Դաս 03: Զանգվածներ և մեթոդներ

push()

push մեթոդը ավելացնում է նոր տարր(եր) զանգվածի վերջում և վերադարձնում է նոր երկարությունը:

```
const fruits = ["խնձոր", "բանան"];
fruits.push("ելակ");
console.log(fruits); // ["խնձոր", "բանան", "ելակ"]
```

pop()

pop մեթոդը հեռացնում է զանգվածի վերջին տարրը և վերադարձնում է այն:

```
const fruits = ["խնձոր", "բանան", "ելակ"];
const lastFruit = fruits.pop();
console.log(fruits); // ["խնձոր", "բանան"]
console.log(lastFruit); // "ելակ"
```

shift()

shift մեթոդը հեռացնում է առաջին տարրը զանգվածից և վերադարձնում է այն:

```
const fruits = ["խնձոր", "բանան", "ելակ"];
const firstFruit = fruits.shift();
console.log(fruits); // ["բանան", "ելակ"]
console.log(firstFruit); // "խնձոր"
```

unshift()

unshift մեթոդը ավելացնում է նոր տարր(եր) զանգվածի սկզբում և վերադարձնում է նոր երկարությունը:

```
const fruits = ["բանան", "ելակ"];
fruits.unshift("խնձոր");
console.log(fruits); // ["խնձոր", "բանան", "ելակ"]
```

1. map()

Վերադարձնում է նոր զանգված, որտեղ յուրաքանչյուր տարրին վրա կիրառվում է փոխանցված ֆունկցիան

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6, 8]
```

2. filter()

Ստեղծում է նոր զանգված, որը պարունակում է միայն այն տարրերը, որոնց համար փոխանցված ֆունկցիան վերադարձրել է true

```
const numbers = [1, 2, 3, 4, 5];
const even = numbers.filter(num => num % 2 === 0);
console.log(even); // [2, 4]
```

3. reduce()

Մշակում է զանգվածը և վերադարձնում է մեկ ընդհանուր արժեք կիրառելով փոխանցված ֆունկցիան

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((accumulator, current) => accumulator + current, 0);
console.log(sum); // 10
```

4. forEach()

Կատարում է փոխանցված ֆունկցիան յուրաքանչյուր տարրի համար: Վերադարձնում է undefined:

```
const numbers = [1, 2, 3];
numbers.forEach(num => console.log(num * 2));
// Вывод: 2, 4, 6
```

5. find()

Վերադարձնում է զանգվածի առաջին տարրը, որը բավարարում է ֆունկցիայի պայմաններին: Եթե պայմանները բավարարող տարր չի հայտնաբերվել, վերադարձնում է undefined.

```
const people = [{ name: "Alice", age: 25 }, { name: "Bob", age: 30 }];
const person = people.find(p => p.age === 30);
console.log(person); // { name: "Bob", age: 30 }
```

6. findIndex()

Վերադարձնում է առաջին տարրի ինդեքսը որը բավարարում է ֆունկցիայի պայմանին: Եթե նման տարր չի հայտնաբերվել, վերադարձնում է -1:

```
const numbers = [5, 12, 8, 130, 44];
const index = numbers.findIndex(num => num > 10);
console.log(index); // 1
```

7. some()

Ստուգում է, համապատասխանում է գոնե զանգվածի մեկ տարր ֆունկցիայի փոխանցված տվյալին: վերադարձնում է true կամ false

```
const numbers = [1, 2, 3, 4];
const hasEven = numbers.some(num => num % 2 === 0);
console.log(hasEven); // true
```

8. every()

Ստուգում է, համապատասխանում են զանգվածի բոլոր տարրերը ֆունկցիայի փոխանցված տվյալին: վերադարձնում է true կամ false

```
const numbers = [2, 4, 6];
const allEven = numbers.every(num => num % 2 === 0);
console.log(allEven); // true
```

9. includes()

Ստուգում է պարունակում է արդյոք զանգվածը կոնկրետ արժեք թե ոչ

```
const fruits = ["apple", "banana", "mango"];
console.log(fruits.includes("banana")); // true
```

10. sort()

Սորտավորում է զանգվածը: Թվերի համար պետք է համեմատման ֆունկցիա:

```
const numbers = [4, 2, 8, 1];
numbers.sort((a, b) => a - b);
console.log(numbers); // [1, 2, 4, 8]
```

11. concat()

Միավորում է երկու կամ ավելի զանգվածներ մեկում

```
const arr1 = [1, 2];
const arr2 = [3, 4];
const result = arr1.concat(arr2);
console.log(result); // [1, 2, 3, 4]
```

12. slice()

Վերադարձնում է նոր զանգված, որը պարունակում է սկզբնական զանգվածի այն մասը որը նշվել է

```
const numbers = [1, 2, 3, 4, 5];
const sliced = numbers.slice(1, 3);
console.log(sliced); // [2, 3]
```

13. splice()

Թույլատրում է ավելացնել կամ ջնջել տարրեր զանգվածից, փոփոխելով իր պարունակությունը:

```
const numbers = [1, 2, 3, 4];
numbers.splice(1, 2, 5, 6); // ջնջել 2 տարր 1 ինդեքսից սկսած և ավելացնել 5, 6
console.log(numbers); // [1, 5, 6, 4]
```

14. reverse()

Շրջում է զանգվածը

```
const array = [1, 2, 3, 4, 5];
array.reverse();
console.log(array); // [5, 4, 3, 2, 1]
```

15. at() // 2022 update

Ստանում է բացասական թիվ և ցուցարդում n-երորդ տարրը սկսած զանգվածի վերջից՝ -1 -ը դա զանգվածի վերջից առաջին տարրն է (2022-թ ից):

```
const arr = [1, 2, 3, 4];
console.log(arr.at(-1));
```

16. findLast() findLastIndex() // 2022 update

Աշխատում են ինչպես find, findIndex սակայն զանգվածի վերջից սկսած:

```
const numbers = [1, 2, 3, 4, 5, 3];
console.log(numbers.findLast(num => num === 3)); // 3
console.log(numbers.findLastIndex(num => num === 3)); // 5
```

17. toSorted() // 2023 update

Ստեղծում է զանգվածի սորտավորված կրկնօրինակը:

```
const numbers = [3, 1, 4, 1, 5];
const sorted = numbers.toSorted();
console.log(sorted); // [1, 1, 3, 4, 5]
console.log(numbers); // [3, 1, 4, 1, 5]
```

18. toReversed() // 2023 update

Ստեղծում է զանգվածի հակառակ շրջած կրկնօրինակը:

```
const numbers = [1, 2, 3];
const reversed = numbers.toReversed();
console.log(reversed); // [3, 2, 1]
```

19. toSpliced() // 2023 update

Վերադարձնում է զանգվածի կրկնօրինակը ավելացնելով տարրերը համապատաս խան դիրքերում

```
const numbers = [1, 2, 3, 4];
const spliced = numbers.toSpliced(1, 2, 9, 10);
console.log(spliced); // [1, 9, 10, 4]
```

20. with() // 2023 update

Վերադարձնում է զանգվածի կրկնօրինակը փոխարինելով նշված ինդեքսի արժեքը

```
const numbers = [1, 2, 3];
const updated = numbers.with(1, 9);
console.log(updated); // [1, 9, 3]
```

*** Ասինխրոն ստեղծվող զանգված // 2023 update**

Թույլ է տալիս ասինխրոն եղանակով ստեղծել զանգվածներ, օգտակար է, երբ զանգվածի տվյալները ստացվում են ասինխրոն:

```
async function fetchData() {
  return [1, 2, 3];
}

const result = await Array.fromAsync(fetchData());
console.log(result); // [1, 2, 3]
```