5. Sikllar



Tasavvur qiling: o'qituvchi sizga "daftarga 100 marta 'Men dars qilaman' deb yozing" dedi. Qo'lda yozish uzoq. Dasturchi esa bu ishni **uch qatorli sikl** bilan qiladi. Bu dars sikllarni noldan o'rgatadi.

> **Bu darsda nimani o'rganasiz**
>
> Takrorlash (sikl) nima va nima uchun kerak.
>
> `for` siklining uch qismi.
>
> `while` va `do...while`.
>
> `break` va `continue`.
>
> Sikl yordamida yig'indi va jadval hisoblash.
>
> Ichma-ich sikl.

1.1. Takrorlash nima uchun kerak?

Faraz qiling, 1 dan 5 gacha sonlarni ekranga chiqarish kerak. Sikl yo'q holda:

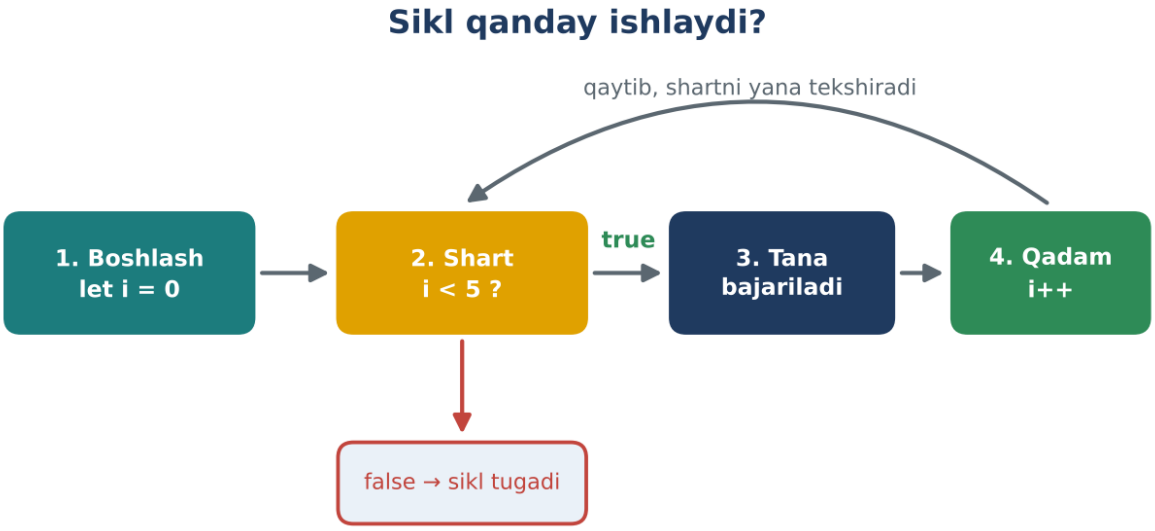
```
console.log(1);
console.log(2);
console.log(3);
```

```
console.log(4);
console.log(5);
```

Natija:

```
1
2
3
4
5
```

5 ta bo'lsa chidasa bo'ladi, lekin 1000 ta bo'lsa-chi? Buning uchun **sikl (loop)** bor: bir kodni berilgan shart bajarilguncha takror-takror ishga tushiradi.



1-rasm. Sikl bosqichlari

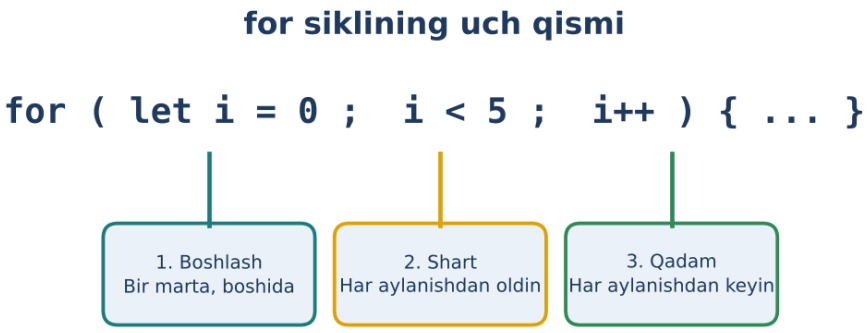
1.2. for sikli

`for` — eng ko'p ishlatiladigan sikl. Aylanishlar soni **oldindan ma'lum** bo'lganda qulay.

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

Natija:

```
1
2
3
4
5
```



2-rasm. for ning uch qismi

1-jadval. for sikli qismlari

Qism	Misol	Ma'nosi
1. Boshlash	`let i = 1`	Hisoblagich yaratiladi. **Bir marta**, siklning boshida
2. Shart	`i <= 5`	Har aylanishdan **oldin** tekshiriladi. `false` bo'lsa, sikl tugaydi
3. Qadam	`i++`	Har aylanishdan **keyin** bajariladi (hisoblagich oshadi)
Tana	`{ console.log(i); }`	Har aylanishda bajariladigan ish

i nima?

`i` — hisoblagich nomi (inglizcha *index* yoki *iteration*). Dasturchilar odatda `i`, `j`, `k` ishlatadi. Siz boshqa nom ham bera olasiz, lekin `i` an'anaviy.

Kod izi: i qanday o'zgaradi

2-jadval. for (let i = 1; i <= 3; i++) kod izi

Aylanish	`i`	`i <= 3`?	Nima chiqadi	Keyin
1	1	true	1	i++ → 2
2	2	true	2	i++ → 3
3	3	true	3	i++ → 4
—	4	**false**	—	Sikl tugadi

Boshqa yo'nalishlar

```
// Kamayish: 5 dan 1 gacha
for (let i = 5; i >= 1; i--) {
```

```
  console.log(i);
}
console.log("Ishga tushdi!");
```

Natija:

```
5
4
3
2
1
Ishga tushdi!
```

```
// Qadam 2: juft sonlar
for (let i = 0; i <= 10; i += 2) {
  console.log(i);
}
```

Natija:

```
0
2
4
6
8
10
```

1.3. Sikl bilan hisoblash

Sikllarning eng foydali ishlatilishi — natijani **bosqichma-bosqich yig'ish**. Buning uchun siklgacha o'zgaruvchi yaratiladi, sikl ichida u yangilanadi.

```
let yigindi = 0;                // boshlang'ich qiymat

for (let i = 1; i <= 10; i++) {
  yigindi += i;                 // har aylanishda i ni qo'shamiz
}

console.log("1 dan 10 gacha yig'indi:", yigindi);
```

Natija:

```
1 dan 10 gacha yig'indi: 55
```

3-jadval. Yig'indi qanday yig'iladi

Aylanish	`i`	`yigindi` (keyin)
1	1	0 + 1 = 1
2	2	1 + 2 = 3
3	3	3 + 3 = 6
4	4	6 + 4 = 10
…	…	…

Aylanish	`i`	`yigindi` (keyin)
10	10	**55**

```
// Faktorial: 5! = 1*2*3*4*5
let natija = 1;                  // ko'paytmada boshlang'ich qiymat 1
for (let i = 1; i <= 5; i++) {
  natija *= i;
}
console.log("5! =", natija);
```

Natija:

```
5! = 120
```

> **Eslab qoling**
>
> Yig'indi uchun boshlang'ich qiymat **0**, ko'paytma uchun **1** bo'ladi.
> Ko'paytmani 0 dan boshlasangiz, natija doim 0 chiqadi.

1.4. while sikli

`while` ("... bo'lguncha") siklida aylanishlar soni oldindan noma'lum bo'lishi mumkin. U faqat **shart** oladi.

```
let i = 1;                       // boshlash sikldan tashqarida

while (i <= 5) {                 // shart
  console.log(i);
  i++;                           // qadam sikl ichida
}
```

Natija:

```
1
2
3
4
5
```

```
// Pul ikki baravar bo'lguncha
let pul = 100;
let yil = 0;

while (pul < 200) {
  pul = pul * 1.1;   // har yili 10 foiz
  yil++;
}
console.log(`${yil} yildan keyin pul ${Math.round(pul)} bo'ldi`);
```

Natija:

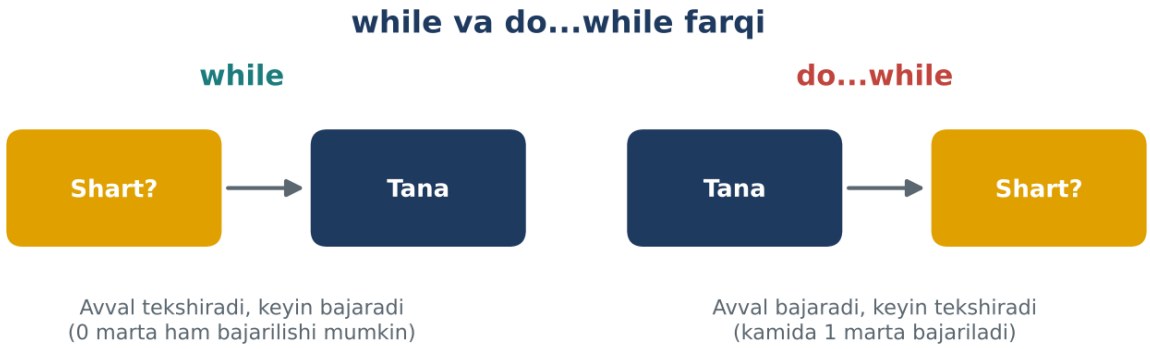
```
8 yildan keyin pul 214 bo'ldi
```

Cheksiz sikl!

Agar shart hech qachon `false` bo'lmasa, sikl **abadiy** ishlaydi va dastur qotib qoladi. `while (i <= 5)` da `i++` ni unutsangiz, aynan shunday bo'ladi. Qotib qolsa, brauzer sahifasini yoping, terminalda `Ctrl + C` bosing.

1.5. do...while sikli

`do...while` da tana **avval** bajariladi, shart **keyin** tekshiriladi. Shuning uchun tana kamida **bir marta** albatta ishlaydi.



3-rasm. while va do...while farqi

```
let son = 10;

while (son < 5) {
  console.log("while ishladi");   // shart false, bajarilmaydi
}

do {
  console.log("do...while ishladi");   // kamida bir marta bajariladi
} while (son < 5);
```

Natija:

```
do...while ishladi
```

1.6. break va continue

Ba'zan siklni **vaqtidan oldin to'xtatish** yoki ayrim aylanishni **o'tkazib yuborish** kerak.

4-jadval. break va continue

Buyruq	Vazifasi	Hayotiy misol

`break`	Siklni butunlay to'xtatadi	Kerakli narsani topdingiz, qidirishni to'xtatasiz
`continue`	Joriy aylanishni tashlab, keyingisiga o'tadi	Yomon olmani o'tkazib yuborasiz, qolganini ko'rasiz



4-rasm. continue va break

```
for (let i = 1; i <= 6; i++) {
  if (i === 3) continue;   // 3 ni o'tkazib yubor
  if (i === 5) break;      // 5 da to'xta
  console.log(i);
}
console.log("Sikl tugadi");
```

Natija:

```
1
2
4
Sikl tugadi
```

1.7. Sikl va massiv (qisqacha)

Massiv — bir nechta qiymatni ketma-ket saqlaydigan ro'yxat. Uni 7-mavzuda batafsil o'rganamiz. Hozircha sikl bilan o'qishni ko'ramiz:

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

// 1) indeks bilan
for (let i = 0; i < mevalar.length; i++) {
  console.log(i, mevalar[i]);
}

// 2) for...of (qulayroq)
for (const meva of mevalar) {
  console.log(meva);
}
```

Natija:

```
0 olma
1 anor
2 uzum
```

```
olma
anor
uzum
```

Eslatma

Massiv indeksi **0 dan** boshlanadi. Shuning uchun sikl `i < mevalar.length` (qat'iy kichik) bilan yoziladi, <= bilan emas.

1.8. Ichma-ich sikl

Sikl ichida yana sikl yozish mumkin. Tashqi sikl bir marta aylansa, ichki sikl **to'liq** aylanib chiqadi.



5-rasm. Ichma-ich sikl jadvali

```
for (let i = 1; i <= 3; i++) {
  for (let j = 1; j <= 3; j++) {
    console.log(`${i} x ${j} = ${i * j}`);
  }
}
```

Natija:

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
3 x 1 = 3
3 x 2 = 6
```

```
3 x 3 = 9
```

1.9. Mini loyihalar

1) Ko'paytirish jadvali

jadval.js

```
const son = 7;
for (let i = 1; i <= 10; i++) {
  console.log(`${son} x ${i} = ${son * i}`);
}
```

Natija:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

2) Yulduzchalardan uchburchak

uchburchak.js

```
for (let qator = 1; qator <= 5; qator++) {
  console.log("*".repeat(qator));
}
```

Natija:

```
*
**
***
****
*****
```

3) Juft sonlar yig'indisi

juft.js

```
let yigindi = 0;
for (let i = 1; i <= 100; i++) {
  if (i % 2 === 0) {
    yigindi += i;
  }
}
console.log("1 dan 100 gacha juft sonlar yig'indisi:", yigindi);
```

Natija:

```
1 dan 100 gacha juft sonlar yig'indisi: 2550
```

Qo'shimcha vazifa

Teskari uchburchak chiqaring (5 ta yulduzdan boshlab 1 tagacha kamayib borsin).

1.10. Eng ko'p uchraydigan xatolar

5-jadval. Tipik xatolar

Xato	Sababi	To'g'ri yozuv
`while (i < 5)` da `i++` yo'q	Cheksiz sikl	Sikl ichida `i++` qo'shing
`for (let i = 0; i <= 5; i++)` dan 5 ta element kutish	6 marta aylanadi (0, 1, 2, 3, 4, 5)	`i < 5` yozing
Massivda `i <= mevalar.length`	Oxirida mavjud bo'lmagan element (`undefined`)	`i < mevalar.length`
`for (let i = 0; i < 5; i--)`	`i` kamayadi, shart doim `true`: cheksiz sikl	`i++` yozing
Ko'paytmada `let natija = 0`	Hammasi nolga ko'payadi	`let natija = 1`

1.11. Mashqlar

1. 1 dan 10 gacha sonlarni `for` bilan chiqaring.
2. 10 dan 1 gacha kamayib chiqaring.
3. 1 dan 20 gacha toq sonlarni chiqaring.
4. 1 dan 100 gacha sonlar yig'indisini hisoblang.
5. `while` yordamida 2 ning darajalarini 1000 dan oshmaguncha chiqaring (1, 2, 4, 8, ...).
6. `for` va `continue` bilan 1 dan 15 gacha 5 ga bo'linuvchi sonlarni o'tkazib yuborib chiqaring.
7. `for` va `break` bilan 1 dan boshlab, kvadrati 50 dan oshgan birinchi sonni toping.
8. Ko'paytirish jadvalining 1 dan 3 gacha bo'lgan qismini ichma-ich sikl bilan chiqaring.

Javoblar

4) 5050. 5) 1, 2, 4, 8, 16, 32, 64, 128, 256, 512.

7) 8 (8×8 = 64 > 50; 7×7 = 49 hali kichik).

1.12. O'zingizni tekshiring

6-jadval. Qisqa test

№	Savol	Javob
1	Sikl nima?	Kodni bir necha marta takrorlash
2	`for` ning uch qismi?	Boshlash, shart, qadam
3	`while` va `do...while` farqi?	`do...while` kamida bir marta ishlaydi
4	`break` nima qiladi?	Siklni to'xtatadi
5	`continue` nima qiladi?	Joriy aylanishni o'tkazib yuboradi
6	Cheksiz sikl nima?	Sharti hech qachon `false` bo'lmaydigan sikl
7	Massiv indeksi nechadan boshlanadi?	0 dan
8	Yig'indi uchun boshlang'ich qiymat?	0

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

1.13. Kichik lug'at

7-jadval. Lug'at

Atama	Sodda ma'nosi
Sikl (loop)	Kodni takror-takror bajaruvchi konstruksiya
Iteratsiya	Siklning bitta aylanishi
Hisoblagich	Aylanishlarni sanaydigan o'zgaruvchi (`i`)
Tana	Sikl ichidagi `{ }` kod
Cheksiz sikl	To'xtamaydigan sikl
Ichma-ich sikl	Sikl ichidagi sikl

Xulosa

Sikl bir ishni ko'p marta bajaradi. `for` — aylanishlar soni ma'lum bo'lganda, `while` — shart bo'yicha, `do...while` — kamida bir marta. `break` to'xtatadi, `continue` o'tkazadi. Cheksiz siklga e'tibor bering: qadam (`i++`) ni unutmang.

Chuqurroq o'rganmoqchimisiz?

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