

Fonksiyonlar ve modüler kod

Uzayıp giden bir `draw`'ı, her biri tek iş yapan adlandırılmış parçalara böleceğiz.

Tanım ve çağrı

Parametre

return

Yardımcı çizim fonksiyonları

Modüler oyun kodu

SORU

draw fonksiyonunuz 60 satır oldu.
**Tahtanın rengini değiştirmek
istiyorsunuz.**

İlgili satırı bulmak ne kadar sürer? Tahta, skor ve hedef kodu
birbirine karışmışsa?



Her işe bir ad verin

sketch.js

Kopyala

Büyüt

```
function draw() {  
  arkaPlanCiz();  
  tahtaCiz();  
  skorYaz();  
}
```

Fonksiyon (function)

Belirli bir görevi yerine getiren, isimlendirilmiş ve gerektiğinde tekrar çağrılabilen kod bloğudur.

- `draw` artık ne yaptığını bir liste gibi söylüyor.
- Tahta bozuksa bütün `draw` 'a değil, `tahtaCiz` 'e bakarsınız.
- Değişiklik tek noktada yapılır; kopyalanmış satırlarda unutulmuş güncelleme kalmaz.

Aslında fonksiyonları zaten kullanıyoruz

Hazır gelenler

`rect()`, `fill()`, `dist()`,
`random()`. Onları biz
çağırıyoruz.

p5.js'in çağırdıkları

`setup()`, `draw()`,
`mousePressed()`. Biz
yazıyoruz, p5.js belirli anlarda
kendisi çağırıyor.

Bugün: kendi fonksiyonlarımız

Adını biz koyacağız, biz
yazacağız, biz çağıracağız.

Matematikteki $f(x)$ gibi: bir kara kutu



Girdi verirsiniz, içerideki kural çalışır, bir sonuç çıkar. `dist(x1, y1, x2, y2)` tam böyledir: formülü ezberlemesiniz de sonucu kullanırsınız.

Önce şu soruyu sorun

Bu fonksiyon ne alır, ne verir, ne yapar? İçini satır satır bilmek şart değil.

Tanımlamak ve çağırmak

Tanım

Kopyala

Büyüt

```
function fonksiyonAdi() {
  // gövde – çağrıldığında çalışan satırlar
}
```

Çağrı

Kopyala

Büyüt

```
fonksiyonAdi();
```

Kavram	Biçim	Anlamı
Tanım	<code>function hedefCiz() { ... }</code>	Fonksiyonun adı ve gövdesi. Tek başına hiçbir şey çalıştırmaz.
Çağrı	<code>hedefCiz();</code>	Tanımlanan bloğu şimdi çalıştırır. Parantezler şart.
p5.js yerleşik	<code>setup()</code> , <code>draw()</code>	p5.js belirli anlarda otomatik çağırır.

İlk iki fonksiyonumuz

sketch.js

Kopyala

Düzenle

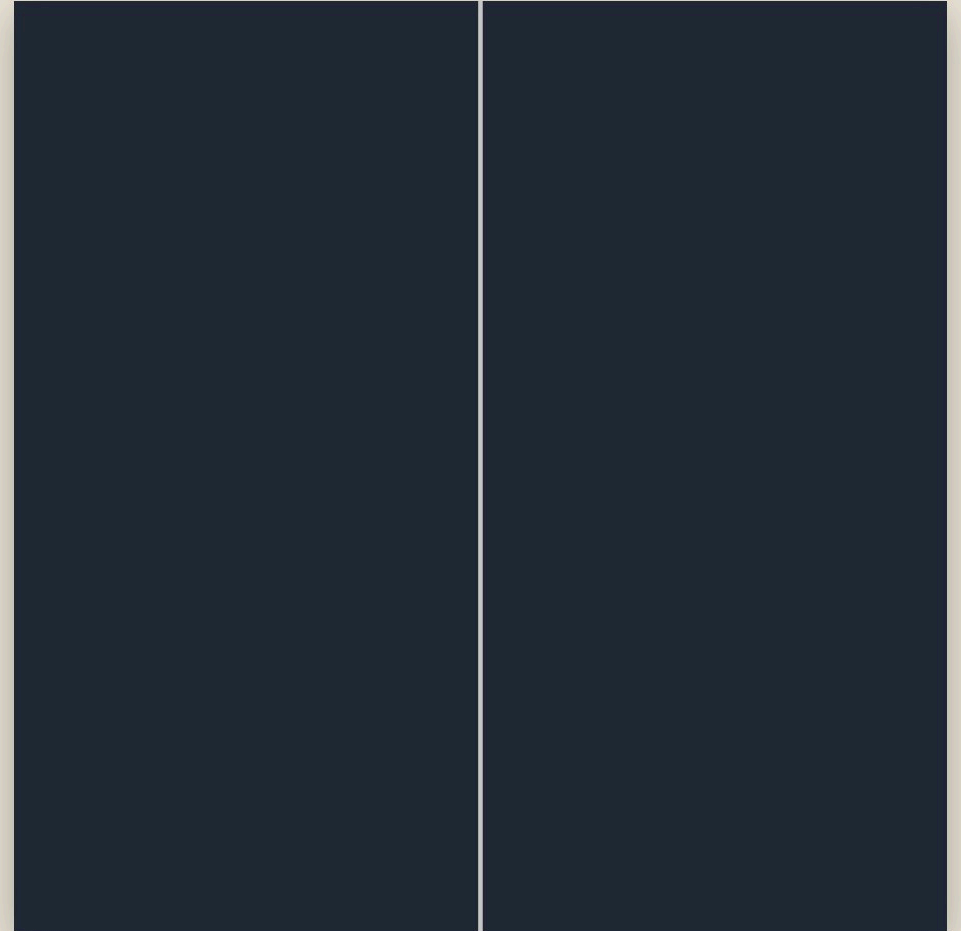
Büyüt

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   arkaPlanCiz();  
7   ortaCizgiCiz();  
8 }  
9  
10 function arkaPlanCiz() {  
11   background(35, 40, 55);  
12 }  
13  
14 function ortaCizgiCiz() {  
15   stroke(200);  
16   strokeWeight(2);  
17   line(width / 2, 0, width / 2, height);  
18   noStroke();  
19 }
```

● Çıktı

Çalıştır

Büyüt



Çağrı izi: sırayla kim çalışıyor?

Adım	Çalışan	Not
1	<code>draw()</code>	p5.js tetikler
2	<code>arkaPlanCiz()</code>	<code>draw</code> içinden çağrılır
3	<code>ortaCizgiCiz()</code>	<code>draw</code> içinden çağrılır
4	(frame biter)	Sonraki frame'de 1'e dönülür

Çağrı yığını (call stack)

Fonksiyonlar iç içe çağrılabilir. `tahtaCiz` bir döngüde `hucreCiz`'i çağırır; her `hucreCiz` bitince kontrol `tahtaCiz`'e döner.

Tanım nerede durmalı?

`function ad() { ... }` tanımları, dosyanın neresinde olursa olsun çalışmadan önce tanınır. Çağrı tanımdan önce yazılsa da çalışır.

SORU

draw yerine cizimYap yazarsak ne olur?

sketch.js

Kopyala

Büyüt

```
function cizimYap() {  
  background(220);  
  ellipse(200, 200, 80, 80);  
}
```

Hiçbir şey çizilmez

Ad sabit

`setup`, `draw` ve `mousePressed` adlarını p5.js tanır ve belirli anlarda bu adlardaki fonksiyonları arar. `cizimYap` diye birini aramaz; onu çağıran olmaz.

Kendi fonksiyonlarınızda

Ad sizin seçiminizdir. camelCase örfünü kullanıyoruz: `hedefCiz`, `skorGuncelle`.

İyi bir fonksiyon nasıl olur?

Tek, net bir iş

Değer alır, çoğu zaman değer üretir ya da tek bir işi tamamlar. "Hem skor hesapla hem dosyaya yaz hem menüyü çiz" bakımı zorlaştırır.

Yan etki

`hedefCiz` tuvale çizer; bu bir yan etkidir ama görevi tek ve anlaşılır. Dosya, ağ, konsol gibi işler çoğunlukla başka katmanlara bırakılır.

Dış değişkeni değiştirmek

Bir fonksiyonun global bir değişkeni güncellemesi de yan etkidir. İdeal olan `return` ile sonuç vermektir; yine de olay fonksiyonlarında sık görülür.

Fonksiyonlar ekip işini kolaylaştırır

- Biri `tahtaCiz` 'i yazarken diğeri `skorYaz` üzerinde çalışabilir.
- Üzerinde anlaşılması gereken tek şey **arayüz**: ad, parametreler, dönüş.
- İki değişiklik aynı satırlarda çakışmaz.

Kara kutu burada da işe yarar

Arkadaşınızın yazdığı fonksiyonun içini bilmeden, ne aldığını ve ne verdiğini biliyorsanız onu çağırabilirsiniz. Tıpkı `dist` gibi.

Uzun bir `draw`'ı bölün

 10 dk

1. Daha önce yazdığınız uzun bir `draw()` seçin (örneğin Daire Tıklama).
2. En az iki anlamlı fonksiyona bölün: `arkaplanCiz`, `skorYaz` gibi.
3. Her fonksiyonun tek sorumluluğunu bir satırla yazın.

SORU

**Biri kırmızı, biri mavi iki dikdörtgen
çizeceğiz. İki ayrı fonksiyon mu
yazalım?**

`kirmiziDikdortgenCiz()` ve `maviDikdortgenCiz()` ... Peki ya on
farklı renk ve konum olursa?



Parametre: fonksiyona dışarıdan değer vermek

sketch.js

Kopyala

Düzenle

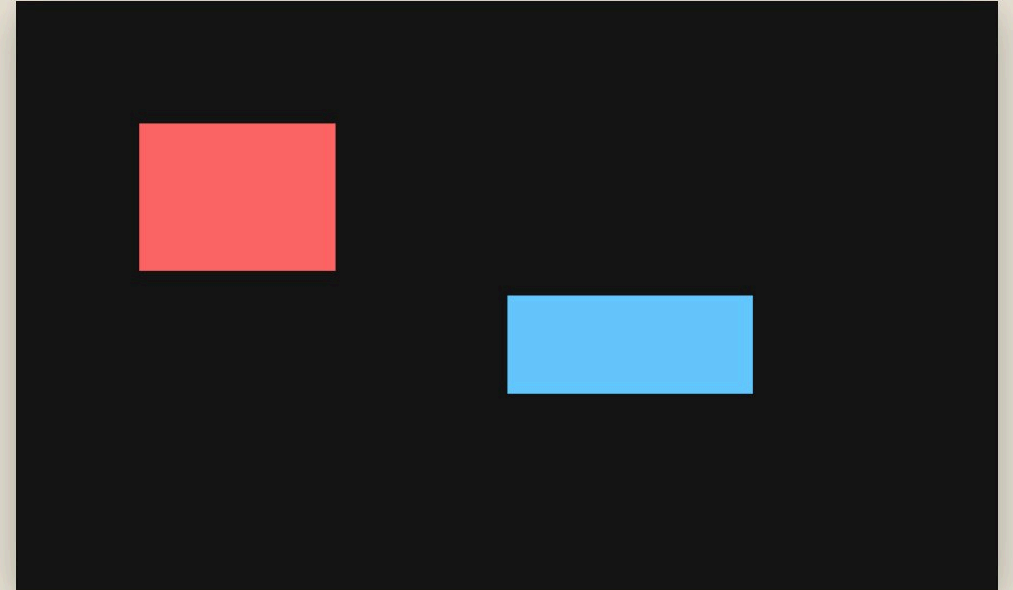
Büyüt

```
1 function dikdortgenCiz(x, y, w, h, renkR, renkG, renkB) {  
2   fill(renkR, renkG, renkB);  
3   noStroke();  
4   rect(x, y, w, h);  
5 }  
6  
7 function draw() {  
8   background(20);  
9   dikdortgenCiz(50, 50, 80, 60, 255, 100, 100); // kırmızımsı  
10  dikdortgenCiz(200, 120, 100, 40, 100, 200, 255); // mavimsi  
11 }  
12  
13 function setup() {  
14   createCanvas(400, 240);  
15 }
```

Çıktı

Çalıştır

Büyüt



Tek fonksiyon, farklı değerlerle iki kez çağrılıyor. Üçüncü bir dikdörtgen eklemek tek satır.

Parametre ve argüman

Parametre (parameter)

Fonksiyon **tanımında** parantez içine yazılan yer tutucu. Dışarıdan gelecek değeri bekler.

```
function dikdortgenCiz(x, y, w, h)
```

Argüman (argument)

Çağrı sırasında verilen somut değer.

```
dikdortgenCiz(50, 50, 80, 60)
```

Sıra önemli

İlk argüman ilk parametreye gider: ilk `50` `x`'e, ikinci `50` `y`'ye. Argüman eksikse karşılığı olmayan parametreler `undefined` olur; fazlaysa fazlası yok sayılır.

SORU

`dist()` bize bir sayı veriyor. Kendi fonksiyonumuz da bir şey geri verebilir mi?

sketch.js

Kopyala

Büyüt

```
let sonuc = topla(3, 5);
```

`sonuc` kutusuna 8 girmesini istiyoruz.

return : sonucu çağırana geri gönder

sketch.js

Kopyala

Büyüt

```
1 let sonuc = topla(3, 5); // sonuc değişkeni 8 olur
2
3 function topla(a, b) {
4   return a + b;
5 }
```

Dönüş değeri (return value)

Fonksiyonun `return` ile geri gönderdiği sonuç. Çağrı ifadesinin **yerine geçer**.

Biçim	Anlamı	Örnek
<code>return;</code>	Değer yok; fonksiyon hemen biter.	<code>mousePressed</code> içinde erken çıkış
<code>return değer;</code>	Hesaplanan değeri çağırana verir.	<code>return a + b;</code>

Boolean döndüren fonksiyon: `acikKareMi`

sketch.js

Kopyala

Düzenle

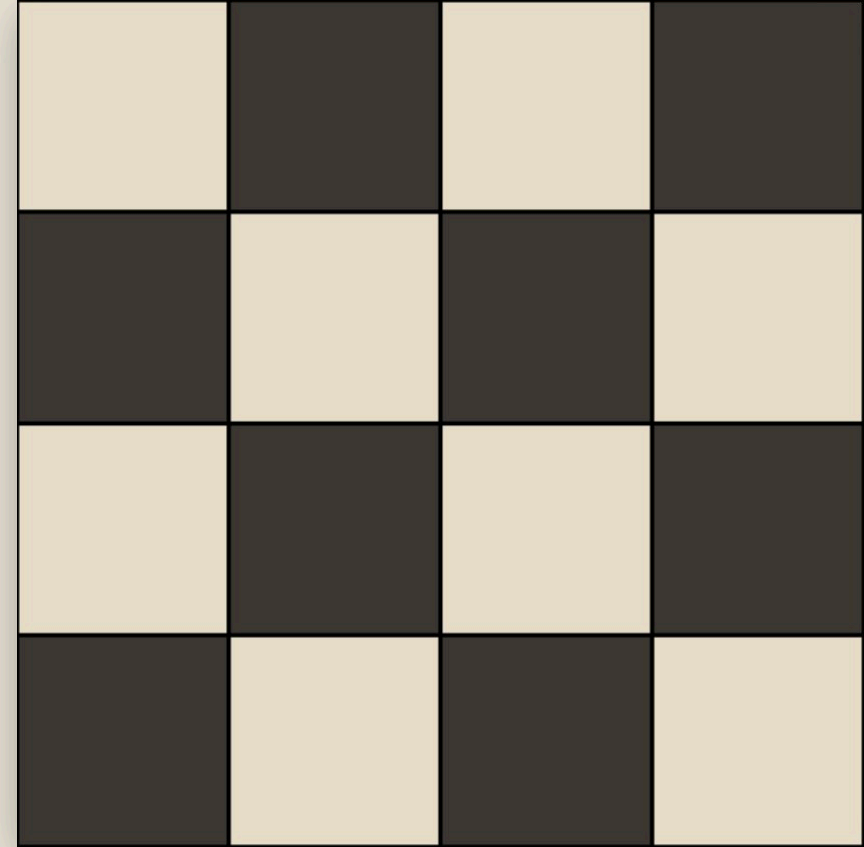
Büyüt

```
1 function acikKareMi(satirSutunToplami) {  
2   return (satirSutunToplami % 2 === 0);  
3 }  
4  
5 function draw() {  
6   background(50);  
7   for (let satir = 0; satir < 4; satir = satir + 1) {  
8     for (let sutun = 0; sutun < 4; sutun = sutun + 1) {  
9       if (acikKareMi(satir + sutun)) {  
10        fill(230, 220, 200);  
11      } else {  
12        fill(60, 55, 50);  
13      }  
14      rect(sutun * 50, satir * 50, 50, 50);  
15    }  
16  }  
17 }  
18  
19 function setup() {  
20   createCanvas(200, 200);  
21 }
```

Çıktı

Çalıştır

Büyüt



`acikKareMi(3) → false`, `acikKareMi(6) → true`. Dönen değer doğrudan `if` koşulunda kullanılıyor.

Oyun mantığında `return` : içinde mi?

sketch.js

Kopyala

Büyüt

```
1 function mesafeHesapla(x1, y1, x2, y2) {  
2   return dist(x1, y1, x2, y2);  
3 }  
4  
5 function icindeMi(fareX, fareY, merkezX, merkezY, yaricap) {  
6   let m = mesafeHesapla(fareX, fareY, merkezX, merkezY);  
7   return m < yaricap;  
8 }
```

`icindeMi(mouseX, mouseY, hedefX, hedefY, 40)` ifadesi `true` ya da `false` üretir.

Tanıdık geldi mi?

Daire Tıklama'daki `dist(...) < hedefYaricap` satırının fonksiyona taşınmış hâli. Okuması kolay: `if (icindeMi(...))`.

Bir fonksiyon başka bir fonksiyonu çağırabilir: `icindeMi`, `mesafeHesapla` 'yı kullanıyor.

“İçinde mi?” testini fonksiyona taşıyın

 20 dk

1. Bölüm 1'deki iki örneği açın: fare yaklaşınca renk değiştiren daire ve Daire Tıklama.
2. “Nokta dairenin içinde mi?” denetimini `icindeMi` fonksiyonuna taşıyın; eski yerlerde bu fonksiyonu çağırın.
3. Davranış değişmemeli.
4. İsterseniz: genişliği ve yüksekliği farklı bir elips için ayrı bir fonksiyon yazmayı deneyin.

Tahtayı fonksiyonlarla çizmek (1/3)

sketch.js · satır 1-14

Kopyala

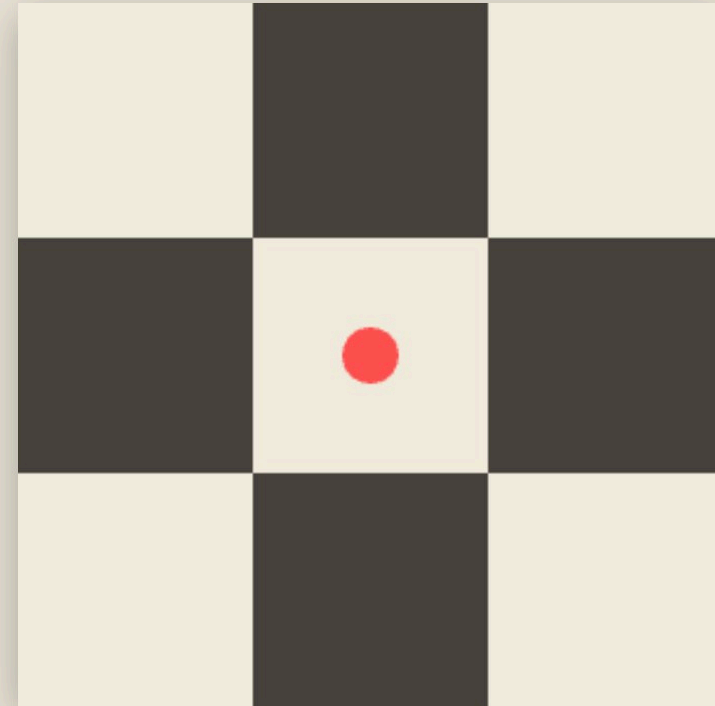
Büyüt

```
1 const HUCRE = 50;  
2 const SUTUN = 3;  
3 const SATIR = 3;  
4  
5 function setup() {  
6   createCanvas(SUTUN * HUCRE, SATIR * HUCRE);  
7   noStroke();  
8 }  
9  
10 function draw() {  
11   background(30, 35, 45);  
12   tahtaCiz(0, 0, SATIR, SUTUN, HUCRE);  
13   merkezIsaretiCiz(width / 2, height / 2);  
14 }
```

● Çıktı

Çalıştır

Büyüt



draw yalnızca iki satır: tahtayı çiz, merkez işaretini çiz.

Tahtayı fonksiyonlarla çizmek (2/3)

sketch.js · satır 16-28

Kopyala

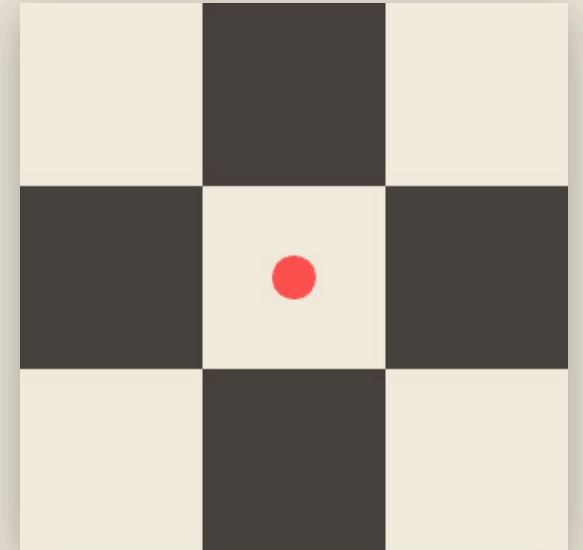
Büyüt

```
16 function tahtaCiz(baslangicX, baslangicY, satirSayisi, sutunSayisi, hucreBoyutu) {  
17   for (let satir = 0; satir < satirSayisi; satir = satir + 1) {  
18     for (let sutun = 0; sutun < sutunSayisi; sutun = sutun + 1) {  
19       hucreCiz(  
20         baslangicX + sutun * hucreBoyutu,  
21         baslangicY + satir * hucreBoyutu,  
22         hucreBoyutu,  
23         satir,  
24         sutun  
25       );  
26     }  
27   }  
28 }
```

Çıktı

Çalıştır

Büyüt



`tahtaCiz` tahtayı gezer; her hücrenin konumunu hesaplayıp işi `hucreCiz` 'e bırakır.

Tahtayı fonksiyonlarla çizmek (3/3)

sketch.js · satır 30-42

Kopyala

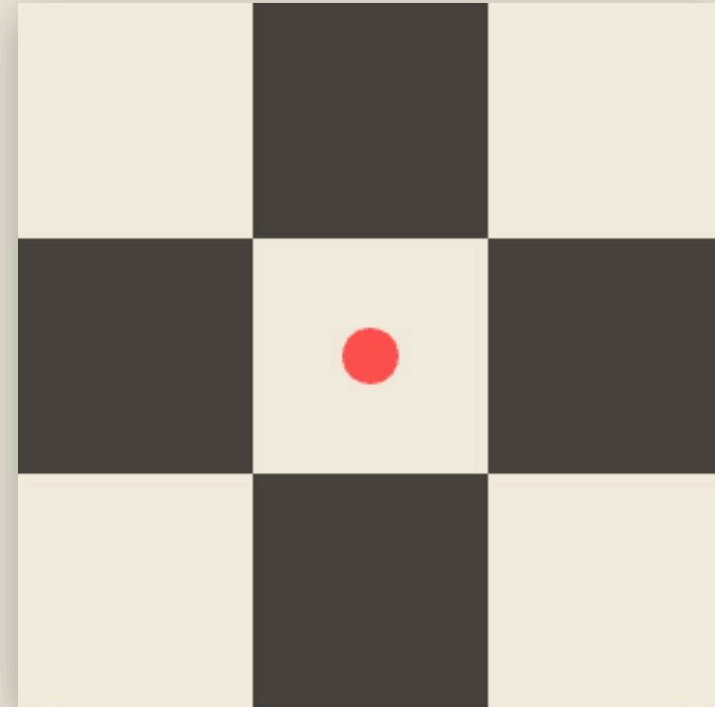
Büyüt

```
30 function hucreCiz(x, y, boyut, satir, sutun) {  
31   if ((satir + sutun) % 2 === 0) {  
32     fill(240, 235, 220);  
33   } else {  
34     fill(70, 65, 60);  
35   }  
36   rect(x, y, boyut, boyut);  
37 }  
38  
39 function merkezIsaretiCiz(merkezX, merkezY) {  
40   fill(255, 80, 80);  
41   ellipse(merkezX, merkezY, 12, 12);  
42 }
```

Çıktı

Çalıştır

Büyüt



`tahtaCiz` farklı argümanlarla yeniden çağrılabilir; yalnızca boyut ve konum değişir. Bu yapı, ileride yazacağımız XOX oyununun zeminidir.

2 × 2 tahtada çağrı sırası

satir	sutun	hucresiz çağrısı	Hücre rengi
0	0	hucresiz(0, 0, 50, 0, 0)	açık (toplam çift)
0	1	hucresiz(50, 0, 50, 0, 1)	koyu
1	0	hucresiz(0, 50, 50, 1, 0)	koyu
1	1	hucresiz(50, 50, 50, 1, 1)	açık

Her (satır, sütun) ikilisi için `hucresiz` bir kez çağrılır. Toplam çağrı: `satirSayisi * sutunSayisi`.

Modülerlik

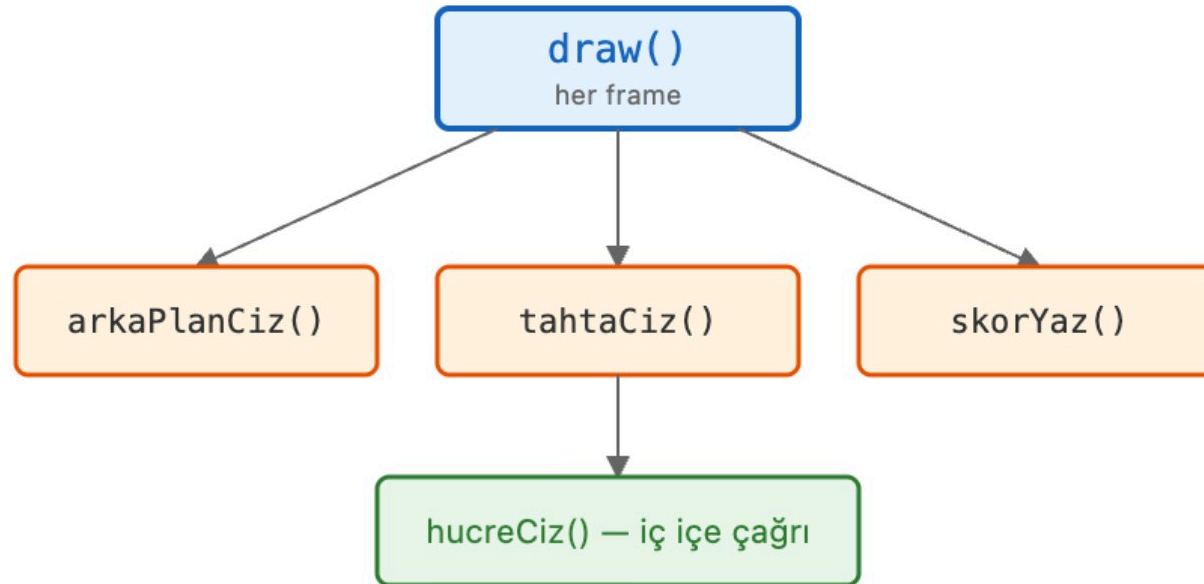
Modülerlik

Programın bağımsız, yeniden kullanılabilir parçalara (fonksiyonlara, ileride sınıflara) ayrılmasıdır.

Görev	Örnek ad	Ne zaman çağrılır?
Çizim	<code>tahtaCiz</code> , <code>hedefCiz</code>	<code>draw()</code> içinde, her frame
Girdi	<code>mousePressed</code> (p5.js olayı)	Fare tıklanınca
Mantık	<code>skorArtir</code> , <code>oyunBittiMi</code>	Olay ya da <code>draw()</code> içinde
Yardımcı hesap	<code>rastgeleKonum</code> , <code>icindeMi</code>	İhtiyaç duyulduğunda

draw işleri dağıtır

Modüler kod — draw() ve yardımcı fonksiyonlar



draw() ne yaptığını listeler; ayrıntı fonksiyon gövdesinde kalır
p5.js otomatik: setup(), draw(), mousePressed() — kullanıcı tanımlı: camelCase ad

draw() görevleri ayrı fonksiyonlara devreder; tahtaCiz de hucreCiz'i çağırır.

Daire Tıklama'yı toparlayalım

sketch.js

Kopyala

Büyüt

```
1 function yeniHedefKonumu() {
2   hedefX = random(50, width - 50);
3   hedefY = random(50, height - 50);
4 }
5
6 function mousePressed() {
7   if (oyunBitti) {
8     return;
9   }
10  let mesafe = dist(mouseX, mouseY, hedefX, hedefY);
11  if (mesafe < hedefYaricap) {
12    skor = skor + 1;
13    if (skor >= HEDEF_SKOR) {
14      oyunBitti = true;
15    } else {
16      yeniHedefKonumu();
17    }
18  }
19 }
```

Aynı iki satır hem `setup`'ta hem `mousePressed`'te vardı. Artık tek bir yerde: `yeniHedefKonumu()`.

Kazanç

Hedefin nasıl seçildiğini değiştirmek isterseniz tek bir fonksiyona dokunursunuz.

Çalışan kodu davranışını bozmadan düzenlemeye **refaktör** denir.

İleriye bakış: XOX taşı çizmek

sketch.js

Kopyala

Büyüt

```
1 function tasCiz(satir, sutun, hucre, oyuncu) {
2   let x = sutun * hucre + hucre / 2;
3   let y = satir * hucre + hucre / 2;
4   textAlign(CENTER, CENTER);
5   textSize(hucre * 0.55);
6   fill(30);
7   text(oyuncu, x, y);
8 }
9
10 function xoxTahtasiniCiz(tahta, hucre) {
11   for (let satir = 0; satir < 3; satir = satir + 1) {
12     for (let sutun = 0; sutun < 3; sutun = sutun + 1) {
13       if (tahta[satir][sutun] !== "") {
14         tasCiz(satir, sutun, hucre, tahta[satir][sutun]);
15       }
16     }
17   }
18 }
```

- `tasCiz` bir hücrenin ortasına "X" ya da "O" yazar.
- `xoxTahtasiniCiz` tahtayı gezer; boş hücrelerde `tasCiz` 'i çağırır.
- Veri (`tahta`) ile görüntü (`tasCiz`) birbirinden ayrı.
- `tahta[satir][sutun]` yazımını diziler konusunda öğreneceğiz.

Örnek olay: 40 satırlık `draw`

3 × 3 tahta çizen kod tek bir `draw()` içinde 40 satıra ulaşmış. Tahtanın rengi değiştirilmek isteniyor ama ilgili satırlar bulunamıyor.

1. `return;` ile erken çıkış ve `return değer;` arasındaki fark ne?
2. Aynı fonksiyonu farklı argümanlarla çağırmak kod tekrarını neden azaltır?
3. `draw()` içinde kaç yardımcı fonksiyon çağrısı makul sayılır?
4. Bu kodu hangi fonksiyonlara bölerdiniz? Taş çizimi nereye gider?

Yapay zekâyla: kara kutu ile ekipçe sketch (1/2)

⌚ 40-50 dk

1. **Çizim fonksiyonlarını yazdırın.** Yapay zekâdan `gunesCiz`, `insanCiz`, `agacCiz`, `evCiz`, `kusCiz` isteyin. Hepsi `(merkezX, merkezY)` alsın, yalnızca temel şekillerle çizsin, sabit piksel adresi kullanmasın.
2. **Seçim fonksiyonunu yazdırın.** Global `let secili = "gunes"` olsun. `keyPressed` tuşa göre değiştirsin: G güneş, I insan, A ağaç, E ev, K kuş.

Klavyeyi henüz işlemedik, sorun değil

`keyPressed` 'in içini bilmeniz gerekmiyor. Sözleşme belli: doğru tuşa basınca `secili` doğru metin olur. Bu bir kara kutu.

Yapay zekâyla: kara kutu ile ekipçe sketch (2/2)

sketch.js · satır 13-28

Kopyala

Büyüt

```
13  if (mouseX < 40 && mouseY < 40) {  
14    background(230); // temizle  
15    return;  
16  }  
17  if (secili === "gunes") {  
18    gunesCiz(mouseX, mouseY);  
19  } else if (secili === "insan") {  
20    insanCiz(mouseX, mouseY);  
21  } else if (secili === "agac") {  
22    agacCiz(mouseX, mouseY);  
23  } else if (secili === "ev") {  
24    evCiz(mouseX, mouseY);  
25  } else if (secili === "kus") {  
26    kusCiz(mouseX, mouseY);  
27  }  
28 }
```

- 3. Sizin işiniz:** tıklayınca seçili şekli fare konumuna çizmek. Sol üst köşeye tıklamak tuvali temizler.
- 4.** Tuşla seçin, tıklayarak yerleştirin: birkaç ağaç, bir ev, güneş...
- 5. Eleştirel bakın:** kara kutu işe yaradı mı? Global `secili` 'yi değiştirmek ideal mi? Yapay zekâ hangi şekli bozuk çizdi, neden?

Kod zamanı

 30 dk

1. `yildizCiz(x, y)` yazın (basit bir artı işareti de olur). Tuvale 5 farklı yerde çağırın.
2. `kareAlan(kenar)` yazın; alanı **döndürsün**. Sonucu ekrana yazdırın.
3. `buyukOlan(a, b)` yazın; iki sayıdan büyük olanı döndürsün.
4. `tuvalIcindeMi(x, y)` yazın; boolean döndürsün. Fare tuvalin içindeyken arka plan yeşil, dışındayken gri olsun.
5. **Zorlu:** `cicekCiz(x, y, yaprakSayisi)`. Yaprakları bir döngü, `cos` ve `sin` ile dizin.

Ekrana ne yazılır?

sketch.js

Kopyala

Büyüt

```
1 function ikiKat(n) {  
2   n * 2;  
3 }  
4  
5 function draw() {  
6   background(240);  
7   text(ikiKat(5), 20, 40);  
8 }
```

A 10

B 5

☒ undefined

D Hata verir

Hesap yapılıyor ama sonuç geri gönderilmiyor: `return` yok. `return` olmayan fonksiyon `undefined` verir.

Hangi dikdörtgen çizilir?

sketch.js

Kopyala

Büyüt

```
1 function kutu(x, y, w, h) {  
2   rect(x, y, w, h);  
3 }  
4  
5 kutu(100, 20, 50);
```

- ☐ A (100, 20) konumunda 50 × 50 bir kare
- ☒ B Düzgün çizilmez: h parametresi undefined kalır
- ☐ C Program hata verip durur

Dört parametre var, üç argüman verildi. JavaScript hata vermez; eksik parametre `undefined` olur. Hata sessizdir.

"B" konsola yazılır mı?

sketch.js

Kopyala

Büyüt

```
1 function kontrol(can) {  
2   console.log("A");  
3   if (can <= 0) {  
4     return;  
5   }  
6   console.log("B");  
7 }  
8  
9 kontrol(0);
```

☐ A Evet☒ Hayır

can 0 olduğu için return çalışır ve fonksiyon orada biter. Yalnızca "A" yazılır.

Bugün ne öğrendik?

- ✓ Fonksiyon: adı olan, tekrar çağrılabilen kod bloğu. Tanım ayrı, çağrı ayrı.
- ✓ `setup`, `draw`, `mousePressed` adları sabittir; p5.js onları kendisi çağırır.
- ✓ Parametre tanımdaki yer tutucu, argüman çağrıdaki değerdir. Sıra önemlidir.
- ✓ `return;` erken çıkar; `return değer;` sonucu çağırana verir.
- ✓ Modüler kod: çizim, girdi, mantık ve yardımcı hesap ayrı fonksiyonlarda.
- ✓ Fonksiyon bir kara kutudur: ne aldığını ve ne verdiğini bilmek yeter.

SIRADAKİ DERS

Bu x hangi x?

Global ve yerel değişkenler, kapsam ve gölgeleme.

Sonraki sunum →

← Önceki

