7. Massivlar



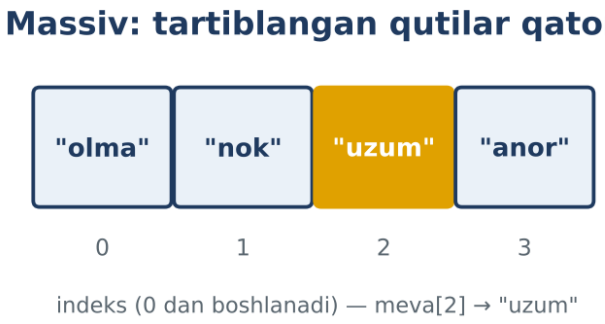
Massiv — **ko'p qiymatni tartib bilan saqlaydigan ro'yxat**. Bu dars massivni noldan, kundalik hayotdan misollar bilan tushuntiradi.

> **Bu darsda nimani o'rganasiz**
>
> Massiv nima va nima uchun kerak.
>
> Massiv yaratish va indeks bilan elementga murojaat.
>
> Element qo'shish va olib tashlash: `push`, `pop`, `shift`, `unshift`.
>
> `length` xossasi.
>
> Sikl bilan massivni aylanib chiqish.
>
> Massivda qidirish: `includes`, `indexOf`.

1.1. Massiv nima?

Tasavvur qiling, sizda 30 ta talaba ismi bor. 30 ta alohida o'zgaruvchi (`talaba1`, `talaba2`, ...) yaratish juda noqulay. Buning o'rniga **bitta ro'yxat** — massiv — yaratiladi.

Massiv — bu **raqamlangan qutilar qatori**. Har qutida bitta qiymat turadi, har qutining o'z raqami bor. Bu raqam **indeks** deyiladi va **0 dan** boshlanadi.



1-rasm. Massiv — indekslangan qutilar

Eng muhim qoida

Indeks **0 dan** boshlanadi, 1 dan emas! Birinchi element — `arr[0]`, ikkinchisi — `arr[1]`. Oxirgisining indeksi: `uzunlik - 1`.

1.2. Massiv yaratish va elementga murojaat

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

console.log(mevalar);
console.log(mevalar[0]);     // birinchi
console.log(mevalar[2]);     // uchinchi
console.log(mevalar[3]);     // oxirgi
console.log(mevalar[10]);    // yo'q joy: undefined
```

Natija:

```
[ 'olma', 'nok', 'uzum', 'anor' ]
olma
uzum
anor
undefined
```

Massivda **har qanday tur** bo'lishi mumkin: matn, son, mantiqiy qiymat, hatto aralash:

```
const sonlar = [10, 20, 30];
const aralash = ["Ali", 25, true, null];
const bosh = [];                    // bo'sh massiv
console.log(sonlar, aralash, bosh);
```

Natija:

```
[ 10, 20, 30 ] [ 'Ali', 25, true, null ] []
```

Elementni o'zgartirish

```
const ranglar = ["qizil", "yashil", "ko'k"];
ranglar[1] = "sariq";
console.log(ranglar);
```

Natija:

```
[ 'qizil', 'sariq', "ko'k" ]
```

> **const va massiv**
>
> `const` bilan e'lon qilingan massivning **elementlarini o'zgartirish mumkin**. Taqiqlangan narsa — o'zgaruvchiga butunlay **boshqa** massiv tayinlash (`ranglar = []` xato beradi).

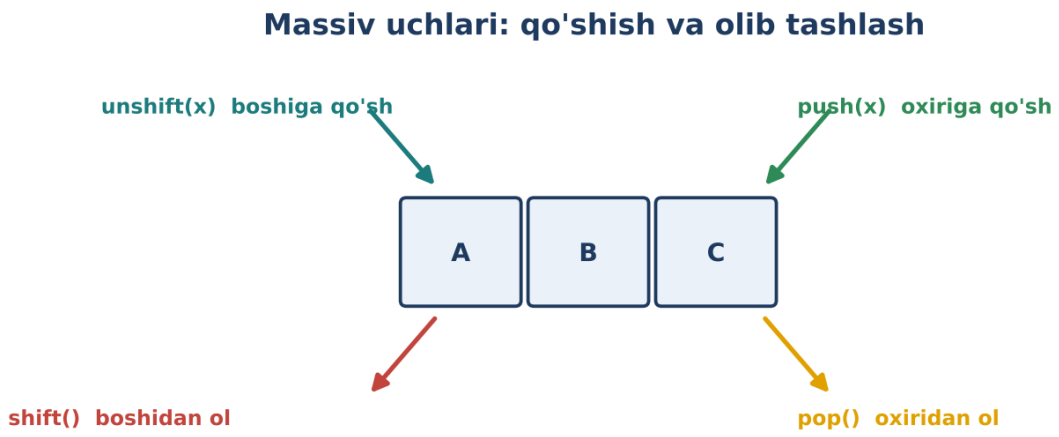
1.3. length — uzunlik

```
const mevalar = ["olma", "nok", "uzum"];
console.log(mevalar.length);                  // 3
console.log(mevalar[mevalar.length - 1]);  // oxirgi element
console.log([].length);                       // bo'sh massiv: 0
```

Natija:

```
3
uzum
0
```

1.4. Element qo'shish va olib tashlash



2-rasm. Massivning ikki uchi

1-jadval. To'rt asosiy metod

Metod	Nima qiladi	Qaytaradi
`push(x)`	**Oxiriga** qo'shadi	Yangi uzunlik

Metod	Nima qiladi	Qaytaradi
`pop()`	**Oxiridan** olib tashlaydi	Olib tashlangan element
`unshift(x)`	**Boshiga** qo'shadi	Yangi uzunlik
`shift()`	**Boshidan** olib tashlaydi	Olib tashlangan element

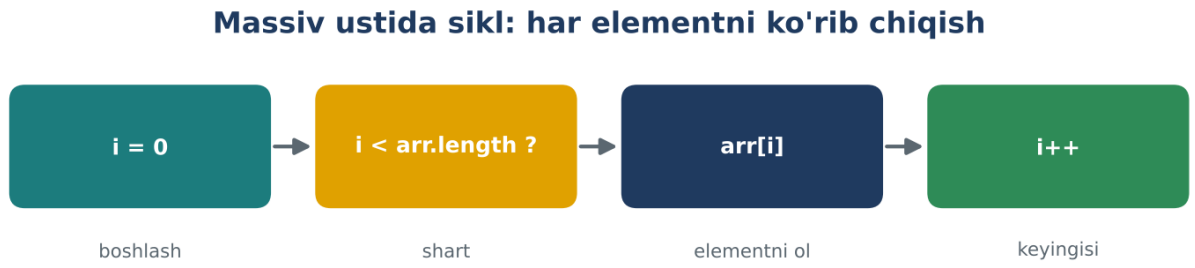
```
const royxat = ["B", "C"];

royxat.push("D");         // oxiriga
console.log(royxat);
royxat.unshift("A");      // boshiga
console.log(royxat);

const olingan1 = royxat.pop();     // oxiridan
const olingan2 = royxat.shift();   // boshidan
console.log(olingan1, olingan2, royxat);
```

Natija:

```
[ 'B', 'C', 'D' ]
[ 'A', 'B', 'C', 'D' ]
D A [ 'B', 'C' ]
```

1.5. Massivni aylanib chiqish

Massivning har bir elementi bilan ish qilish uchun sikldan foydalanamiz. Indeks 0 dan `length - 1` gacha o'zgaradi.



3-rasm. Massiv ustida for sikli

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

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

Natija:

```
0: olma
1: nok
2: uzum
```

Yanada qulay usul — `for...of`: indeks kerak bo'lmasa, to'g'ridan-to'g'ri elementlarni oladi.

```
const mevalar = ["olma", "nok", "uzum"];
for (const meva of mevalar) {
  console.log(meva);
}
```

Natija:

```
olma
nok
uzum
```

Yig'indi va o'rtacha

```
const baholar = [85, 70, 92, 60, 78];
let jami = 0;
for (const b of baholar) {
  jami += b;
}
console.log("Jami:", jami);
console.log("O'rtacha:", jami / baholar.length);
```

Natija:

```
Jami: 385
O'rtacha: 77
```

1.6. Massivda qidirish

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

console.log(mevalar.includes("nok"));     // bormi? true/false
console.log(mevalar.includes("anor"));
console.log(mevalar.indexOf("uzum"));     // qaysi indeksda?
console.log(mevalar.indexOf("anor"));     // topilmasa: -1
```

Natija:

```
true
false
2
-1
```

> **Eslab qoling**
>
> `indexOf` topilmasa **-1** qaytaradi. Shuning uchun "bormi?" degan savol uchun `includes` qulayroq.

1.6.1. Massiv va funksiya

Massivni funksiyaga **argument** sifatida berish mumkin. Funksiya natijani `return` bilan qaytaradi:

```
function yigindi(arr) {
  let j = 0;
  for (const x of arr) j += x;
  return j;
}
function ortacha(arr) {
  return arr.length === 0 ? 0 : yigindi(arr) / arr.length;
}

const baholar = [90, 75, 60, 85];
console.log(yigindi(baholar));
console.log(ortacha(baholar));
console.log(ortacha([]));          // bo'sh massivda 0 ga bo'lmaslik uchun tek
```

Natija:

```
310
77.5
0
```

1.6.2. Massiv bilan ishlash: tezkor shpargalka

Tezkor shpargalka

Men xohlayman...	Yozaman	Natija
Birinchi elementni olish	`arr[0]`	Birinchi qiymat
Oxirgi elementni olish	`arr[arr.length - 1]`	Oxirgi qiymat
Elementlar sonini bilish	`arr.length`	Son
Oxiriga qo'shish	`arr.push(x)`	Massiv o'sadi
Oxiridan olib tashlash	`arr.pop()`	Olingan element
Bor-yo'qligini tekshirish	`arr.includes(x)`	`true` / `false`
Joyini topish	`arr.indexOf(x)`	Indeks yoki -1
Hammasini ko'rib chiqish	`for (const x of arr)`	Har bir element

Massivni qo'lda o'zgartirish: bosqichma-bosqich

Quyidagi dastur ro'yxatga element qo'shadi, o'chiradi va har qadamda holatni ko'rsatadi:

```
const guruh = ["Ali", "Vali"];
console.log("1:", guruh);

guruh.push("Zara");
console.log("2: push", guruh);

guruh.unshift("Bobur");
```

```
console.log("3: unshift", guruh);

guruh.pop();
console.log("4: pop", guruh);

guruh.shift();
console.log("5: shift", guruh, "uzunlik:", guruh.length);
```

Natija:

```
1: [ 'Ali', 'Vali' ]
2: push [ 'Ali', 'Vali', 'Zara' ]
3: unshift [ 'Bobur', 'Ali', 'Vali', 'Zara' ]
4: pop [ 'Bobur', 'Ali', 'Vali' ]
5: shift [ 'Ali', 'Vali' ] uzunlik: 2
```

Bo'sh massivni tekshirish

```
const bosh = [];
const toliq = [1];

if (bosh.length === 0) console.log("bosh massiv");
console.log(toliq.length > 0 ? "elementlar bor" : "bo'sh");
console.log(bosh.pop());           // bo'sh massivdan pop ? undefined
```

Natija:

```
bosh massiv
elementlar bor
undefined
```

> **Maslahat**
>
> Massiv bilan ishlashdan oldin uning bo'shligini tekshiring: bo'sh massivda `arr[0]` yoki `pop()` natijasi `undefined` bo'ladi, bu keyingi hisob-kitoblarda `NaN` ga olib kelishi mumkin.

1.7. Kod izi

```
const arr = [3, 6, 9];
arr.push(12);
arr[0] = 1;
arr.pop();
console.log(arr, arr.length);
```

2-jadval. Kod izi

Qadam	Massiv holati	Izoh
1	`[3, 6, 9]`	Boshlang'ich
2	`[3, 6, 9, 12]`	`push` oxiriga qo'shdi
3	`[1, 6, 9, 12]`	0-indeks o'zgardi

Qadam	Massiv holati	Izoh
4	`[1, 6, 9]`	pop oxirgisini oldi
5	`[1, 6, 9]`, uzunlik 3	Chop etildi

1.8. Mini loyihalar

massiv_loyiha.js

```
// 1) Xaridlar ro'yxati
const xaridlar = [];
xaridlar.push("non");
xaridlar.push("sut");
xaridlar.push("tuxum");
console.log(`Jami ${xaridlar.length} ta: ${xaridlar.join(", ")}`);

// 2) Eng katta son
const sonlar = [12, 45, 7, 89, 23];
let eng = sonlar[0];
for (const s of sonlar) {
  if (s > eng) eng = s;
}
console.log("Eng katta:", eng);

// 3) Juft sonlarni sanash
let juft = 0;
for (const s of sonlar) {
  if (s % 2 === 0) juft++;
}
console.log("Juftlar soni:", juft);

// 4) Teskari tartibda chop etish
const kun = ["Du", "Se", "Chor", "Pay"];
for (let i = kun.length - 1; i >= 0; i--) {
  console.log(kun[i]);
}
```

Natija:

```
Jami 3 ta: non, sut, tuxum
Eng katta: 89
Juftlar soni: 1
Pay
Chor
Se
Du
```

1.9. Eng ko'p uchraydigan xatolar

3-jadval. Tipik xatolar

Xato	Sababi	To'g'ri yozuv

`arr[arr.length]` ni oxirgi deb o'ylash	Oxirgi indeks `length - 1`	`arr[arr.length - 1]`
Sikl `i <= arr.length` bilan	Oxirida `undefined` olinadi	`i < arr.length`
Indeksni 1 dan boshlash	Indeks 0 dan boshlanadi	Birinchi: `arr[0]`
`push` natijasini massiv deb o'ylash	`push` yangi **uzunlik** qaytaradi	`arr.push(x)` ni alohida yozing
`const` massivni o'zgartirib bo'lmaydi deb o'ylash	Faqat qayta tayinlash taqiqlangan	Elementlar o'zgartirilishi mumkin

1.10. Mashqlar

1. Sevimli 5 ta filmingiz ro'yxatini massivga yozing va uchinchisini chop eting.
2. `[4, 8, 15, 16, 23, 42]` massivning oxirgi elementini `length` yordamida oling.
3. Bo'sh massivga 3 ta ism `push` qiling, so'ng birinchisini `shift` bilan olib tashlang.
4. `[2, 4, 6, 8]` massiv elementlari yig'indisini sikl bilan toping.
5. Massivdagi eng kichik sonni toping.
6. Massivdagi elementlarni teskari tartibda chop eting.
7. Massivda `"Ali"` bor-yo'qligini `includes` bilan tekshiring.
8. Massivdagi 50 dan katta sonlar sonini sanang.
9. Foydalanuvchidan kelgan ismlar ro'yxatida takrorlanuvchi ismni topuvchi funksiya yozing.
10. `[1, 2, 3, 4, 5]` dan faqat toq sonlardan iborat yangi massiv yig'ing (`push` bilan).

Javoblar

2) `arr[arr.length - 1]` → 42. 4) 20. 9) yangi bo'sh massiv oching, sikl ichida `if (x % 2 !== 0) yangi.push(x)` → `[1, 3, 5]`.

1.11. O'zingizni tekshiring

4-jadval. Qisqa test

№	Savol	Javob
1	Massiv indeksi nechadan boshlanadi?	0 dan
2	Oxirgi element indeksi?	`length - 1`
3	Oxiriga qo'shish metodi?	`push`
4	Boshidan olib tashlash metodi?	`shift`

№	Savol	Javob
5	Mavjud bo'lmagan indeks natijasi?	`undefined`
6	`indexOf` topa olmasa?	`-1`
7	Massiv elementlarini `for...of` bilan olish?	`for (const x of arr)`
8	Bo'sh massivning uzunligi?	0

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

1.12. Kichik lug'at

5-jadval. Lug'at

Atama	Sodda ma'nosi
Massiv (array)	Tartiblangan qiymatlar ro'yxati
Element	Massivdagi bitta qiymat
Indeks	Elementning tartib raqami (0 dan boshlanadi)
length	Elementlar soni
Iteratsiya	Elementlarni birma-bir ko'rib chiqish

Xulosa

Massiv — indekslangan ro'yxat. Indeks 0 dan boshlanadi. `push`/`pop` oxirida, `unshift`/`shift` boshida ishlaydi. Massivni sikl bilan aylanib, yig'indi, eng katta qiymat kabi masalalarni yechamiz.

Chuqurroq o'rganmoqchimisiz?

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