

Bu x hangi x?

Bir değişken nerede doğar, nerede yaşar, nerede görünmez olur?

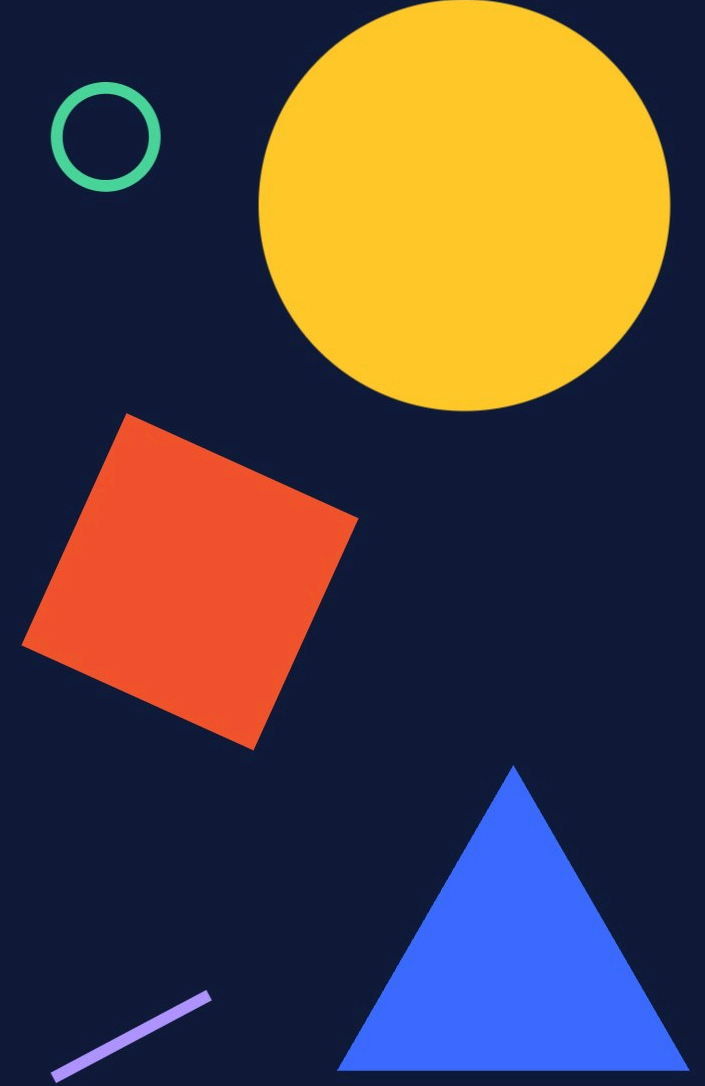
Kapsam

Global

Yerel

Argüman kopyalanır

Gölgeleme



SORU

Sınıfta iki Ahmet var. “Ahmet!” diye seslenince hangisi döner?

Programda da aynı adı taşıyan iki değişken olabilir. Hangisinin kastedildiğini belirleyen bir kural olmalı.



Kapsam: adın geçerli olduğu bölge

Kapsam (scope)

Bir değişken adının geçerli olduğu kod bölgesidir. Bu bölgenin dışında aynı ad ya tanımsızdır ya da başka bir değişkeni gösterir.

- Bir değişkenin adı, yalnızca tanımlandığı bölgede geçerlidir.
- Yanlış yerde tanımlanan skor, frame bitince yok olur.
- Yanlış adlandırılan parametre, global skoru güncellemiyormuş gibi görünür.

Üç tanım yeri

Nerede tanımlanır?	Tür	Örnek	Ne kadar yaşar?
<code>setup</code> / <code>draw</code> dışında, en üstte	Global	<code>skor</code> , <code>hedefX</code> , <code>oyunBitti</code>	Sketch çalıştığı sürece
Fonksiyon gövdesinde <code>let</code> / <code>const</code>	Yerel	Geçici hesap, döngü değişkeni	Fonksiyon bitene kadar
Fonksiyon parametresi	Yerel	<code>dikdortgenCiz(x, y, w, h)</code> içindeki <code>x</code>	Fonksiyon bitene kadar

Global değişkenler: herkes görür, değer kalır

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let skor = 0;           // global – setup/draw dışında
2 let hedefX;
3 let hedefY;
4
5 function setup() {
6   createCanvas(400, 400);
7   hedefX = width / 2;
8   hedefY = height / 2;
9 }
10
11 function draw() {
12   background(240);
13   text("Skor: " + skor, 20, 30); // global skor okunur
14   circle(hedefX, hedefY, 40);
15 }
16
17 function skorArtir(artisMiktari) {
18   skor = skor + artisMiktari; // global skor güncellenir
19 }
20
21 function mousePressed() {
22   skorArtir(1);
23 }
```

Çıktı

Çalıştır

Büyüt

Skor: 0



Tuvale tıklayın. `draw` her frame baştan çalışsa da `skor` silinmiyor. Oyun durumu bu yüzden en üstte tutulur.

Herkesin dokunabildiği kutu

Bütün değişkenleri global yaptım.
Her yerden erişiliyor, çok rahat!



projenin ilk günü

Skor neden -40 oldu?



Üç hafta sonra

Bu kutuya on dört fonksiyon
dokunabiliyor. Hangisi bozdu?



dedektiflik başlıyor

Global değişken gereklidir ama az olmalıdır: yalnızca gerçekten herkesin bilmesi gereken oyun durumu için.

Yerel değişkenler: fonksiyonla doğar, fonksiyonla ölür

sketch.js

Kopyala

Büyüt

```
1 let skor = 0; // global – doğru yer
2
3 function draw() {
4   let geciciX = mouseX; // yerel – yalnızca bu frame’de geçerli
5   // let skor = 0;      // YANLIŞ: her frame sıfırdan başlar, skor birikmez
6   background(240);
7   text(skor, 20, 30);
8   circle(geciciX, 200, 20);
9 }
```

- `geciciX` her frame fare konumundan yeniden hesaplanır. Global tutmaya gerek yok.
- `skor` birikmeli; bu yüzden global.
- Parametreler de yereldir.

`draw` **içindeki** `let`

Her frame’in başında yeniden yaratılır, sonunda yok olur. Kalıcı oyun durumu için kullanılamaz.

Kutular ne zaman doğar, ne zaman ölür?

● Çıktı

Çalıştır

Büyüt

GLOBAL: program boyunca yaşar

skor	5
------	---

```
function puanEkle() { ... } ← şu an çalışıyor
```

bonus	5
-------	---

```
let bonus = 5;  
skor = skor + bonus;  
global kutuya yazıldı: değer kalıcı
```

bonus her çağrıda yeniden yaratılır ve fonksiyon bitince yok olur. **skor** ise çağrılar arasında değerini korur; bu yüzden birikebilir.

SORU

Ekranda hangi iki sayı görünür?

sketch.js

Kopyala

Büyüt

```
1 function konumYaz(x) {  
2   // buradaki x, çağıranın x'i değil – kopya (yerel parametre)  
3   x = x + 50;  
4   text(x, 20, 40);  
5 }  
6  
7 function draw() {  
8   background(255);  
9   let x = 100;           // draw'ın yerel x'i  
10  konumYaz(x);           // argüman: 100 → parametreye kopyalanır  
11  text(x, 20, 80);       // hâlâ 100 – konumYaz içindeki atama burayı değiştirmez  
12 }
```

150 ve 100

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function konumYaz(x) {  
2   // buradaki x, çağıranın x'i değil – kopya (yerel parametre)  
3   x = x + 50;  
4   text(x, 20, 40);  
5 }  
6  
7 function draw() {  
8   background(255);  
9   let x = 100;      // draw'ın yerel x'i  
10  konumYaz(x);      // argüman: 100 → parametreye kopyalanır  
11  text(x, 20, 80);  // hâlâ 100 – konumYaz içindeki atama burayı değiştirmez  
12 }  
13  
14 function setup() {  
15   createCanvas(300, 120);  
16   textSize(28);  
17 }
```

● Çıktı

Çalıştır

Büyüt

150
100

Adları aynı olsa da iki ayrı kutu var. Çağrı anında değer **kopyalanır**.

Argüman, parametreye kopyalanır



Sonuç

Fonksiyonun içinde parametreyi değiştirmek, çağırıcı taraftaki değişkeni **değiştirmez**.

Karışmasın diye

Adları bilerek farklı seçebilirsiniz: `function konumYaz(ekranX)`. Davranış aynıdır, okuması kolaylaşır.

SORU

Skor artmıyor. Hata nerede?

sketch.js

Kopyala

Büyüt

```
1 let skor = 10; // global
2
3 function skorArtir(skor) { // parametre adı da skor – gölgeleme
4     skor = skor + 1;       // yalnızca yerel parametre artar
5     // global skor hâlâ 10
6 }
7
8 function draw() {
9     skorArtir(skor);       // argüman: 10 → yerel parametreye kopyalanır
10    text(skor, 20, 30);    // hâlâ 10 – beklenen artış olmaz
11 }
```

Gölgeleme (shadowing)

Gölgeleme

Parametre ya da yerel `let` adı, global bir değişkenle aynı olursa fonksiyonun içinde **yerel olan kazanır**. Global, o adla görünmez olur.

Burada ne oldu?

Parametrenin adı da `skor`. Artan şey yerel kopya; global `skor` hâlâ 10.

Düzeltilmiş

Kopyala

Büyüt

```
1 let skor = 10;
2
3 function skorArtir(artisMiktari) {
4   skor = skor + artisMiktari; // global skor bilinçli güncellenir
5 }
6
7 function draw() {
8   // ...
9   skorArtir(1);
10 }
```

Parametreye `artisMiktari` gibi ayırt edici bir ad verin ya da globali parametresiz, doğrudan güncelleyin. İkisini aynı adla karıştırmayın.

İki düğme, iki skor: hangisi artar?

● Çıktı

Çalıştır

Büyüt

Parametresiz: globali günceller

```
function skorArtir() {  
  skor = skor + 1;  
}
```

global skor: 10

skorArtir çağır

Parametrenin adı da skor: gölgeleme

```
function skorArtir(skor) {  
  skor = skor + 1;  
}
```

global skor: 10

skorArtir çağır

İki düğmeye de birkaç kez tıklayın. Sağdaki fonksiyon her çağrıda 11'i hesaplar ama sonucu yerel kopyaya yazar; kopya fonksiyonla birlikte yok olur.

Bir oyunda kim global, kim yerel?

Global: oyunun durumu

- skor , can , seviye
- hedefX , hedefY
- oyunBitti

Frame'ler arasında hatırlanması gerekenler.

Yerel: geçici hesap

- Bu tıklamadaki mesafe
- Döngü sayacı i
- Bir çizimde kullanılan renk

İş bitince kimsenin ihtiyaç duymadıkları.

Kaydet ve yükle

Bir oyunu kaydetmek, global durumun bir dosyaya yazılmasıdır. Yerel değişkenler kaydedilmez; yüklenince yeniden hesaplanırlar.

Kendi cümlelerinize

1. Skor neden `draw()` içinde `let` ile tanımlanmamalı?
2. Parametre adı global bir değişkenle aynı olursa ne olur?
3. İç içe iki `for` döngüsünde ikisine de `i` demek neden sorun çıkarıyordu? Bunun bugünkü konuyla ilgisi ne?
4. Her şeyi global yapsak ne kaybederiz?

Kod zamanı

 25 dk

1. Bir `can` değişkeni tutun (başlangıç 3). `canAzalt(miktar)` fonksiyonu yazın; her tıklamada 1 azaltsın, ekranda gösterin.
2. Parametrenin adını bilerek `can` yapın. Ne oldu? Neden?
3. Global hiç kullanmadan çözün: `yeniCan(eskiCan, miktar)` yeni değeri **döndürsün**; çağıran `can = yeniCan(can, 1);` yazsın.
4. İki çözümü karşılaştırın: hangisinde fonksiyon tek başına daha anlaşılır?

Konsola ne yazılır?

sketch.js

Kopyala

Büyüt

```
1 let a = 1;
2
3 function degistir() {
4   let a = 5;
5   a = a + 1;
6 }
7
8 degistir();
9 console.log(a);
```



1

B 5**C** 6**D** Hata verir

Fonksiyonun içindeki `let a` yeni, yerel bir kutu açar ve globali gölgeler. Global `a`'ya hiç dokunulmaz.

Peki bu sefer?

sketch.js

Kopyala

Büyüt

```
1 let a = 1;
2
3 function degistir() {
4   a = 5;
5   a = a + 1;
6 }
7
8 degistir();
9 console.log(a);
```

A 1**B** 5☒ **✓** 6**D** Hata verir

let yok; yeni kutu açılmıyor. Fonksiyon doğrudan global a'yı değiştiriyor. Tek bir sözcük bütün farkı yaratıyor.

Bu kod neden hata verir?

sketch.js

Kopyala

Büyüt

```
1 function draw() {  
2   background(240);  
3   for (let i = 0; i < 5; i++) {  
4     rect(i * 50, 80, 40, 40);  
5   }  
6   text(i, 20, 30);  
7 }
```

A

text sayı yazamaz.

**i** yalnızca **for** bloğunun içinde yaşar; dışarıda tanımsızdır.**C**

Döngü hiç bitmez.

let blok kapsamlıdır: süslü parantezin içinde tanımlanan, dışarıdan görünmez.

Merak edenler için kaynaklar

PATT VİRA

Pass by Value vs Pass by Reference

Argümanın parametreye kopyalanması; “kopya mı, kendisi mi?” sorusu.

[pattvira.com > intro-to-creative-coding > pass-by-value-vs-pass-by-reference](https://pattvira.com/intro-to-creative-coding/pass-by-value-vs-pass-by-reference)

JAVASCRIPT.INFO

Variables

let, const ve adlandırma üzerine ayrıntılı yazılı kaynak (İngilizce).

javascript.info/variables

MDN

let

Blok kapsamının resmî tanımı ve örnekleri (İngilizce).

[developer.mozilla.org > JavaScript > Statements > let](https://developer.mozilla.org/JavaScript/Statements/let)

THE CODING TRAIN

Parameters and Arguments

Parametrelerin yerel olduğunu örnekle gösteren video.

[thecodingtrain.com > 5-functions > 2-arguments](https://thecodingtrain.com/5-functions/2-arguments)

Bugün ne öğrendik?

- ✓ Kapsam: bir adın geçerli olduğu bölge.
- ✓ Global değişkenler en üstte tanımlanır; frame'ler arasında yaşar, her fonksiyondan görülür.
- ✓ Yerel değişkenler ve parametreler fonksiyon bitince yok olur.
- ✓ Argümanın değeri parametreye kopyalanır; parametreyi değiştirmek çağıranı etkilemez.
- ✓ Aynı adlı yerel, globali gölgeler. Parametrelere ayırt edici adlar verin.

SIRADAKİ DERS

Oyun ne kadar hızlı akıyor?

Oyun döngüsü, FPS ve frameRate.

Sonraki sunum →

← Önceki