

Değişkenler ve operatörler

Programın bir şeyi aklında tutmasını ve doğru mu yanlış mı diye sormasını öğreneceğiz.

Değişken nedir?

let ve const

number, string, boolean

==, &&, ||, !

SORU

Oyunda skor şu an 7. Program bunu nerede tutuyor?

`draw` her frame baştan çalışıyor. Bir sonraki frame'de skorun 7 olduğunu nereden bilecek?



Değişken: üstünde adı yazan bir kutu



- Kutunun bir **adı** var: skor .
- İçinde bir **değer** duruyor: 7.
- İsteddiğimiz zaman içine bakabiliriz. Buna **okumak** denir.
- İstedğimiz zaman içindekini değiştirebiliriz. Buna **atamak** denir.

Değişken

Değişken (variable)

Program çalışırken değeri okunabilen, değiştirilebilen ya da sabit tutulabilen, isimlendirilmiş bellek alanıdır.

sketch.js

Kopyala

Büyüt

```
let skor = 0;
```

let

Yeni bir kutu aç.

skor

Kutunun adı.

=

İçine şunu koy.

0

İlk değer.

Kutunun adı aynı kalır, içi değişir



`skor = skor + 1` **nasıl okunur?**

Önce sağ taraf hesaplanır: kutunun içine 1 ekle. Sonra sonuç aynı kutuya geri yazılır. Bu bir denklem değil, bir **komut**.

Skor her frame 1 artsın

sketch.js

Kopyala

Düzenle

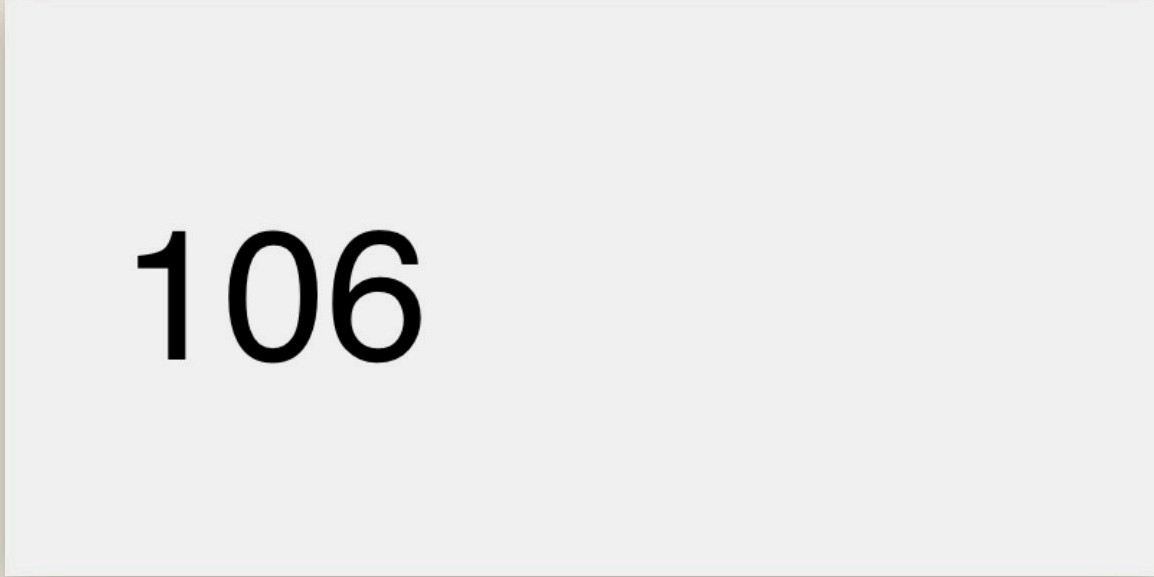
Büyüt

```
1 let skor = 0;
2
3 function setup() {
4   createCanvas(400, 200);
5 }
6
7 function draw() {
8   background(240);
9   skor = skor + 1;
10
11   fill(0);
12   textSize(64);
13   text(skor, 40, 125);
14 }
```

Çıktı

Çalıştır

Büyüt



Kutu en üstte, bir kez açılıyor. `draw` her frame aynı kutuya bakıp içeriğini 1 artırıyor.

SORU

**let skor = 0; satırını draw 'ın
içine taşırsak ne görürüz?**

Sayı yine artar mı? Tahmininizi söyleyin, sonra deneyelim.



Ekranı hep 1 yazar

sketch.js

Kopyala

Düzenle

Büyüt

```
1 function setup() {  
2   createCanvas(400, 200);  
3 }  
4  
5 function draw() {  
6   background(240);  
7   let skor = 0;  
8   skor = skor + 1;  
9  
10  fill(0);  
11  textSize(64);  
12  text(skor, 40, 125);  
13 }
```

Çıktı

Çalıştır

Büyüt

1

Her frame yeni bir kutu

`draw` içinde açılan kutu, frame bitince çöpe gider. Oyun boyunca yaşaması gereken değerleri `setup` ve `draw` 'ın **dışında**, en üstte tanımlayın.

Kutu açmanın üç yolu

let

İçi sonradan **değişebilen** kutu.
Skor, konum, can.

const

Kilitli kutu. Bir kez değer verilir,
sonra **değiştirilemez**. Hedef
skor, tuval ölçüsü.

var

Eski sözdizimi. Beklenmedik
davranışlara yol açabilir. Bu
derste kullanmıyoruz.

`let` ve `const` **blok kapsamlıdır**: süslü parantez `{ }` içinde tanımlanırsa dışarıdan erişilemez. `var` ise fonksiyon kapsamlıdır.

Dört kutu, üç tür

sketch.js

Kopyala

Büyüt

```
1 let skor = 0; // Tam sayı (number)
2 let oyunAdi = "Daire Tıklama"; // Metin (string)
3 let oyunBitti = false; // Mantıksal değer (boolean)
4 const HEDEF_SKOR = 10; // Sabit – sonradan değiştirilmez (BÜYÜK_HARF örfi)
```

skor

0

number

oyunAdi

"Daire Tıklama"

string

oyunBitti

false

boolean

HEDEF_SKOR 

10

number · sabit

Üç temel veri türü

Tür	Ne tutar?	Örnek	Oyunda nerede?
number	Tam ya da ondalıklı sayı	42 , 3.14 , -7	Konum, boyut, skor, can
string	Metin (tırnak içinde)	"Merhaba" , 'p5.js'	Skor yazısı, "Kazandın!" mesajı
boolean	Doğru ya da yanlış	true , false	oyunBitti , hedefVuruldu

JavaScript'te `undefined` , `null` , `object` gibi başka türler de var. Şimdilik bu üçü yeterli.

Kutuya iyi bir ad koymak

Anlamlı adlar

`hedefX`, `topHizi`, `kalanSure`

Belirsiz adlar

`a`, `b`, `temp1` : kodu izlemeyi zorlaştırır.

`x` ve `y` gibi tek harfli adlar koordinat için yaygındır.

camelCase (deve notasyonu)

İlk kelime küçük harfle başlar; sonraki kelimelerin ilk harfi büyüktür. Arada boşluk ya da alt çizgi yoktur.

Sabitlerde BÜYÜK_HARF

Tüm harfler büyük, kelimeler alt çizgiyle ayrılır:

`HEDEF_SKOR`, `MAKS_CAN`.

İkisi de dilin zorunlu kuralı değil, topluluğun benimsediği bir **örf** (convention). Örneklerimizde hep bu düzeni kullanacağız.

SORU

`x = 5` ile `x === 5` aynı şey mi?

Biri bir şey yapıyor, diğeri bir şey soruyor. Hangisi hangisi?



Tek eşittir atar, üç eşittir sorar

X = 5

Atama "x kutusuna 5 koy." Kutunun içi değişir.

X === 5

Karşılaştırma "x'in içindeki 5 mi?" Kutu değişmez; cevap `true` ya da `false` olur.

Dikkat

Eşitlik sorusu için `==` de vardır ama tür dönüşümü yapar: `"5" == 5` sonucu `true` çıkar. Örneklerimizde hem değeri hem türü kontrol eden `===` kullanıyoruz.

Metinle (string) neler yapılır?

İşlem	Yazımı	Örnek	Sonuç
Birleştirme	<code>"a" + "b"</code>	<code>"Skor: " + 7</code>	<code>"Skor: 7"</code>
Uzunluk	<code>metin.length</code>	<code>"p5.js".length</code>	<code>5</code>
Büyük / küçük harf	<code>.toUpperCase()</code> / <code>.toLowerCase()</code>	<code>"Hi".toUpperCase()</code>	<code>"HI"</code>
Eşitlik	<code>===</code>	<code>durum === "bitti"</code>	<code>true</code> ya da <code>false</code>

Mini oyunlarımız çoğunlukla sayı ve boolean ile ilerler. Yine de skor yazısı, seviye adı ya da "Kazandın!" gibi mesajlar string'dir.

Skoru ekrana yazmak

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let skor = 7;
2 let oyunAdi = "Daire Tıklama";
3
4 function setup() {
5   createCanvas(400, 200);
6 }
7
8 function draw() {
9   background(240);
10  fill(0);
11
12  let mesaj = "Skor: " + skor; // "Skor: 7"
13  text(mesaj, 20, 40);
14
15  if (oyunAdi.length > 0) {
16    text(oyunAdi.toUpperCase(), 20, 70);
17  }
18 }
```

Çıktı

Çalıştır

Büyüt

Skor: 7

DAIRE TIKLAMA

- Skor `number` olarak tutulur; ekrana yazarken metinle birleştirilir. Metnin rengi `fill` ile ayarlanır.
- Sayıyı metne çevirmenin başka yolları: `String(skor)` ya da `skor + ""`. Tersi (`Number("10")`, `parseInt`) şimdilik gerekmiyor.

Karşılaştırma operatörleri

Her karşılaştırmanın sonucu bir boolean'dır: `true` ya da `false`.

Operatör	Sorduğu soru	Örnek	Sonuç
<code>===</code>	Tam eşit mi? (değer + tür)	<code>skor === 10</code>	skor 10 ise <code>true</code> ; 7 ise <code>false</code>
<code>!==</code>	Eşit değil mi?	<code>durum !== "bitti"</code>	durum <code>"oynuyor"</code> ise <code>true</code>
<code><</code>	Küçük mü?	<code>x < 100</code>	x 50 ise <code>true</code> ; 150 ise <code>false</code>
<code>></code>	Büyük mü?	<code>y > 200</code>	y 250 ise <code>true</code> ; 100 ise <code>false</code>
<code><=</code>	Küçük ya da eşit mi?	<code>skor <= 5</code>	skor 5 ise <code>true</code> ; 7 ise <code>false</code>
<code>>=</code>	Büyük ya da eşit mi?	<code>can >= 1</code>	can 1 ise <code>true</code> ; 0 ise <code>false</code>

`==` ile `===` farkı: tür dönüşümü

`"10"` bir **string**, `10` ise bir **number**. Görünüşte aynılar ama türleri farklı.

- `==` karşılaştırmadan önce türleri birbirine çevirmeye çalışır (**type coercion**).
- `===` hem değere hem türe bakar.
- Aynı ayırım "eşit değil" için de geçerli: `!=` dönüştürür, `!==` dönüştürmez.

İfade	Sonuç
<code>"10" == 10</code>	true
<code>"10" === 10</code>	false
<code>10 === 10</code>	true
<code>"10" != 10</code>	false
<code>"10" !== 10</code>	true

Sürpriz eşleşmeleri önlemek için örneklerimizde `===` ve `!==` kullanıyoruz.

SORU

“Canım varsa ve oyun bitmediyse devam et.”

Burada iki ayrı soru var. İkisini tek bir soruda nasıl birleştiririz?



Mantıksal operatörler

Karşılaştırmaların ürettiği `true` / `false` değerlerini birleştirirler. Sonuç yine bir boolean'dır.

Operatör	Anlamı	Örnek	Sonuç
<code>&&</code>	VE: iki koşul da doğru olmalı	<code>x > 0 && x < 400</code>	x 150 iken <code>true</code> ; 500 iken <code>false</code>
<code> </code>	VEYA: en az biri doğru yeter	<code>can === 0 sure === 0</code>	can 0, sure 30 iken <code>true</code>
<code>!</code>	DEĞİL: değeri tersine çevirir	<code>!oyunBitti</code>	oyunBitti <code>false</code> ise <code>true</code>

Sorunun cevabı: `can > 0 && !oyunBitti`

İki yeni sözcük

Operand

Bir operatörün üzerinde işlem yaptığı değer ya da ifadedir.

- `skor === 10` : operatör `===` ; operandlar `skor` ve `10` .
- `x > 0 && x < 400` : `&&` 'nin operandları iki koşul.
- `!oyunBitti` : `!` tek operand alır.

Doğruluk tablosu

Bir ifadenin, farklı girdi değerlerinde `true` mu `false` mu olduğunu satır satır gösteren tablodur.

- İki operand varsa $2 \times 2 = 4$ **durum**.
- Tek operand varsa **2 durum**.

&& (VE): ikisi de doğru olmalı

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

Oyunda: `can > 0 && !oyunBitti`

can	oyunBitti	can > 0	!oyunBitti	Sonuç
3	false	true	true	true
0	false	false	true	false
3	true	true	false	false

|| (VEYA): en az biri doğru yeter

A	B	A B
true	true	true
true	false	true
false	true	true
false	false	false

Oyunda: `can === 0 || sure === 0`

can	sure	can === 0	sure === 0	Sonuç
0	45	true	false	true
2	0	false	true	true
2	45	false	false	false

! (DEĞİL): tersine çevir

A	!A
true	false
false	true

- ! tek bir boolean değeri tersine çevirir.
- !oyunBitti : oyun bitmediyse true , bittiyse false .
- "Oyun bitmediyse" diye okunur.

Üç tablo yan yana

A && B (VE)

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

Yalnızca ikisi de true → true

A || B (VEYA)

A	B	A B
true	true	true
true	false	true
false	true	true
false	false	false

En az biri true → true

!A (DEĞİL)

A	!A
true	false
false	true

tersine çevirir

İki operandlı operatörlerde 4 durum, ! için 2 durum.

SORU

**Matematikte $20 < x < 60$ yazarız.
JavaScript'te de çalışır mı?**

x = 70 olsun. 70, 20 ile 60 arasında değil. Sizce sonuç ne çıkar?



Çalışır ama yanlış cevap verir

- 1 JavaScript önce `20 < 70` sorusunu çözer: `true`.
- 2 Sonra `true < 60` sorusunu sorar.
- 3 Sonuç: `true`. Oysa 70 aralıkta değil!

sketch.js

Kopyala

Büyüt

```
1 // Yanlış beklenti (matematik yazımı - JS'te güvenilir değil):  
2 // 20 < x < 60  
3  
4 // Doğru:  
5 20 < x && x < 60
```

Aralık her zaman `&&` ile yazılır

Tuvalin içinde mi? `x > 0 && x < width`

Python'da olur, burada olmaz

Bazı diller zincirli karşılaştırmaya izin verir.

Matematikte geçerli görünen her yazım kodda aynı anlama gelmez; dilin kuralları esastır.

Mantıksal operatörlerle şekil boyamak: &&

sketch.js · satır 18-42

Kopyala

Büyüt

```

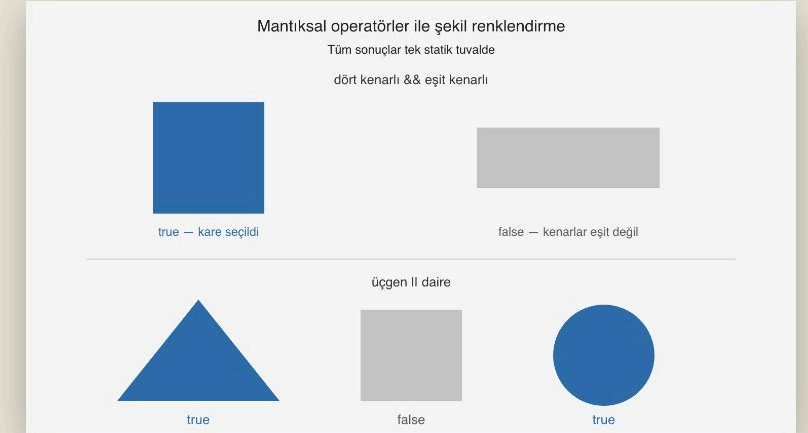
18 let secili = color(45, 110, 170);
19 let diger = color(195);
20
21 // && : dört kenarlı VE eşit kenarlı
22 fill(55);
23 textSize(13);
24 text("dört kenarlı && eşit kenarlı", 380, 82);
25
26 let dortKenar = true;
27 let esitKenar = true;
28 if (dortKenar && esitKenar) {
29   fill(secili);
30 } else {
31   fill(diger);
32 }
33 rect(125, 100, 110, 110); // kare
34
35 dortKenar = true;
36 esitKenar = false;
37 if (dortKenar && esitKenar) {
38   fill(secili);
39 } else {
40   fill(diger);
41 }
42 rect(445, 125, 180, 60); // dikdörtgen

```

Çıktı

Çalıştır

Büyüt



Mantıksal operatörlerle şekil boyamak: | |

sketch.js · satır 58-82

Kopyala

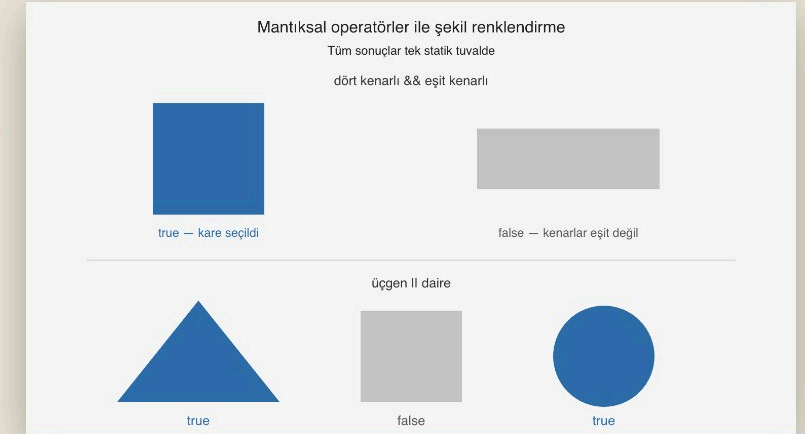
Büyüt

```
58 let ucgenMi = true;
59 let daireMi = false;
60 if (ucgenMi || daireMi) {
61   fill(secili);
62 } else {
63   fill(diger);
64 }
65 triangle(90, 395, 170, 295, 250, 395);
66
67 ucgenMi = false;
68 daireMi = false;
69 if (ucgenMi || daireMi) {
70   fill(secili);
71 } else {
72   fill(diger);
73 }
74 rect(330, 305, 100, 90); // kare
75
76 ucgenMi = false;
77 daireMi = true;
78 if (ucgenMi || daireMi) {
79   fill(secili);
80 } else {
81   fill(diger);
82 }
```

Çıktı

Çalıştır

Büyüt



Bu örnekteki iki yeni komut

Komut	Ne yapar?
<code>triangle(x1, y1, x2, y2, x3, y3)</code>	Verilen üç köşe noktasını birleştirerek üçgen çizer.
<code>color(r, g, b)</code>	Bir RGB rengini değer olarak oluşturur. Değişkende saklanır; <code>fill</code> ve <code>stroke</code> ile kullanılır.

sketch.js

Kopyala

Büyüt

```
let secili = color(45, 110, 170); // rengi kutuya koy; çizim yapmaz
fill(secili);                    // dolgu rengi olarak kullan
```

Rengi bir kez tanımlayıp birçok yerde kullanabilirsiniz. Değiştirmek istediğinizde tek satır değişir.

İkiden fazla koşulu birleştirmek

Biçim	Örnek	Ne zaman <code>true</code> ?
Üçlü VE	<code>skor >= 5 && can > 0 && !oyunBitti</code>	Üçü birden doğruysa
Üçlü VEYA	<code>can === 0 sure === 0 skor >= 100</code>	En az biri doğruysa
VE + VEYA	<code>(skor >= 10 && can > 0) oyunBitti</code>	Skor ve can koşulu tutarsa ya da oyun bitmişse
Tuval sınırı	<code>x > 0 && x < 400 && y > 0 && y < 400</code>	Nokta tuvalin içindeyse

Sıra ve parantez

İfade soldan sağa değerlendirilir; `&&`, `||` 'den önce işlenir. Karışıklığı önlemek için parantez kullanın.

Kısa devre: ikinci soruya hiç bakılmayabilir

&&

İlk koşul `false` ise sonuç zaten `false`. İkinciye bakılmaz.

`can > 0 && skor === hedefSkor`: `can` 0 ise skor kontrol edilmez.

||

İlk koşul `true` ise sonuç zaten `true`. İkinciye bakılmaz.

Neden önemli?

Bu davranışa **kısa devre** (short-circuit) denir. Gereksiz karşılaştırmayı atlar. Ama ikinci tarafta bir iş yapılıyorsa (skoru artırmak, bir fonksiyon çağırmak) o kod **hiç çalışmaz**.

Beş ifadenin her biri true mu, false mu?

sketch.js · satır 1-6

Kopyala

Büyüt

```
1 let x = 150;  
2 let y = 200;  
3 let skor = 7;  
4 let can = 2;  
5 let sure = 0;  
6 let oyunBitti = false;
```

sketch.js

Kopyala

Büyüt

```
x > 0 && x < 400 && y > 0 && y < 400  
skor >= 10 && !oyunBitti  
can > 0 && skor < 10 && !oyunBitti  
can === 0 || sure === 0  
(skor >= 10 && can > 0) || oyunBitti
```

Cevaplar tuvalde

sketch.js · satır 17-30

Kopyala

Büyüt

```
17 // Dört koşullu VE: nokta tuval içinde mi?
18 let tuvalIcinde = (x > 0 && x < 400 && y > 0 && y < 400); // true
19
20 // İkili VE: skor yeterli mi ve oyun bitmedi mi?
21 let kazandi = (skor >= 10 && !oyunBitti); // false – skor henüz 7
22
23 // Üçlü VE: oyun devam edebilir mi?
24 let oyunDevam = (can > 0 && skor < 10 && !oyunBitti); // true
25
26 // VEYA: oyun bitti mi? (can veya süre tükendi)
27 let oyunSonu = (can === 0 || sure === 0); // true – sure 0
28
29 // Parantezli karışım: kazanma veya zorunlu bitiş
30 let ekranMesaji = (skor >= 10 && can > 0) || oyunBitti; // false
```

Çıktı

Çalıştır

Büyüt

```
tuvalIcinde: true
kazandi: false
oyunDevam: true
oyunSonu: true
ekranMesaji: false
```

Kutularla oynayın

 15 dk

1. Adınızı bir `let`, doğum yılınızı başka bir `let` ile tutun. Tuvale "Ada, 2006" gibi tek satır yazdırın.
2. Yaşınızı bu yıldan doğum yılınızı çıkararak hesaplayıp ekrana yazdırın.
3. En üstte `let x = 0;` tanımlayın. Her frame 2 artsın ve bir daire o konumda dursun. Tuvalin sonuna gelince başa dönsün.
4. `const HIZ = 2;` tanımlayıp 2 yerine onu kullanın. Sonra `draw` içinde `HIZ = 5;` yazmayı deneyin. Konsol ne diyor?

Zorlu görev: tabloyu siz doldurun

🕒 15 dk

a	b	!(a && b)	!a !b
true	true	?	?
true	false	?	?
false	true	?	?
false	false	?	?

1. Tabloyu önce kâğıtta doldurun.
2. Son iki sütunda ne fark ettiniz?
3. Dört durumu bir sketch'te değişkenlerle kurup `text` ile ekrana yazdırarak doğrulayın.
4. "Oyun devam ediyor" ifadesinin tersini iki farklı biçimde yazın.

Sonunda **b** kaç olur?

sketch.js

Kopyala

Büyüt

```
1 let a = 5;  
2 let b = a;  
3 a = 10;
```

☒ 5☐ B 10☐ C 15☐ D Hata verir

`b = a`, `a`'nın o andaki değerini `b` kutusuna **kopyalar**.
Kutular birbirine bağlanmaz.

"Skor: " + 3 + 4 ifadesinin sonucu ne?

A "Skor: 7"

✓ "Skor: 34"

C 7

D Hata verir

Soldan sağa: önce "Skor: " + 3 bir metin olur, sonra ona 4 eklenir. Toplamak istiyorsanız parantez: "Skor: " + (3 + 4) .

can = 0, sure = 30, oyunBitti = false. Sonuç ne?

sketch.js

Kopyala

Büyüt

```
can > 0 || sure > 0 && !oyunBitti
```

☒ true☐ false

&& önce işlenir: `sure > 0 && !oyunBitti` → `true`. Sonra `false || true` → `true`. Parantez yazsaydık tartışmaya gerek kalmazdı.

Bugün ne öğrendik?

- ✓ Değişken, adı olan bir kutudur. `let` değişebilir, `const` değişmez.
- ✓ Kalıcı değerler `setup` ve `draw` 'ın dışında tanımlanır.
- ✓ Üç temel tür: number, string, boolean.
- ✓ `=` atar; `===` sorar. Karşılaştırmanın sonucu hep boolean'dır.
- ✓ `&&` ikisi de, `||` en az biri, `!` tersi.
- ✓ Aralık `20 < x && x < 60` diye yazılır; zincir yazılmaz.

SIRADAKİ DERS

Program nasıl karar verir?

if, else if ve else.

Sonraki sunum →

← Önceki