

Değişkenler (variables)

1. **let** yas = 45; (sonradan değiştirilebilir)
2. **const** okul = “İstanbul Teknik Üniversitesi”; (sabit değer)
3. **var** sınıf = “5-A” (sonradan değiştirilebilir)

Primitive(Basit) Veri Tipleri

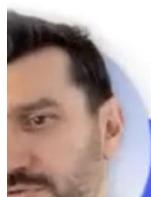
- 1. String (Metin) `const sehir = "İstanbul";`
 - 2. Number (Sayı) `let yas = 46;`
 - 3. Boolean (true / false) `let kapiAcik = false;`
 - 4. Null `let id = null;`
 - 5. Undefined `let alisverisSepeti;`

Koşullu İfadeler

```
if (rastgele > 0.5) {  
    console.log("tura!")  
}
```

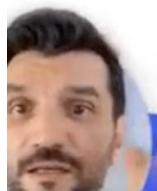
```
if (rastgele > 0.5) {  
    console.log("tura!");  
} else {  
    console.log("yazı!");  
}
```

```
if (yas > 60) {  
    console.log('%25 indirim');  
} else if (yas < 15) {  
    console.log('%50 indirim');  
} else {  
    console.log('İndirim yok :(');  
}
```



Operatörler

1. Aritmetik operatörler (+, -, %, / v.b.)
2. Karşılaştırma operatörleri (==, ===, !=, !==, >=, < v.b.)
3. Mantıksal operatörler (!, &&, || v.b.)



Extra

1. `console.log("Yaşım " + yas)` içindeki bilgiyi console'a yazdırır.
2. `Math.random()` 0 ile 1 arasında bir sayı döner. Rastgele bir sayı üretebiliriz.
3. `const parola = prompt("Parolayı söyle!")` ile kullanıcıdan bilgi alabiliriz.
4. Truthy/Falsy ifadeler: Javascript, boolean değer görmeyi umduğu yerlerde başka tipte veriler görürse, bu değerleri boolean gibi yorumlar.

```
let altınSayisi = 100;

if (altınSayisi) {
  console.log("Mağara kapanmalı");
}
```



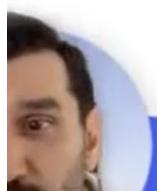
Fonksiyonlar

```
function yaziTura() {  
    let rastgele = Math.random();  
    if (rastgele > 0.5) {  
        console.log("tura!");  
    } else {  
        console.log("yazı!");  
    }  
}
```



Fonksiyon sonucunu dışa aktarma: **return**

```
function yaziTura() {  
    let rastgele = Math.random();  
    if (rastgele > 0.5) {  
        return "tura!";  
    } else {  
        return "yazı!";  
    }  
}  
  
const sonuc = yaziTura();  
  
console.log('Oyun sonucu ' + sonuc);
```



Fonksiyon sonucunu dışa aktarma: **return**

```
function yaziTura() {  
    let rastgele = Math.random();  
    if (rastgele > 0.5) {  
        return "tura!";  
    } else {  
        return "yazı!";  
    }  
}  
  
const sonuc = yaziTura();  
  
console.log('Oyun sonucu ' + sonuc);
```



Hata mesajları: Google'layarak çözebiliriz.

```
function selamVer(isim) {  
    cons selam = "Merhaba" + isim;  
    console.log(selam);
```

```
> function selamVer(isim) {  
    cons selam = "Merhaba" + isim;  
    console.log(selam);  
}
```

✖ Uncaught SyntaxError: Unexpected identifier 'selam'

A screenshot of a Google search results page. The search query "Uncaught SyntaxError: Unexpected identifier" is entered in the search bar. The results show approximately 92,900 results. The top result is a snippet from MDN Web Docs explaining the error, stating it's typically thrown due to misspellings or missing punctuation in JavaScript keywords like let, class, function, const, etc. The snippet is dated Nov 15, 2022.

Google

Uncaught SyntaxError: Unexpected identifier

All Videos Images News Books More Tools

About 92,900 results (0.52 seconds)

The "Uncaught SyntaxError: Unexpected identifier" error is typically thrown as a result of a misspelt keyword, missing operator or missing comma. Keywords in JavaScript include words such as `let` , `class` , `function` , `const` . These words have a special meaning in the language and are part of the syntax.

Nov 15, 2022



Fonksiyonlarda parametreler ve argümanlar

```
function uyarıGoster(mesaj, isim) {  
    console.log(mesaj + " " + isim);  
}
```

```
uyarıGoster("Yine bekleriz", "Ahmet")
```

Dikkat: Parametre sıralaması önemli



Scope

```
let yas = 35;

function yasimiYazdir() {
    const isim = "ali";
    console.log("Benim adım " + isim);
    console.log(yas + " yaşındayım.");

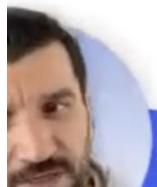
    if (yas > 18) {
        let ehliyetAlabilirMi = true;
        console.log("Ehliyet alabilecek yaştayım.")
    }
}
```

- Her scope bir üst scope'a erişebilir.
- En üst scope'a global scope diyoruz.



Extra

1. `Math.random()` 0 ile 1 arasında bir sayı döner. Rastgele bir sayı üretebiliriz.
2. `Math.random() * n` 0 ile n arasında bir sayı üretir.
3. `Math.floor(5.999)` ile sayılar aşağı yuvarlanır. (sonuç: 5)



Metod Kavramı ve Basit String Metodları

1. Math.random()
2. console.log()

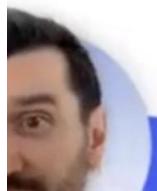
String length	String trim()
String slice()	String trimStart()
String substring()	String trimEnd()
String substr()	String padStart()
String replace()	String padEnd()
String replaceAll()	String charAt()
String toUpperCase()	String charCodeAt()
String toLowerCase()	String split()
String concat()	



Array'ler

```
const katilimci1 = "ömer";
const katilimci2 = "melisa";
const katilimci3 = "aslı";
const katilimci4 = "tuğrul";
const katilimci5 = "gökhan";
```

```
const katilimcilar = ["ömer", "melisa", "aslı", "tuğrul", "gökhan"]
```



Array'lerde Elemanlara Erişim

```
const katilimcilar = ["ömer", "serhan", "emre", "tuğrul", "gökhan"]
```

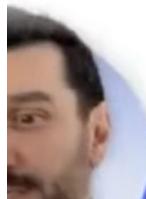


1. index'teki elemana erişim

```
katilimcilar[1]. // "serhan"
```

1. index'teki elemanın değerini değiştirme

```
katilimcilar[1] = "melek";
```



Temel Array Metodları

Array length	Array join()
Array toString()	Array delete()
Array pop()	Array concat()
Array push()	Array flat()
Array shift()	Array splice()
Array unshift()	Array slice()

```
const takipcilerim = ["ilkokul öğretmenim", "teyzem", "Şeyma SubAŞı"]
takipcilerim.length
// 3
```

```
const sinifListesi = ["ahmet", "ayşe", "furkan", "gözde"];
sinifListesi.push("levent");
// ["ahmet", "ayşe", "furkan", "gözde", "levent"]
```





Döngüler (loops)

```
for (let i = 0; i < 3; i++) {  
    console.log(i);  
}  
  
//0  
//1  
//2
```

```
const hedefSayi = Math.ceil(Math.random() * 25);  
let tahminSayisi = 0;  
  
while (tahminSayisi < 6) {  
    const tahmin = prompt("Sence sayı kaç?");  
    tahminSayisi = tahminSayisi + 1;  
}
```



Ufak bir ek bilgi:

`break` komutu for döngülerinde de döngüden çıkmak için kullanılabilir.



Array'lerin Döngüler ile Kullanımı

```
const katilimcilar = ["ömer", "melisa", "aslı", "tuğrul", "gökhan"]

console.log("Tebrikler! " + katilimcilar[0]);
console.log("Tebrikler! " + katilimcilar[1]);
console.log("Tebrikler! " + katilimcilar[2]);
console.log("Tebrikler! " + katilimcilar[3]);
console.log("Tebrikler! " + katilimcilar[4]);

for(let i = 0; i < 5; i++) {
    console.log("Tebrikler!" + katilimcilar[i])
}
```

```
for(let i = 0; i < katilimciler.length; i++) {
    console.log("Tebrikler!" + katilimcilar[i])
}
```



Kompleks Veri Türlerinin Farklılıkları

```
> const name = "Emre";
<- undefined
> name = "Hakan";
! ▶ TypeError: Attempted to assign to readonly property.
```

```
> const ulkeler = ["fas", "japonya"];
<- undefined
> ulkeler[1] = "Almanya"
<- "Almanya" = $1
> console.log(ulkeler)
E ["fas", "Almanya"] (2)
```

```
> const sehirler = ["İstanbul", "Ankara"];
<- undefined
> const sehirler = ["İzmir", "Adana"];
! ▶ SyntaxError: Can't create duplicate variable: 'sehirler'
```



Basit veri tipleri

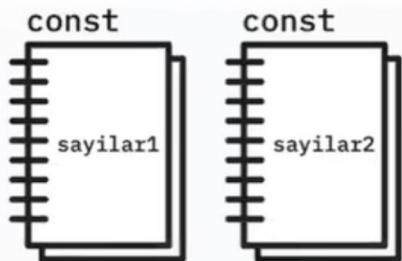


Kompleks veri tipleri



Kompleks Veri Türlerinde Eşitlik Kontrolü

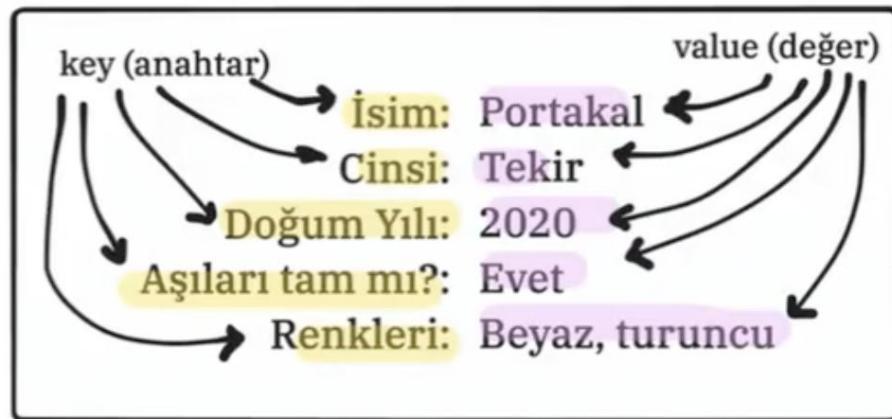
```
> const sayilar1 = [1, 2, 3];
  const sayilar2 = [1, 2, 3];
  sayilar1 === sayilar2
< false
```



```
const ulkeler2 = ulkeler;
```



Objeler



```
const kedim = {  
  isim: "Portakal",  
  cinsi: "Tekir",  
  "Doğum Yılı": 2020,  
  "Aşları tam mı?": true,  
  renkleri: ["Beyaz", "Turuncu"]  
}
```

Objelerin özelliklerine erişmek

kedim.isim

kedim["Doğum Yılı"]

Obje değerlerini eklemek, değiştirmek ve silmek

```
const kedim = {  
    isim: "Portakal",  
    cinsi: "Tekir",  
    "Doğum Yılı": 2020,  
    "Aşıları tam mı?": true,  
    renkleri: ["Beyaz", "Turuncu"]  
}
```

Öncesi

```
kedim["Göz Rengi"] = "Yeşil"  
  
kedim["Doğum Yılı"] = 2019  
  
delete kedim["cinsi"]
```

```
const kedim = {  
    isim: "Portakal",  
    "Doğum Yılı": 2019,  
    "Aşıları tam mı?": true,  
    renkleri: ["Beyaz", "Turuncu"],  
    "Göz Rengi": "Yeşil"  
}
```

Sonrası



Ecran Resmi

Objelere metod eklemek ve **this** ile obje değerlerine erişmek

```
const farem = {  
  ...  
  hizDegistir: function(degisimYonu) {  
    }  
}
```

```
const fare = {  
  renk: "siyah",  
  hiz: 5,  
  solTik: function() {  
    return "sol tuşa tıklandı"  
  },  
  hizRaporu: function() {  
    return "fare hızı: " + this.hiz  
  },  
  hizlan: function() {  
    if(this.hiz < 10) {  
      this.hiz = this.hiz + 1  
    }  
    return "yeni hız: " + this.hiz  
  }  
}
```

Spread Operatörü ile Array ve Obje Kopyalama

- Array kopyalama
- Birden fazla array'i kopyalama
- Obje kopyalama
- Obje kopyalarken yeni bir özellik ekleme veya var olanı değiştirme
- Birden fazla objeyi tek objede kopyalama

```
const tumSehirler = [...yurtici]
```

```
const tumSehirler = [...yurtici, ...yurtdisi]
```

```
const telefon2 = {...telefon}
```

```
const telefon2 = {...telefon, nfc: true}
```

```
const sinemYeniKimlik = {...sinemKimlik, ...sinemEhliyet};
```

Yeni Javascript (ES6) Özellikleri

1. Ternary If: Üçleme, Koşul operatörü

```
onay ? console.log("Silindi") : console.log("İptal edildi")
```

2. Backtick ile stringlerin içine javascript yazma

```
alert(`Hoşgeldin ${kullaniciAdi}! ${sepetMesaji}`)
```

3. Fonksiyon parametrelerinde default değer belirleme

```
function indirimYap(urun, fiyat, yuzde = 20) {  
    const indirim = (fiyat * yuzde)/100;  
    const indirimliFiyat = fiyat - indirim;  
    return "İndirimli " + urun + " fiyatı " + indirimliFiyat;  
}
```

Fonksiyonların Farklı Yazım Türleri

1. Klasik yöntem: Function Declaration

```
function karesiniAl(sayi) {  
    return sayi * sayi;  
}
```

2. Anonim fonksiyonlar: Function Expression

```
const karesiniAl2 = function(sayi) {  
    return sayi * sayi;  
}
```

3. Arrow fonksiyonlar: Arrow Functions

```
const karesiniAl3 = (sayi) => {  
    return sayi * sayi;  
}
```

```
const karesiniAl4 = (sayi) => sayi * sayi;
```

Fonksiyonların Farklı Yazım Türleri

1. Klasik yöntem: Function Declaration

```
function karesiniAl(sayi) {  
    return sayi * sayi;  
}
```

2. Anonim fonksiyonlar: Function Expression

```
const karesiniAl2 = function(sayi) {  
    return sayi * sayi;  
}
```

3. Arrow fonksiyonlar: Arrow Functions

```
const karesiniAl3 = (sayi) => {  
    return sayi * sayi;  
}
```

```
const karesiniAl4 = (sayi) => sayi * sayi;
```

Callback Fonksiyonlar

```
function mailGonder(baslik) {  
    //  
    // burada mail gönderme işlemlerini yapan kodlar var  
    //  
    console.log(baslik + " başlıklı email gönderildi");  
}  
  
function satinAl(urun, callback) {  
    //  
    // burada ödeme işlemlerini yapan kodlar var  
    // ödeme başarılıysa, odemeAlindi true, değilse false  
    //  
    if (odemeAlindi) {  
        console.log("ödeme yapıldı");  
        callback(urun + " siparişiniz onaylandı");  
    } else {  
        console.log("ödeme başarısız");  
    }  
}
```

```
satinAl('mavi gömlek', mailGonder);
```



Async Fonksiyonlar: Örn: setTimeout()

Normalde kodlar satır satır, sırayla çalışır. Ama, bazı durumlarda -özellikle zaman alan durumlarda (örn: diske veri yazma), kodların paralel olarak çalışmasını isteyebiliriz. Bunun için async fonksiyonlar kullanırız.

```
function bilgiVer() {  
    console.log("Giriş başarılı, profiline yönlendiriliyorsun");  
}  
setTimeout(bilgiVer, 3000)
```

* *setTimeout* *async* bir fonksiyondur. Yukarıdaki örnekte *console*'a log 3000ms=3sn sonra basılır.



Higher Order Fonksiyonlar

Higher-order fonksiyonlar, başka fonksiyonları parametre olarak alan veya yeni bir fonksiyon return eden fonksiyonlardır.

Callback metodları kullanılan fonksiyonlar bu gruba girer.

İleri Array Metodları

- 1. .map(fonksiyon)**
 - a. yeni bir array döner
 - b. array'in her elemanı için bir fonksiyon çalıştırır
 - c. orjinal array'i değiştirmez
- 2. .filter(fonksiyon)**
 - a. yeni bir array döner
 - b. verilen koşula uyan her elemanı geri döner
 - c. orjinal array'i değiştirmez
- 3. .reduce(fonksiyon, başlangıç değeri)**
 - a. başlangıç değerini ilk elemana iletir
 - b. array'in her elemanına bir önceki elemanda hesaplanmış değeri iletir
 - c. çalıştırılan fonksiyonun en son sonucunu döner
 - d. orjinal array'i değiştirmez

İleri Array Metodları

4. .sort(fonksiyon)
 - a. orjinal array'i değiştirir
 - b. array'in elemanlarını verilen fonksiyona göre sıralar
 - c. fonksiyon verilmez ise string olarak sıralar



Hatırlayalım: Arrow Function

Klasik gösterim:

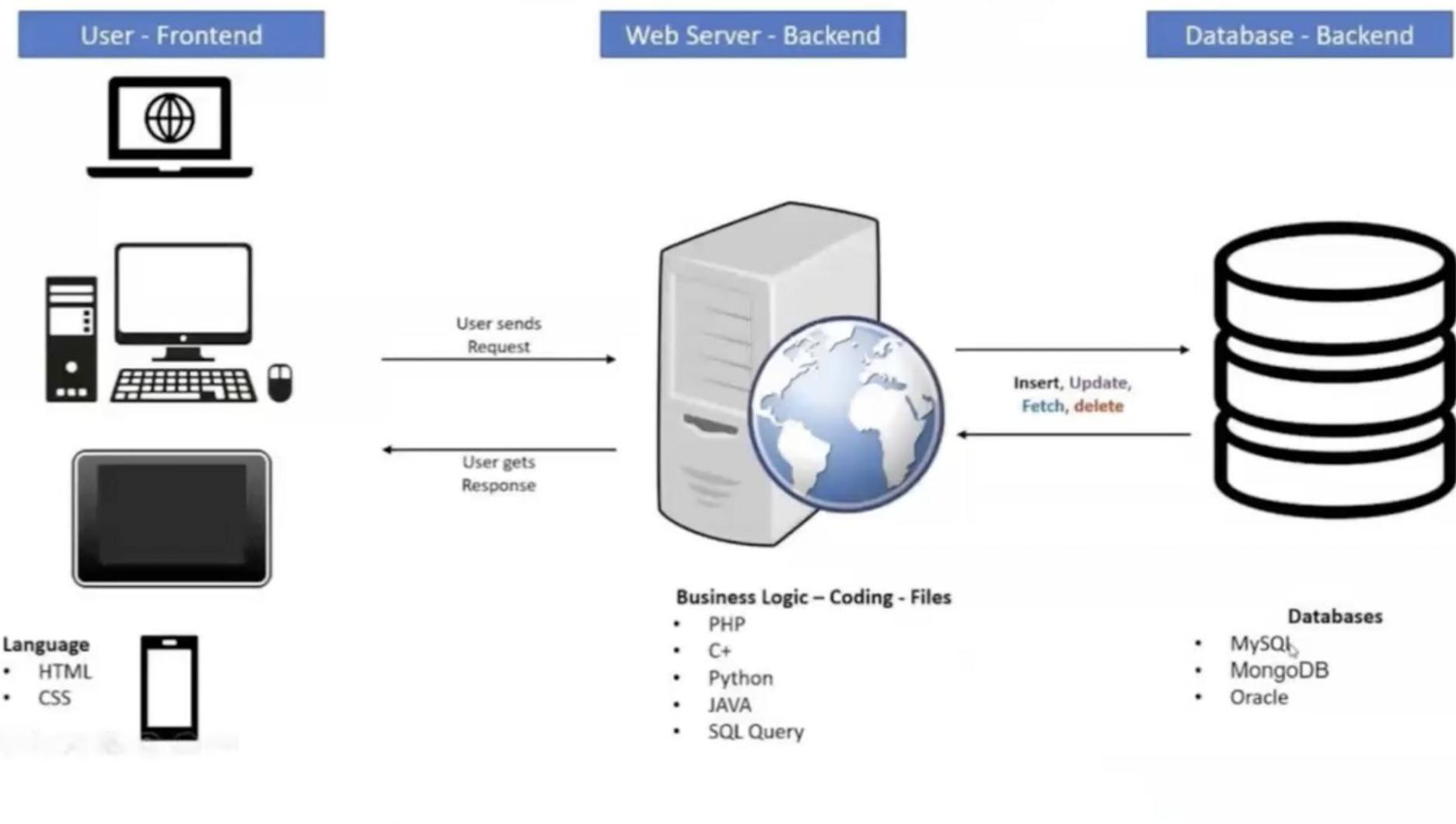
```
const karesiniAl3 = (sayi) => {  
    return sayi * sayi;  
}
```

Tek satır gösterim:

```
const karesiniAl4 = (sayi) => sayi * sayi;
```

İnternet Nasıl Çalışır

Data Flow in Software



JAVASCRIPT

* Değişkenler

- * tanımlama: let, const, var
- * değer atama
- * değerine erişmek
- * değerini değiştirmek
- * camelCase yazılır, türkçe karakter kullanılmaz
- * türleri: basit ve complex (array, object)

* if, else if, else

* Operatör: karşılaştırma ve mantıksal

* Döngüler: while, for, for of, for in

* fonksiyonlar

- * arrow, anonim, klasik

```
const topla = (num1, num2) => {
```

```
    //kodlar buraya
```

```
}
```

```
const topla = function (num1, num2) {
```

```
}
```

```
function topla (num1, num2) {
```

```
}
```

* callback function

* higher order function

* parametre tanımla

* parametrelere başlangıç değeri atama

* Async, sync, await

* timeout

* Array

* index'ler 0'dan başlar

* elemene erişme, değer atama: arr[index]

* tanımlama arr = []

* temel metodlar: .push, .shift, .unshift, .pop, .concat, .length, .split, .slice, .join, .splice, .indexOf, .includes

* ileri array metodları: .reduce, .forEach, .map, .filter, .sort, .find, .some, .every

* Ternary if: koşul ? doğru ise : yanlış ise

* Object

* obje metodları tanımlama :anonim fonksiyon

* this kavramı

* property

* key&value

* Object.keys(obj), Object.values(obj)

* dot notation, bracket notation

```
const user = {
```

```
    name: "Emre",
```

```
    "last name": "Şahiner",
```

```
}
```

```
const key = "name"
```

```
user[key] //name değerini vermez
```

```
user["name"] -> "Emre"
```

* Backtick: `\${word}` kelimesi \${word.length} karakter uzunluğundadır.

```
const arr = []
```

```
({[]}{()}{[]}{[]}{[]}{()}{}})
```



- 1- her elemanı tek tek al. (for)
- 2- if (açılış parantezi) ise array'e push'la
else
 if(arr'in son elemanın kapanışı mı?)
 true => array'den pop'la
 false => return false
- 3- if arr.length > 0 ise return false
değil ise return true