


```
const longitude = response.body.features[0].center[0]; // boylam bilgisi
features bir array döndüren 5 bölgeden ilkinin aldık
const latitude = response.body.features[0].center[1]; // enlem bilgisi
center da bir array enlem ve boylam bilgisini alırız
console.log("Enlem : " + latitude + "Boylam : " + longitude);
});
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Enlem : -6.224825Boylam : 106.934716
PS C:\Users\user\Desktop\WeatherApp>
```

Tahmin ettiğimiz gibi gerçekleşmedi 134 konum bilgisini için Endonezya da bir lokasyon buldu.

```
api.mapbox.com/geocoding/v5/mapbox.places/134.json?access_token=pk.eyJ1ljoib3psZW1hcnNsYW5ubilislmEiOiJbHUzZWYyM...
```

```
{
  "type": "FeatureCollection",
  "query": [
    "134"
  ],
  "features": [
    {
      "id": "postcode.1371751",
      "type": "Feature",
      "place_type": [
        "postcode"
      ],
      "relevance": 1,
      "properties": {
        "mapbox_id": "dX3uOm1eH8sYzpGtZVu"
      },
      "text": "13460",
      "place_name": "13460, Jakarta, Indonesia",
      "bbox": [
        106.92419,
        -6.237353,
        106.960829,
        -6.21599
      ],
      "center": [
        106.934716,
        -6.224825
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          106.934716,
          -6.224825
        ]
      }
    }
  ]
}
```

Biraz daha saçma bir konum bilgisi girelim. Direkt olarak hata alırız.

```
api.mapbox.com/geocoding/v5/mapbox.places/1fefw4.json?access_token=pk.eyJ1ljoib3psZW1hcnNsYW5ubilislmEiOiJbHUzZW...
```

```
{
  "type": "FeatureCollection",
  "query": [
    "1fefw4"
  ],
  "features": [],
  "attribution": "NOTICE: © 2024 Mapbox and its suppliers. All rights reserved. Use of this data is subject to the Mapbox Term information it contains may not be retained. POI(s) provided by Foursquare."
}
```

Geocoding servisine şehir bilgisi değil de saçma bir değer girdiğimizde features arrayin boş olarak geri döndürüldüğünü görmüş olduk. Bu kısımda features arrayi boş mu değil mi diye kontrol yaparak düzenleme yapalım. Hata var mı yok mu kontrolü yapalım.

Çıktı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Enlem : 34.053691 Boylam : -118.242766
```

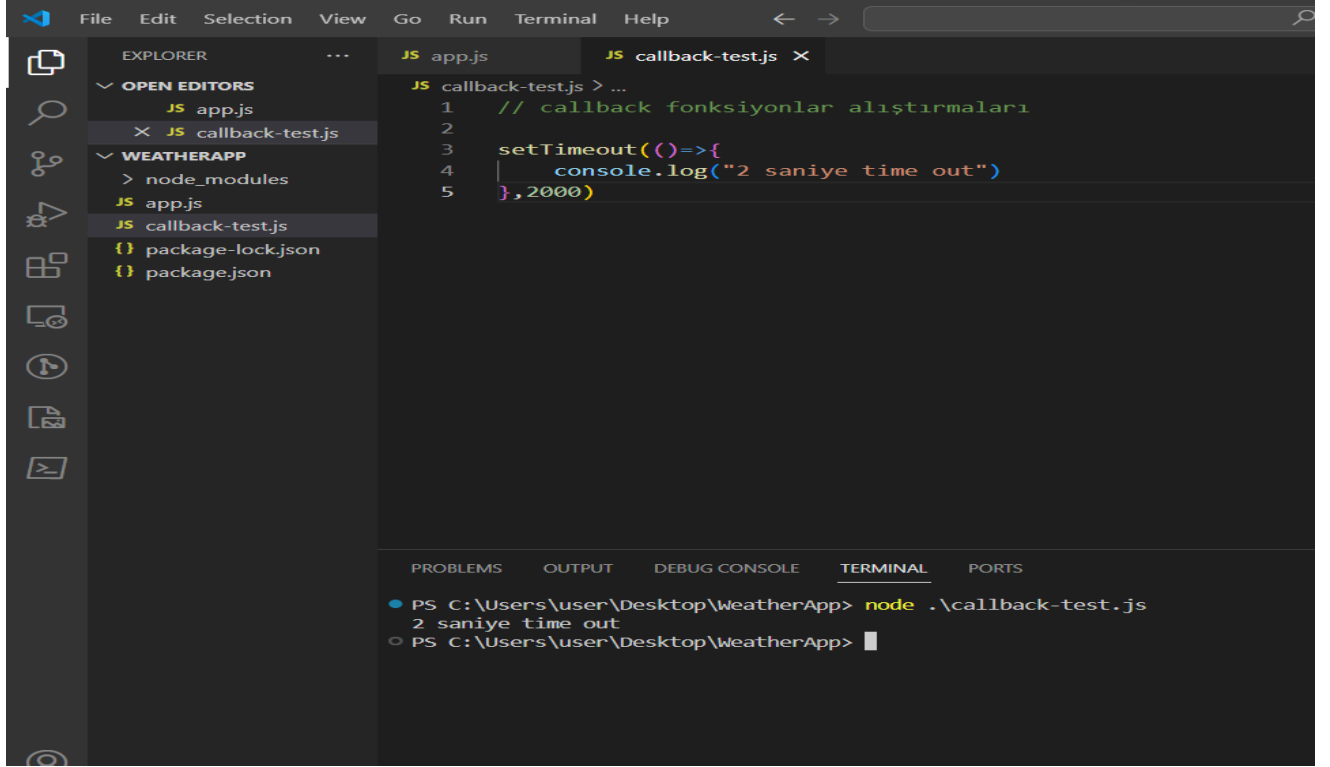
```
const geocodeUrl =
  "https://api.mapbox.com/geocoding/v5/mapbox.places/f5rgfer.json?access_
  token=pk.eyJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJJbHUzZWZyMDAwZDN2MmtwZGlvMz
  QzeG8wIn0.SB2-Lj5qXuozz89bPnWUU0";
```

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Belirttiğiniz konum bilgisi bulunamadı
```

Eğer ağ adaptörünü kapatırsak da if koşulu çalışacaktır.

Callback-test.js isminde deneme dosyası oluşturalım ve callback fonksiyonları ile alıştırma yapalım.

setTimeout() fonksiyonu tanımlayalım bu fonksiyon içinde çalışmasını beklediğimiz fonksiyonu kendimiz tanımlarız. Callback fonksiyonların tamamı asenkron değildir senkron olarak da tanımlanabilir. Bu fonksiyon asenkronudur.



```
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
  JS app.js
  X JS callback-test.js
WEATHERAPP
  > node_modules
  JS app.js
  JS callback-test.js
  {} package-lock.json
  {} package.json

JS callback-test.js > ...
1 // callback fonksiyonlar alıştırması
2
3 setTimeout(()=>{
4   console.log("2 saniye time out")
5 },2000)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\Desktop\WeatherApp> node .\callback-test.js
2 saniye time out
PS C:\Users\user\Desktop\WeatherApp> 
```

Başka bir örneğe bakalım. Bu kodu çalıştırmamıza gerek yok sadece örnek olması için yazıldı.

```
7 const names= ["ali", "berk", "canan"]
8
9 // karakter uzunluğu 4 e eşit veya küçük olanları filtrelesin
10 // filter fonksiyonu bizim bir fonksiyon tanımlamamızı bekler biz de senkron bir callback fonksiyon tanımlayalım.
11
12 const shortNames= names.filter((name)=>{
13   return name.length <=4
14 })
```

Örnek yapmaya devam edelim yazdığımız tüm kodları yorum satırına alalım ve bu kodları ekleyelim çalıştıralım.

```
const geocode=(address, callback)=> {
  // senkron bir şekilde başlayalım uzaktaki bir web servisine
  bağlanmayacak veriyi biz verelim.
  const data={
    latitude:0,
    longitude:0
```

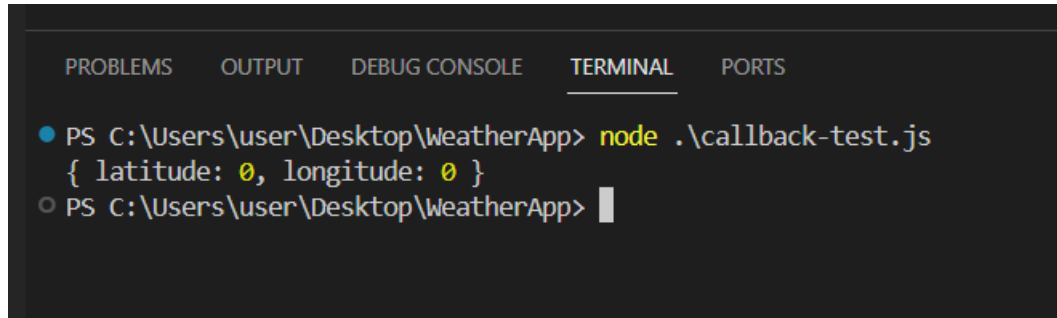
```

    } // enlem ve boylam bilgisini geriye döndürelim.

    return data // geriye verileri return ile döndürürüz
}
const data = geocode("bursa")
console.log(data)

```

Çıktısı:



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\Users\user\Desktop\WeatherApp> node .\callback-test.js
  { latitude: 0, longitude: 0 }
○ PS C:\Users\user\Desktop\WeatherApp>

```

Enlem ve boylam bilgisini getirdiğini gözlemledik çünkü bu kod şu an senkron olarak çalışıyor peki bunu asenkron hale getirelim.

Callback fonksiyonu içine setTimeout fonksiyonunu eklersem asenkron bir hale gelebilir çünkü setTimeout bir asenkron fonksiyondur.

```

const geocode=(address, callback)=> {

    setTimeout(()=>{    // asenkron fonk olan setTimeoutu ekleyelim
        const data={
            latitude:0,
            longitude:0
        } // enlem ve boylam bilgisini geriye döndürelim.
        return data // geriye verileri return ile döndürürüz

    },2000)
}
const data = geocode("bursa")
console.log(data)

```

Çıktısı:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\Users\user\Desktop\WeatherApp> node .\callback-test.js
undefined
● PS C:\Users\user\Desktop\WeatherApp> █
```

Buradaki problem fonksiyon çalışırken return data dedik ve elindeki datayı döndürmeye çalışıyor ama buradaki fonksiyon 2 saniye gecikmeli çalışacaktır bu yüzden çıktı undefined olarak verilir.

Çalıştırabilmek için:

```
const geocode=(address, callback)=> {

    setTimeout(()=>{    // asenkron fonk olan setTimeoutu ekleyelim
        const data={
            latitude:0,
            longitude:0
        } // enlem ve boylam bilgisini geriye döndürelim.
        callback(data) //asenkon olduğu için callback ile veriler geri
        gönderilir.

    },2000)
}
// bu kısımda da data ifadesini göndermemiz lazım
const data = geocode("bursa",(data)=>{
    console.log(data)
})
```

Çıktısı:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\user\Desktop\WeatherApp> node .\callback-test.js
{ latitude: 0, longitude: 0 }
PS C:\Users\user\Desktop\WeatherApp> █
```

Özetlemek gerekirse eğer fonksiyonumuz senkron çalışıyorsa return yapabiliriz ama asenkron ise Return kullanamayız callback olarak değiştirmemiz lazım.

Bir örnek daha yapalım. Toplama fonksiyonu tanımlayalım 3 parametre alıyor ilk iki parametre toplamak istediğimiz sayılar ve üçüncü parametresi ise bir callback fonksiyonudur.

```
const add =(a, b, callback)=>{
    // asenkron hale getirmek için setTimeout fonksiyonu kullanıldı.
    // not: request fonksiyonunda asenkron bir fonksiyondur.
    setTimeout(() => {
        callback(a+b)    // asenkron olduğu için return kullanamayız
        // callback çağrılır.
    }, 2000);
}
// iki saniye sonra sayıları toplar ve ekrana basar.
add(1,4,(sum)=>{
    console.log(sum)
})
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node .\callback-test.js
5
PS C:\Users\user\Desktop\WeatherApp>
```

Örneklerimiz bu kadar yeterlidir tekrardan app.js dosyasına gidelim burada request ile sadece bir lokasyon için sorgu yapabiliyoruz ama farklı lokasyonlar için aynı sorguyu çalıştırmak istediğimizde kodumuzu kopyalayıp yapıştırmamız gerekecektir bu işlemde projemiz de kod tekrarına neden olacaktır. Bu sorgulama işlemini bir fonksiyon haline getirmemiz lazım. Uygulamamamızın bulunduğu klasör içinde utils isminde bir klasör açalım. Utils klasörü içinde de Geocode.js isimli yeni bir dosya açalım ve burada fonksiyon tanımlama işlemlerini gerçekleştirelim import ve export işlemlerini gerçekleştirip app.js içinde bu fonksiyonu çağıralım.

Geocode.js içinde bulunan kodlar:

```
// fonksiyona geocode ismi verelim.

const request=require("postman-request") // request işlemi
gerçekleştireceğimiz için bunu kullanmak zorundayız
const geocode =(adress,callback)=>{

    // adres bilgisini url içine eklememiz lazım ve ek olarak url de boşluk
    bulunmamalı
    // adres bilginizde boşluk veya özel karakter bulunuyorsa bunu
    encodeURIComponent(address) ile almanız lazım
    const url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" +
    encodeURIComponent(address) +
    ".json?access_token=pk.eyJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJJbHUzZWYyMDAw
eDN2MmtwZG1vMzQzeG8wIn0.SB2-Lj5qXuozz89bPnWUUQ";
```

```

// fonksiyonun çağrıldığı yerde ne yapılacağına karar vermek daha doğru
olur bu yüzden console.log yapılmamalı callback yapılmalıdır.
// callback kullanırken geriye ya error döndürürüz ya da göndermek
istediğimiz mesajdır.
request({ url: url, json: true }, (error, response) => { // hata ve
sonuc gelebilir bu yüzden error ve response kullandık
// hata mekanizması oluşturalım
if(error){
    //console.log("Geocoding servisine bağlanamadı")
    // callback error ve response bekler ama bu kısımda response
bulunmadığı için undefined yazalım
    callback("Geocoding servisine bağlanamadı",undefined)
}
else if(response.body.features.length===0){ // servise bağlanmıştır
ama konumda hata vardır
    //console.log("Belirttiğiniz konum bilgisi bulunamadı")
    callback("Belirttiğiniz konum bilgisi bulunamadı",undefined)
}
else{
    const longitude = response.body.features[0].center[0]; // boylam
bilgisi features bir array döndüren 5 bölgeden ilkinin aldık
    const latitude = response.body.features[0].center[1]; // enlem
bilgisi center da bir array enlem ve boylam bilgisini alırız
    //console.log("Enlem : " + latitude + "Boylam : " + longitude)
    callback(undefined,{ // responseyi obje haline getirelim
        longitude: response.body.features[0].center[0],
        latitude: response.body.features[0].center[1],
        location:response.body.features[0].place_name})
    }
});
}
// bu fonksiyonu app.js de çağırabilmem için export etmemiz lazım.
module.exports=geocode

```

app.js içinden bu geocode fonksiyonunu çağıralım ama çağırma işlemine geçmeden önce app.js içine geocode.js dosyasını dahil edebilmek için require anahtar kelimesini kullanmamız lazım.

App.js içinde yazan kodlar:

```

// önce import işlemini kullanalım.

const request = require('postman-request');

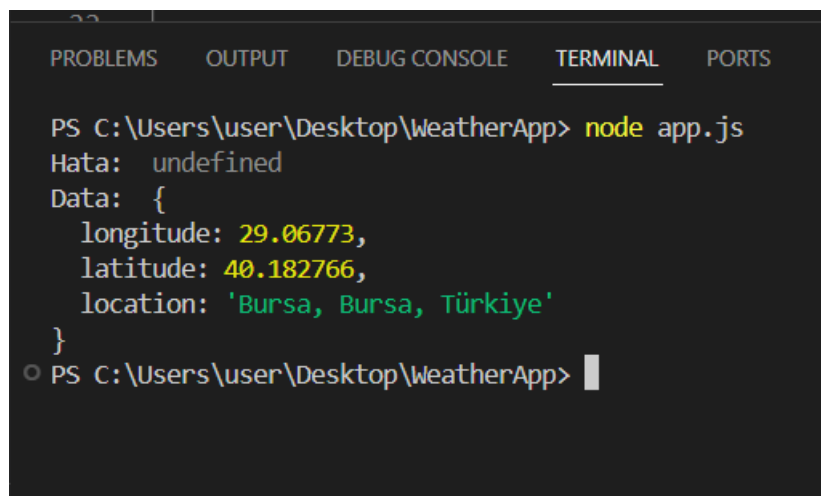
const geocode= require('./utils/geocode')

```



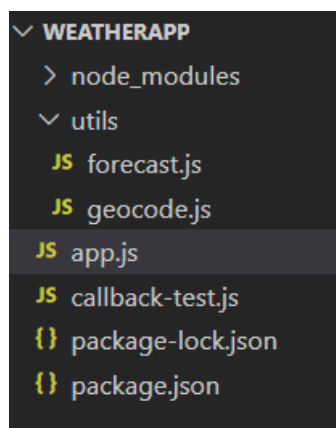
```
// callback fonksiyonunu çağıralım ilk olarak adres parametresi verildi  
daha sonra callback fonksiyonu verildi.  
geocode("bursa",(error,data)=>{  
    console.log("Hata: ",error)  
    console.log("Data: ",data )  
}))
```

Çıktısı:



```
PS C:\Users\user\Desktop\WeatherApp> node app.js  
Hata: undefined  
Data: {  
  longitude: 29.06773,  
  latitude: 40.182766,  
  location: 'Bursa, Bursa, Türkiye'  
}  
PS C:\Users\user\Desktop\WeatherApp>
```

Diğer web servisimiz için bu işlemleri yapabilmek adına util klasörünün içinde forecast.js dosyası açalım.



App.js içinde yorum satırlarında bulunan weatherstack ile ilgili kodları kopyalayın ve forecast.js dosyasına yapıştırın.

Forecast.js içinde yazan kodlar:

```

const request= require("postman-request")

const forecast=(longitude, latitude, callback)=>{
  // url içinde enlem ve boylam bilgisini parametre alacak şekilde
  değiştirdik
  const url
  ="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
  1ac35fa762&query=" + latitude+ longitude + "&units=s"
  //Requesti göndermek için kullanacağımız fonksiyon error ile
  belirlenecek ve sorgudan gelen cevap response ile tutulacak
  request({ url: url, json:true }, (error, response) => {
    // console.log(
    //   "Hava sıcaklığı:" +
    //   response.body.current.temperature +
    //   " Hissedilen:" +
    //   response.body.current.feelslike
    //   );
    if (error) { // direkt bir hata olduğu durumda mesela ethernet
    bağlantısı kesilirse
      //console.log("Hava durumu servisine bağlantı kurulamadı.");
      callback kullanıcaz console.log kullanamayız
      callback("Hava durumu servisine bağlantı kurulamadı.", undefined)
    }
    else if(response.body.error){ // sorgu başarılı ama gelen cevapta
    bir hata varsa mesela enlem ve boylam bilgisi sorguda yoksa
      // url yanlış girildiyse
      // console.log("Girilen konum bilgisi bulunamadı.")
      callback("Girilen konum bilgisi bulunamadı.",undefined)
    }
    else { //hiçbir hata durumu oluşmamışsa
      callback(undefined,"Hava sıcaklığı"+
      response.body.current.temperature+"Hissedilen: "+
      response.body.current.feelslike)
      console.log(
        "Hava sıcaklığı:" +
        response.body.current.temperature +
        "Hissedilen:" +
        response.body.current.feelslike
      );
    }
  })
}
module.exports=forecast

```

App.js içinde yazan kodlar:

```
// önce import işlemini kullanalım.

const request = require('postman-request');

const geocode= require('./utils/geocode')

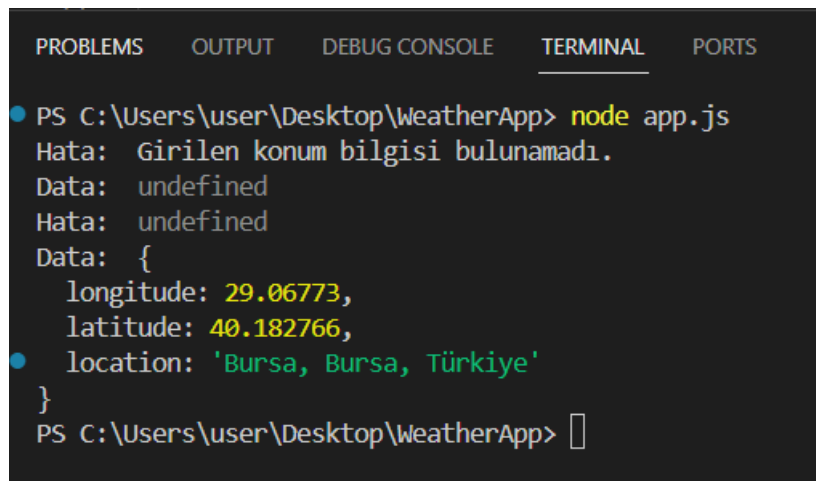
const forecast=require("./utils/forecast")

// callback fonksiyonunu çağıralım ilk olarak adres parameteri verildi
daha sonra callback fonksiyonu verildi.
geocode("bursa",(error,data)=>{
    console.log("Hata: ",error)
    console.log("Data: ",data )
})

// callback şeklinde tanımlanan forecast fonksiyonunu çağıralım

forecast(-75.7088, 44.1545, (error,data)=>{
    console.log("Hata: ",error)
    console.log("Data: ",data)
})
```

Çıktısı:



```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hata:  Girilen konum bilgisi bulunamadı.
Data:  undefined
Hata:  undefined
Data:  {
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
PS C:\Users\user\Desktop\WeatherApp> 
```

Bu iki servis arasında bir sıralama var enlem ve boylam bilgisini geocode ile alıp daha sonra hava durumu sorgusu yapabiliyorum. Önce geocode servisi çalışmalı elde edilen bilgi hava durumu servisine verilmeli. Sıralama yapabilmek için callback chaining işlemi yapılır önce geocode çağırılmalı. App.js içinde düzenlemeler yapalım.

```
// önce import işlemini kullanalım.
```

```

const request = require('postman-request');

const geocode= require('./utils/geocode')

const forecast=require("./utils/forecast")

// callback fonksiyonunu çağıralım ilk olarak adres parameteri verildi
daha sonra callback fonksiyonu verildi.
geocode("bursa", (error,data)=>{
    if(error){
        return console.log("Hata: ", error) // hata varsa forecaste
        gitmeden uygulamdan çıkabilmesi için return attık
    }
    forecast(data.longituude, data.latitude,
(error,forecastData)=>{
        if(error){
            console.log("Hata: ",error)
        }
        console.log("Data: ",forecastData)
    })
})
})

```

Çıktısı:

```

PS C:\Users\user\Desktop\WeatherApp> node app.js
Hata: Girilen konum bilgisi bulunamadı.
Data: undefined
PS C:\Users\user\Desktop\WeatherApp> 

```

App.js içinde konum bilgisini alalım.

```

// önce import işlemini kullanalım.

const request = require('postman-request');

const geocode= require('./utils/geocode')

const forecast=require("./utils/forecast")

// callback fonksiyonunu çağıralım ilk olarak adres parameteri verildi
daha sonra callback fonksiyonu verildi.

```

```

geocode("bursa", (error,data)=>{
    if(error){
        return console.log("Hata: ", error) // hata varsa forecaste
        gitmeden uygulamdan çıkabilmesi için return attık
    }
    forecast(data.longituude, data.latitude,
(error,forecastData)=>{
        if(error){
            console.log("Hata: ",error)
        }
        console.log(data.location)
        console.log("Data: ",forecastData)

    })
})
})

```

Çıktısı:

```

PS C:\Users\user\Desktop\WeatherApp> node app.js
Hata: Girilen konum bilgisi bulunamadı.
Bursa, Bursa, Türkiye
Data: undefined
PS C:\Users\user\Desktop\WeatherApp> 

```

Adres bilgisini parametre olarak vermek için şu işlemler yapılır. Önce const address değişkeni oluşturulur ve böylece kullanıcı terminalden adres bilgisini girebilecektir.

Terminalden adresin girilip girilmediğini kontrol edelim. Bunun için app.js içinde bazı değişiklikler yapalım.

```

// önce import işlemini kullanalım.

const request = require('postman-request');

const geocode= require('./utils/geocode')

const forecast=require("./utils/forecast")

const address=process.argv[2]
if(!address){
    console.log("Lütfen adres bilgisini giriniz!!!")
}
else{

```

```

geocode(address, (error, data)=>{
  if(error){
    return console.log(error)
  }
  forecast(data.latitude, data.longitude, (error, forecastData)=>{
    if(error){
      return console.log(error)
    }
    console.log(data.location)
    console.log(forecastData)
  })
})
})
}

```

Forecast.js içinde yazan kodlar

```

const request= require("postman-request")

const forecast=(longitude, latitude, callback)=>{
  const url =
    "http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&query="
    + latitude+
    ","+
    longitude
  request({ url: url, json:true }, (error, response) => {

    if (error) {
      callback("Hava durumu servisine bağlantı kurulamadı.", undefined)
    }
    else if(response.body.error){
      callback("Girilen konum bilgisi bulunamadı.",undefined)
    }
    else {
      callback(undefined,"Hava sıcaklığı: "
      +response.body.current.temperature+
      " Hissedilen: "+ response.body.current.feelslike)
    }
  })
}
module.exports=forecast

```

Geocode.js içinde yazan kodlar:

```

const request=require("postman-request")
const geocode =(address,callback)=>{
const url =
"https://api.mapbox.com/geocoding/v5/mapbox.places/"+encodeURIComponent
(address) +
".json?access_token=pk.eyJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJJbHUzZWYMDAw
eDN2MmtwZGlzMzQzeG8wIn0.SB2-Lj5qXuozz89bPnWUUQ";

request({ url: url, json: true }, (error, response) => {

  if(error){

    callback("Geocoding servisine bağlanamadı",undefined)
  }
  else if(response.body.features.length===0){
    callback("Belirttiğiniz konum bilgisi bulunamadı",undefined)
  }
  else{
    callback(undefined,{
      longitude: response.body.features[0].center[0],
      latitude: response.body.features[0].center[1],
      location:response.body.features[0].place_name})
  }
});
}
module.exports=geocode

```

Çıktısı:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\user\Desktop\WeatherApp> node app.js ankara
Ankara, Ankara, Türkiye
Hava sıcaklığı: 25 Hissedilen: 24
● PS C:\Users\user\Desktop\WeatherApp> node app.js
Lütfen adres bilgisini giriniz!!!
○ PS C:\Users\user\Desktop\WeatherApp> 

```