8. Massiv metodlari



Massivni `for` sikli bilan aylanish ishlaydi, lekin zamonaviy JavaScriptda **massiv metodlari** kodni qisqa va o'qilishi oson qiladi. Bu dars ularni noldan tushuntiradi.

> **Bu darsda nimani o'rganasiz**
>
> Nega sikl o'rniga metodlar kerak.
>
> `forEach`: har element bilan ish qilish.
>
> `map`: har elementni o'zgartirib yangi massiv olish.
>
> `filter`: shartga mos elementlarni tanlash.
>
> `find`: birinchi mosini topish.
>
> `some` va `every`: ha/yo'q savollari.
>
> Metodlarni ketma-ket ulash.

1.1. Nega massiv metodlari?

Tasavvur qiling, sizga sonlar ro'yxati berildi va har birini 2 ga ko'paytirish kerak. Sikl bilan:

```
const sonlar = [1, 2, 3, 4];
const natija = [];
for (let i = 0; i < sonlar.length; i++) {
  natija.push(sonlar[i] * 2);
}
console.log(natija);
```

Natija:

```
[ 2, 4, 6, 8 ]
```

Bir xil ishni `map` metodi bilan **bitta qatorda** bajarish mumkin:

```
const sonlar = [1, 2, 3, 4];
const natija = sonlar.map((x) => x * 2);
console.log(natija);
```

Natija:

```
[ 2, 4, 6, 8 ]
```

Metod nomi **nima qilinayotganini** o'zi aytib turadi: `map` — har bir elementni o'zgartirib **yangi** massiv beradi, `filter` — keraklilarini **tanlab** oladi. Ikkalasi ham asl massivga tegmaydi, shuning uchun kod o'qilishi oson bo'ladi.

> **Callback nima?**
>
> Bu metodlar ichiga **funksiya** beriladi (`x => x * 2`). Bu funksiya **callback** deyiladi: metod uni har bir element uchun o'zi chaqiradi. Funksiya mavzusidan eslang: `(x) => x * 2` — "x ni 2 ga ko'paytir" degani.



1-rasm. Qaysi metodni tanlash kerak?

1.2. forEach — har bir element bilan ish qilish

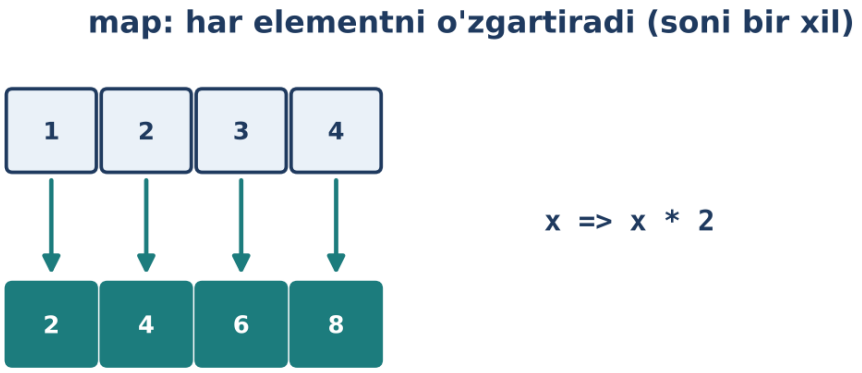
`forEach` massivning har bir elementi uchun callbackni chaqiradi. U **hech narsa qaytarmaydi** (`undefined`), faqat "ish qiladi" (masalan, chop etadi).

```
const mevalar = ["olma", "nok", "uzum"];

mevalar.forEach((meva) => {
  console.log("Meva:", meva);
});

// Callback 3 ta narsa oladi: element, indeks, massivning o'zi
mevalar.forEach((meva, i) => {
  console.log(`${i + 1}. ${meva}`);
});
```

Natija:

```
Meva: olma
Meva: nok
Meva: uzum
1. olma
2. nok
3. uzum
```

1.3. map — har birini o'zgartirish



2-rasm. map: uzunlik o'zgarmaydi

`map` har bir elementga callbackni qo'llaydi va **natijalardan yangi massiv** yig'adi. Yangi massivning uzunligi **asl bilan bir xil**. Asl massiv o'zgarmaydi.

```
const sonlar = [1, 2, 3, 4];
console.log(sonlar.map((x) => x * x));        // kvadratlar
console.log(sonlar.map((x) => x + 100));
console.log(sonlar.map((x) => `?${x}`));
console.log(sonlar);                          // asl o'zgarmadi

const ismlar = ["ali", "vali", "zara"];
console.log(ismlar.map((s) => s.toUpperCase()));
console.log(ismlar.map((s) => s.length));
```

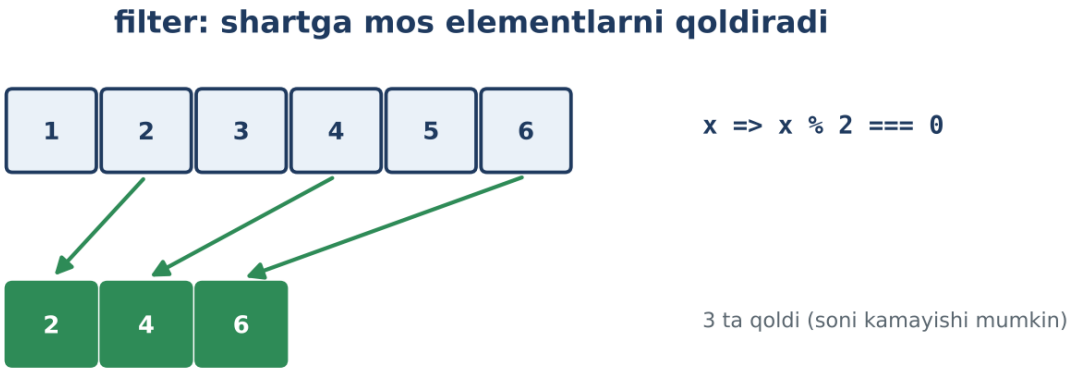
Natija:

```
[ 1, 4, 9, 16 ]
[ 101, 102, 103, 104 ]
[ '?1', '?2', '?3', '?4' ]
[ 1, 2, 3, 4 ]
[ 'ALI', 'VALI', 'ZARA' ]
[ 3, 4, 4 ]
```

map va forEach farqi

`map` — **yangi massiv qaytaradi**; `forEach` — hech narsa qaytarmaydi. Natija kerak bo'lsa, `map` ishlating. Faqat chop etish uchun `forEach`.

1.4. filter — tanlab olish



3-rasm. filter: shartga mos qoladi

`filter` callbackdan `true` qaytgan elementlarni **yangi massivga** yig'adi. Natija asl massivdan qisqa (yoki teng) bo'ladi.

```
const sonlar = [5, 12, 8, 130, 44];

console.log(sonlar.filter((x) => x > 10));
console.log(sonlar.filter((x) => x % 2 === 0));
console.log(sonlar.filter((x) => x > 1000));   // hech biri mos kelmasa: []

const sozlar = ["olma", "anor", "nok", "uzum", "anjir"];
console.log(sozlar.filter((s) => s.startsWith("a")));
console.log(sozlar.filter((s) => s.length > 3));
```

Natija:

```
[ 12, 130, 44 ]
[ 12, 8, 130, 44 ]
[]
[ 'anor', 'anjir' ]
[ 'olma', 'anor', 'uzum', 'anjir' ]
```

1.5. find va findIndex

`find` shartga mos **birinchi elementni** qaytaradi (topilmasa `undefined`). `findIndex` esa uning **indeksini** (topilmasa `-1`).

```
const sonlar = [5, 12, 8, 130, 44];

console.log(sonlar.find((x) => x > 10));        // 12
console.log(sonlar.findIndex((x) => x > 10));   // 1
console.log(sonlar.find((x) => x > 1000));      // undefined
console.log(sonlar.findIndex((x) => x > 1000)); // -1
```

Natija:

```
12
1
undefined
-1
```

1-jadval. filter, find, findIndex

Metod	Nima qaytaradi?	Topilmasa
`filter`	**Hamma** mos elementlar (massiv)	`[]`
`find`	**Birinchi** mos element	`undefined`
`findIndex`	Birinchi mos elementning indeksi	`-1`

1.6. some va every

Bular **ha/yo'q** savoliga `true` yoki `false` qaytaradi:

- `some` — **kamida bitta** element shartga mosmi?
- `every` — **hamma** element shartga mosmi?

```
const yoshlar = [18, 22, 15, 30];

console.log(yoshlar.some((y) => y < 18));     // kamida bittasi voyaga yetmaga
console.log(yoshlar.every((y) => y >= 18));   // hammasi voyaga yetganmi?
console.log(yoshlar.every((y) => y > 0));

console.log([].some((x) => x > 0));           // bo'sh massivda some ?
 false
console.log([].every((x) => x > 0));          // bo'sh massivda every ?
 true
```

Natija:

```
true
false
true
false
true
```

1.7. Obyektlar massivi bilan ishlash

Real dasturlarda ma'lumotlar ko'pincha **obyektlar massivi** ko'rinishida bo'ladi. Metodlar shu yerda ayniqsa foydali:

users.js

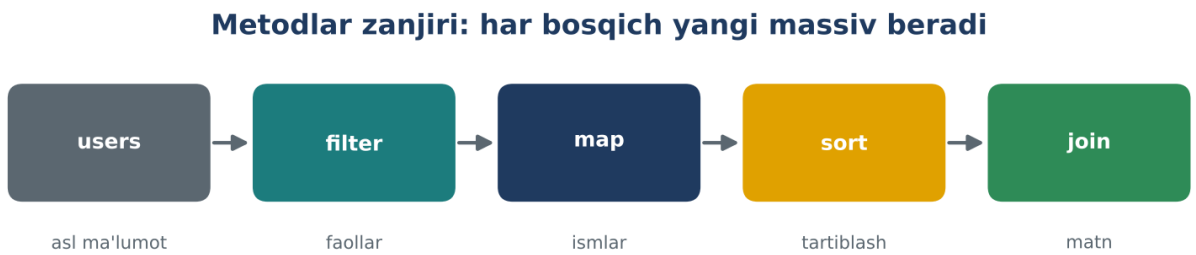
```
const users = [
  { ism: "Ali", yosh: 25, faol: true, ball: 85 },
  { ism: "Vali", yosh: 17, faol: false, ball: 60 },
  { ism: "Zara", yosh: 31, faol: true, ball: 92 },
  { ism: "Bobur", yosh: 22, faol: true, ball: 48 },
];

// Faqat ismlar
console.log(users.map((u) => u.ism));

// Faollar
console.log(users.filter((u) => u.faol).map((u) => u.ism));

// Zara ni topish
console.log(users.find((u) => u.ism === "Zara"));

// Hamma faolmi? Kamida bittasi 90 balldan yuqorimi?
console.log(users.every((u) => u.faol), users.some((u) => u.ball > 90));
```

Natija:

```
[ 'Ali', 'Vali', 'Zara', 'Bobur' ]
[ 'Ali', 'Zara', 'Bobur' ]
{ ism: 'Zara', yosh: 31, faol: true, ball: 92 }
false true
```

1.8. Metodlarni zanjirga ulash

`map` va `filter` **yangi massiv** qaytargani uchun ularni nuqta bilan **ketma-ket ulash** mumkin. Bu "quvur" (pipeline) kabi ishlaydi: har bosqich oldingi natijani qayta ishlaydi.



4-rasm. Metodlar zanjiri

```
const users = [
  { ism: "Ali", yosh: 25, faol: true, ball: 85 },
  { ism: "Vali", yosh: 17, faol: false, ball: 60 },
  { ism: "Zara", yosh: 31, faol: true, ball: 92 },
  { ism: "Bobur", yosh: 22, faol: true, ball: 48 },
];

const natija = users
```

```
  .filter((u) => u.faol)         // 1) faollarni ol
  .map((u) => u.ism);            // 2) faqat ismlarini ol
console.log(natija);

const kattalar = users.filter((u) => u.yosh >= 18).map((u) => `${u.ism} (${u.y
console.log(kattalar);
```

Natija:

```
[ 'Ali', 'Zara', 'Bobur' ]
[ 'Ali (25)', 'Zara (31)', 'Bobur (22)' ]
```

1.9. Kod izi

```
const arr = [1, 2, 3, 4, 5, 6];
const r = arr.filter((x) => x % 2 === 0).map((x) => x * 10);
console.log(r);
```

2-jadval. Zanjir izi

Qadam	Amal	Natija
1	`filter(x => x % 2 === 0)`: juftlarni qoldir	`[2, 4, 6]`
2	`map(x => x * 10)`: har birini 10 ga ko'paytir	`[20, 40, 60]`
3	`console.log(r)`	`[20, 40, 60]`

1.10. Mini loyihalar

loyiha.js

```
// 1) Narxlarga QQS (12%) qo'shish
const narxlar = [10000, 25000, 5000];
console.log(narxlar.map((n) => n * 1.12));

// 2) Uzun so'zlar
const gap = "JavaScript juda qiziqarli dasturlash tili".split(" ");
console.log(gap.filter((s) => s.length > 5));

// 3) Ismlardan salomlashuv ro'yxati
console.log(["Ali", "Vali"].map((i) => `Salom, ${i}!`));

// 4) O'tgan talabalar (ball >= 55)
const talabalar = [{ ism: "A", ball: 70 }, { ism: "B", ball: 40 }, { ism: "C",
const otgan = talabalar.filter((t) => t.ball >= 55).map((t) => t.ism);
console.log("O'tganlar:", otgan.join(", "));

// 5) Manfiy sonlarni topish
console.log([3, -1, 4, -5, 9].filter((x) => x < 0));
```

Natija:

```
[ 11200.000000000002, 28000.000000000004, 5600.000000000001 ]
[ 'JavaScript', 'qiziqarli', 'dasturlash' ]
[ 'Salom, Ali!', 'Salom, Vali!' ]
```

```
O'tganlar: A, C
[ -1, -5 ]
```

1.11. Eng ko'p uchraydigan xatolar

3-jadval. Tipik xatolar

Xato	Sababi	To'g'ri yozuv
`map` ichida `return` unutish (`{}` bilan)	Natija `[undefined, ...]` bo'ladi	`x => { return x * 2; }` yoki `x => x * 2`
`forEach` natijasini o'zgaruvchiga olish	U doim `undefined` qaytaradi	Natija kerak bo'lsa — `map`
`filter` o'rniga `find` ishlatib, massiv kutish	`find` bitta element qaytaradi	Hammasi kerak bo'lsa — `filter`
`find` natijasini tekshirmaslik	Topilmasa `undefined`, keyin xato	`if (natija) { ... }`
Callbackda shartni `=` bilan yozish	`=` tayinlash, `===` taqqoslash	`x === 5`

```
const a = [1, 2, 3];
console.log(a.map((x) => { x * 2; }));     // return yo'q — undefined lar
console.log(a.map((x) => { return x * 2; }));
```

Natija:

```
[ undefined, undefined, undefined ]
[ 2, 4, 6 ]
```

1.12. Mashqlar

1. `[1, 2, 3, 4, 5]` massivining har bir elementiga 10 ni qo'shing.
2. `["ali", "vali"]` ni bosh harfli ko'rinishga keltiring (`charAt(0).toUpperCase() + slice(1)`).
3. `[3, 8, 12, 5, 20]` dan 10 dan kattalarni tanlang.
4. `[2, 4, 6, 7, 8]` da birinchi toq sonni toping (`find`).
5. `[10, 20, 30]` hammasi 5 dan kattami? (`every`).
6. `[1, 2, 3]` da kamida bittasi 3 ga tengmi? (`some`).
7. Talabalar massivi berilgan: faqat ismlari bo'lgan massivni oling.
8. Talabalarning faqat o'tganlarini (ball >= 55) tanlab, ismlarini chiqaring.

Javoblar

1) `arr.map(x => x + 10)`. 3) `arr.filter(x => x > 10)` → `[12, 20]`. 4) `arr.find(x => x % 2 !== 0)` → 7.

5) `true`. 6) `true`.

1.13. O'zingizni tekshiring

4-jadval. Qisqa test

№	Savol	Javob
1	`map` nima qaytaradi?	Yangi massiv (uzunligi teng)
2	`filter` nima qaytaradi?	Mos elementlardan yangi massiv
3	`forEach` nima qaytaradi?	`undefined`
4	`find` topa olmasa?	`undefined`
5	`some` va `every` farqi?	Kamida bittasi / hammasi
6	Metodlar asl massivni o'zgartiradimi?	`map/filter/find` — yo'q
7	Callback nima?	Metodga beriladigan funksiya
8	Zanjir qanday yoziladi?	`arr.filter(...).map(...)`

6 va undan ko'p to'g'ri javob — **2-darajaga** tayyorsiz.

1.14. Kichik lug'at

5-jadval. Lug'at

Atama	Sodda ma'nosi
Callback	Metodga argument sifatida berilgan funksiya
Predikat	`true/false` qaytaradigan callback (`filter`, `find`, `some` da)
Zanjir (chaining)	Metodlarni nuqta bilan ketma-ket ulash
Iteratsiya	Elementlarni birma-bir ko'rib chiqish

Xulosa

`map` o'zgartiradi, `filter` tanlaydi, `find` topadi, `some/every` savol beradi, `forEach` ish qiladi. Ular asl massivni o'zgartirmaydi va zanjirga ulanadi.

Chuqurroq o'rganmoqchimisiz?

Bu mavzuning o'rta va murakkab darajalari JavaScript adaptiv kursida — daraja testi sizga mos materialni o'zi tanlaydi, oxirida sertifikat.