

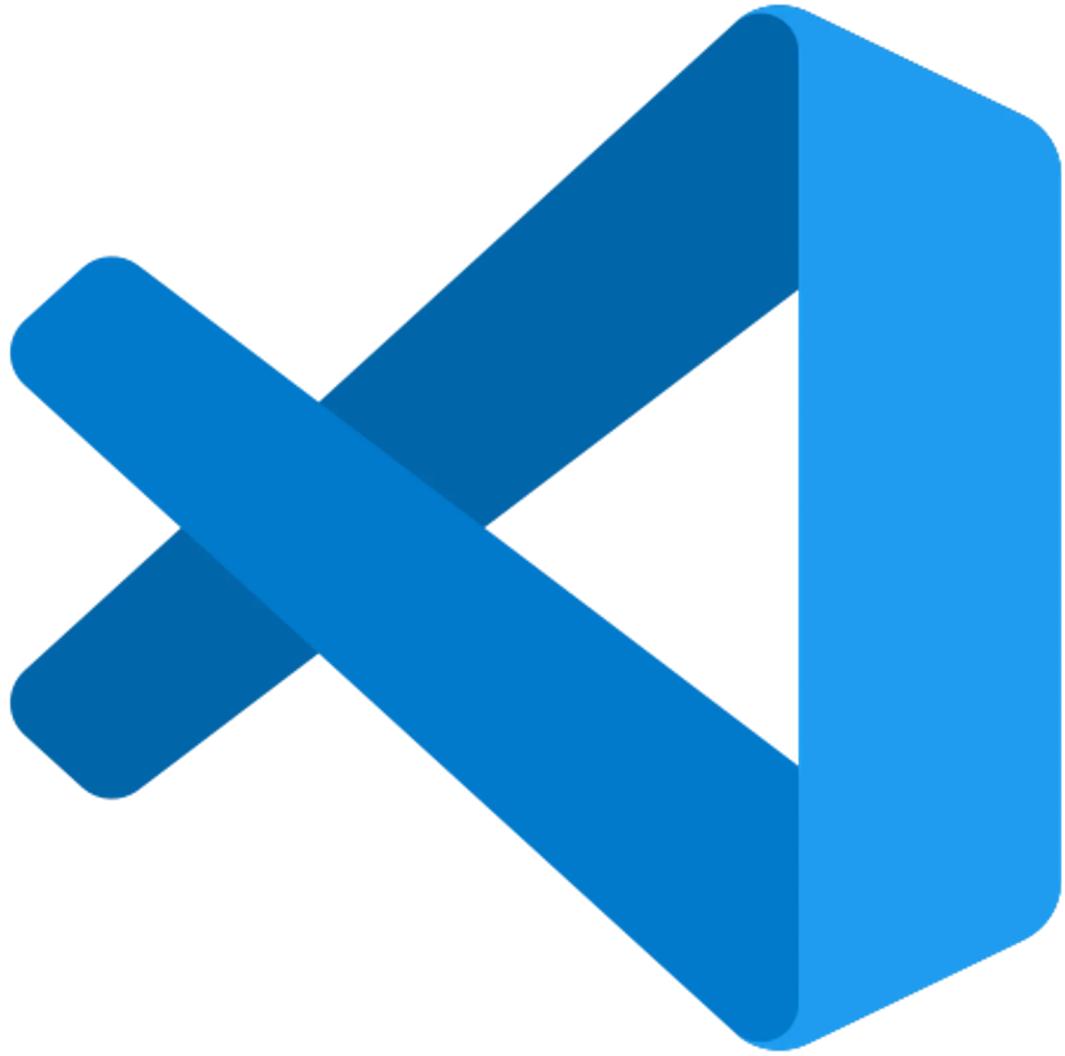


JavaScript

▼ JavaScript o'zi nima?

JavaScript - bu dasturlash tili bo'lib, asosan veb-sahifalarni dinamik va interaktiv qilish uchun ishlataladi.

Bizga nimalar kerak





Node.js: JavaScript kodini server tomonda ishlatalish imkonini beradi. Node.js yordamida siz JavaScript dasturlarini ishga tushirishingiz va test qilishingiz mumkin.

▼ Web sahifamizga JavaScriptni ulash

Web sahifasiga JavaScript ulashning bir nechta usuli bor.

1. HTML ichida (Inline Script)

Siz JavaScriptni bevosita `<script>` tegi ichida yozishingiz mumkin:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        <title>JavaScriptni ulash</title>
</head>
<body>
    <h1>JavaScriptni HTML ga ulash</h1>
```

```
<p>Bu sahifada JavaScript ishlamoqda.</p>

<script>
    console.log("Salom, dunyo!");
    alert("JavaScript ishladi!");
</script>
</body>
</html>
```

2. Tashqi Fayl orqali (External Script)

JavaScript kodini alohida `.js` faylga yozib, uni sahifaga ulashingiz mumkin.

1-qadam: JavaScript kodini alohida faylga yozing:

`script.js` fayli:

```
console.log("Tashqi fayldan salom!");
alert("Tashqi fayldan JavaScript ishladi!");
```

2-qadam: HTML faylida ulang:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        <title>Tashqi Scriptni ulash</title>
</head>
<body>
    <h1>Tashqi JavaScript fayli ulanishi</h1>
    <p>Tashqi fayldagi kod ishlayapti.</p>

    <script src="script.js"></script>
</body>
</html>
```

3. Body ichida yoki `<head>` da

- Agar `<head>` ichida script ulasangiz, sahifa to'liq yuklanib bo'lishini kutish uchun `defer` yoki `async` atributidan foydalanishingiz kerak:

```
<script src="script.js" defer></script>
```

Yoki `body` oxirida ulashingiz mumkin:

```
<body>
  ...
<script src="script.js"></script>
</body>
```

Qaysi usulni tanlash kerak?

- Tez ishlash uchun:** Tashqi fayldan foydalanish va `<script>` tegi `<body>` oxirida bo'lishi kerak.
- Kichik loyihalar uchun:** Inline script ishlatishtingiz mumkin.
- Katta loyihalar uchun:** Tashqi JavaScript fayllarini ulash tavsiya qilinadi, chunki ular kodni tartibli va boshqarishni osonlashtiradi.

▼ var,let,const

JavaScriptda `var`, `let`, va `const` o'zgaruvchilarni e'lon qilishda ishlataladi.

1. `var`

`var` - eski usulda o'zgaruvchi e'lon qilish uchun ishlataladi (ES5 dan oldingi davr).

2. `let`

`let` - o'zgaruvchilarni e'lon qilish uchun tavsiya etiladigan usul (ES6 dan boshlab)

3. `const`

`const` - o'zgarmas (constant) qiymatni saqlash uchun ishlataladi.

`var`, `let`, `const` taqqoslash

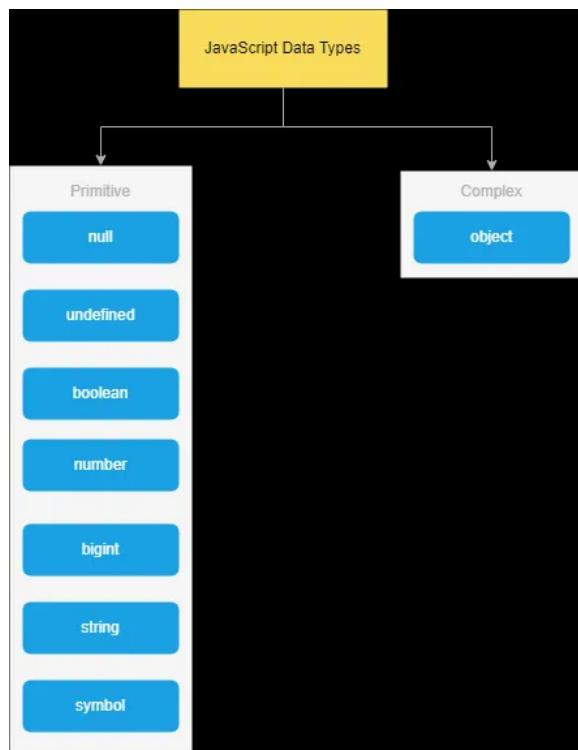
Xususiyatlar	<code>var</code>	<code>let</code>	<code>const</code>
Doirasi (Scope)	Funksiya	Blok	Blok
Hoisting	Ha, qiymati <code>undefined</code>	Ha, lekin TDZ bor	Ha, lekin TDZ bor
Qayta e'lon qilish	Ha	Yo'q	Yo'q
Qiymatni o'zgartirish	Ha	Ha	Yo'q
Blokdan tashqarida mavjudligi	Ha	Yo'q	Yo'q

Qachon qaysi biridan foydalanish kerak?

1. `var` : Eski kodni qo'llab-quvvatlash uchun ishlating.
2. `let` : O'zgaruvchan qiymatlar uchun ishlating.
3. `const` : O'zgarmas qiymatlar yoki obyektlar uchun ishlating (afzal).

▼ JavaScriptda ma'lumot turlari (data types)

JavaScriptda ma'lumot turlari (data types) ikkita asosiy turga bo'linadi:
Primitive (asosiy) va **Non-primitive (murakkab)** ma'lumot turlari.



1. Primitive Data Types (Asosiy ma'lumot turlari)

Primitive ma'lumotlar JavaScriptning oddiy turlaridir. Ular o'zgarmas (immutable) va faqat bitta qiymatni saqlaydi.

Ma'lumot turi	Tavsifi	Misol
Number	Raqamli qiymatlar, butun yoki qoldiqqli	<code>42</code> , <code>3.14</code> , <code>-7</code>
BigInt	Juda katta butun sonlar	<code>12345678901234567890n</code>
String	Matnlar, <code>"</code> , <code>'</code> yoki `` (backtick) ichida	<code>"Salom"</code> , <code>'JavaScript'</code>
Boolean	True/False qiymatlar	<code>true</code> , <code>false</code>
Undefined	Qiymat belgilanmagan	<code>let x; // undefined</code>
Null	Bo'sh yoki qiymatsiz	<code>let y = null;</code>
Symbol	Unikal identifikator yaratadi	<code>Symbol('id')</code>

2. Non-Primitive Data Types (Murakkab ma'lumot turlari)

Non-primitive ma'lumot turlari murakkab strukturalarni ifodalaydi. Ular qiymatni emas, balki ma'lumot joylashgan manzilni (reference) saqlaydi.

Ma'lumot turi	Tavsifi	Misol
Object	Kalit-qiymat juftligi bo'lgan tuzilmalar	{ name: "Ali", age: 25 }
Array	Tartibli elementlar ro'yxati	[1, 2, 3, 4, 5]
Function	Kod blokini o'zida saqlovchi obyekt	function greet() {}

Ma'lumot turini tekshirish (`typeof` operatori)

```
console.log(typeof 42);           // "number"
console.log(typeof "Hello");      // "string"
console.log(typeof true);         // "boolean"
console.log(typeof undefined);    // "undefined"
console.log(typeof null);         // "object" (bu eski xato,
console.log(typeof Symbol("id")); // "symbol"
console.log(typeof { a: 1 });      // "object"
console.log(typeof [1, 2, 3]);     // "object" (Array ham obje
console.log(typeof function(){}); // "function"
```

▼ prompt, alert, console.log

JavaScriptda `prompt`, `alert`, va `console.log` foydalanuvchi bilan muloqot qilish va ma'lumotni namoyish qilish uchun ishlatiladi.

1. `prompt`

`prompt` - foydalanuvchidan ma'lumot (kirish) olish uchun ishlatiladi.

Xususiyatlari:

- Foydalanuvchiga matnli maydon ko'rsatadi.
- Foydalanuvchi kiritgan qiymatni qaytaradi (har doim **string** sifatida).
- Agar foydalanuvchi hech narsa kiritmasa, `null` qaytaradi.

Sintaksis:

```
let userInput = prompt(message, defaultValue);
```

- `message` - foydalanuvchiga ko'rsatiladigan matn.
- `defaultValue` (*ixtiyoriy*) - kiritish maydonidagi standart qiymat.

2. `alert`

`alert` - foydalanuvchiga oddiy ogohlantirish oynasi ko'rsatadi.

Xususiyatlari:

- Foydalanuvchiga faqat xabar ko'rsatiladi.
- Foydalanuvchi oynani yopgunga qadar boshqa hech narsa qilish imkoniyati yo'q.
- Hech qanday qiymat qaytarmaydi.

Sintaksis:

```
alert(message);
```

`message` - foydalanuvchiga ko'rsatiladigan matn.

3. `console.log`

`console.log` - dasturchi uchun ma'lumotni konsolda chiqarish uchun ishlataladi. Bu ko'pincha kodni tekshirish va debugging maqsadida ishlataladi.

Xususiyatlari:

- Konsolda qiymat yoki ma'lumotni namoyish qiladi.
- Har qanday ma'lumot turini chiqarishi mumkin (raqam, string, obyekt, massiv, va hokazo).
- Foydalanuvchi uchun ko'rinxaydi, faqat brauzerning **console** bo'limida ko'rinxadi (Developer Tools).

Sintaksis:

```
console.log(message);
```

`message` - konsolda ko'rsatiladigan ma'lumot.

▼ Math

JavaScriptda `Math` obyekti matematik operatsiyalarni bajarish uchun ishlataladi. Bu obyekt ichida ko'plab funksiyalar va konstantalar mavjud. `Math` obyektidan foydalanish uchun uni to'g'ridan-to'g'ri chaqirish mumkin (hech qanday e'lon qilish talab qilinmaydi).

Math obyektining asosiy funksiyalari

1. Raqamlarni yaxlitlash

- `Math.round()`: Eng yaqin butun songa yaxlitlaydi.

```
console.log(Math.round(4.5)); // 5  
console.log(Math.round(4.4)); // 4
```

- `Math.ceil()`: Sonni yuqoriga qarab butun songa yaxlitlaydi.

```
console.log(Math.ceil(4.1)); // 5  
console.log(Math.ceil(-4.1)); // -4
```

- `Math.floor()`: Sonni pastga qarab butun songa yaxlitlaydi.

```
console.log(Math.floor(4.9)); // 4  
console.log(Math.floor(-4.9)); // -5
```

- `Math.trunc()`: Butun qismni ajratib oladi (kasrni olib tashlaydi).

```
console.log(Math.trunc(4.9)); // 4  
console.log(Math.trunc(-4.9)); // -4
```

2. Maksimum va minimum qiymatlarni topish

- `Math.max()`: Berilgan qiymatlar orasidan maksimalini qaytaradi.

```
console.log(Math.max(10, 20, 30)); // 30
```

- `Math.min()`: Berilgan qiymatlar orasidan minimalini qaytaradi.

```
console.log(Math.min(10, 20, 30)); // 10
```

3. Tasodifiy sonlar

- `Math.random()`: 0 va 1 oralig'idagi (1 kirmaydi) tasodifiy kasr sonni qaytaradi.

```
console.log(Math.random()); // Masalan: 0.123456789
```

- Tasodifiy butun son yaratish:

```
// 1 dan 10 gacha tasodifiy butun son
let randomInt = Math.floor(Math.random() * 10) + 1;
console.log(randomInt);
```

4. Eksponent va ildizlar

- `Math.pow(base, exponent)`: Sonning darajasini hisoblaydi.

```
console.log(Math.pow(2, 3)); // 8
```

- `Math.sqrt()`: Kvadrat ildizni hisoblaydi.

```
console.log(Math.sqrt(16)); // 4
```

- `Math.cbrt()`: Kub ildizni hisoblaydi.

```
console.log(Math.cbrt(27)); // 3
```

5. Mutlaq qiymat va belgilar

- `Math.abs()`: Sonning mutlaq qiymatini qaytaradi (musbatga aylantiradi).

```
console.log(Math.abs(-10)); // 10
```

▼ 1-topshiriq

1. **3** ta o'zgaruvchi berilgan. **0**'zgaruvchilar qiymati **31, 18**
2. Konsolga **50** va **10** ni yangi qatorlarda yozing
3. Klaviaturadan foydalanuvchi ismini so'rang. Console ga '**A**'
4. Klaviaturadan foydalanuvchining sevimli futbol komandasini
5. Klaviaturadan kiritilgan raqamning oldingi va keyingi qiymatni hisoblang.
6. Do'kondan n kg nok, s kg sabzi, b kg bodring sotib olib uchqiling.
7. **A** va **B** sonlar berilgan ularning o'rtacha qiymatini hisoblansin.
8. **A** va **B** sonlar berilgan ularning o'rta geometric qiymatini hisoblansin.
9. Kvadrat tomoni berilgan. Uning perimetri hisoblansin. **P =**
10. Kvadratning tomoni berilgan uning yuzasi topilsin.
11. To'g'rito'rtburchakning tomonlari **A** va **B** berilgan. Uning diametri **D** hisoblang.
12. Diametr **D** berilgan. Uning uzunligi **R** ni hisoblang. Pi ni **3.14** qo'shing.
13. Kub tomon a berilgan. Kubning hajmi topilsin va uning yuzasi.
14. Parallipedning tomonlari **a, b, c** berilgan. Uning hajmini hisoblang.
15. Aylananing radiusi **R** berilgan. Uning uzunligi **L** va yuzasi.
16. **A** va **B** qiymat berilgan. **0**'rta arifmetikini toping.
17. **A** va **B** berilgan. ularning o'rta geometriyasini toping.
18. **A** va **B** berilgan. ularning yig'indisi, ayirmasi, ko'paytma, hisoblang.
19. Uchburchakning a va b tomoni berilgan. Uchburchakning **3-**
20. Aylaning yuzi **S** berilgan. Diametri **D** va uzunligi **L** topilganda.
21. **X** va **Y** nuqtalar berilgan. ularning orasidagi masofa topilganda.
22. **A** va **B** lar berilgan bo'lib, ularning qiymatlari almashti.
23. **A, B, C** berilgan. **A** bilan **B**, **B** bilan **C**, **C** bilan **A** ning qisqashi.

24. **X** ning qiymati berilgan holatda. **Y** topilsin
25. **X** ning qiymati berilgan holatda **Y** topilsin
26. **X** kg konfet **A** so'm bo'lsa, **1** kg va **Y** kg konfet qancha tur

▼ if, else

JavaScriptda `if` va `else` shartli operatorlari yordamida turli shartlarga qarab kodni boshqarish mumkin. Bu operatorlar ma'lum bir shart (condition) bajarilsa yoki bajarilmasa, qanday kod bajarilishini belgilaydi.

1. `if`

`if` operatori shartni tekshiradi va agar shart `true` (rost) bo'lsa, tegishli kodni bajaradi.

Sintaksis:

```
if (condition) {  
    // Shart rost bo'lsa, bu kod bajariladi  
}
```

2. `if-else`

`else` blok shart yolg'on bo'lsa (`false`) bajariladigan kodni belgilaydi.

Sintaksis:

```
if (condition) {  
    // Shart rost bo'lsa, bu kod bajariladi  
} else {  
    // Shart yolg'on bo'lsa, bu kod bajariladi  
}
```

3. `if-else if-else`

Bir nechta shartlarni ketma-ket tekshirish uchun `else if` ishlataladi.

Sintaksis:

```
if (condition1) {  
    // Shart1 rost bo'lsa, bu kod bajariladi  
} else if (condition2) {  
    // Shart2 rost bo'lsa, bu kod bajariladi  
} else {  
    // Yuqoridagi barcha shartlar yolg'on bo'lsa, bu kod bajariladi  
}
```

▼ Ternary operator

Ternary operator `if-else` blokiga muqobil hisoblanadi va bir qatorda yoziladi.
Ternary operatorni ishlatish sintaksisi sodda va tez.

Ternary operator sintaksisi:

```
javascript condition ? expressionIfTrue : expressionIfFalse;
```

- `condition` - Tekshiriladigan shart.
- `expressionIfTrue` - Shart `true` bo'lsa bajariladigan qiymat yoki amal.
- `expressionIfFalse` - Shart `false` bo'lsa bajariladigan qiymat yoki amal.

1.Bir nechta ternary operatorlar

Ternary operator ichma-ich ishlatilishi mumkin:

```
let score = 85;  
let grade = (score >= 90) ? "A" :  
            (score >= 80) ? "B" :  
            (score >= 70) ? "C" :  
            (score >= 60) ? "D" : "F";  
console.log(grade); // "B"
```

Ternary operatorning afzalliklari:

1. **Qisqalik**: Kodni qisqartiradi va o'qilishi osonlashtiradi.
2. **Oddiy shartlar uchun ideal**: Oddiy `if-else` bloklar o'rniiga ishlatalish mumkin.

Ternary operatorni noto'g'ri ishlatalishdan ehtiyyot bo'ling

- Juda uzun va murakkab shartlar uchun ishlatalish tavsiya etilmaydi.
- Haddan tashqari ko'p ichma-ich ternary operatorlar kodni chalkashtirishi mumkin.

▼ switch, case

JavaScriptda `switch` operatori bir shartni bir nechta qiymatga solishtirib, tegishli kodni bajarish uchun ishlataladi. Bu operator bir nechta `if-else` shartlarini sodda va o'qilishi oson shaklda yozishga yordam beradi.

`switch` Sintaksisi:

```
switch (expression) {
    case value1:
        // Agar expression value1 ga teng bo'lsa, bu kod bajariladi
        break;
    case value2:
        // Agar expression value2 ga teng bo'lsa, bu kod bajariladi
        break;
    default:
        // Yuqoridagi shartlarning hech biri bajarilmasa, bu kod ishlaydi
}
```

- `expression`: Tekshiriladigan qiymat (masalan, o'zgaruvchi yoki ifoda).

- `case` : Har bir holat (qiymat) uchun kod bloki.
 - `break` : Kodni to'xtatadi va `switch` operatoridan chiqadi.
 - `default` : Hech bir `case` mos kelmasa bajariladigan kod.
-

`switch` Misollar

1. Oddiy `switch-case` misoli

```
let day = 3;

switch (day) {
    case 1:
        console.log("Dushanba");
        break;
    case 2:
        console.log("Seshanba");
        break;
    case 3:
        console.log("Chorshanba");
        break;
    case 4:
        console.log("Payshanba");
        break;
    case 5:
        console.log("Juma");
        break;
    default:
        console.log("Dam olish kuni");
}
```

Agar `day` qiymati 3 bo'lsa, `"Chorshanba"` chiqadi.

2. Bir nechta `case` bir xil kodni ishlatsi

Bir nechta `case` qiymatlari uchun bir xil kodni ishlatsish mumkin:

```

let color = "red";

switch (color) {
    case "red":
    case "blue":
    case "green":
        console.log("Bu asosiy rang.");
        break;
    default:
        console.log("Bu asosiy rang emas.");
}

```

Agar `color` qiymati `"red"`, `"blue"`, yoki `"green"` bo'lsa, `"Bu asosiy rang."` chiqadi.

5. Ifoda (`expression`) ishlatalish

`switch` ichida murakkab ifoda ham ishlatalish mumkin:

```

let x = 10;

switch (true) {
    case (x > 0 && x <= 10):
        console.log("Son 1 dan 10 gacha.");
        break;
    case (x > 10 && x <= 20):
        console.log("Son 11 dan 20 gacha.");
        break;
    default:
        console.log("Son 0 yoki 20 dan katta.");
}

```

`switch` va `if-else` taqqoslash

- `if-else` murakkab shartlar uchun mos.

- `switch` bir qiymatni ko'p holatlar bilan solishtirishda qulay.

Agar shartlar murakkab bo'lsa, `if-else` ishlatalish tavsiya etiladi. Ammo ko'p qiymatlarni tekshirish kerak bo'lsa, `switch-case` ishlatalish samaraliroq! 😊

▼ 2-topshiriq

2-topshiriq

1. Eldor ning yoshi **X**, Aziza ning yoshi – **Y**. Ularning o'rtaci
2. **A** soni berilgan. Quyidagi gapni to'g'ri yoki notog'riliqi
3. **A** soni berilgan. Quyidagi gapni to'g'ri yoki notog'riliqi
4. **A** soni berilgan. Quyidagi gapni to'g'ri yoki notog'riliqi
5. **A** va **B** sonlar berilgan. Quyidagi gapni tekshiring: "**A > 2**"
6. **A**, **B** va **C** sonlar beriglan. Quyidagi ifoda to'g'riliini isb
7. **A** va **B** sonlar berilgan. Quyidagi ifodani tekshiring: "**A < B**"
8. **A** soni berilgan. Agarda musbat bo'lsa qiymati birga oshir
9. **A** soni berilgan. Agarda musbat bo'lsa qiymati birga oshir
10. **A** soni berilgan. Agarda musbat bo'lsa qiymati ikki baroba
11. **A** va **B** sonlar berilgan. Ularning orasidan kattasini hisob
12. **A**, **B** va **C** sonlar berilgan. Ularning orasidan eng kichkina
13. **N** soni berilgan. **N** ga to'g'ri keluvchi hafta kunini chop
14. **M** soni berilgan. **M** ga to'g'ri keluvchi oy nomini chop eti
15. Bokschining vazni berilgan. Vazn qiymatiga qarab. Yengil

▼ `for` tsikli, `break`, va `continue`

JavaScriptda `for` tsikli takrorlanuvchi amallarni bajarish uchun ishlataladi. `break` va `continue` operatorlari esa tsiklning ishini boshqarishda ishlataladi.

1. `for` tsikli

`for` tsikli ma'lum bir shart to'g'ri bo'lgunga qadar kodni takrorlaydi.

Sintaksis:

```
for (initialization; condition; increment/decrement) {  
    // Takrorlanadigan kod  
}
```

- **initialization**: Boshlang'ich qiymat (bir marta ishlaydi).
- **condition**: Shart (har safar tsikl boshlanishida tekshiriladi).
- **increment/decrement**: Har bir iteratsiyadan keyin bajariladi.

2. **break** operatori

break tsiklni to'xtatadi va undan chiqib ketadi.

Misol: **break** ishlatalish

```
for (let i = 0; i < 10; i++) {  
    if (i === 5) {  
        break; // 5 ga yetganda tsikl to'xtaydi  
    }  
    console.log("Qiymat:", i);  
}  
// Natija:  
// Qiymat: 0  
// Qiymat: 1  
// Qiymat: 2  
// Qiymat: 3  
// Qiymat: 4
```

3. **continue** operatori

continue tsiklning hozirgi iteratsiyasini tugatib, keyingi iteratsiyaga o'tadi.

Misol: **continue** ishlatalish

```

for (let i = 0; i < 10; i++) {
    if (i % 2 === 0) {
        continue; // Juft sonlarni o'tkazib yuboradi
    }
    console.log("Toq son:", i);
}
// Natija:
// Toq son: 1
// Toq son: 3
// Toq son: 5
// Toq son: 7
// Toq son: 9

```

4. Ichma-ich (nested) for tsikllar

Bir tsikl ichida yana bir tsikl bo'lishi mumkin.

Misol:

```

for (let i = 1; i <= 3; i++) {
    for (let j = 1; j <= 3; j++) {
        console.log(`i = ${i}, j = ${j}`);
    }
}
// Natija:
// i = 1, j = 1
// i = 1, j = 2
// i = 1, j = 3
// i = 2, j = 1
// i = 2, j = 2
// i = 2, j = 3
// i = 3, j = 1
// i = 3, j = 2
// i = 3, j = 3

```

5. Shartsiz `for` tsikli

Agar shart va o'sish/dekremet aniqlanmasa, tsikl cheksiz davom etadi.

Bunday holatda `break` ishlatalishi kerak.

Misol: Cheksiz tsikl va `break`

```
let i = 0;
for (;;) {
    console.log("Qiymat:", i);
    i++;
    if (i === 5) {
        break; // Tsiklni 5 ga yetganda to'xtatadi
    }
}
// Natija:
// Qiymat: 0
// Qiymat: 1
// Qiymat: 2
// Qiymat: 3
// Qiymat: 4
```

6. `for` tsiklni boshqa operatorlar bilan birga ishlatalish

Misol: `continue` va `break` birga

```
for (let i = 1; i <= 10; i++) {
    if (i % 2 === 0) {
        continue; // Juft sonlarni o'tkazib yuboradi
    }
    if (i > 7) {
        break; // i 7 dan oshganda tsiklni to'xtatadi
    }
}
```

```

        console.log("Toq son:", i);
    }
    // Natija:
    // Toq son: 1
    // Toq son: 3
    // Toq son: 5
    // Toq son: 7

```

Asosiy farqlar: **break** va **continue**

Operator	Vazifasi	Qachon ishlataladi
break	Tsiklni to'xtatadi	Shart bajarilganda tsiklni yakunlash uchun
continue	Hozirgi iteratsiyani o'tkazib yuboradi	Beglangan shartlarda iteratsiyani o'tkazib yuborish uchun

Bu operatorlar yordamida tsikllarni yanada aniq boshqarish mumkin. 😊

Misollar:

- n gacha bo'lgan sonlar yigindisini chiqaring
- n bo'lgan sonlarning ko'paytimasi
- n bo'lgan sonlarning juftlarini ko'paytimasi
- a va b orasidagigilar ko'paytimasi
- birdan 10 karagacha karra jadvalini chiqaring.

▼ while

JavaScriptda **while** tsikli biror shart to'g'ri bo'lsa, kodni takrorlash uchun ishlataladi. Tsikl davom etishi uchun shart **true** bo'lishi kerak. **while** tsikli shartni tekshirgandan keyin ishlaydi, ya'ni **shart oldin tekshiriladi**. Agar shart dastlab **false** bo'lsa, kod bir marta ham ishlamaydi.

while tsikli sintaksisi:

```
while (condition) {  
    // Takrorlanadigan kod  
}
```

- `condition`: Shart (agar true bo'lsa, tsikl davom etadi).
- Kod blokidagi amallar `condition` true bo'lgunga qadar takrorlanadi.

Oddiy `while` tsikli misoli:

```
let i = 0;  
  
while (i < 5) {  
    console.log(i); // i qiymatini chiqaradi  
    i++; // i ni oshiradi  
}  
// Natija:  
// 0  
// 1  
// 2  
// 3  
// 4
```

do...while tsikli

Agar `while` tsiklida shart avval tekshirilsa, `do...while` tsiklida shart oxirida tekshiriladi. Bu demak, `do...while` tsikli kamida bitta marta ishlaydi, hatto shart **false** bo'lsa ham.

Sintaksis:

```
do {  
    // Takrorlanadigan kod  
} while (condition);
```

Misol: `do...while` tsikli:

```
let i = 0;

do {
    console.log(i); // i qiymatini chiqaradi
    i++;
} while (i < 5);
// Natija:
// 0
// 1
// 2
// 3
// 4
```

`while` va `do...while` taqqoslash

Tsikl tipi	Sintaksis	Shart tekshiruvni	Misol
<code>while</code>	<code>while (condition)</code>	Tsikl boshlanishida tekshiriladi	Tsikl shart true bo'lgunga qadar ishlaydi.
<code>do...while</code>	<code>do { ... } while (condition)</code>	Tsikl oxirida tekshiriladi	Kamida bitta marta ishlaydi, hatto shart false bo'lsa ham.

`while` tsikli takrorlanuvchi amallarni bajarish uchun juda foydali, ayniqsa tsiklini faqat shart bajarilguncha ishlatish kerak bo'lса. `do...while` tsikli esa kamida bir marta bajariladigan kodni yozish uchun ishlatiladi. 😊

▼ 3-topshiriq

for siki operatoriga oid masalalar

For1. k va n butun sonlari berilgan ($n > 0$): k sonini n marta chiqaruvchi programma tuzilsin.

For2. a va b butun sonlari berilgan ($a < b$): a va b sonlari orasidagi barcha butun sonlari (a va b ni ham) chiqaruvchi va chiqarilgan sonlari chiqaruvchi programma tuzilsin. (a va b xam chiqarilsin).

For3. a va b butun sonlari berilgan ($a < b$): a va b sonlari orasidagi barcha butun sonlari (a va b dan tashqari) kamayish tarfibida chiqaruvchi va chiqarilgan sonlari chiqaruvchi programma tuzilsin.

For4. Bir kg konfelingan nani berilgan (haqiqiy son). 1, 2, ..., 10 kg konfelingan nani chiqaruvchi programma tuzilsin.

For5. Bir kg konfelingan nani berilgan (haqiqiy son). 0,1, 0,2, ..., 0,9, 1 kg konfelingan nani chiqaruvchi programma tuzilsin.

For6. Bir kg konfelingan nani berilgan (haqiqiy son). 1,2, 1,4, ..., 2 kg konfelingan nani chiqaruvchi programma tuzilsin.

For7. a va b butun sonlari berilgan ($a < b$): a dan b gacha bo'lgan barcha butun sonlar yig'indisini chiqaruvchi programma tuzilsin.

For8. a va b butun sonlari berilgan ($a < b$): a dan b gacha bo'lgan barcha butun sonlar ko'paytmasini chiqaruvchi programma tuzilsin.

For9. a va b butun sonlari berilgan ($a < b$): a dan b gacha bo'lgan barcha butun sonlar kvadratining yig'indisini chiqaruvchi programma tuzilsin.

For10. n butun soni berilgan ($n > 0$). Quyidagi yig'indini hisoblovchi programma tuzilsin.
 $S = 1 + 1/2 + 1/3 + \dots + 1/n$

For11. n butun soni berilgan ($n > 0$). Quyidagi yig'indini hisoblovchi programma tuzilsin.
 $S = n^2 \cdot (n+1)^2 \cdot (n+2)^2 \cdot \dots \cdot (2n)^2$

For12. n butun soni berilgan ($n > 0$). Quyidagi ko'paytmani hisoblovchi programma tuzilsin.
 $S = 1 \cdot 1 \cdot 2 \cdot 1 \cdot 3 \cdot \dots \cdot (n \cdot n)$

For13. n butun soni berilgan ($n > 0$). Quyidagi yig'indini hisoblovchi programma tuzilsin.
 $S = 1 \cdot 1 \cdot 2 \cdot 1 \cdot 3 \cdot \dots \cdot (n \cdot q)$ shu shartda ishoralar almashsa keladi. Shart operatoridan foydalalmagan!

For14. n butun soni berilgan ($n > 0$). Shu sonning kvadratini quyidagi formula asosida hisoblovchi programma tuzilsin.
 $n^2 = 1 + 3 + 5 + \dots + (2n - 1)$
 har bir qo'shiluvchidan keyin natijani ekranga chiqarib boring. Natijda ekranida 1 dan n gacha bo'lgan sonlar kvadrati chiqariladi.

For15. n butun soni va haqiqiy soni berilgan ($n > 0$). a ning n – darajesini aniqlovchi programma tuzilsin.
 $a^n = a \cdot a \cdot \dots \cdot a$.

For16. n butun soni va haqiqiy soni berilgan ($n > 0$). Bir sikdan foydalanan a ning 1 dan n gacha bo'lgan barcha darajalarini chiqaruvchi programma tuzilsin.

For17. n butun soni va haqiqiy soni berilgan ($n > 0$). Bir sikdan foydalanan quyidagi a ning 1 dan n gacha bo'lgan barcha darajalarini chiqaruvchi va yig'indini hisoblovchi programma tuzilsin.
 $1 + a + a^2 + a^3 + \dots + a^n$

For18. n butun soni va haqiqiy soni berilgan ($n > 0$). Bir sikdan foydalanan quyidagi a ning 1 dan n gacha bo'lgan barcha darajalarini chiqaruvchi va yig'indini hisoblovchi programma tuzilsin.
 $1 - a + a^2 - a^3 + \dots - (-1)^n a^n$

Shart operatoridan foydalanimas.

For19. n butun soni berilgan ($n > 0$). Birdan n gacha bo'lgan sonlar ko'paytmasini chiqaruvchi programma tuzilsin. $n! = 1 \cdot 2 \cdot \dots \cdot n$
 Birdan n gacha bo'lgan sonlar ko'paytmasini faktorial deylladi.

For20. n butun soni berilgan ($n > 0$). Bir sikdan foydalangan holda quyidagi yig'indini hisoblovchi programma tuzilsin.
 $1 + 2! + 3! + \dots + n!$

For21. n butun soni berilgan ($n > 0$). Bir sikdan foydalangan holda quyidagi yig'indini hisoblovchi programma tuzilsin. (Olingan natija taxminan e^x ga yaqinlashad)

$$1 + x + x^2 / (2!) + x^3 / (3!) + \dots + x^n / (n!)$$

For22. n butun soni va x haqiqiy soni berilgan ($n > 0$). Quyidagi yig'indini hisoblovchi programma tuzilsin. (Olingan natija taxminan $\sin(x)$ ga yaqinlashad)

$$x - x^3 / (3!) + x^5 / (5!) - \dots - (-1)^{(n-1)} x^{2n+1} / ((2n+1)!)$$

For24. n butun soni va x haqiqiy soni berilgan ($n > 0$). Quyidagi yig'indini hisoblovchi programma tuzilsin. (Olingan natija taxminan $\cos(x)$ ga yaqinlashad)

$$1 - x^2 / (2!) + x^4 / (4!) - \dots - (-1)^n x^{2n} / ((2n)!)$$

For25. n butun soni va x haqiqiy soni berilgan ($n > 0, |x| < 1$). Quyidagi yig'indini hisoblovchi programma tuzilsin. $x - x^2 / 2 + x^3 / 3 - \dots - (-1)^{n-1} x^n / n$

For26. n butun soni va x haqiqiy soni berilgan ($n > 0, |x| < 1$). Quyidagi yig'indini hisoblovchi programma tuzilsin. $x - x^3 / 3 + x^5 / 5 - \dots - (-1)^{n-1} x^{2n+1} / (2n+1)$

For27. n butun soni va x haqiqiy soni berilgan ($n > 0, |x| < 1$). Quyidagi yig'indini hisoblovchi programma tuzilsin. $x + 1 \cdot x^3 / (2 \cdot 3) + 1 \cdot 3 \cdot x^5 / (2 \cdot 4 \cdot 5) + \dots + 1 \cdot 3 \cdot \dots \cdot (2n-1) \cdot x^{2n+1} / ((2 \cdot 4 \cdot \dots \cdot (2n)) \cdot (2n+1))$

For28. n butun soni va x haqiqiy soni berilgan ($n > 0, |x| < 1$). Quyidagi yig'indini hisoblovchi programma tuzilsin. $1 + x / 2 - 1 \cdot x^2 / (2 \cdot 4) + 1 \cdot 3 \cdot x^4 / (2 \cdot 4 \cdot 6) - \dots + (-1)^{n-1} \cdot 1 \cdot 3 \cdot \dots \cdot (2n-3) \cdot x^{2n} / ((2 \cdot 4 \cdot \dots \cdot (2n)))$

For29. n butun soni va sonlar o'rda 2 ta A, B nutga berilgan. (A, B haqiqiy son). [A, B] kesmani teng n ta kesmaga bo'ling. [A, B] kesmada ajratilgan barcha nuqtalarni chiqaring.

For30. n butun soni va sonlar o'rda 2 ta A, B nutga berilgan. (A, B haqiqiy son). [A, B] kesmani teng n ta kesmaga bo'ling. [A, B] kesmada ajratilgan barcha nuqtalar uchun $F(x) = 1 - \sin(x)$ funktsiya oyjmatini hisoblang.

For31. n butun soni berilgan ($n > 0$). Quyidagi ketma – ketlikning daslatbiki n ta hadini chiqaruvchi programma tuzilsin.

$$A_0 = 2; \quad A_K = 2 + 1/A_{K-1}; \quad K = 1, 2, \dots$$

For32. n butun soni berilgan ($n > 0$). Quyidagi ketma – ketlikning daslatbiki n ta hadini chiqaruvchi programma tuzilsin.

$$A_0 = 1; \quad A_K = (A_{K-1} + 1) / K; \quad K = 1, 2, \dots$$

For33. n butun soni berilgan ($n > 1$). Fibonachi ketma – ketlikning daslatbiki n ta hadini chiqaruvchi programma tuzilsin.

$$F_1 = 1; \quad F_2 = 1; \quad F_K = F_{K-2} + F_{K-1}; \quad K = 3, 4, \dots$$

For34. n butun soni berilgan ($n > 1$). Quyidagi ketma – ketlikning daslatbiki n ta hadini chiqaruvchi programma tuzilsin.

$$A_1 = 1; \quad A_2 = 2; \quad A_K = (A_{K-2} + 2 \cdot A_{K-1}) / 3; \quad K = 3, 4, \dots$$

For35. n butun soni berilgan ($n > 2$). Quyidagi ketma – ketlikning daslatbiki n ta hadini chiqaruvchi programma tuzilsin.

$$A_1 = 1; \quad A_2 = 2; \quad A_3 = 3; \quad A_K = A_{K-3} + A_{K-2} - 2 \cdot A_{K-3}; \quad K = 4, 5, \dots$$

Ichma – Ich ochilgen siklар

For36. N va K butun sonlari berilgan. Quyidagi yig'indini chiqaruvchi programma tuzilsin.
 $1^k + 2^k + \dots + N^k$

For37. N butun soni berilgan. Quyidagi yig'indini chiqaruvchi programma tuzilsin.
 $1^k + 2^k + \dots + N^k$

For38. N butun soni berilgan. Quyidagi yig'indini chiqaruvchi programma tuzilsin.
 $1^k + 2^{2k} + \dots + N^k$

For39. A va B butun soni berilgan ($A < B$). A va B sonlari orasidagi barcha butun sonlari chiqaruvchi programma tuzilsin. Bunda har bir son o'zining qiyomaticha chiqarilishi. Ya'niz son 3 marta chiqariladi.

For40. A va B butun soni berilgan ($A < B$). A va B sonlari orasidagi barcha butun sonlari chiqaruvchi programma tuzilsin. Bunda A soni 1 marta, (A + 1) soni 2 marta chiqariladi va yakazo.

Shart siki operatorlari

While1. A va B butun musbat sonlari berilgan ($A > B$). A usunlikdagi kesmada maksimal darajada B kesma joylashtirilgan. A kesmaning bo'sh qismini aniqlovchi programma tuzilsin. Ko'paytirish va bo'lish amallarini ishlatmang.

While2. A va B butun musbat sonlari berilgan ($A > B$). A usunlikdagi kesmada B kesmada nechta joylashtirish mumkinligini aniqlovchi programma tuzilsin. Ko'paytirish va bo'lish amallarini ishlatmang.

While3. N va K butun musbat sonlari berilgan. Faqat ayirish va qo'shish amallarini ishlatib N sonini K soniga bo'lgandagi qoldiq va butun qismini aniqlovchi programma tuzilsin.

While4. n butun soni berilgan ($n > 0$). Agar n soni 3 ning darajasi bo'lsa "3 – ning darajasi", aks xolda "3 – ning darajasi emas" degan natija chiqaruvchi programma tuzilsin. Qoldiqli bo'lish va bo'lish amallarini ishlatmang.

While5. 2 sonining qandaydir darajasini bildiruvchi n butun soni berilgan ($n > 0$). $n = 2^k$, k ni aniqlovchi programma tuzilsin.

While6. n natural soni berilgan ($n > 0$). Quyidagi ifodani hisoblovchi programma tuzilsin:
 $n! = n * (n - 2) * (n - 4) \dots$

Agar n juft bo'lsa oxirgi ko'payuchini 2, toq bo'lsa 1 bo'ladi.

While7. n natural soni berilgan ($n > 0$). Kvadrati n dan katta bo'ladigan eng kichik butun k sonini ($k^2 > n$) aniqlovchi programma tuzilsin. Ildizdan chiqaruvchi funksiyadan foydalanhmsg.

While8. n natural soni berilgan ($n > 0$). Kvadrati n dan katta bo'lmagan eng katta butun k sonini ($k^2 \leq n$) aniqlovchi programma tuzilsin. Ildizdan chiqaruvchi funksiyadan foydalanhmsg.

While9. n natural soni berilgan ($n > 1$). $3^k > n$ shartni qanoatlantiruvchi eng kichik butun k sonini aniqlovchi programma tuzilsin.

While10. n natural soni berilgan ($n > 1$). $3^k \leq n$ shartni qanoatlantiruvchi eng katta butun k sonini aniqlovchi programma tuzilsin.

While11. n natural soni berilgan ($n > 1$). $(1 + 2 + 3 + \dots + k) \geq n$ shart bajariladigan eng kichik k sonini aniqlovchi programma tuzilsin. 1 dan k gacha bo'lgan yig'indi ham ekranga chiqarilsin.

While12. n natural soni berilgan ($n > 1$). $(1 + 2 + 3 + \dots + k) \leq n$ shart bajariladigan eng katta k sonini aniqlovchi programma tuzilsin. 1 dan k gacha bo'lgan yig'indi ham ekranga chiqarilsin.

While13. a soni berilgan ($a > 1$). $(1 + 1/2 + 1/3 + \dots + 1/k) \geq a$ shart bajariladigan eng kichik k sonini aniqlovchi programma tuzilsin. Yig'indi ham ekranga chiqarilsin.

While14. a soni berilgan ($a > 1$). $(1 + 1/2 + 1/3 + \dots + 1/k) \leq a$ shart bajariladigan eng katta k sonini aniqlovchi programma tuzilsin. Yig'indi ham ekranga chiqarilsin.

While15. Bankda boshlang'ich S so'm qo'yildi. Har oyda bor bo'lgan summa p foizga oshadi ($0 < p < 25$). Necha oydan keyin boshlang'ich qiyomat 2 martada ko'p bo'lishini hisoblovchi programma tuzilsin. Necha oy k – butun son. Bankda hosil bo'lgan summa haqiqiyson ekranga chiqarilsin.

While16. Sportsmen birinchini 10 km yugirib boshladi. Keyingi kunlari bir oldingi kunga nisbatan p foiz ko'p yugurdi ($0 < p < 50$). Sportsmenning necha kundan keyin jami yugurgan masogasi 200 km dan oshadi? Jami kunlar soni va masofani (butun son) chiqaruvchi programma tuzilsin.

While17. n va m butun musbat sonlari berilgan ($n > 0$). n sonini m soniga bo'lib butun va qoldiq qismalarini bo'lish va qoldiqni olish amallarini ishlatsmasdan topuvchi programma tuzilsin.

While18. n butun soni berilgan ($n > 0$). Bo'lib butun va qoldiq qismalarini aniqlash orqali, berilgan son raqamlarini teskari taritibda chiqaruvchi programma tuzilsin.

While19. n butun soni berilgan ($n > 0$). Bo'lib butun va qoldiq qismalarini aniqlash orqali, berilgan son raqamlarini yig'indisini va raqamlari sonini chiqaruvchi programma tuzilsin.

While20. n butun soni berilgan ($n > 0$). Bo'lib butun va qoldiq qismalarini aniqlash orqali, berilgan son raqamlarining orasida 2 raqami bor – yo'qligini aniqlovchi programma tuzilsin.

While21. n butun soni berilgan ($n > 0$). Bo'lib butun va qoldiq qismalarini aniqlash orqali, berilgan son raqamlarining orasida toq raqamlar bor – yo'qligini aniqlovchi programma tuzilsin.

While22. n butun soni berilgan ($n > 1$). N sonini tub yoki tub emasligini aniqlovchi programma tuzilsin.

While23. a va b butun musbat sonlari berilgan. Berilgan sonlarning eng katta umumiy bo'luvchisini aniqlovchi programma tuzilsin.

While24. n butun soni berilgan ($n > 1$). n sonidan katta bo'lgan birinchini Fibonachchi sonini aniqlovchi programma tuzilsin. Fibonachchi sonlari quyidagi qonuniyat asosida topiladi.
 $F_1 = 1; F_2 = 1; F_k = F_{k-1} + F_{k-2}; k = 3, 4, \dots$

While25. n butun soni berilgan ($n > 1$). n sonidan katta bo'lgan birinchini Fibonachchi sonini aniqlovchi programma tuzilsin. Fibonachchi sonlari while24 masalasida berilgan.

While26. Fibonachchi soni bo'lgan n butun soni berilgan ($n > 1$). (Fibonachchi sonlari while24 masalasida berilgan.) n sonidan bir oldingi va bir keyingi Fibonachchi sonlarni chiqaruvchi programma tuzilsin.

While27. Fibonachchi soni bo'lgan n butun soni berilgan ($n > 1$). (Fibonachchi sonlari while24 masalasida berilgan.) n soni Fibonachchi ketma – ketligining nechanchi xadi ekanini chiqaruvchi programma tuzilsin.

While28. e haqiqiy musbat soni berilgan. Ketma - ketlik xadrlari quyidagicha aniqlanadi:
 $a_1=2; a_k = 2 + 1 / a_{k-1}; k = 2, 3, \dots$

$|a_k - a_{k-1}| < e$ shartni qanoatlantiruvchi eng kichik k sonini aniqlovchi programma tuzilsin.
 a_k va a_{k-1} ham ekranga chiqarilsin.

While29. e haqiqiy musbat soni berilgan. Ketma - ketlik xadrlari quyidagicha aniqlanadi:

$a_1=1; a_2=2; a_k = (a_{k-2} + 2 * a_{k-1}) / 3; k = 3, 4, \dots$

$|a_k - a_{k-1}| < e$ shartni qanoatlantiruvchi eng kichik k sonini aniqlovchi programma tuzilsin.
 a_k va a_{k-1} ham ekranga chiqarilsin.

While30. A, B, C musbat butun sonlari berilgan. A x B to'rtburchak ichida tomoni C bo'lgan kvadratdan nechtasi sig'ishini aniqlovchi programma tuzilsin. Ko'paytirish va bo'lish amallarini ishlatmang.

▼ massiv(array). push, pop, unshift, shift

JavaScriptda

`massiv` (array) — bu ma'lumotlarning tartiblangan to'plami. Massiv bir nechta qiymatlarni saqlash uchun ishlataladi va bu qiymatlar turli xil bo'lishi mumkin: raqamlar, matnlar, boshqa massivlar va hokazo. Massivlar ko'pincha indekslangan bo'ladi, ya'ni har bir element o'zining o'ziga xos indeksiga ega.

Massivning sintaksisi

Massivni yaratish uchun kvadrat qavslar (`[]`) ishlataladi. Elementlar vergul bilan ajratiladi.

Sintaksis:

```
let array = [element1, element2, element3, ...];
```

Massivning xususiyatlari:

- Indekslar:** Massivning har bir elementi indeks bilan bog'langan. Indekslar 0 dan boshlanadi.
 - Misol:** `numbers[0]` massivning birinchi elementiga, ya'ni `1` ga teng bo'ladi.
- Dinamik uzunlik:** Massivning uzunligi avtomatik tarzda o'zgaradi.
 - Misol:** `numbers.length` massivning elementlar sonini beradi.

Massiv metodlari:

1. `push()` — Massiv oxiriga element qo'shish

```
let fruits = ["apple", "banana"];
fruits.push("cherry");
console.log(fruits); // Natija: ["apple", "banana", "cher
```

2. `pop()` — Massivning oxiridan element olib tashlash

```
let fruits = ["apple", "banana", "cherry"];
let removed = fruits.pop();
console.log(fruits); // Natija: ["apple", "banana"]
console.log(removed); // Natija: "cherry"
```

3. `unshift()` — Massivning boshiga element qo'shish

```
let fruits = ["banana", "cherry"];
fruits.unshift("apple");
console.log(fruits); // Natija: ["apple", "banana", "cherry"]
```

s Massivning boshidan element olib tashlash

```
let fruits = ["apple", "banana", "cherry"];
let removed = fruits.shift();
console.log(fruits); // Natija: ["banana", "cherry"]
console.log(removed); // Natija: "apple"
```

5. `length` — Massivning uzunligini olish

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits.length); // Natija: 3
```

Massivlar bilan ishlashning asosiy metodlari:

Metod	Vazifasi	Misol
<code>push()</code>	Massivning oxiriga element qo'shadi	<code>array.push(5)</code>
<code>pop()</code>	Massivning oxiridan element olib tashlaydi	<code>array.pop()</code>
<code>unshift()</code>	Massivning boshiga element qo'shadi	<code>array.unshift(0)</code>

<code>shift()</code>	Massivning boshidan element olib tashlaydi	<code>array.shift()</code>
<code>indexof()</code>	Elementning indeksini topadi	<code>array.indexOf("apple")</code>

- Massiv elementlari yig'indisini hisoblash.
- Massiv elementlari ko'paytimasini hisoblash.
- Userdan massiv o'lchamini n kiritishini so'raymiz. n ta massivdan iborat tasodifiy qiymatlarga ega massiv elementlarini yig'indisini hisoblang.
- Berilgan massivni boshidagi va ohiridagi elementlarni joyini o'zgartishi
- Tasodifiy qiymatlar bilan to'ldirilgan massive elementlarini orasidagi eng kattasini toping.

▼ String (matnlar)

JavaScriptda `String` (matnlar) belgilar ketma-ketligini ifodalash uchun ishlataladi. Ularning asosiy vazifasi turli xil matnlar bilan bog'liq operatsiyalarni amalga oshirishdir.

String — JavaScriptda belgilar ketma-ketligini ifodalovchi asosiy ma'lumot turi. Stringlar qo'shtirnoq (`" "`), bitta tirnoq (`' '`), yoki backtick (`` ``) ichida yoziladi.

String xususiyatlari

1. `length` — stringning uzunligini (belgilar sonini) qaytaradi.

```
let text = "Hello, World!";
console.log(text.length); // Natija: 13
```

String metodlari

1. `toUpperCase()` va `toLowerCase()`

Matnni katta yoki kichik harflarga o'zgartirish:

```
let text = "JavaScript";
console.log(text.toUpperCase()); // Natija: "JAVASCRIPT"
```

```
console.log(text.toLowerCase()); // Natija: "javascript"
```

2. `charAt()`

Stringdagi belgilarning indeks bo'yicha qiymati:

```
let text = "Hello";
console.log(text.charAt(1));           // Natija: "e"
```

3. `indexOf()` va `lastIndexOf()`

`indexOf()` Belgilangan elementning **birinchi uchragan** indeksini qaytaradi. `lastIndexOf()` Belgilangan elementning **oxirgi uchragan** indeksini qaytaradi.

```
let text = "JavaScript is fun";
console.log(text.indexOf("is"));        // Natija: 11
console.log(text.lastIndexOf("a"));    // Natija: 3
```

4. `slice()`

Stringdan kichik bir qismni ajratib oladi:

- `slice(start, end)`: Belgilangan oralig'idagi qismini qaytaradi.

```
let text = "Hello, World!";
console.log(text.slice(0, 5));          // Natija: "Hello"
```

5. `replace()` va `replaceAll()`

- `replace()`
- **Vazifasi:** Berilgan string ichidagi **birinchi uchragan** qidiruv qiymatini boshqa qiymat bilan almashtiradi.
- `replaceAll()`

- **Vazifasi:** Berilgan string ichidagi **barcha uchragan** qidiruv qiymatini boshqa qiymat bilan almashtiradi.

```
string.replace(searchValue, newValue)  
string.replaceAll(searchValue, newValue)
```

```
let text = "I love JavaScript";  
console.log(text.replace("JavaScript", "coding")); // Natija: "I love coding"
```

```
let text2 = "cat, cat, cat";  
console.log(text2.replaceAll("cat", "dog")); // Natija: "dog, dog, dog"
```

6. `trim()`, `trimStart()`, va `trimEnd()`

String boshidagi va oxiridagi bo'shliqlarni olib tashlash:

```
let text = "Hello, World! ";  
console.log(text.trim()); // Natija: "Hello, World!"  
console.log(text.trimStart()); // Natija: "Hello, World!"  
  
console.log(text.trimEnd()); // Natija: "Hello, World!"
```

7. `split()`

Stringni belgiga qarab massivga aylantiradi:

```
let text = "apple, banana, cherry";  
let fruits = text.split(", ");  
console.log(fruits); // Natija: ["apple", "banana", "cherry"]
```

8. `includes()`

Stringning ma'lum bir qiymatni o'z ichiga olganini, boshlanganini yoki tugaganini tekshiradi:

```
let text = "JavaScript is fun";
console.log(text.includes("fun"));           // Natija: true
```

9. Template Literals (Shablonlar)

String ichida o'zgaruvchilarni qo'shish va ko'p qatorli matnlarni yaratish uchun qulay:

```
let name = "Alice";
let age = 30;

let introduction = `My name is ${name}, and I am ${age} years old.`;
console.log(introduction);
// Natija: "My name is Alice, and I am 30 years old."
```

10. `join()`

`join()` metodi massiv (array) ichidagi elementlarni birlashtirib, bitta stringga aylantiradi.

```
array.join(separator)
```

- `separator` (*ixtiyoriy*): Elementlarni birlashtirishda ishlataladigan ajratgich (masalan, vergul, bo'sh joy yoki boshqa belgi). Agar berilmasa, **default qiymat** — vergul `,`.
- **Matnlarni massivdan yig'ish:**

```
let words = ["Hello", "World"];
console.log(words.join(" "));
```

```
// Natija: "Hello World"
```

Stringlar bilan ishlashning asosiy metodlari

Metod	Vazifasi	Misol
<code>toUpperCase()</code>	Katta harflarga aylantiradi	<code>text.toUpperCase()</code>
<code>toLowerCase()</code>	Kichik harflarga aylantiradi	<code>text.toLowerCase()</code>
<code>indexOf()</code>	Beglining indeksini qaytaradi	<code>text.indexOf("word")</code>
<code>slice()</code>	Stringning qismini kesib oladi	<code>text.slice(0, 5)</code>
<code>trim()</code>	String boshidagi/oxiridagi bo'shliqlarni olib tashlaydi	<code>text.trim()</code>
<code>split()</code>	Stringni massivga aylantiradi	<code>text.split(",")</code>
<code>replace()</code>	Matndagi bir qiymatni boshqasiga almashtiradi	<code>text.replace("old", "new")</code>

Stringlar bilan ishlashda muhim tushunchalar

1. **O'zgarmaslik:** Stringlar **immutable** (o'zgarmas). Ya'ni, mavjud stringni to'g'ridan-to'g'ri o'zgartirib bo'lmaydi. Faqat yangi string yaratiladi.

```
let text = "Hello";
text[0] = "h"; // Bu ishlamaydi
console.log(text); // Natija: "Hello"
```

- prompt orqali kiritilgan matndagi "a" harflari sonini toping.
- berilgan lotin matindagi b,s,p,l hariflarini kiril variantiga almashtiring.

Xulosa

Stringlar JavaScriptda matnlarni saqlash va qayta ishlashda asosiy vosita hisoblanadi. Ularning ko'plab funksiyalari va metodlari mavjud bo'lib, dasturlashda qulaylik yaratadi. 😊

▼ Function

Function (Funktsiya) — bu ma'lum bir vazifani bajaradigan, bir necha qator kodni o'z ichiga olgan blokdir. Funksiyalarni yaratish va chaqirish orqali biz kodni qayta ishlatish, tartibga solish va modular qilish imkoniyatiga ega bo'lamiz.

Sintaksis:

```
function functionName(parameter1, parameter2) {  
    // Kod bloklari  
    return result;  
}
```

1. Function Declaration

Funktsiyani yaratishning eng oddiy usuli:

```
function myFunction() {  
    console.log("Salom, dunyo!");  
}
```

- `function`: Funktsiya yaratish uchun kalit so'z.
- `myFunction`: Funktsiyaning nomi.
- `{}`: Funktsiya ichidagi kod bloklari.

2. Function Expression

Funktsiya ifodasi o'zgaruvchilarga funksiya qiymatini tayinlash orqali ishlaydi. Bu funktsiyalar anonim bo'lishi mumkin.

```
const myFunction = function() {
    console.log("Salom, dunyo!");
};
```

3. Arrow Function (Yuqori darajadagi funktsiya)

ES6 bilan tanish bo'lgan **arrow function** sintaksisi yanada qisqaroq va o'qish oson. Bu metod funktsiyani qisqaroq ifodalash uchun ishlatiladi.

```
const myFunction = () => {
    console.log("Salom, dunyo!");
};
```

Bir qatorli funktsiyalar uchun qisqaroq sintaksis:

```
const add = (a, b) => a + b;

console.log(add(2, 3)); // Natija: 5
```

Funktsiya qaytargan qiymat (return)

Funktsiya natijasini qaytarish uchun **return** kalit so'zidan foydalанилди. Bu funktsiyaning bajarilishi to'xtaydi va qiymat qaytariladi.

```
function add(a, b) {
    return a + b;
}

let result = add(2, 3);
console.log(result); // Natija: 5
```

- ✓ berilgan massiv uzunligidagi tasodifiy qiymatga ega massiv elementlari yig'indisini chiqaring.

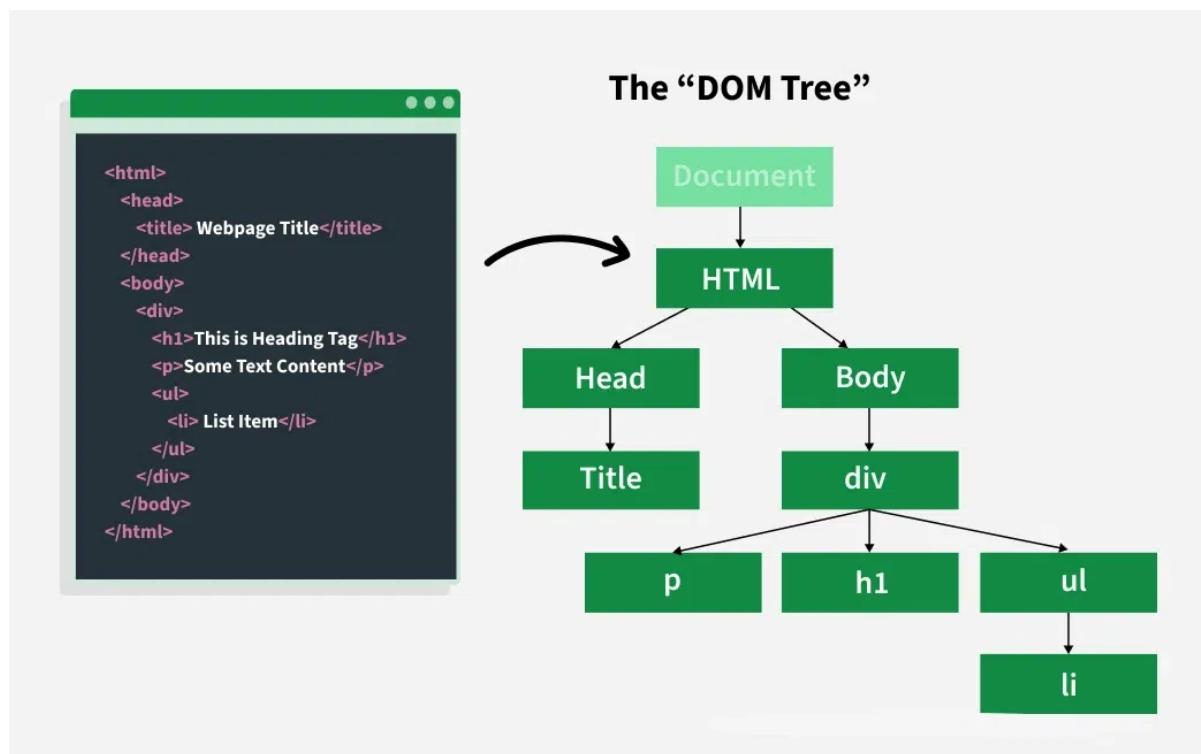
- Berilgan sonning barcha bo'luvchilarini aniqlaydigan funksiya yozing

▼ 5-topshiriq

1. Funktsiya yordamida ikkita sonning yig'indisini hisoblash.
2. Funktsiya yordamida sonni kvadratga oshiring.
3. Funktsiya yordamida so'zning uzunligini qaytarish.
4. Funktsiya yordamida massivdagi eng katta sonni toping.
5. Funktsiya yordamida massivdagi juft sonlar yig'indisini hisoblash.
6. Berilgan sonning barcha bo'luvchilarini aniqlaydigan funksiya yozing

▼ DOM

DOM (Document Object Model) — bu HTML yoki XML hujjatlarining strukturasini JavaScript orqali boshqarishga imkon beradigan interfeysdir. DOM sahifangizning har bir elementini (teglar, atributlar, matn) obyektlar sifatida ko'rsatadi va ular bilan manipulyatsiya qilishga imkon beradi.



▼ **getElementById()**

`getElementById()` metodini ishlatib, HTML hujjatidagi elementni uning `id` atributi orqali tanlash mumkin. Bu metod faqat bitta elementni qaytaradi, chunki `id` atributi yagona bo'lishi kerak.

Sintaksis:

```
document.getElementById("elementID");
```

▼ `querySelector()`

`querySelector()` metodi CSS selektoriga mos keladigan birinchi elementni qaytaradi. Bu metod HTML hujjatida bir nechta elementlar bo'lsa, faqat birinchi topilganini qaytaradi.

Sintaksis:

```
document.querySelector("CSS selector");
```

▼ `innerHTML`

`innerHTML` xususiyati elementning ichidagi HTML tarkibini o'zgartirish yoki olish uchun ishlataladi. Bu xususiyat orqali elementning HTML kodini to'liq o'zgartirish mumkin.

Sintaksis:

```
element.innerHTML = "Yangi HTML tarkibi";
let content = element.innerHTML; // HTML tarkibini olish
```

▼ `textContent`

`textContent` xususiyati elementning ichidagi matnni olish yoki o'zgartirish uchun ishlataladi. Bu xususiyat HTML teglarini inobatga olmaydi, faqat matnni qaytaradi.

Sintaksis:

```
element.textContent = "Yangi matn";
```

```
let content = element.textContent; // Matnni olish
```

Farqlarni taqqoslash:

Xususiyat/Metod	Tavsif	Xavfsizlik	HTML yoki Matnni qaytaradi
<code>getElementById()</code>	<code>id</code> bo'yicha elementni tanlash	Yuqori xavfsizlik	Matn yoki elementni qaytaradi
<code>querySelector()</code>	CSS selektori orqali elementni tanlash	O'rtacha xavfsizlik	Matn yoki elementni qaytaradi
<code>innerHTML</code>	Elementning ichidagi HTML tarkibini o'zgartirish	Xavfli (XSS)	HTML va matnni qaytaradi
<code>textContent</code>	Elementning ichidagi matnni o'zgartirish	Yuqori xavfsizlik	Faqat matnni qaytaradi

- har bir onClick bo'lganida son birga oshib div ichiga songacha bo'lgan qiymatlar yig'indisini chiqaring.
- user input ichiga yozgan matnlardagi unlilarni sonini chiqaring.

▼ **style,classList,keypress,anonim function,forEach**

1. `style`

`style` DOM elementlarning inline CSS uslublarini boshqarish uchun ishlataladi.

```
const element = document.getElementById('box');
element.style.backgroundColor = 'blue';
element.style.padding = '10px';
```

⚡ maslahat:

- Har doim inline `style` dan foydalanishdan qoching. Yaxshisi, klasslar (`classList`) orqali uslublarni boshqaring.

- Inline style dinamik o'zgarishlar uchun yaxshi, lekin kodni boshqarish qiyinlashadi.

2. `classList`

`classList` yordamida elementning CSS klasslarini boshqarish oson bo'ladi.

```
const element = document.querySelector('.box');
element.classList.add('active'); // Klass qo'shish
element.classList.remove('hidden'); // Klass olib tashlash
element.classList.toggle('dark-mode'); // Klassni o'zgartirish
```

⚡ maslahat:

`toggle` usuli bilan interaktiv funksionalliklarni (masalan, temani o'zgartirish) juda samarali tarzda amalga oshirish mumkin.

3. Anonim funksiyalar

Anonim funksiyalar nomlanmagan funksiyalardir. Ular odatda bir marta lik vazifalar uchun ishlatiladi.

⚡ maslahat:

- Arrow funksiyalarni qisqa, kontekst talab qilinmaydigan vazifalar uchun ishlating.
- Nomlangan funksiyalar esa debugging va qayta ishlatish uchun qulay.

4. `forEach`

`forEach` massiv elementlarini ketma-ket ko'rib chiqish uchun ishlatiladi.

```
const numbers = [1, 2, 3];
numbers.forEach((num, index) => {
    console.log(`Index: ${index}, Value: ${num}`);
});
```

maslahat:

- `forEach` qiymatlarni o'zgartirish uchun ishlatalmaydi. Modifikatsiyalar uchun `map` yoki `reduce` dan foydalaning.
- Asinxron funksiyalar bilan ishlashda `forEach` muammoli bo'lishi mumkin. O'rniغا `for...of` tavsiya qilinadi.

5. keypress

`keypress` endi tavsiya qilinmaydi (deprecated). Buning o'rniغا `keydown` yoki `keyup` ishlatalish kerak.

```
document.addEventListener('keydown', (event) => {
    if (event.key === 'Enter') {
        console.log('Enter bosildi!');
    }
});
```

maslahat:

- `event.key` ni ishlatib, maxsus tugmalarni tekshirish mumkin.
- Kodni yaxshi boshqarish uchun `event.preventDefault()` va `event.stopPropagation()` dan foydalanishni unutmang.

input bor massiv uzunligini qabul qiladi. Massivni tasodifiy sonlar bilan to'ldiring. Massivni ul ichiga li qilib chiqaring

▼ Keyboard Events

1. Keyboard Events Turlari:

`keydown`

- Tugma bosilgan zahoti ishga tushadi.

- Tugmani qo'yib yubormaguncha qayta-qayta ishlayveradi (repeat mode).
- Tugma kodlarini (`event.key`, `event.code`) olish uchun ishlatiladi.

keypress (deprecated)

- Eski versiyalarda ishlatilgan. Endi zamonaviy brauzerlarda ishlatish tavsiya etilmaydi.
- Faqat matnli tugmalar uchun ishlaydi (`a-z`, `0-9`, va hokazo).

keyup

- Tugma qo'yib yuborilgandan keyin ishga tushadi.
- Bosish tugagandan keyingi holatni qayd etish uchun qulay.

2. Hodisa obyekti (event)

Klaviatura hodisalari paytida, `event` obyektidan foydalanib quyidagi ma'lumotlarni olish mumkin:

Muhim xususiyatlar:

1. `event.key`

Bosilgan tugmaning qiymatini (masalan, `a`, `1`, `Shift`) qaytaradi.

2. `event.code`

Tugmaning fizik joylashuvini qaytaradi (masalan, `KeyA`, `Digit1`).

3. `event.altKey`, `event.ctrlKey`, `event.shiftKey`

Agar `Alt`, `Ctrl` yoki `Shift` tugmalari bosilgan bo'lsa, `true` qaytaradi.

4. `event.repeat`

Agar tugma bosilgan holda uzoq ushlab turilgan bo'lsa, `true` qaytaradi.

3. Foydalanish misollari

Bosilgan tugmani konsolda ko'rsatish

```
document.addEventListener('keydown', (event) => {
    console.log(`Bosilgan tugma: ${event.key}`);
    console.log(`Tugma kodi: ${event.code}`);
});
```

5. Keyboard Events bilan bog'liq muhim maslahatlar

- `preventDefault()` dan foydalaning: Brauzerning default harakatlarini (masalan, `Ctrl+S` saqlashni) bloklash uchun.
- **Global hodisalar uchun ehtiyyot bo'ling:** `document` yoki `window` ga keyboard hodisalarini qo'shishda, ko'p ishlov beruvchi kodlar sahifani sekinlashtirishi mumkin.
- **Mobil qurilmalar:** Ko'p klaviatura hodisalari mobil qurilmalarda ishlamaydi, chunki ular sensor bilan boshqariladi.

▼ createElement, setAttribute, append, oninput

document.createElement

`createElement` metodi DOM'da yangi HTML elementini yaratish uchun ishlataladi. Bu element hali sahifaga qo'shilmaydi, balki avval yaratiladi va keyinchalik boshqa elementlar bilan birlashtiriladi.

Sintaksis:

```
const newElement = document.createElement(tagName);
```

Misol:

```
const newDiv = document.createElement('div');
newDiv.textContent = "Salom, bu yangi div!";
document.body.appendChild(newDiv);
```

Kattaroq kontekst:

- **Qachon ishlataladi:** Dinamik ravishda elementlarni yaratish, masalan, foydalanuvchi interaktiv elementlarini qo'shish.
- **Optimallik:** Dastlab element yaratiladi va kerak bo'lsa DOM'ga kiritiladi.

setAttribute

`setAttribute` metodi elementning atributlarini o'rnatish yoki o'zgartirish uchun ishlataladi.

Sintaksis:

```
element.setAttribute(attributeName, attributeValue);
```

Misol:

```
const link = document.createElement('a');
link.setAttribute('href', 'https://example.com');
link.setAttribute('target', '_blank');
link.textContent = "Example saytga o'tish";
document.body.appendChild(link);
```

Kattaroq kontekst:

- **Foydasi:** Dinamik ravishda atributlar qo'shish yoki mavjudlarini o'zgartirish.
- **Misol:** Class qo'shish, data-* atributlarini belgilash.

4. append

`append` metodi DOM'ga yangi element qo'shish uchun ishlataladi. U `appendChild` dan ko'ra qulayroq, chunki bir vaqtning o'zida bir nechta tugun yoki tekstlarni qo'shishi mumkin.

Sintaksis:

```
parentElement.append(...nodesOrStrings);
```

Misol:

```
const container = document.getElementById('container');
const newParagraph = document.createElement('p');
newParagraph.textContent = "Bu yangi paragraf.";

const additionalText = " Qo'shimcha matn.";
container.append(newParagraph, additionalText);
```

Kattaroq kontekst:

- **Foydasi:** `appendChild` faqat bitta elementni qabul qiladi, lekin `append` bir nechta element va tekstni birgalikda qo'shadi.
- **Cheklov:** `append` qaytadigan qiymat yo'q.

oninput

`oninput` — bu HTML elementida foydalanuvchi biror ma'lumot kiritganda (input event) ishga tushadigan voqeа (event) handler hisoblanadi. Bu hodisa `keyup`, `keydown`, va `change` hodisalariga o'xshash, lekin har bir kiritilgan harf yoki o'zgarishda avtomatik ravishda ishga tushadi.

Foydalanish:

```
<input type="text" id="name" oninput="handleInput(event)" />
<script>
    function handleInput(event) {
        console.log(`Hozirgi qiymat: ${event.target.value}`);
    }
</script>
```

Kattaroq kontekst:

- **oninput foydasi:** Foydalanuvchi ma'lumot kiritayotgan vaqtida real vaqtida o'zgarishlarni kuzatish mumkin.
- **Misol:** Form validation, dynamic search, yoki qiymatni tahlil qilish.

Bularni birlashtirish misoli:

```
const input = document.createElement('input');
input.setAttribute('type', 'text');
input.setAttribute('placeholder', 'Ismingizni kriting');
input.oninput = (event) => {
    const output = document.getElementById('output');
    output.textContent = `Hozirgi qiymat: ${event.target.value}`;
};

const output = document.createElement('p');
output.setAttribute('id', 'output');

document.body.append(input, output);
```

Natija: Foydalanuvchi input kiritgan sari, paragraflar tagida avtomatik ravishda qiymat aks etadi.

Bu texnikalar JavaScript orqali DOM bilan ishlashni yanada moslashuvchan qiladi.

Mahsulot ro'yxatini shakilantiruvchi dastur. + tugmasi orqali yangi mahsulot uchun input hosil qilish lozim. Saqlash tugmasini bosish orqali barcha inputlar valuelarida massiv qilish lozim

```

index.html
14     display: flex;
15     flex-direction: column;
16     gap: 10px;
17     padding: 10px;
18   }
19   input {
20     width: 100%;
21   }
22 </style>
23 </head>
24 <body>
25   <div id="inps">
26
27   </div>
28   <hr>
29   <button id="plus">+</button>
30   <button id="add">Saqlash</button>
31
32   <script src="./app.js"></script>
33 </body>
34 </html>

```

```

app.js
47 let addBtn = document.getElementById('add')
48
49 const createInput = () => {
50   let input = document.createElement('input')
51   input.setAttribute('type', 'text')
52   input.onkeypress = (e) => {
53     console.log('bosildi', e.key)
54     if (e.key === 'Enter') {
55       createInput()
56     }
57   }
58   inps.append(input)
59   input.focus()
60 }
61
62 plusBtn.onclick = createInput
63 let values = []
64 addBtn.onclick = () => {
65   let inputs = document.querySelectorAll('input')
66   inputs.forEach(input => {
67     values.push(input.value)
68   })
69   console.log(values)
70 }
71

```

▼ setTimeOut, clearTimeout, setInterval, clearInterval

setTimeout va **clearTimeout**

1. **setTimeout**

- **Ma'nosi:** Biror kodni yoki funksiyani ma'lum bir kechikishdan so'ng bajarish uchun ishlataladi. Kechikish millisekundlarda ko'rsatiladi.
- **Sintaksis:**

```

const timeoutID = setTimeout(callback, delay, arg1, arg
2, ...);

```

- **callback**: Kechikishdan keyin bajariladigan funksiya.
- **delay**: Kechikish vaqt (millisekundlarda).
- **arg1, arg2, ...**: Funksiyaga uzatiladigan qo'shimcha argumentlar.

Misol:

```
const sayHello = () => console.log('Hello, world!');  
const timeoutID = setTimeout(sayHello, 2000); // 2 soniyada  
n keyin ishlaydi
```

2. `clearTimeout`

- **Ma'nosi:** `setTimeout` yordamida rejalashtirilgan funksiyani bajarilishidan oldin to'xtatish uchun ishlatiladi.
- **Sintaksis:**

```
clearTimeout(timeoutID);
```

Misol:

```
const timeoutID = setTimeout(() => console.log('Bu bajarilm  
aydi'), 3000);  
clearTimeout(timeoutID); // setTimeout bekor qilinadi
```

`setInterval` va `clearInterval`

1. `setInterval`

- **Ma'nosi:** Funksiyani belgilangan vaqt oralig'ida doimiy ravishda bajarib turish uchun ishlatiladi.
- **Sintaksis:**

```
const intervalID = setInterval(callback, delay, arg1, ar  
g2, ...);
```

- `callback`: Har bir intervalda bajariladigan funksiya.
- `delay`: Har bir bajarilish oralig'i (millisekundlarda).

Misol:

```
const greet = () => console.log('Hello again!');  
const intervalID = setInterval(greet, 1000); // Har 1 soniy  
ada ishlaydi
```

2. `clearInterval`

- **Ma'nosi:** `setInterval` yordamida ishlayotgan intervalni to'xtatish uchun ishlataladi.
- **Sintaksis:**

```
clearInterval(intervalID);
```

Misol:

```
const intervalID = setInterval(() => console.log('Bu bir ne  
cha marta ishlaydi'), 1000);  
  
setTimeout(() => {  
    clearInterval(intervalID); // Intervalni 5 soniyadan keyi  
n to'xtatadi  
    console.log('Interval to\'xtatildi');  
}, 5000);
```

1. **setTimeout va setInterval asinxron:** Ikkalasi ham asinxron mexanizm bo'lib, **Event Loop** yordamida boshqariladi.
2. **Callback Hell:** Haddan tashqari `setTimeout` yoki `setInterval` dan foydalanish kodni chalkashtirib yuborishi mumkin. Buning o'rniغا, **Promise** yoki **async/await** usullariga o'tish kerak.
3. **Performance:** `setInterval` funksiyasi ma'lum bir kechikish tugashi bilanoq keyingi ishni boshlaydi, lekin `callback` ichidagi kod uzoq vaqt talab qilsa, interval kechikishi mumkin.
4. **Alternativ usullar:**

- `setTimeout` ni rekursiv chaqirish orqali `setInterval` ning o'rnini bosishingiz mumkin. Bu ko'proq boshqaruv imkoniyatini beradi:

```
const repeat = () => {
  console.log('Recursive Timeout');
  setTimeout(repeat, 1000); // Har 1 soniyada qayta chaq
  iriladi
};
repeat();
```

5. **Memory Leaks:** `setTimeout` yoki `setInterval` ni noto'g'ri boshqarish, ularni vaqtida to'xtatmaslik **memory leak**ga olib kelishi mumkin. Har doim `clearTimeout` va `clearInterval` dan foydalaning.

▼ filter,map,some,every

`filter`

Maqsad: Massivdan faqat shartga mos keladigan elementlarni qaytaradi.

Foydali xususiyatlari:

- Asl massivni o'zgartirmaydi.
- Yangi massiv qaytaradi.

Kod namunasi:

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

Izoh: Bu kodda `filter` faqat juft sonlarni qaytardi. Filtrlashda har bir elementga shart qo'llanadi.

`map`

Maqsad: Har bir elementni o'zgartirish va yangi massiv yaratish.

Foydali xususiyatlari:

- Asl massivni o'zgartirmaydi.
- Har bir element ustida amallar bajariladi.

Kod namunasi:

```
const numbers = [1, 2, 3, 4, 5];
const squares = numbers.map(num => num ** 2);
console.log(squares); // [1, 4, 9, 16, 25]
```

Izoh: Bu kodda `map` har bir sonni kvadratga ko'tardi. Yangi massiv hosil qilindi.

some

Maqsad: Massivda kamida bitta element shartni qanoatlantirsa, `true` qaytaradi.

Foydali xususiyatlari:

- Tezlikni oshirish uchun shart bajarilishi bilanoq to'xtaydi.
- Boolean qiymat qaytaradi.

Kod namunasi:

```
const numbers = [1, 3, 5, 7];
const hasEven = numbers.some(num => num % 2 === 0);
console.log(hasEven); // false
```

Izoh: Bu kodda `some` massivda juft son borligini tekshirdi. Shart bajarilmadi, natija `false`.

every

Maqsad: Massivning barcha elementlari shartni qanoatlantirsa, `true` qaytaradi.

Foydali xususiyatlari:

- Boolean qiymat qaytaradi.
- Shart bajarilmagan birinchi elementda to'xtaydi.

Kod namunasi:

```
const numbers = [2, 4, 6, 8];
const allEven = numbers.every(num => num % 2 === 0);
console.log(allEven); // true
```

Izoh: Bu kodda `every` barcha sonlarning juft ekanini tasdiqladi, natija `true`.

Katta farqlari

Metod	Maqsad	Natija
filter	Elementlarni filtrlash	Yangi massiv
map	Elementlarni o'zgartirish	Yangi massiv
some	Kamida bitta shart bajarilsa	<code>true</code> yoki <code>false</code>
every	Hammasi shartni bajarsa	<code>true</code> yoki <code>false</code>

1. `splice()`

`splice()` metodi arrayning tarkibini o'zgartiradi (original arrayni o'zgartiradi). U ma'lum indeksdan boshlab elementlarni qo'shish, o'chirish yoki almashtirish uchun ishlatiladi.

Sintaksis:

```
array.splice(start, deleteCount, item1, item2, ...);
```

Parametrlar:

- `start`: Boshlash indeksi (array qayerdan o'zgartiriladi).
- `deleteCount`: O'chiriladigan elementlar soni.
- `item1, item2, ...` (*ixtiyoriy*): O'chirilgan joyga qo'shiladigan yangi elementlar.

Misol:

1. Elementlarni o'chirish:

```
let fruits = ["apple", "banana", "cherry", "date"];
fruits.splice(1, 2); // "banana" va "cherry" o'chiriladi
console.log(fruits); // ["apple", "date"]
```

2. Elementlarni qo'shish:

```
let colors = ["red", "blue"];
colors.splice(1, 0, "green", "yellow"); // 1-indeksga ya
ngi elementlar qo'shiladi
console.log(colors); // ["red", "green", "yellow", "blu
e"]
```

3. Elementlarni almashtirish:

```
let numbers = [1, 2, 3, 4];
numbers.splice(1, 2, 9, 8); // 1-indeksdan boshlab 2 ta
element o'chiriladi va o'rniiga 9 va 8 qo'shiladi
console.log(numbers); // [1, 9, 8, 4]
```

2. `slice()`

`slice()` metodi arraydan bir qismni ajratib oladi va yangi array qaytaradi (original arrayni o'zgartirmaydi).

Sintaksis:

```
array.slice(start, end);
```

Parametrlar:

- `start`: Boshlash indeksi (array qayerdan kesiladi).
- `end` (*ixtiyoriy*): Kesishni to'xtatish indeksi (bu indeks kiritilmaydi).

Misol:

1. Arraydan qismni ajratib olish:

```
let fruits = ["apple", "banana", "cherry", "date"];
let slicedFruits = fruits.slice(1, 3); // 1-indeksdan 3-
gacha (3 kiritilmaydi)
console.log(slicedFruits); // ["banana", "cherry"]
```

2. Oxirigacha kesish:

```
let numbers = [10, 20, 30, 40];
let part = numbers.slice(2); // 2-indeksdan boshlab oxir
igacha
console.log(part); // [30, 40]
```

3. Original array o'zgarmasligi:

```
let colors = ["red", "green", "blue"];
let newColors = colors.slice(0, 2);
console.log(newColors); // ["red", "green"]
console.log(colors); // ["red", "green", "blue"]
```

Farqlari:

Xususiyat	splice	slice
Ta'siri	Original arrayni o'zgartiradi	Original arrayni o'zgartirmaydi
Qo'llanilishi	Elementlarni qo'shish, o'chirish, almashtirish	Arraydan qismni ajratib olish uchun
Qaytadigan qiymati	O'chirilgan elementlardan yangi array	Ajratilgan elementlardan yangi array

Q'o'shimcha Misol:

```
let arr = [1, 2, 3, 4, 5];
```

```

// splice bilan
arr.splice(1, 2); // 1-indeksdan boshlab 2 ta element o'chi
rildi
console.log(arr); // [1, 4, 5]

// slice bilan
let slicedArr = arr.slice(0, 2); // 0 dan 2-gacha kesiladi
console.log(slicedArr); // [1, 4]
console.log(arr); // [1, 4, 5]

```

- **Иммутация** tamoyiliga amal qiling: Bu metodlar asl massivni o'zgartirmaydi, bu esa funksional dasturlash prinsiplarini saqlashga yordam beradi.
- Kodni optimallashtirish uchun kerakli metodni aniq tanlang. Masalan, `filter` va `map` kombinatsiyasi o'rniغا ko'p hollarda faqat `reduce` dan foydalanish mumkin.
- Katta massivlarda ishlaganda, yuqori samaradorlik uchun shartni minimal murakkablikda yozing.

[JS Array Cheat Sheet - Dark.pdf](#)

[JS Array Cheat Sheet - Light.pdf](#)

▼ Object

JavaScript'dagi **Object** — bu kalit-qiyamat (key-value) juftliklarini saqlash uchun ishlataladigan asosiy ma'lumot turi. JavaScript'da deyarli hamma narsa obyekt, shu jumladan massivlar, funksiyalar, va boshqa maxsus turlar.

Object nima?

Object — kalit-qiyamat juftliklarini saqlaydigan struktura. Kalitlar (keys) odatda qator (string) yoki `Symbol`, qiyamatlar esa istalgan turdag'i bo'lishi mumkin (raqam, qator, obyekt, funksiya va hokazo).

Object yaratish usullari

1. Literal sintaksis (eng oddiy usul):

```
const person = {  
    name: "Ali",  
    age: 25,  
    isStudent: true,  
};
```

2. `new Object()` orqali:

```
const person = new Object();  
person.name = "Ali";  
person.age = 25;
```

Object'ga xos asosiy xususiyatlar

1. Kalit-qiyomat juftliklari:

Obyektdagi har bir element kalit (key) va qiymatdan (value) iborat.

```
const obj = { key1: "value1", key2: 42 };  
console.log(obj.key1); // "value1"  
console.log(obj["key2"]); // 42
```

2. Dynamic xususiyatlar qo'shish yoki o'chirish:

```
const user = {};  
user.name = "Ali"; // Qo'shish  
delete user.name; // O'chirish
```

3. Kalitlarga murojaat qilishning ikki usuli:

- Nuqta operatori (`.`):

```
console.log(user.name);
```

- Qavslı yozuv (`[]`):

```
console.log(user["name"]);
```

Object'ga xos foydali metodlar

1. `Object.keys()`

Obyektdagi barcha kalitlarni massiv sifatida qaytaradi.

```
const obj = { name: "Ali", age: 25 };
console.log(Object.keys(obj)); // ["name", "age"]
```

2. `Object.values()`

Obyektdagi barcha qiymatlarni massiv sifatida qaytaradi.

```
console.log(Object.values(obj)); // ["Ali", 25]
```

3. `Object.entries()`

Kalit-qiymat juftliklarini massiv sifatida qaytaradi.

```
console.log(Object.entries(obj)); // [["name", "Ali"], ["age", 25]]
```

4. `Object.assign()`

Bir obyektni boshqasiga nusxalash yoki birlashtirish uchun ishlataladi.

```
const target = { a: 1 };
const source = { b: 2 };
Object.assign(target, source);
console.log(target); // { a: 1, b: 2 }
```

Obyektning prototipi va xususiyatlar merosxo'rligi

JavaScript'da barcha obyektlar `Object.prototype` ni meroq qilib oladi. Shu sababli obyektga `toString()`, `hasOwnProperty()` kabi metodlar avtomatik kiradi.

```
const obj = {};
console.log(obj.toString()); // [object Object]
```

Advanced: Obyektni chuqr nusxalash (Deep Copy)

Agar obyekt ichida boshqa obyektlar bo'lsa, chuqr nusxa olish uchun rekursiya yoki maxsus kutubxonalaridan foydalanish kerak.

```
const obj = { a: 1, b: { c: 2 } };
const deepCopy = JSON.parse(JSON.stringify(obj));
```

- Language kriting shunga qarab salomlashing.

```
const language = {
    uz: 'Salom',
    eng: 'Hello',
    ru: 'Привет',
};

let lang = prompt('Langyage kriting:');

console.log(language[lang] || 'Bunday til mavjuda emas');
```

- input lotincha matin kriting.object orqali har bir harifni kirilchaga o'zgartish kerak.

```
<textarea id="text"></textarea>
<br />
<button id="btn">BTN</button>
<div id="res"></div>
```

```
// JS
const lotinToKiril = {
    a: 'а',
    б: 'б',
    с: 'ц',
    д: 'д',
    е: 'е',
    ф: 'ф',
    г: 'г',
    х: 'х',
    и: 'и',
    ж: 'ж',
    к: 'к',
    л: 'л',
    м: 'м',
    н: 'н',
    о: 'о',
    п: 'п',
    ќ: 'ќ',
    р: 'р',
    с: 'с',
    т: 'т',
    у: 'у',
    в: 'в',
    х: 'х',
    э: 'й',
    з: 'з',
    ш: 'ш',
    ч: 'ч',
    ѕ: 'ё',
    ю: 'ю',
    я: 'я',
    љ: 'њ',
    џ: 'Ѡ'}
```

```

};

const txt = document.getElementById('text');
const res = document.getElementById('res');
const btn = document.getElementById('btn');

btn.addEventListener('click', () => {
    let arr = txt.value.split('');
    arr = arr.map(h => lotinToKiril[h] || h);
    res.innerHTML = arr.join('');
});

```

Xulosa

- **Object** JavaScript'da asosiy ma'lumot turi bo'lib, u turli xil tuzilmalarni ifodalash va ma'lumotlarni tashkil qilish uchun ishlataladi.
- **Object.keys()**, **Object.values()**, **Object.entries()** kabi metodlar ishlashni osonlashtiradi.
- Prototipga asoslangan merosxo'rlik va dinamik xususiyatlar bilan obyektlar qulay va kuchli vositadir.

<https://www.figma.com/file/Po3BOChutKMWCxG4kgjaPp/todolistAPP?node-id=0%3A1>

▼ localStorage

localStorage — JavaScript'ning brauzerda ma'lumotlarni saqlash uchun foydalaniladigan *Web Storage API* qismi bo'lib, u foydalanuvchi brauzerida ma'lumotlarni saqlashga imkon beradi. Ushbu ma'lumotlar quyidagi xususiyatlarga ega:

1. Key-value formatida ma'lumot saqlaydi:

Har bir ma'lumotni **key** (kalit) va **value** (qiymat) juftligi sifatida saqlaydi. Kalit va qiymatlar har doim satr (**string**) formatida bo'ladi.

2. Doimiy saqlash:

`localStorage` ma'lumotlari brauzer yopilib, qaytadan ochilgandan keyin ham o'chib ketmaydi. U faqat dasturchi yoki foydalanuvchi tomonidan o'chirilishi mumkin.

3. Sessiyaga bog'liq emas:

`sessionStorage` dan farqli o'laroq, ma'lumotlar biror sessiyaga bog'liq bo'lmaydi va foydalanuvchi qurilmasida qachonlardir foydalanish uchun saqlanadi.

4. Cheklangan hajm:

Odatda brauzerlarda `localStorage` uchun maksimal 5-10 MB ma'lumot saqlash limiti mavjud.

`localStorage` bilan ishlash metodlari

1. Ma'lumot saqlash (`setItem`)

```
localStorage.setItem('key', 'value');
```

- Misol:

```
localStorage.setItem('username', 'Ali');
```

2. Ma'lumotni o'qish (`getItem`)

```
const value = localStorage.getItem('key');
```

- Misol:

```
const username = localStorage.getItem('username');
console.log(username); // 'Ali'
```

3. Ma'lumotni o'chirish (`removeItem`)

```
localStorage.removeItem('key');
```

- Misol:

```
localStorage.removeItem('username');
```

4. Barcha ma'lumotlarni tozalash (`clear`)

```
localStorage.clear();
```

5. Kalitlarning nomlarini olish (`key`)

```
const keyName = localStorage.key(index);
```

- Misol:

```
console.log(localStorage.key(0)); // 'username'
```

6. Saqlangan ma'lumotlarning soni (`length`)

```
console.log(localStorage.length); // 1
```

Ma'lumotni ob'ekt sifatida saqlash

`localStorage` faqat `string` formatida ma'lumot saqlaydi. Ob'ekt yoki massiv saqlash uchun `JSON.stringify` va o'qishda `JSON.parse` ishlataladi:

- Ob'ekt saqlash:

```
const user = { id: 1, name: 'Ali', age: 25 };
localStorage.setItem('user', JSON.stringify(user));
```

- Ob'ektni o'qish:

```
const user = JSON.parse(localStorage.getItem('user'));
console.log(user.name); // 'Ali'
```

Foydalanishdagi xavfsizlik

1. Xavfsizlik:

- `localStorage` ma'lumotlari brauzerdan tashqariga shifrlanmagan holda saqlanadi. Shuning uchun nozik ma'lumotlarni (`parol`, `token`) saqlash uchun tavsiya etilmaydi.

2. **Same-Origin Policy:** Har bir veb-sayt o'z domeniga xos `localStorage` ga ega. Bir domenning `localStorage` ma'lumotlarini boshqa domen ko'ra olmaydi.

Performance va cheklovlar

1. Sinxron xususiyat:

`localStorage` operatsiyalari sinxron ravishda ishlaydi. Katta hajmdagi ma'lumotlar saqlash yoki o'qish sekinlashuvga olib kelishi mumkin.

2. Asinxronlik kerak bo'lsa:

Agar katta hajmdagi yoki kechiktirilgan saqlashni xohlasangiz, `IndexedDB` yoki server tomonidan boshqariladigan ma'lumotlar bazalaridan foydalanish yaxshiroq.

Qachon `localStorage` dan foydalanish kerak?

- Kichik va doimiy ma'lumotlarni saqlash kerak bo'lsa (masalan, foydalanuvchi sozlamalari).
- Sessiya davomida yoki undan keyin ham ma'lumotlar saqlanishini xohlasangiz.

Qachon foydalanmaslik kerak?

- Katta hajmdagi yoki shifrlashni talab qiluvchi ma'lumotlarni saqlash kerak bo'lsa.
- Sinxron operatsiyalar sekin ishlashga olib keladigan holatlarda.

- input va button bor input ga ma'lumot yozib button bosilganida inputdagি ma'lumotni olib bitta obj yig' va uni arrayga push qil shu arrayni localStorega saqla va shu yig'ilyatgan arraylarni localStoredan olib div ga forEach qilib chiqarib ber

```
<div>
  <input type="text" id="dataInput" placeholder="Ma'lumot kirish"/>
  <button id="saveButton">Saqlash</button>
</div>
<div id="output"></div>

<script src="script.js"></script>
```

```
// HTML elementlarini olish
const input = document.getElementById('dataInput');
const saveButton = document.getElementById('saveButton');
const outputDiv = document.getElementById('output');

// LocalStorage'dan massivni olish yoki bo'sh massiv yaratish
let dataArray = JSON.parse(localStorage.getItem('dataArray'))

// LocalStorage'dagi ma'lumotlarni sahifaga chiqarish
function renderData() {
    outputDiv.innerHTML = ''; // Divni tozalash
    dataArray.forEach((item, index) => {
        const div = document.createElement('div');
        div.textContent = `${index + 1}. ${item.text}`;
        outputDiv.appendChild(div);
    });
}

// Button bosilganda massivga obyekt qo'shish va LocalStoragega qo'shish
saveButton.addEventListener('click', () => {
    const inputValue = input.value.trim();
    if (inputValue) {
```

```
// Obyekt yaratish va massivga qo'shish
const newObj = { text: inputValue };
dataArray.push(newObj);

// LocalStorage'ga saqlash
localStorage.setItem('dataArray', JSON.stringify(dataArray));

// Sahifaga chiqarish
renderData();

// Inputni tozalash
input.value = '';
} else {
    alert("Iltimos, ma'lumot kriting!");
}
});

// Dastlabki render
renderData();
```