

Diziler: birçok değer, tek bir ad

Elli düşmanı elli ayrı değişkende tutmadan yönetmeyi öğreneceğiz.

Dizi ve indeks

for ile dolaşmak

Paralel diziler

push ve pop

Yığın

SORU

Ekranda 50 düşman var. Konumlarını nerede tutacaksınız?

sketch.js

Kopyala

Büyüt

```
let x1 = 60;  
let x2 = 340;  
let x3 = 260;  
// ... x50 ?
```

Peki hepsini 2 piksel sağa kaydırmak için kaç satır yazacaksınız?

Dizi: yan yana duran, numaralı kutular

sketch.js

Kopyala

Büyüt

```
let skorlar = [10, 25, 40];
```

10	25	40
0	1	2

üstte değerler • altta indeksler

Dizi (array)

Köşeli parantez `[]` ile tanımlanan, indeksle erişilen **sıralı** veri koleksiyonudur.

Dizi tanımlamak

sketch.js

Kopyala

Büyüt

```
1 let skorlar = [10, 25, 40];           // Sayı dizisi
2 let renkler = [255, 120, 80];         // RGB bileşenleri
3 let mesajlar = ["dogru", "yuksek", "dusuk"]; // Metin dizisi
4 let bosDizi = [];                     // Boş dizi
```

İfade	Anlamı
<code>[]</code>	Boş dizi oluşturur
<code>[a, b, c]</code>	Başlangıç elemanlarıyla dizi
<code>dizi[0]</code>	İlk eleman (indeks 0)
<code>dizi.length</code>	Eleman sayısı

Envanter çubuğu bir dizidir

● Çıktı

Çalıştır

Büyüt

Fareyi bir kutunun üstüne getirin



Dokuz kutu, tek bir ad. Hangi kutuyu istediğinizi bir sayıyla söylüyorsunuz.

Oyunlarda dizi her yerde

Envanter

Minecraft'ın alt çubuğunda 9 yuva var; her biri numaralı.

Yüksek skor tablosu

Atari salonlarındaki ilk 10 listesi: sıralı on sayı.

Sıradaki parçalar

Tetris'te "sonraki" kutusunda bekleyen parçalar bir sırada durur.

Düşmanlar, mermiler

Space Invaders'taki her uzaylı, her mermi bir listenin elemanı.

Ortak nokta: aynı türden **birçok** değer ve onları sırayla işleme ihtiyacı.

SORU

skorlar[1] hangi değeri verir?

sketch.js

Kopyala

Büyüt

```
let skorlar = [10, 25, 40];
```

Peki skorlar[3] ?

25. Çünkü saymaya 0'dan başlıyoruz

10	25	40	?
0	1	2	3

`skorlar.length` $\equiv$ 3 · son geçerli indeks 2

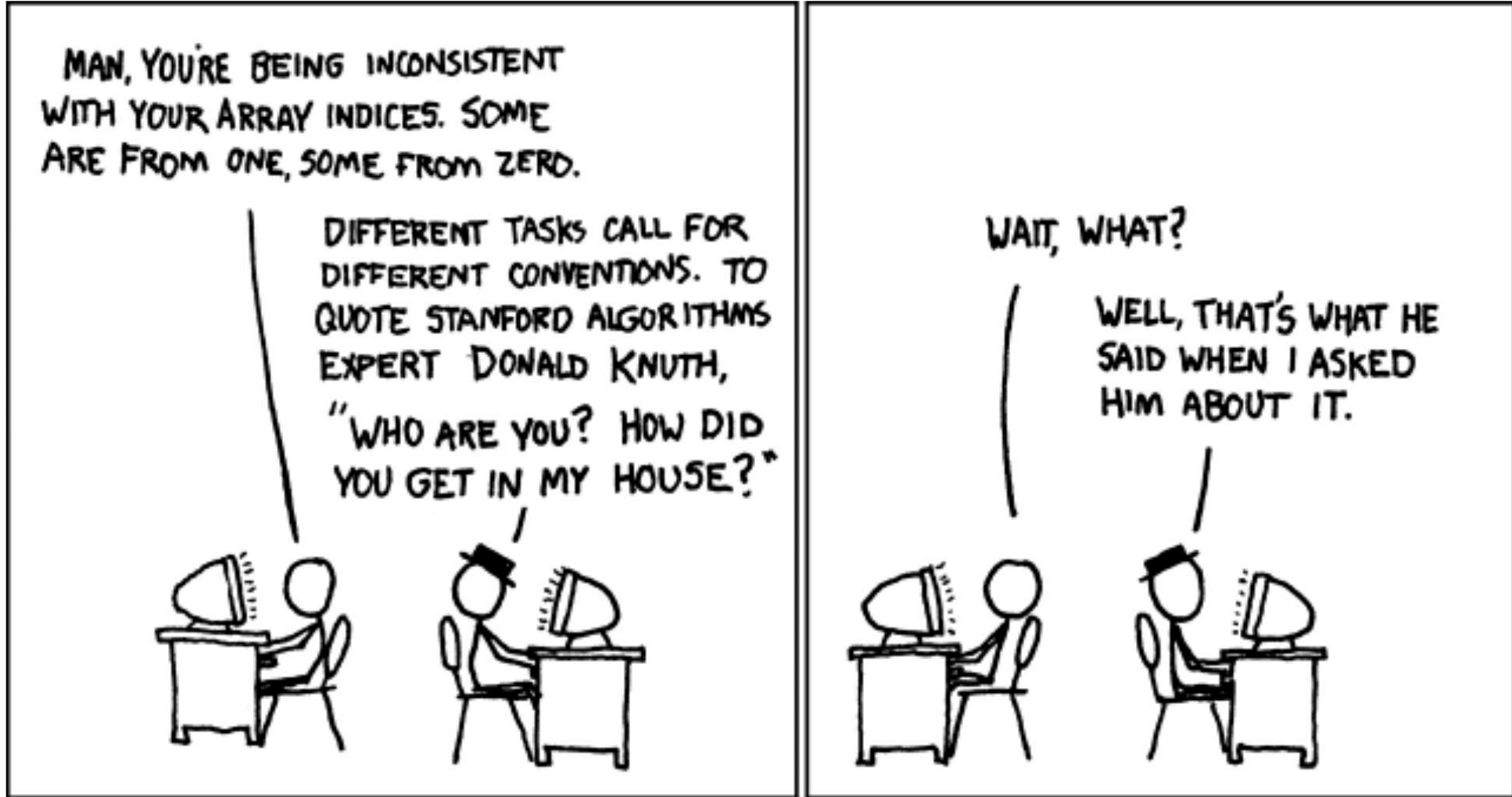
İndeksler 0'dan başlar

`skorlar[1]` **ikinci** elemandır.

Dördüncü kutu yok

`skorlar[3]` hata vermez; `undefined` döner.
Hata sessizdir.

Programcıların bitmeyen tartışması



"İndekslerinde tutarsızsın: bazıları birden, bazıları sıfırdan başlıyor." · Kaynak: xkcd.com/163, Randall Munroe (CC BY-NC 2.5)

Olmayan indekse yazmak: delikli dizi

sketch.js

Kopyala

Büyüt

```
1 let a = [1];  
2 a[3] = 44;  
3 // a.length → 4; a[1] ve a[2] boş (empty); okununca undefined
```

1	boş	boş	44
0	1	2	3

Boş slot (hole)

Araya hiç doldurulmamış hücreler girer.

Okununca `undefined` verirler. Konsolda `[1, <2 empty slots>, 44]` gibi görünür.

`push` ya da 0, 1, 2... sırasıyla atama boşluk bırakmaz. Oyun listelerinde bu delikleri oluşturmayın.

Dizi, ilgili değerleri bir arada taşır

İndeks	renkler[indeks]	Anlamı
0	255	Kırmızı
1	120	Yeşil
2	80	Mavi
3	undefined	Karşılığı yok

sketch.js

Kopyala

Büyüt

```
let renkler = [255, 120, 80];  
fill(renkler[0], renkler[1], renkler[2]);
```

`fill(255, 120, 80)` ile aynı sonuç.

Karışık türler

JavaScript aynı dizide farklı türlere izin verir: `[1, "iki", true]`. Oyun kodunda tek tür tutmak okumayı kolaylaştırır.

Kâğıt kalem

 5 dk

sketch.js

Kopyala

Büyüt

```
let skorlar = [10, 25, 40];
```

1. `skorlar[0]` ve `skorlar[2]` nedir? `skorlar[3]` neden sorunlu?
2. `skorlar.length` kaçtır? Son geçerli indeks hangisi?
3. Üç skoru alt alta yazdıracak `for` döngüsünün koşulunu yazın: `i < ...`

Elemanı okumak ve değiştirmek

sketch.js

Kopyala

Büyüt

```
1 let tahminler = [7, 12, 5];  
2 tahminler[0] = 8;           // İlk elemanı 8 yap  
3 let ortanca = tahminler[1]; // 12
```

7	12	5
0	1	2

↓ tahminler[0] = 8

8	12	5
0	1	2

Her hücre kendi başına bir değişken gibidir: okunur, üstüne yazılır. Diğer hücreler etkilenmez.

for ile diziyi dolaşmak

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let tahminler = [7, 12, 5, 9];
2
3 function setup() {
4   createCanvas(400, 300);
5 }
6
7 function draw() {
8   background(240);
9   fill(30);
10  textSize(18);
11  textAlign(LEFT, TOP);
12
13  for (let i = 0; i < tahminler.length; i = i + 1) {
14    let metin = (i + 1) + ". tahmin: " + tahminler[i];
15    text(metin, 20, 20 + i * 28);
16  }
17 }
```

● Çıktı

Çalıştır

Büyüt

```
1. tahmin: 7
2. tahmin: 12
3. tahmin: 5
4. tahmin: 9
```

`i < tahminler.length` koşulu indeksi dizinin sınırları içinde tutar. Diziye bir sayı daha ekleyin: döngüye dokunmanız gerekmez.

SORU

Koşulu `i <= tahminler.length` yaparsak?

sketch.js

Kopyala

Büyüt

```
for (let i = 0; i <= tahminler.length; i = i + 1) {  
  text(tahminler[i], 20, 20 + i * 28);  
}
```

Dizide 4 eleman var. Döngü kaç tur atar? Son turda ne okunur?

Bir tur fazla: son satırda `undefined`

i	tahminler[i]
0	7
1	12
2	5
3	9
4	undefined

Sınırdaki bir fazla (off-by-one)

`length` eleman sayısıdır: 4. Son geçerli indeks ise `length - 1`: 3.

Kalıp

```
for (let i = 0; i < dizi.length; i++)
```

Küçüktür, küçük eşit değil.

SORU

Her hedefin bir x 'i ve bir y 'si var. Tek bir dizi yeter mi?

Ekranda üç hedef daire olacak. Altı sayıyı nasıl düzenlersiniz ki hangisinin hangi daireye ait olduğu karışmasın?



Paralel diziler: aynı indeks, aynı nesne

hedefXListesi

80	200	320
0	1	2

hedefYListesi

100	150	200
0	1	2

1 numaralı hedef: (200, 150)

Paralel dizi

Bir nesnenin özelliklerini ayrı dizilerde, **aynı indekste** tutma modeli. `hedefXListesi[i]` ile `hedefYListesi[i]` aynı hedefin koordinatlarıdır.

İleride nesneler ve iki boyutlu diziler gibi başka yollar da göreceğiz. Şimdilik paralel dizi yeterli.

Konum listesinden hedef çizmek

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let hedefXListesi = [80, 200, 320];
2 let hedefYListesi = [100, 150, 200];
3 const YARICAP = 25;
4
5 function setup() {
6   createCanvas(400, 300);
7   noStroke();
8 }
9
10 function draw() {
11   background(25, 30, 45);
12   fill(255, 200, 50);
13
14   for (let i = 0; i < hedefXListesi.length; i = i + 1) {
15     ellipse(hedefXListesi[i], hedefYListesi[i], YARICAP * 2, YARICAP * 2);
16   }
17 }
```

● Çıktı

Çalıştır

Büyüt



Aynı `i` hem `x`'i hem `y`'yi okur; döngü her turda bir hedef çizer. İki diziye birer sayı daha ekleyin: dördüncü hedef kendiliğinden belirir.

Biz ve düşmanlar

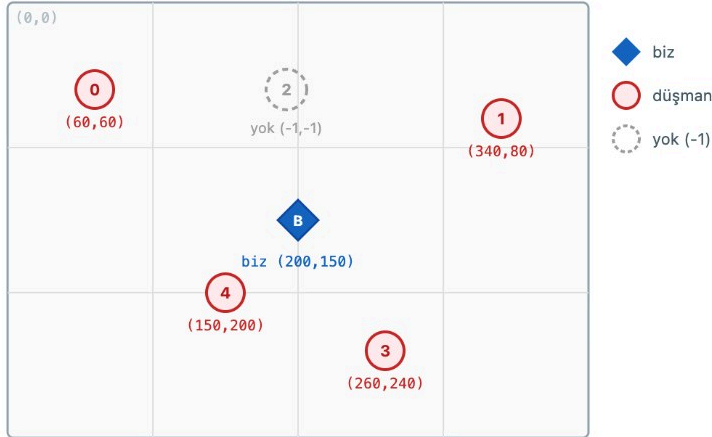
15 dk

Düşmanlar — paralel diziler + 2B sahne

	[0]	[1]	[2]	[3]	[4]
dusmanX	60	340	-1	260	150
dusmanY	60	80	-1	240	200

[2]: X ve Y ikisi de -1

Aynı indeks = bir düşman: (dusmanX[i], dusmanY[i]). Boş slotta ikisi de -1; sahnede çizilmez.
biz = (200, 150): tek nokta — dizi gerekmez. Düşmanlar çok → paralel dizi.



sketch.js

Kopyala

Büyüt

```
let bizX = 200, bizY = 150;
let dusmanX = [60, 340, -1, 260, 150];
let dusmanY = [60, 80, -1, 240, 200]; // [2] ikisi de -1 → yok
```

1. `dusmanX.length` kaç? Sahnede gerçekten kaç düşman var?
2. `(dusmanX[0], dusmanY[0])` neresi? `[2]` neden sahnede yok?
3. Bize en yakın düşman hangi indekste? `for` ile `-1` olanları atlayıp `dist` ile en küçüğü bulun.

SORU

Oyuncu her tıkladığında listeye yeni bir eleman eklenecek.

Dizinin kaç eleman tutacağını baştan bilmiyoruz. Çalışırken nasıl büyütürüz?



push() ve pop()

push()

Dizinin **sonuna** eleman ekler. `length` bir artar.

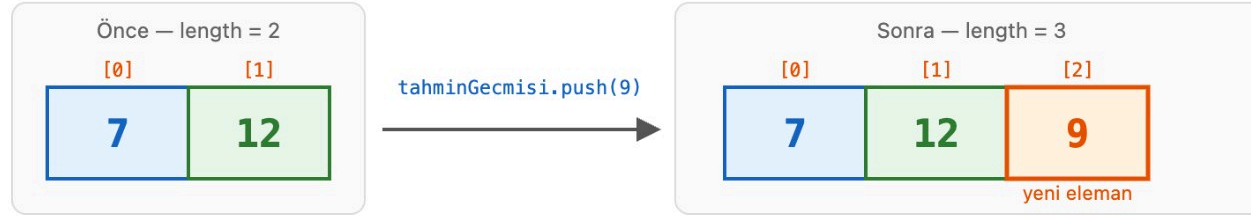
pop()

Dizinin **son** elemanını çıkarır ve o değeri döndürür. `length` bir azalır.

İşlem	Örnek	Önce	Sonra
Sona ekle	<code>liste.push(42)</code>	<code>[10, 20]</code>	<code>[10, 20, 42]</code>
Sondan çıkar	<code>liste.pop()</code>	<code>[10, 20, 42]</code>	<code>[10, 20]</code> ; 42 döner
Eleman sayısı	<code>liste.length</code>	<code>[10, 20]</code>	2

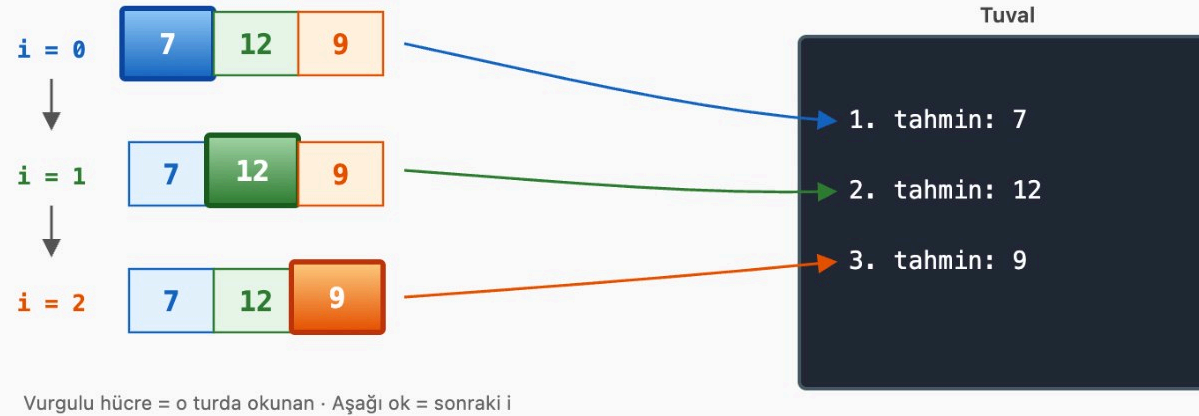
push(9) : önce ve sonra

tahminGecmisi.push(9) — sona ekleme



for ile dolaşma

```
for (let i = 0; i < tahminGecmisi.length; i = i + 1) {  
  text((i + 1) + ". tahmin: " + tahminGecmisi[i], 30, 80 + i * 24);  
}
```



Üstte dizi büyüyor; altta `for` döngüsü her turda bir sonraki hücreyi okuyor.

Tahmin geçmişi: her tıklamada bir eleman

sketch.js

Kopyala

Düzenle

Büyüt

```
1 let tahminGecmisi = [];  
2  
3 function setup() {  
4   createCanvas(400, 320);  
5 }  
6  
7 function draw() {  
8   background(240);  
9   fill(30);  
10  textSize(18);  
11  textAlign(LEFT, TOP);  
12  text("Tıklayın · eleman sayısı: " + tahminGecmisi.length, 20, 12);  
13  for (let i = 0; i < tahminGecmisi.length; i = i + 1) {  
14    text((i + 1) + ". tahmin: " + tahminGecmisi[i], 20, 44 + i * 26);  
15  }  
16 }  
17  
18 function mousePressed() {  
19   let yeniTahmin = floor(random(1, 11)); // 1-10 arası tam sayı  
20   tahminGecmisi.push(yeniTahmin);  
21 }
```

● Çıktı

Çalıştır

Büyüt

Tıklayın · eleman sayısı: 0

floor() : aşağı yuvarla

floor()

Verilen sayıyı aşağı yuvarlayan p5.js fonksiyonu. Tam sayı elde etmek için `random()` ile birlikte kullanılır.

İfade	Sonuç
<code>random(1, 11)</code>	1.00 ... 10.99
<code>floor(7.93)</code>	7
<code>floor(random(1, 11))</code>	1, 2, ... 10

Oyunlarda

Zar atmak: `floor(random(1, 7))`. Rastgele bir düşman türü seçmek, rastgele bir hücreye ödül koymak hep aynı kalıptır.

Rastgele sayı demişken

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

"Adil bir zar atışıyla seçildi. Rastgele olduğu garanti." · Kaynak: xkcd.com/221, Randall Munroe (CC BY-NC 2.5)

pop() adım adım

Adım	İşlem	liste	Dönen
Başlangıç	—	[10, 20, 42]	—
1	liste.pop()	[10, 20]	42
2	liste.pop()	[10]	20
3	liste.pop()	[]	10
4	liste.pop()	[]	undefined

Boş dizide pop()

Hata vermez ama anlamlı bir değer de vermez. Önce `length > 0` diye bakın.

push da bir değer döndürür

Eklemeden sonraki yeni eleman sayısını. Örneklerimizde kullanmıyoruz; var olduğunu bilmek yeter.

SORU

Bir tabak yığınınından tabak alırken hangisini alırsınız?

En alttakini mi, en üsttekini mi? En üstteki, yığına ne zaman konmuştu?



Yığın (stack): son giren ilk çıkar

Yığın (stack)

`push` ve `pop` birlikte kullanıldığında dizi bir yığın gibi davranır. Kural **LIFO**'dur: **last in, first out**. En son eklenen, ilk çıkarılır.

Ayrı bir `Stack` sınıfı şart değil; sıradan bir dizi ve `push` / `pop` çoğu zaman yeter.

Tanıdık bir yığın

Fonksiyon çağrıları da bir **çağrı yığını** üzerinde tutulur: en son çağrılan fonksiyon önce biter. Derleyiciler ve yorumlayıcılar bu modeli kullanır.

Yığın oyunlarda nerede?

Geri al (Undo)

Son hamleler bir yığında durur. Ctrl+Z bir `pop` 'tur. Super Mario Maker'ın seviye düzenleyicisi, bulmaca oyunları.

Menü yığını

Duraklat → Ayarlar → Kontroller. Kapatınca bir üst menüye dönersiniz: `pop` .

Kart destesi

Solitaire'de desteden kart çekmek üstten `pop` ; kart koymak `push` .

Envanter

"Son alınan eşyayı geri ver" gibi kısa geçmişler.

Sık kullanılan bir kalıp: en büyüğü bulmak

sketch.js

Kopyala

Büyüt

```
1 function enBuyuk(degerler) {
2   if (degerler.length === 0) {
3     return 0;
4   }
5   let maks = degerler[0];
6   for (let i = 1; i < degerler.length; i = i + 1) {
7     if (degerler[i] > maks) {
8       maks = degerler[i];
9     }
10  }
11  return maks;
12 }
```

- İlk elemanı “şimdilik en büyük” kabul et.
- Geri kalanları tek tek gez; daha büyüğünü görürsen onu tut.
- Boş dizi için önceden karar ver: burada 0 dönüyor.
- `enBuyuk(skorlar)` en yüksek skoru verir.

`for` + karşılaştırma kalıbı toplam, ortalama, en küçük gibi birçok özet hesapta tekrar eder.

Mini kule savunma: tıkla, kule kur

sketch.js · satır 26-40

Kopyala

Büyüt

```
26   for (let i = 0; i < kuleX.length; i = i + 1) {  
27     fill(70, 110, 200);  
28     circle(kuleX[i], kuleY[i], 26); // kule govdesi  
29     fill(255);  
30     circle(kuleX[i], kuleY[i], 10); // namlu  
31   }  
32 }  
33  
34 function mousePressed() {  
35   if (mouseX < 72 && mouseY < 36) {  
36     // GERİ AL - Sıra Sizde'de pop eklenecek  
37   }  
38   kuleX.push(mouseX);  
39   kuleY.push(mouseY);  
40 }
```

Çıktı

Çalıştır

Büyüt

GERİ AL

Haritaya tıkla: kule kur
GERİ AL: son kuleyi kaldır

Kule sayısı: 0

Her tıklama, konumu iki diziye birden `push` eder. Birkaç kule kurun, sonra sol üstteki GERİ AL'a basın.

SORU

GERİ AL'a bastınız. Kule kalkmadı, üstelik yeni bir kule eklendi. Neden?

sketch.js · satır 34-40

Kopyala

Büyüt

```
34 function mousePressed() {  
35     if (mouseX < 72 && mouseY < 36) {  
36         // GERİ AL – Sıra Sizde'de pop eklenecek  
37     }  
38     kuleX.push(mouseX);  
39     kuleY.push(mouseY);  
40 }
```

GERİ AL'ı çalıştırın

 20 dk

- Kâğıtta:** `kuleX = [10, 20]` , `kuleY = [30, 40]` . (50, 60)'a bir kule kurun, sonra bir kez geri alın. Diziler nasıl kalır?
- GERİ AL dalında, `length > 0` ise **her iki dizide** `pop` çağırın.
- Birkaç kule kurup geri alın. Ekrandaki kule sayısı `kuleX.length` ile uyumlu mu?

İpucu

Yalnızca `if` 'in içine `pop` yazmak yetmez. GERİ AL'a basınca alttaki `push` satırına ulaşılmaması gerekir.

Yapay zekâyla: dolaşma hatası

 15-20 dk

1. `i <= dizi.length` yazan hatalı bir döngü düşünün. Ne olur? Önce kendiniz yazın.
2. Bu hatayı ve düzeltmesini bir yapay zekâya sorun; gerekçesini sizinkiyle karşılaştırın.
3. Kısa bir diziyle editörde hem hatalı hem doğru sınırı deneyin.

Kod zamanı

 30 dk

1. Beş yıldızın konumunu paralel dizilerde tutup çizin. Tıklanan yere yeni bir yıldız eklensin.
2. `ortalama(degerler)` fonksiyonu yazın; skor listesinin ortalamasını döndürsün.
3. Kule oyununda en fazla 8 kule kurulabilsin; 9. tıklama yok sayılsın ve ekranda “Kule hakkı bitti” yazsın.
4. **Zorlu:** “Biz ve düşmanlar” sahnesini çizin; bize en yakın düşmanı farklı renkte gösterin. Biz fareyi takip edelim.

Sonunda **d** nasıl görünür?

sketch.js

Kopyala

Büyüt

```
1 let d = [3, 6];  
2 d.push(9);  
3 d.pop();  
4 d.pop();  
5 d.push(1);
```

A [3, 6, 1]☒ [3, 1]**C** [1, 3]**D** [3, 6, 9, 1]

[3, 6, 9] → [3, 6] → [3] → [3, 1]. Ekleme de çıkarma da hep sondan.

`liste[liste.length]` neyi verir?

A Son elemanı

B İlk elemanı

☒ `undefined`

D Eleman sayısını

Son eleman `liste[liste.length - 1]`'dir. `length` her zaman son geçerli indeksten bir fazladır.

Ekranda ne olur?

sketch.js

Kopyala

Büyüt

```
1 let xs = [50, 150, 250];
2 let ys = [80, 80];
3
4 for (let i = 0; i < xs.length; i++) {
5   circle(xs[i], ys[i], 40);
6 }
```

A Üç daire çizilir.



İki daire düzgün çizilir; üçüncüsü çizilemez.

C Hiçbir şey çizilmez.

Üçüncü turda `ys[2]` `undefined`. Paralel dizilerin uzunlukları her zaman eşit olmalı.

Merak edenler için kaynaklar

PATT VİRA

Array

Dizileri görsel örneklerle, sakın bir tempoda anlatıyor.

pattvira.com/intro-to-creative-coding/array

PATT VİRA

Changing Size of Array

Diziyi çalışırken büyütme ve küçültme: bugünkü

`push` / `pop`.

pattvira.com/intro-to-creative-coding/changing-size-of-array

THE CODING TRAIN

7.1 Arrays

Daniel Shiffman'ın dizilere girişi.

[thecodingtrain.com > 7-arrays > 1-arrays](https://thecodingtrain.com/7-arrays/1-arrays)

THE CODING TRAIN

7.2 Arrays and Loops

Diziyi `for` ile dolaşmak.

[thecodingtrain.com > 7-arrays > 2-arrays-loops](https://thecodingtrain.com/7-arrays/2-arrays-loops)

Bugün ne öğrendik?

- ✓ Dizi: köşeli parantezle tanımlanan, indeksle erişilen sıralı koleksiyon.
- ✓ İndeksler 0'dan başlar; son geçerli indeks `length - 1`.
- ✓ Dolaşma kalıbı: `for (let i = 0; i < dizi.length; i++)`.
- ✓ Paralel diziler: aynı indeks aynı nesneyi gösterir; uzunluklar eşit kalmalı.
- ✓ `push` sona ekler, `pop` sondan çıkarır; birlikte bir yığın (LIFO) oluştururlar.
- ✓ Boş dizide `pop` ve sınır dışı indeks `undefined` verir; hata sessizdir.

SIRADAKİ DERS

Beş yuvalı bir envanter

Sabit kapasiteli diziler ve paralel dizilerle balon sahnesi.

Sonraki sunum →

← Önceki

