

Döngüler: for ve while

Aynı satırı yüz kez yazmadan yüz şekil çizmeyi öğreneceğiz.

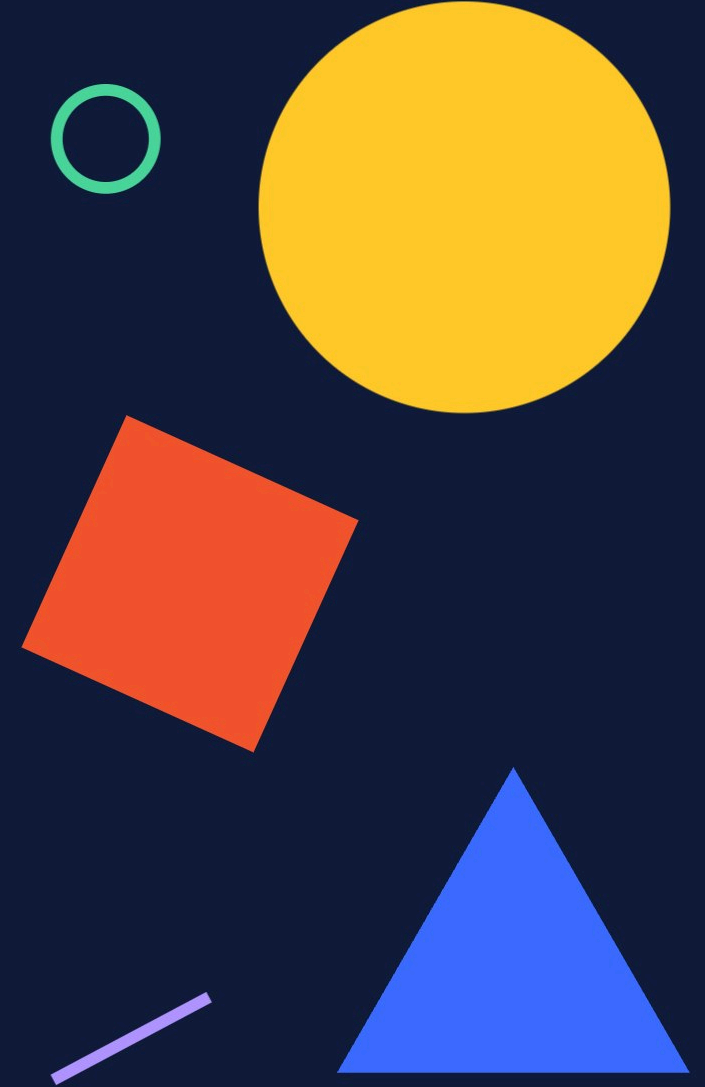
Neden döngü?

for

while

Hangisi ne zaman?

break ve continue



SORU

Yan yana 5 kare böyle çiziliyor. 100 kare için ne yaparsınız?

sketch.js

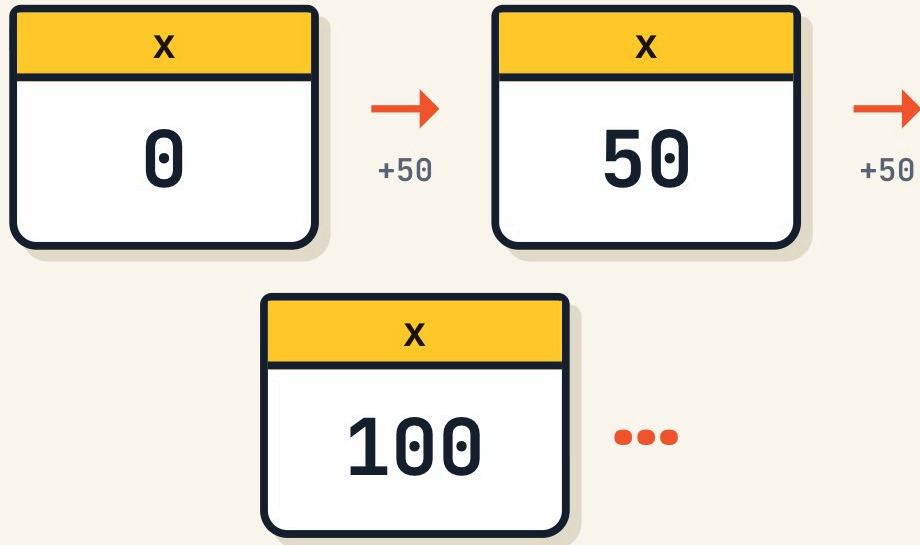
Kopyala

Büyüt

```
1 rect(0, 0, 40, 40);  
2 rect(50, 0, 40, 40);  
3 rect(100, 0, 40, 40);  
4 rect(150, 0, 40, 40);  
5 rect(200, 0, 40, 40);
```

Bu beş satırda değişen ne, hep aynı kalan ne?

Değişen tek şey x: her seferinde 50 artıyor



Döngü (loop)

Aynı ya da benzer işlemleri, bir döngü değişkeni ve bir koşul yardımıyla otomatik olarak tekrarlayan programlama yapısıdır.

Örüntüyü tek bir değişkene bağlarız; değişken her turda güncellenir. On kare de yüz kare de aynı üç satırdır.

for döngüsü

sketch.js

Kopyala

Büyüt

```
for (başlangıç; koşul; artış) {  
    // döngü gövdesi – koşul true olduğu sürece tekrarlanır  
}
```

for döngüsü

Başlangıç değeri, devam koşulu ve artış adımı en başta, noktalı virgülle ayrılmış üç parçada yazılır. Tekrar sayısı genellikle önceden bellidir.

Döngü değişkeni

Tur sayısını ya da konumu izleyen değişkendir. Çoğu zaman `i`, `j` ya da `satır`, `sutun` gibi anlamlı adlar kullanılır.

Üç parça, üç farklı an

Sıra	Parça	Ne zaman çalışır?	Görevi	Örnek
1	Başlangıç	Bir kez, en başta	Değişkene ilk değeri verir	<code>let i = 0</code>
2	Koşul	Her turdan önce	<code>true</code> ise gövde çalışır; <code>false</code> ise döngü biter	<code>i < 5</code>
3	Güncelleme	Her turun sonunda	Değişkeni bir sonraki tura hazırlar	<code>i++</code>



Güncelleme gövdeden **sonra** gelir; gövde içinde `i` hâlâ o turun değeridir.

Üç parçayı konsolda izleyelim

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3   console.log("başlangıç çalışacak");  
4   for (let i = 0; i < 3; i++) {  
5     console.log("gövde, i =", i);  
6   }  
7   console.log("döngü bitti");  
8 }  
9  
10 function draw() {  
11   background(240);  
12 }
```

● Konsol

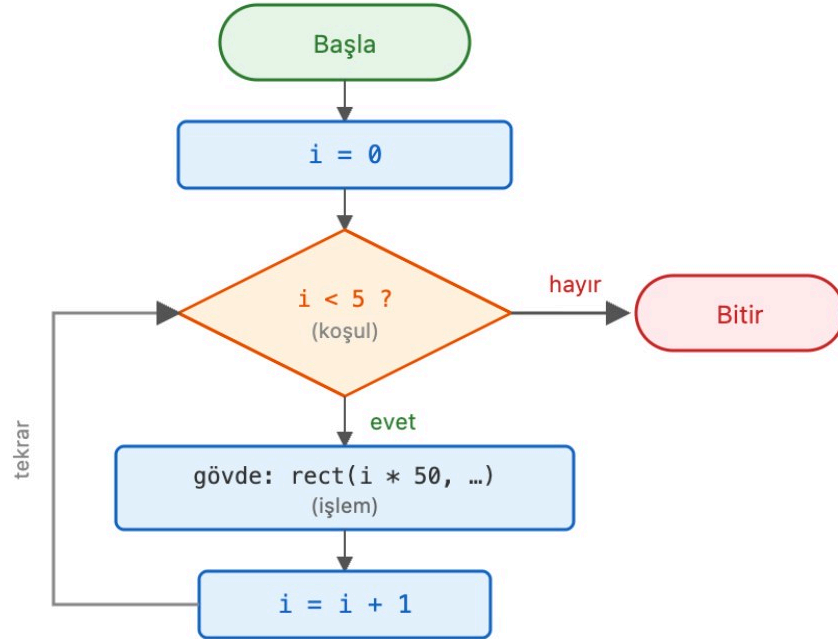
Çalıştır

Büyüt

```
başlangıç çalışacak  
gövde, i = 0  
gövde, i = 1  
gövde, i = 2  
döngü bitti
```

for döngüsünün akışı

for (let i = 0; i < 5; i = i + 1)



Elmas = koşul

Dikdörtgen = işlem

Yuvarlak = başla/bitir

Koşul false olunca döngü biter; evet dalında gövde çalışır, i artar, koşul yeniden sorulur.

1 Başlangıç: $i = 0$

2 Koşul: $i < 5$ mi?

3 Evetse gövde çalışır.

4 Artış: $i = i + 1$

5 Koşula geri dön. Hayırsa döngü biter.

Beş kare, tek rect

sketch.js

Kopyala

Düzenle

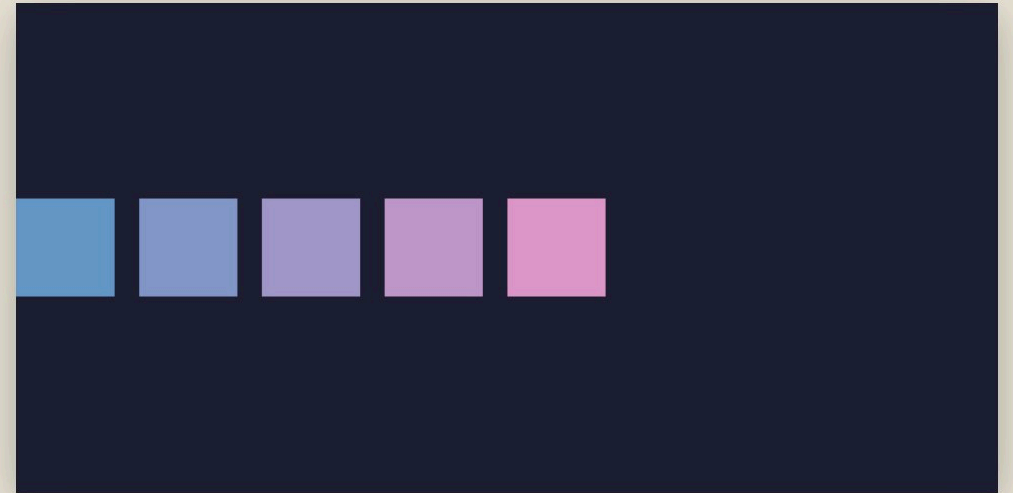
Büyüt

```
1 function setup() {  
2   createCanvas(400, 200);  
3   noStroke();  
4 }  
5  
6 function draw() {  
7   background(30, 30, 50);  
8  
9   for (let i = 0; i < 5; i = i + 1) {  
10     fill(100 + i * 30, 150, 200);  
11     rect(i * 50, 80, 40, 40);  
12   }  
13 }
```

Çıktı

Çalıştır

Büyüt



i sırayla 0, 1, 2, 3, 4 olur. Her değerde `rect(i * 50, ...)` farklı bir x konumuna çizer. Renk de **i**'ye bağlı.

Tur tur izleyelim

Tur	Koşul <code>i < 5</code>	i	Çizilen x	Tur sonu
1	true (0 < 5)	0	0	i → 1
2	true (1 < 5)	1	50	i → 2
3	true (2 < 5)	2	100	i → 3
4	true (3 < 5)	3	150	i → 4
5	true (4 < 5)	4	200	i → 5
6	false (5 < 5)	—	döngü biter	—

Koşul `false` olunca gövde bir kez daha çalışmaz; program döngüden sonraki satıra geçer.

Kısa yazımlar: `++`, `--`, `+=`, `-=`

Uzun yazım	Kısayol	Anlamı
<code>i = i + 1</code>	<code>i++</code>	i bir artar
<code>i = i - 1</code>	<code>i--</code>	i bir azalır
<code>x = x + 10</code>	<code>x += 10</code>	x on artar
<code>skor = skor - 1</code>	<code>skor -= 1</code>	skor bir azalır

Yeni bir kural değil

Yalnızca uzun yazımın kısa hâli.

Dikkat

`++` ve `--` tek birimlik adımdır. Beşer beşer gitmek için `i += 5`. `i++++` diye bir yazım yok.

Üçüncü parça her zaman “artış” olmak zorunda değil

sketch.js

Kopyala

Büyüt

```
for (let i = 0; i < 5; i = i + 1) { /* ... */ }  
for (let i = 0; i < 5; i++) { /* ... */ } // aynı anlam  
for (let satir = 5; satir >= 0; satir--) { /* ... */ } // alttan üste
```

Güncelleme

Bu parçaya alışkanlıkla “artış” denir. Ama azaltma (`i--`), ikişer ya da beşer adım (`i += 2`, `i += 5`) da olur. Önemli olan, tur sonunda değişkenin bir sonraki tura hazır olmasıdır.

SORU

`i < 5` yerine `i <= 5` yazarsak kaç kare çizilir?

sketch.js

Kopyala

Büyüt

```
for (let i = 0; i <= 5; i++) {  
  rect(i * 50, 80, 40, 40);  
}
```

Altı kare: sınırda bir fazla

sketch.js

Kopyala

Düzenle

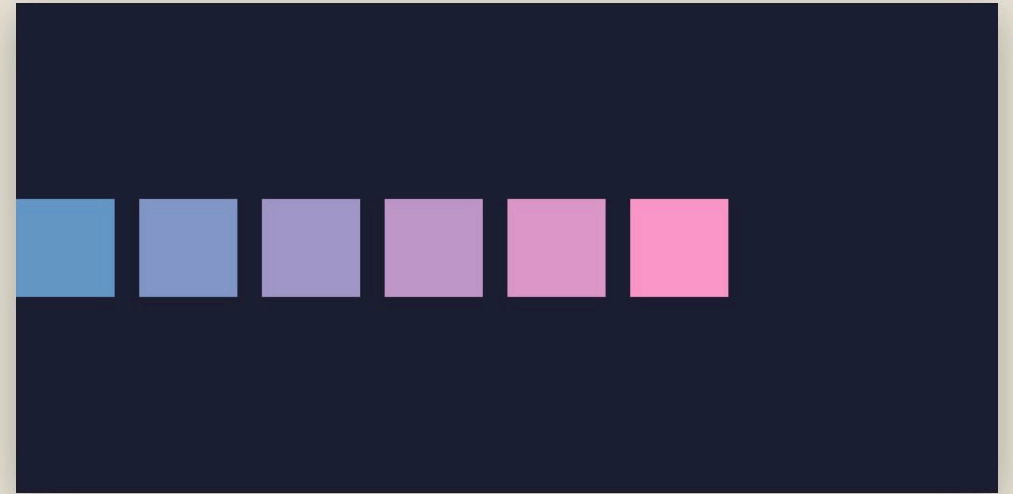
Büyüt

```
1 function setup() {  
2   createCanvas(400, 200);  
3   noStroke();  
4 }  
5  
6 function draw() {  
7   background(30, 30, 50);  
8  
9   for (let i = 0; i <= 5; i++) {  
10     fill(100 + i * 30, 150, 200);  
11     rect(i * 50, 80, 40, 40);  
12   }  
13 }
```

Çıktı

Çalıştır

Büyüt



En yaygın döngü hatası

`i < 5` beş tur üretir (0, 1, 2, 3, 4). `i <= 5` altı tur üretir. Izgara ve dizilerde bu tek fark sık sık hataya yol açar.

Parçaları boş bırakmak mümkün

Yazım	Anlamı
<code>for (let i = 0; i < 5;)</code>	Artış yok. Gövdede <code>i++</code> yazılmazsa sonsuz döngü.
<code>for (; i < 5; i++)</code>	Başlangıç yok. <code>i</code> döngüden önce tanımlanmış olmalı.
<code>for (let i = 0; ; i++)</code>	Koşul yok; hep <code>true</code> sayılır. Çıkmak için <code>break</code> gerekir.
<code>for (;;)</code>	Üçü de boş. <code>while (true)</code> ile aynı; bilinçli <code>break</code> şart.

Noktalı virgüller yerinde kalır; aradaki ifade yazılmazsa o parça atlanmış sayılır.

Artış boş: güncelleme gövdede

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3   for (let i = 0; i < 3; ) {  
4     console.log("artış gövdede, i =", i);  
5     i++; // üçüncü parça boş olduğu için burada  
6   }  
7 }  
8  
9 function draw() {  
10  background(240);  
11 }
```

● Konsoİ

Çalıştır

Büyüt

```
artış gövdede, i = 0  
artış gövdede, i = 1  
artış gövdede, i = 2
```

Başlangıç boş: **i** önceden tanımlı

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3   let i = 0; // başlangıç döngü dışında  
4   for (; i < 3; i++) {  
5     console.log("başlangıç dışarıda, i =", i);  
6   }  
7 }  
8  
9 function draw() {  
10  background(240);  
11 }
```

● Konso1

Çalıştır

Büyüt

```
başlangıç dışarıda, i = 0  
başlangıç dışarıda, i = 1  
başlangıç dışarıda, i = 2
```

Koşul boş: çıkış break ile

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3   for (let i = 0; ; i++) {  
4     console.log("koşul yok, i =", i);  
5     if (i >= 2) {  
6       break; // yoksa sonsuz döner  
7     }  
8   }  
9   console.log("break ile çıktık");  
10 }  
11  
12 function draw() {  
13   background(240);  
14 }
```

● Konsol

Çalıştır

Büyüt

```
koşul yok, i = 0  
koşul yok, i = 1  
koşul yok, i = 2  
break ile çıktık
```

for (;;) : üçü de boş

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3   let n = 0;  
4   for (;;) {  
5     console.log("for(;;), n =", n);  
6     n++;  
7     if (n >= 3) {  
8       break;  
9     }  
10  }  
11 }  
12  
13 function draw() {  
14   background(240);  
15 }
```

● Konsoİ

Çalıştır

Büyüt

```
for(;;), n = 0  
for(;;), n = 1  
for(;;), n = 2
```

Boş bırakmak ile unutmak aynı şey değil

Bilinçli boş bırakma nadirdir

Başlangıç düzeyinde üç parçayı da doldurun; okuması daha kolaydır. `for (;;)` ve koşulsuz `for` yalnızca `break` ile birlikte ve `draw()` dışında kullanılır.

Döngü `draw` içindeyse

Gövde her frame'de tekrarlanır. 10 turluk bir `for`, saniyede 60 frame ile gövdeyi saniyede **600 kez** çalıştırır. Yavaşlama varsa tur sayısına ve `draw`'daki iş yüküne birlikte bakın.

Döngü gövdesinde de girinti

- `for` ve `while` gövdesi, `if` gibi süslü parantez içindedir.
- Gövde satırlarını içeri alın: çoğunlukla 2 ya da 4 boşluk.
- İç içe döngülerde hangi satırın hangi döngüye ait olduğu ancak böyle görülür.
- JavaScript girintiyi zorunlu kılmaz; yine de örneklerimizde hep kullanıyoruz. Tab ile boşluğu karıştırmayın.

while döngüsü

sketch.js

Kopyala

Büyüt

```
while (koşul) {  
    // gövde  
}
```

while döngüsü

Koşul başta kontrol edilir; `true` olduğu sürece gövde tekrarlanır. Başlangıç döngüden **önce**, artış gövdenin **içinde** yazılır.

“Şu doğru olduğu sürece devam et” demenin yolu.

while ile sayaç

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(400, 400);  
3  
4   let sayac = 0;  
5   while (sayac < 5) {  
6     console.log("sayac:", sayac);  
7     sayac = sayac + 1;  
8   }  
9   console.log("while bitti");  
10 }  
11  
12 function draw() {  
13   background(220);  
14 }
```

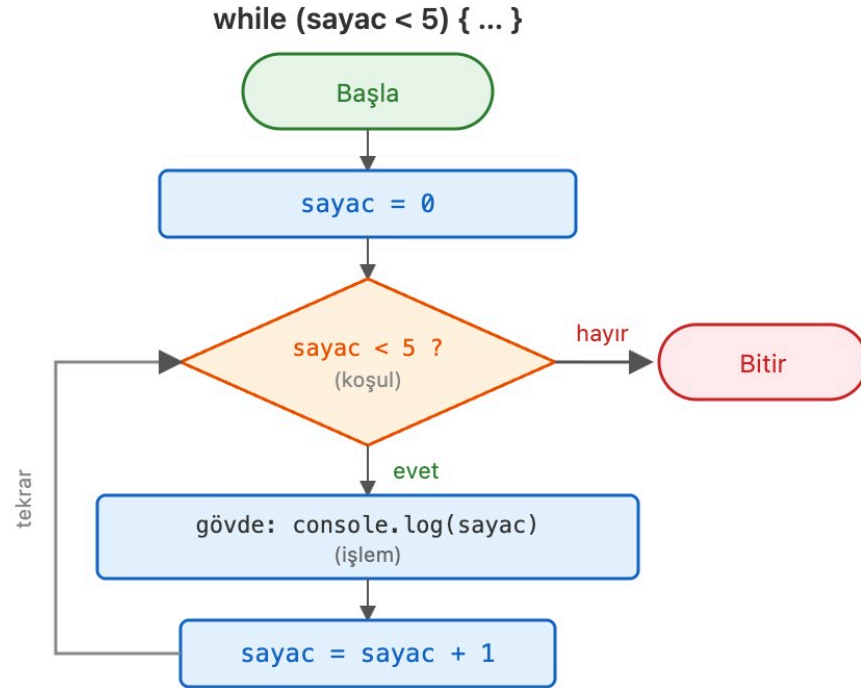
● Konsoİ

Çalıştır

Büyüt

```
sayac: 0  
sayac: 1  
sayac: 2  
sayac: 3  
sayac: 4  
while bitti
```

while akışı ve izleme tablosu



Elmas = koşul
Dikdörtgen = işlem
Yuvarlak = başla/bitir
Başlangıç ve artış for'da başlıkta; while'da gövdede — artış unutulursa sonsuz döngü.

Tur	sayac < 5	Yazılan	Tur sonu
1	true	sayac: 0	1
2	true	sayac: 1	2
...	...	...	...
5	true	sayac: 4	5
6	false	döngü biter	—

for tablosuyla aynı mantık. Tek fark, başlangıcın ve artışın nerede yazıldığı.

SORU

Bu kodu çalıştırırsak ne olur?

sketch.js

Kopyala

Büyüt

```
1 let n = 0;  
2 while (n < 5) {  
3   console.log(n);  
4 }
```

Çalıştırmadan düşünün. Koşul ne zaman `false` olur?

Hiç bitmez: sonsuz döngü

sketch.js

Kopyala

Büyüt

```
1 let n = 0;
2 while (n < 5) {
3   console.log(n);
4   // n artırılmadı → koşul hep true kalır
5 }
```

Çalıştırmayın

`n` hiç artmıyor; koşul hep `true`. Program donar, tarayıcı kilitlenebilir.

Gövdede, koşulu bir gün `false` yapacak bir güncelleme mutlaka olmalı.

`draw` içinde `while (true)` yazmayın

Animasyonu hızlandırmaz, durdurur

`draw` bir frame'i bitirmeden sonraki frame'e geçilemez. Uzun bir `while` o frame'i bitirmez: ekran güncellenmez, fare ve tuş olayları da işlenmez.

Sürekli tekrar zaten var

p5.js `draw()` 'ı her frame kendisi çağırır. Ek bir "sonsuz while" gerekmez.

Aynı iş, iki yazım

for

Kopyala

Büyüt

```
1 for (let i = 0; i < 5; i++) {  
2   console.log("for:", i);  
3 }
```

while

Kopyala

Büyüt

```
1 let i = 0;  
2 while (i < 5) {  
3   console.log("while:", i);  
4   i++;  
5 }
```

for parçası

let i = 0 (başlangıç)

i < 5 (koşul)

i++ (artış)

while'daki karşılığı

Döngüden önceki let i = 0

while (i < 5)

Gövdenin sonundaki i++

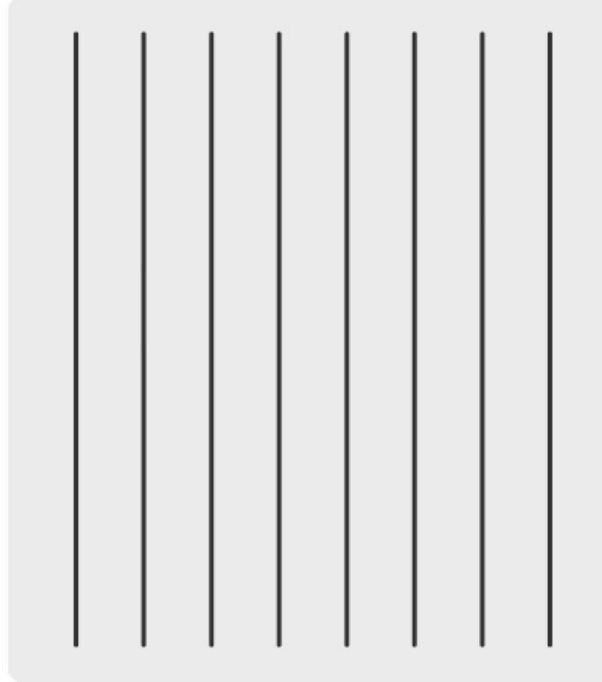
İkisi de konsola 0 1 2 3 4 yazar.

Döngüyle üç basit desen

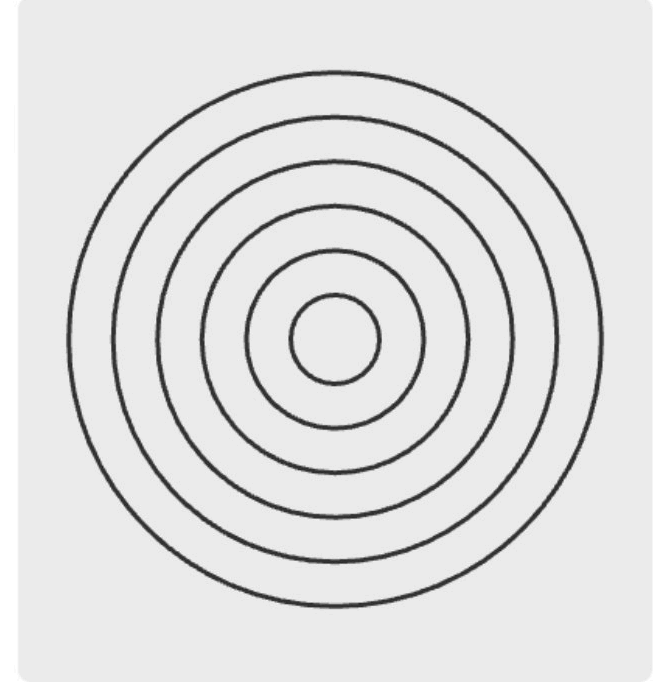
Yatay çizgiler



Dikey çizgiler



Eşmerkezli daireler



Döngü değişkenini konuma ya da boyuta bağlamanın en sade örnekleri.

Yatay çizgiler: for

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(280, 200);  
3 }  
4  
5 function draw() {  
6   background(236);  
7   stroke(40);  
8   for (let i = 0; i < 8; i++) {  
9     let y = 20 + i * 20;  
10    line(20, y, 260, y);  
11  }  
12 }
```

Çıktı

Çalıştır

Büyüt



i her turda y konumunu 20 piksel kaydırır; x uçları sabittir.

Aynı çizgiler: `while`

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(280, 200);  
3 }  
4  
5 function draw() {  
6   background(236);  
7   stroke(40);  
8   let i = 0;  
9   while (i < 8) {  
10     let y = 20 + i * 20;  
11     line(20, y, 260, y);  
12     i++;  
13   }  
14 }
```

● Çıktı

Çalıştır

Büyüt



Dikey çizgiler: değişen artık x

sketch.js

Kopyala

Büyüt

```
1 // for
2 for (let i = 0; i < 8; i++) {
3   let x = 20 + i * 30;
4   line(x, 20, x, 180);
5 }
6
7 // while (eşdeğer)
8 let i = 0;
9 while (i < 8) {
10   let x = 20 + i * 30;
11   line(x, 20, x, 180);
12   i++;
13 }
```

Eşmerkezli daireler

sketch.js

Kopyala

Düzenle

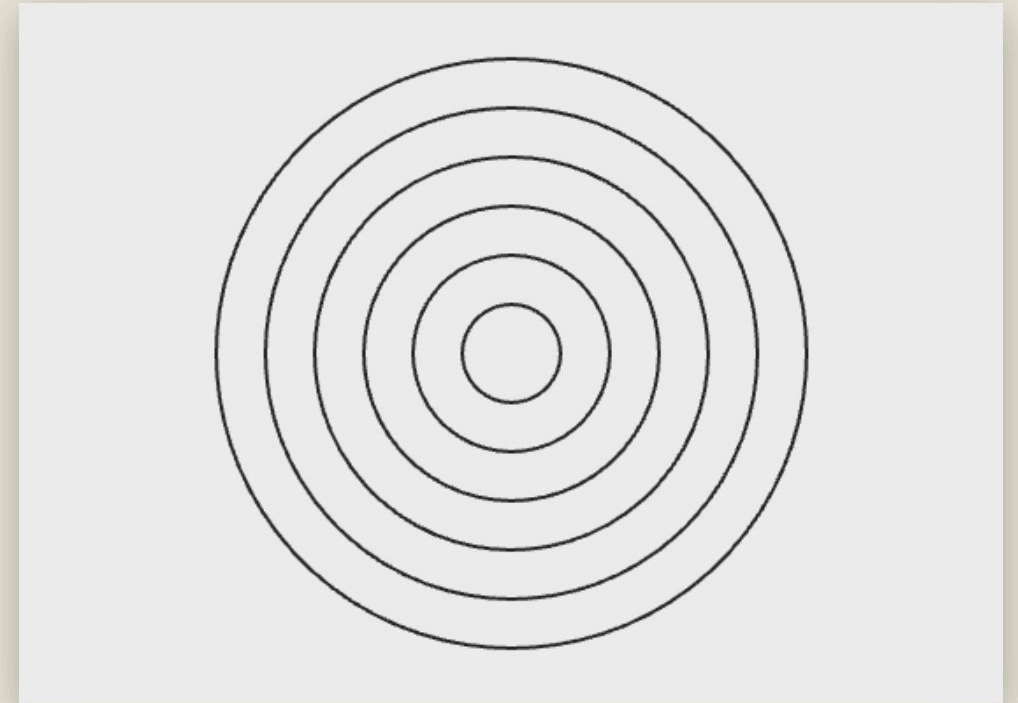
Büyüt

```
1 function setup() {  
2   createCanvas(280, 200);  
3 }  
4  
5 function draw() {  
6   background(236);  
7   noFill();  
8   stroke(40);  
9   for (let i = 1; i <= 6; i++) {  
10     let çap = i * 28;  
11     circle(140, 100, çap);  
12   }  
13 }  
14  
15 // while eşdeğeri:  
16 // let i = 1;  
17 // while (i <= 6) {  
18 //   circle(140, 100, i * 28);  
19 //   i++;  
20 // }
```

Çıktı

Çalıştır

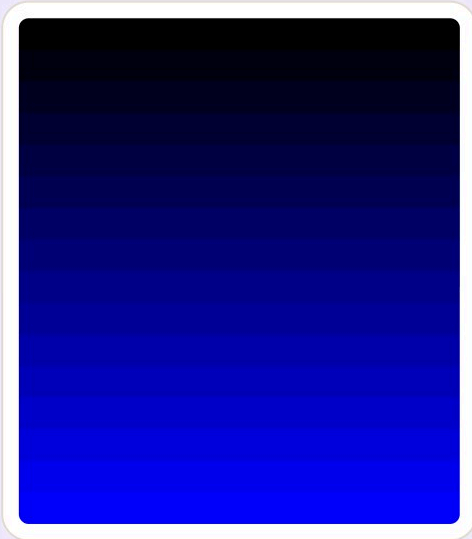
Büyüt



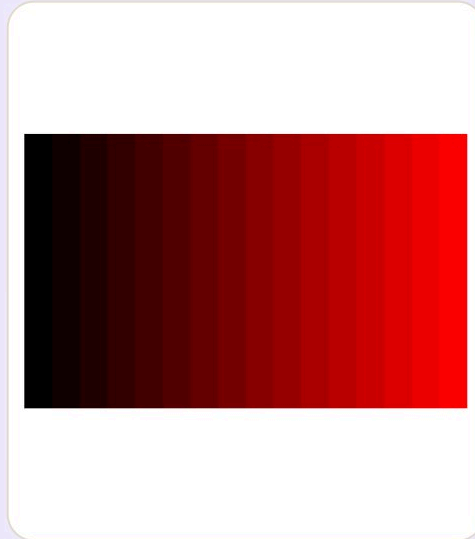
Merkez sabit, çap `i` ile büyüyor. `circle(x, y, çap)` tek bir çap alır.

Tek bir `for` ile renk tayfı

🕒 20 dk



Mavi: üstte 0, altta 255



Kırmızı: solda 0, sağda 255

1. **Mavi:** yatay şeritler; mavi bileşen 0'dan 255'e artsın.
2. **Kırmızı:** dikey şeritler; kırmızı 0'dan 255'e artsın.
3. İç içe döngü gerekmez. Tek tek renk yazmayın.

İpucu

N şerit için $\text{renk} = i * (255 / (N - 1))$. Şerit yüksekliği height / N , genişliği width / N .

SORU

draw zaten tekrar ediyor. Onu döngü gibi kullanabilir miyiz?

background çağırmazsak önceki çizimler silinmez. Her frame'de bir çizgi eklesek ne olur?



Çizgiler birer birer birikir

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let satir = 0;
2
3 function setup() {
4   createCanvas(280, 200);
5   background(236); // yalnızca bir kez
6   stroke(40);
7 }
8
9 function draw() {
10  if (satir < 8) {
11    let y = 20 + satir * 20;
12    line(20, y, 260, y);
13    satir++;
14  }
15  // satir 8 olunca yeni çizgi eklenmez; tuval dolu kalır
16 }
```

● Çıktı

Çalıştır

Büyüt



İlk 8 frame'de her seferinde bir çizgi eklenir. `for` ile yazılan sürüm ise tek frame'de biter.

Üç yaklaşım

Yaklaşım	Ne zaman?
<code>for</code> / <code>while</code> + her frame <code>background</code>	Desen her frame yeniden çizilir. Oyun ve animasyon iskeleti budur.
<code>draw</code> + biriktirme (<code>background</code> yok)	Adım adım biriktirme; basit bir “çizim makinesi”.
<code>draw</code> içinde <code>while (true)</code>	Kullanmayın: frame bitmez, tarayıcı kilitlenir.

Dikkat

Biriktirme örneğinde `background` 'ı `draw` içine koyarsanız tuval her frame silinir ve yalnızca son çizgi kalır.

Hangisi ne zaman?

Durum	Öneri	Gerekçe
Tur sayısı baştan belli (5 kare, 8×8 ızgara)	<code>for</code>	Üç parça tek satırda; artışı unutmak zor.
“Koşul doğru olduğu sürece” (hedef bulunana kadar)	<code>while</code>	Vurgu koşulda; tur sayısı baştan belli olmayabilir.
<code>draw</code> içinde belli sayıda nesne çizmek	<code>for</code>	Frame başına bilinen tur.

Kesin bir kural yok. “Beş kare çiz” için `while` de çalışır; ama `for`, “sayarak tekrarlar” niyetini daha açık gösterir. Seçim, kodu okuyan insan içindir.

`for (;;) ile while (true)` eşdeğerdir; ikisinde de çıkış `break` ile yapılır.

Döngüler oyunlarda ne tekrarlar?

Oyun	Döngünün tekrarladığı iş	Çoğu zaman
Super Mario Bros.	Ekrandaki her düşmanı ve jetonu güncellemek, çizmek	for
Pac-Man	Her hayaleti sırayla hareket ettirmek	for
Tetris	Dolu satırları tarayıp silmek; parçanın bloklarını çizmek	for
Space Invaders	Her gemiyi ve her mermiyi ilerletmek	for

`while` daha çok “can bitene kadar”, “doğru cevap gelene kadar”, “liste boşalana kadar” gibi işlerde öne çıkar.

İki oyun, iki döngü

 10 dk

1. Tablodan ya da bildiğiniz oyunlardan iki örnek seçin.
2. Her biri için yazın: döngü neyi tekrarlıyor?
3. `for` mu `while` mu daha doğal? Birer cümleyle gerekçelendirin.

Erken çıkış: `break` ve `continue`

`break`

Döngüyü **hemen bitirir**. Kalan turlar çalışmaz; program döngüden sonraki satıra geçer.

"Hedef bulundu, yeter."

`continue`

Yalnızca **o turun** geri kalanını atlar. Döngü bir sonraki turla sürer.

"Bu turu atla, sonrakine bak."

İkisi de hem `for` hem `while` içinde kullanılabilir.

break : döngü biter

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3  
4   for (let i = 0; i < 10; i++) {  
5     if (i === 3) {  
6       console.log("break: i =", i, "→ döngü biter");  
7       break;  
8     }  
9     console.log("i =", i);  
10  }  
11  console.log("for sonrası");  
12 }  
13  
14 function draw() {  
15   background(240);  
16 }
```

● Konsol

Çalıştır

Büyüt

```
i = 0  
i = 1  
i = 2  
break: i = 3 → döngü biter  
for sonrası
```

continue : yalnızca bir tur atlanır

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3  
4   for (let i = 0; i < 5; i++) {  
5     if (i === 2) {  
6       console.log("continue: i =", i, "→ bu tur atlandı");  
7       continue;  
8     }  
9     console.log("i =", i);  
10  }  
11 }  
12  
13 function draw() {  
14   background(240);  
15 }
```

● Konsol

Çalıştır

Büyüt

```
i = 0  
i = 1  
continue: i = 2 → bu tur atlandı  
i = 3  
i = 4
```

İç içe döngüde **break** yalnızca içtekini kırar

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3  
4   for (let satir = 0; satir < 3; satir++) {  
5     for (let sutun = 0; sutun < 3; sutun++) {  
6       console.log("bak:", satir, sutun);  
7       if (satir === 1 && sutun === 1) {  
8         console.log("break - yalnızca iç döngü biter");  
9         break;  
10      }  
11    }  
12    console.log("dış tur bitti, satir =", satir);  
13  }  
14 }  
15  
16 function draw() {  
17   background(240);  
18 }
```

● Konsol

Çalıştır

Büyüt

```
bak: 0 0  
bak: 0 1  
bak: 0 2  
dış tur bitti, satir = 0  
bak: 1 0  
bak: 1 1  
break - yalnızca iç döngü biter  
dış tur bitti, satir = 1  
bak: 2 0  
bak: 2 1  
bak: 2 2  
dış tur bitti, satir = 2
```

Dış döngü sürer. İkisini birden kesmek için bir boolean bayrak (**bulundu**) ya da fonksiyon içinde **return** kullanılır.

Kontrollü sonsuz döngü

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let deneme = 0;
2 while (true) {
3   deneme++;
4   console.log("deneme", deneme);
5   if (deneme >= 3) {
6     break; // bilinçli çıkış
7   }
8 }
```

● Konsoİ

Çalıştır

Büyüt

```
deneme 1
deneme 2
deneme 3
```

Bilinçli bir çıkışı olan `while (true)`. `draw` içinde değil, `setup` ya da mantık kodunda kullanılır.

SORU

Konsolda hangi sayılar görünür?

sketch.js

Kopyala

Büyüt

```
1 for (let i = 0; i < 10; i++) {  
2   if (i === 3) {  
3     continue;  
4   }  
5   if (i === 7) {  
6     break;  
7   }  
8   console.log(i);  
9 }
```

Önce kâğıda yazın.

012456

sketch.js

Kopyala

Düzenle

Büyüt

```
1 for (let i = 0; i < 10; i++) {  
2   if (i === 3) {  
3     continue;  
4   }  
5   if (i === 7) {  
6     break;  
7   }  
8   console.log(i);  
9 }
```

● Konsoİ

Çalıştır

Büyüt

```
0  
1  
2  
4  
5  
6
```

3 atlandı ama döngü sürdü. 7'de döngü tamamen bitti; 8 ve 9'a sıra gelmedi.

Yapay zekâyla: for ve while

 15 dk

1. 0'dan 9'a kadar sayıları konsola yazdıran bir `for` isteyin.
2. Eşdeğer bir `while` isteyin. Artış satırını bilerek silin ve ne olduğunu gözleyin. Gerekirse sayfayı yenileyin.
3. Hangisinin daha okunabilir olduğunu bir cümleyle not edin.

Kod zamanı

 30 dk

1. Yan yana 10 daire çizin; her biri bir öncekinden biraz daha büyük olsun.
2. Merdiven: 8 kare sol üstten sağ alta çapraz dizilsin.
3. Geri sayım: konsola 10'dan 1'e kadar sayın, sonra "Başla!" yazın.
4. 1'den 100'e kadar sayıların toplamını bir döngüyle hesaplayıp konsola yazdırın.
5. **Zorlu:** 1'den 30'a kadar sayın; 3'e tam bölünenleri atlayın, 25'i geçince durun.

Konsola ne yazılır?

sketch.js

Kopyala

Büyüt

```
1 for (let i = 10; i > 0; i -= 3) {  
2   console.log(i);  
3 }
```

A 10 7 4☒ **C** 10 7 4 1**C** 10 7 4 1 -2**D** Sonsuz döngü

1 hâlâ 0'dan büyük, yazılır. Sonraki değer -2: koşul yanlış, döngü biter.

Gövde kaç kez çalışır?

sketch.js

Kopyala

Büyüt

```
1 for (let i = 0; i <= 10; i += 2) {  
2   rect(i * 20, 50, 15, 15);  
3 }
```

A 5☒ **✓** 6**C** 10**D** 11

i sırayla 0, 2, 4, 6, 8, 10 olur. `<=` yüzünden 10 da dahil.

Sonunda konsola hangi sayı yazılır?

sketch.js

Kopyala

Büyüt

```
1 let n = 1;
2 while (n < 100) {
3   n = n * 2;
4 }
5 console.log(n);
```

A 64**B** 100☒ **C** 128**D** 200

1, 2, 4, 8, 16, 32, 64... 64 hâlâ 100'den küçük; bir tur daha döner ve 128 olur. Tur sayısı baştan belli değil: `while` 'in doğal yeri.

Bugün ne öğrendik?

- ✓ Döngü, tekrarı bir değişkene ve bir koşula bağlar.
- ✓ `for` (başlangıç; koşul; güncelleme) : başlangıç bir kez, koşul her turdan önce, güncelleme her turun sonunda.
- ✓ `while` : koşul doğru kaldıkça. Güncellemeyi unutursanız sonsuz döngü.
- ✓ Tur sayısı belliyse `for` , vurgu koşuldaysa `while` .
- ✓ `break` döngüyü bitirir; `continue` yalnızca o turu atlar.
- ✓ `draw` içinde `while (true)` yazılmaz.

SIRADAKİ DERS

Döngünün içinde döngü

Izgaralar, dama tahtası ve dalgalanan daireler.

Sonraki sunum →

Tüm sunumlar