeus', reign: 12 });
15  // -> [1,3,5,8,'Ivan',false, {name: 'Commodeus', reign: 12 }]
16
17  // initialize array with 2 empty elements - !!!bad practice!!!
18  let array4 = new Array(2);
19  // -> [undefined, undefined]
```

Access of elements

- Index is the position of the element in the array - **zero based!**
- First index - **0**, last index - **length - 1**
- Arrays in JS are mutable - elements can be changed, arrays can be expanded or shrunk



IT TALENTS Training Camp

Access of elements, array.length

```
1   let arr = [3,3,4,3,6,5,2];
2
3   // gets the first element
4   let firstEl = arr[0];
5   // -> 3
6
7   // how many elements are there? - specially for us - arr.length!
8   let count = arr.length;
9   //-> 7
10
11  // gets the last element -> 2
12  let lastEl = arr[arr.length - 1];
13  // -> 2
14
15  //trying to get non-existent element will result in undefined
16  let nonExistentEl = arr[-2]; // -> undefined
17  let nonExistentEl2 = arr[200]; // -> undefined
18  let nonExistentEl3 = arr[true]; // -> undefined
19  let nonExistentEl4 = arr['bahur']; // -> undefined
```

IT TALENTS
Training Camp

More about arrays

```
1    let arr = [];
2
3    arr[12] = 8;
4    // -> [12 x empty, 8]
```

- Setting an element at index greater than the length will fill with **undefined** the cells between the last existent element and element at index - 1

- Setting an element on negative index is a **very bad practice** - strictly forbidden!
  It results in unexpected behavior.

```
1    let arr = [1, 2, 6.2, 4];
2
3    arr[-1] = 2; // forbidden
4
5    console.log(arr.length);
6    // -> 4 The element on negative index is not considered
7
8    console.log(arr[-1]);
9    // -> 2 The element is stored
```

How to check if a variable is an Array

- **typeof** on Array results in **object**

```
let arr = [1, 2, 6.2, 4];

console.log(Array.isArray(arr));
// -> true

console.log(Array.isArray(true));
// -> false

console.log(Array.isArray('Array'));
// -> false

console.log(Array.isArray(undefined));
// -> false

console.log(Array.isArray(null));
// -> false
```

IT TALENTS
Training Camp

Iterating an Array

- Iterating is the process of **going through all the elements in the Array**

```
1
2   let gradeList = [2, 4, 5, 3, 4, 4, 3, 5];
3
4   // most common use is for loop
5   for (let i = 0; i < gradeList.length; i++) {
6       console.log(gradeList[i]);
7   }
8   // you can also modify the elements
9   for (let i = 0; i < gradeList.length; i++) {
10      gradeList[i] = i * 2 + 3; // some complex business logic
11  }
12
13  // you can use any loop you want
14  let j = 0;
15  while (j < gradeList.length) {
16      console.log(gradeList[j]);// prints to the console
17      j++;
18  }
```

Comparing arrays

- We can't use "==", because **Arrays are reference type**

```
1    // two identical arrays
2    let arr = [1, 2, 6.2, 4];
3    let arr1 = [1, 2, 6.2, 4];
4
5    console.log(arr == arr1);
6    // -> false
```

- To compare properly we have to iterate through the array and compare every element individually

```
5    let areEqual = true;
6    for (let i = 0; i < arr.length; i++) {
7        if(arr[i] !== arr1[i]) {
8            areEqual = false;
9            break;
10       }
11   }
12   console.log(areEqual);
13   // -> true
```

Copying arrays

- We can't use "=", because **Arrays are reference type**

```
1    // two identical arrays
2    let arr = [1, 2, 6.2, 4];
3    let arr1 = arr;
4
5    arr1[0] = 'New value';
6    console.log(arr);
7    // -> ['New value', 2, 6.2, 4]
```

- To copy an Array properly we have to iterate through all the elements and copy them individually

```
5    for (let i = 0; i < arr.length; i++) {
6        arr1[i] = arr1[i];
7    }
8    // arr -> [1, 2, 6.2, 4];
9    // arr1 -> [1, 2, 6.2, 4];
10
11   arr1[0] = 'New Value';
12   console.log(arr);
13   // -> [1, 2, 6.2, 4];
```

Array.unshift(newEle)

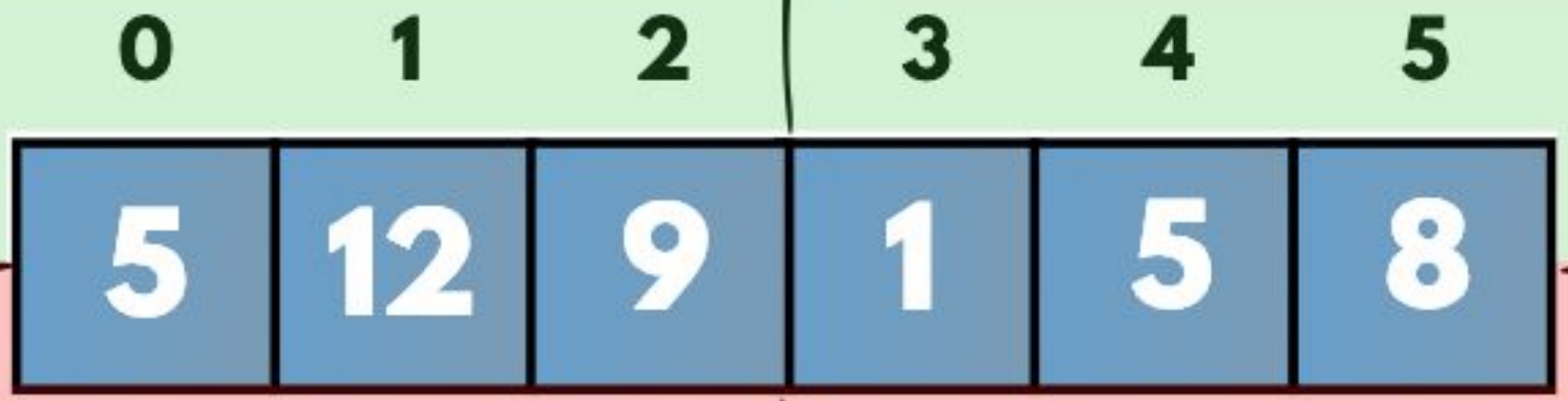
adds at the beginning

```
let arr = [5, 12, 9, 1, 5, 8];
arr.unshift(6);
console.log(arr); // -> [6, 5, 12, 9, 1, 5, 8]
```

Array.push(newEle)

adds at the end

```
let arr = [5, 12, 9, 1, 5, 8];
arr.push(6);
console.log(arr); // -> [5, 12, 9, 1, 5, 8, 6]
```



Array.shift()

removes from the beginning

```
let arr = [5, 12, 9, 1, 5, 8];
arr.shift();
console.log(arr); // -> [12, 9, 1, 5, 8]
```

Array.pop()

removes from the end

```
let arr = [5, 12, 9, 1, 5, 8];
arr.pop();
console.log(arr); // -> [5, 12, 9, 1, 5]
```

Built in operations - Array.slice

```
1   let arr = [1, 2, 6.2, 4];
2   //without arguments
3   let copy = arr.slice() // used to create a copy of the array
4   console.log(copy);
5   // -> [1, 2, 6.2, 4]
6
7   /*
8       start argument defines the beginnning index of
9       extracted elements
10  */
11  let part = arr.slice(2);
12  console.log(part);
13  // -> [6.2, 4]
14
15  /*
16      end argument defines the end index of the
17      extracted elements NOT INCLUSIVE
18  */
19  let part1 = arr.slice(2,3);
20  console.log(part1);
21  // -> [6.2]
```

IT TALENTS
Training Camp

Built in operations - Array.splice

- Modifies the Array in place
- **start** param - defines the starting index at which to start changing the array.
- **deleteCount** - indicating the number of elements in the array to remove from **start**.
- **item1, item2, …** - the elements to add to the array, beginning from start

```
1    let arr = [1, 2, 6.2, 4];
2    /*
3        remove 1 element, starting from index 2
4        and put in its place 9,9,9
5    */
6    arr.splice(2, 1, 9,9,9);
7    console.log(arr);
8    // -> [ 1, 2, 9, 9, 9, 4 ]
```

KEEP CALM THIS IS MY LAST SLIDE!

IT TALENTS
Training Camp