

NodeJs Web Programlama Hafta-5 Raporu

Bugünkü derste geçen hafta yazdığımız koddan devam edildi. Artı olarak hata ayıklama ,callback fonksiyonu ve asenkron erişime bakıldı.

Geocoding servisine şehir ismi yerine rastgele bir şey girersek query altındaki features array'i boş olarak gelir.

Herhangi bir konum girme durumunda o şehir ile ilişkili olan yerleri bir array olarak döndürür.

Bu bağlı olarak kod üzerinde düzenleme yapıldı. Düzenleme yapılan yerde features array' inin boş olup olmamasına bakıldı.

Eğer gelen yanıt error'sa ve error null değilse bir hata vardır. Bu durumda servise bağlanmadığı hakkında uyarı verilir.

Eğer konum bulunamama durumu olduysa bunu else if içerisinde sorgulatmamız gerekir. Bunu da yanıtın (response) body kısmının içerisindeki features array inin içeriğinin boyutundan (length) bulabiliriz. Boyut sıfıra eşitse demekki konuma bağlamada sıkıntı yaşamıştır veya girilen konum yanlıştır. Bunu uygun çıktıyı da yine konsol üzerinden veririz.

Son bir durum kalıyor. Bu da else kısmı. Bu kısım da girilen konumun doğru olup sorunsuz bir biçimde çıktı vermesi durumudur. Bu durumda da enlem ve boylam bilgisini çıktı olarak yazdırırız.

```
const geocodeUrl =
  "https://api.mapbox.com/geocoding/v5/mapbox.places/Bursa.json?access_token=pk.eyJ1Ijoiz3ptYXZjMCI6ImEiOiJjbHU0NDh3M3UxZWZWRuMmpuMHZ3eWRzaHdhIn0.TOJnU1NyRoRbIepClJMs7Q";

request({ url: geocodeUrl, json: true }, (error, response) => {
  if (error) {
    console.log("Geocoding servisine bağlanamadı.");
  } else if (response.body.features.length === 0) {
    console.log("Belirtilen konum bulunamadı.");
  } else {
    const longitude = response.body.features[0].center[0];
    const latitude = response.body.features[0].center[1];
    console.log("Enlem : " + latitude + "\nBoylam : " + longitude);
  }
});
```

Yazılan kodumuzda if-else bloklarının doğru çalışıp çalışmadığını kontrol etmek için ise testler yapabiliriz. Bu testler sırasıyla:

- If durumunu test etmek için ethernet bağlantısı kesilmeli
- Else-if durumunu test etmek için url içerisinde rastgele bir konum girilmeli
- Son durum olan else durumunu test etmek içinse url içinde doğru bir konum girilmeli

Ardından callback fonksiyonlar ile ilgili deneme yapılabilmek için callback-test.js dosyası oluşturuldu.

```
setTimeout(() => {  
  console.log("2 saniye time out");  
}, 2000);
```

Yukarıda bulunan kod bloğu setTimeout() fonksiyonudur. Bu fonksiyon içerisinde girilen milisaniye değeri ile içerisinde yazılan kodu , girilen milisaniye kadar gecikmeyle yazdırır. Fonksiyonun

```
() => {  
  console.log("2 saniye time out");  
},
```

kısmi bir callback fonksiyonudur. Bu fonksiyon asenkron bir fonksiyon olmakla beraber her callback fonksiyonu asenkron olmaz. Senkron olarak da tanımlanabilir.

Mesela bir örnek vermek gerekirse:

```
//senkron bir callback fonksiyon  
const names = ["Ali", "Veli", "Ayşe"];  
  
// karakter uzunluğu 4'e eşit veya 4'den küçükleri getir  
const shortNames = names.filter((name) => {  
  return name.length <= 4;  
});
```

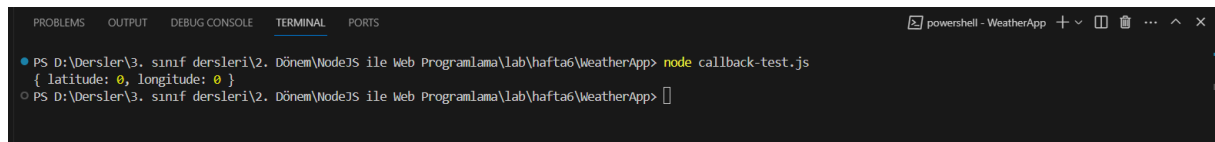
Arrow formatında bir filter fonksiyonu tanımadık. Bu fonksiyon karakter uzunluğu 4'e eşit veya 4'den küçük olan isimleri döndürecek. Bunu da names array' inden teker teker alarak sağlayacak. Yani bu callback fonksiyonu senkron bir fonksiyondur.

Başka bir örneğe bakmak gerekirse:

```
const geocode = (address, callback) => {
  const data = {
    latitude: 0,
    longitude: 0,
  };
  return data;
};

const data = geocode("Bursa");
console.log(data);
```

Kodumu address ve callback olmak üzere iki parametre aldı. İlk olarak uzaktaki web sayfasına bağlanmadan enlem ve boylam bilgileri elden girdik ve return olarak verileri gönderdik. Ardından terminalde çalıştırdık ve çıktı olarak:



```
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node callback-test.js
{ latitude: 0, longitude: 0 }
```

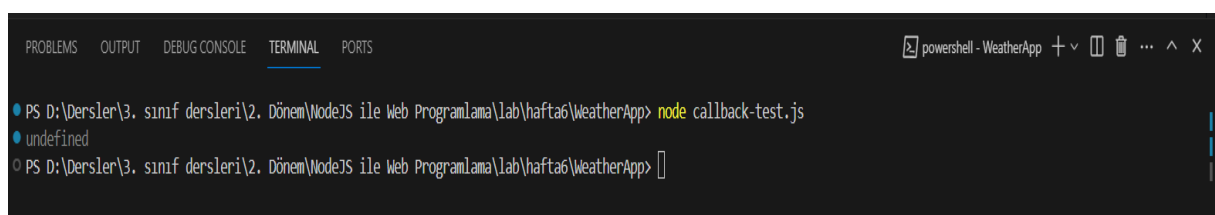
Bu kod senkron haldeydi. Ardından senkron olan bu kodu asenkron hale getirip çalıştırmayı denedik.

O da şu şekilde oldu:

```
const geocode = (address, callback) => {
  setTimeout(() => {
    const data = {
      latitude: 0,
      longitude: 0,
    };
    return data;
  }, 2000);
};

const data = geocode("Bursa");
console.log(data);
```

Asenkron yapmak adına kodumuza setTimeout() fonksiyonunu dahil ettik. 2000 milisaniye gecikme ile az önce yazdığımız kodu parametre olarak alıp çalıştırdık ve tekrardan çalıştırdık. Ve çıktı olarak şunu aldık:



```
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node callback-test.js
undefined
```

<undefined> çıktısı verdi. Bunun sebebi ise kodumuzu her ne kadar asenkron yapsak da fonksiyon çalışırken <return data> ya girer ve değeri döndürmesi gerekir fakat setTimeout() kullandığımızdan 2 saniye gecikmesi gerekir. Böyle olunca da sonuç <undefined> olarak gelir. Bunun yerine <callback(data)> yazarak kullanmayı denedik fakat onda da şu sonucu aldık:

```
12 // },
13
14 const geocode = (address, callback) => {
15   setTimeout(() => {
16     const data = {
17       latitude: 0,
18       longitude: 0,
19     };
20     callback(data);
21   }, 2000);
22 };
23
24 const data = geocode("Bursa");
25 console.log(data);
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + v [] [] ... ^ X

PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node callback-test.js

undefined

D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp\callback-test.js:20
 callback(data);
 ^

TypeError: callback is not a function
 at Timeout._onTimeout (D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp\callback-test.js:20:5)
 at listOnTimeout (node:internal/timers:573:17)
 at process.processTimers (node:internal/timers:514:7)

Node.js v20.11.1

PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp>

Bu yöntemle de hata almış olduk ardından sadece kodu döndürme kısmında değil de çağırma kısmında da değiştirme gitmemiz gerektiğini görmüş olduk. Onun içinse şu şekil bir değişikliğe gittik:

```
const geocode = (address, callback) => {
  setTimeout(() => {
    const data = {
      latitude: 0,
      longitude: 0,
    };
    callback(data);
  }, 2000);
};

// asenkron
geocode("Bursa", (data) => {
  console.log(data);
});
```

Bu seferki halin çıktısı ise şu şekilde oldu:

```
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node callback-test.js
{ latitude: 0, longitude: 0 }
```

Kod başarılı bir şekilde çalıştı. Sebebi ise çağırma kısmında iki parametre aldık. Biri adresimiz olan Bursayken, diğeri de arrow fonksiyon oldu. Bu fonksiyon callback den döndürülecek olan veriyi aldı. Ardından da konsola yazdırma işlemini gerçekleştirdi.

Bir özetleme yapmak gerekirse fonksiyonumuz senkron bir biçimde çalışıyor ise return olarak veriyi döndürebiliriz. Ancak fonksiyonumuz asenkron ise kullanılmaz return yerine callback alır ve diğer duruma artı olarak da callback çağırma fonksiyonu oluşturduk. Bu fonksiyon veri geldiyse ekrana yaz odaklı bir fonksiyondu.

Az önce de yaptığımız gibi bir fonksiyonu asenkron biçimde çalıştırmak için asenkron olan herhangi bir fonksiyon kullanılabiliriz. Biz örneklerimizde setTimeout() fonksiyonunu kullandık. Bunun üzerinden yine başka bir örnek göstermek gerekirse:

```
const add = (a, b, callback) => {
  setTimeout(() => {
    callback(a + b);
  }, 2000);
};

add(1, 4, (result) => {
  console.log(result);
});
```

Kodumuzda add adında bir toplama fonksiyon oluşturduk. Bu fonksiyon 3 parametre aldı. İlk olarak toplanacak sayılardan biri olan a, ve diğeri b son olarak da callback fonksiyonu. Bu fonksiyonu asenkron olarak kullanmak için yine setTimeout() fonksiyonunu kullandık. Fonksiyon içerisinde de callback kullanarak değeri döndürdük. Ardından callback çağırma fonksiyonu ile değerleri girdik ve yine arrow fonksiyonu ile gelen sonucu konsola yazdırdık. Çıktı olarak da şunu aldık:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node
5
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> 
```

Ardından örneklere son verdik. Bundan sonra da app.js dosyasına döndük. Burada yazdığımız kod içerisinde url de değişikliğe giderek belirli yerlerin konumunu, enlem boylam gibi bir çok bilgiye erişebiliyorduk. Fakat her seferinde url üzerinden değişiklik yapıp tekrar tekrar yazmamak adına başka bir dosya üzerinden yazdığımız kodu fonksiyon yapmaya karar verdik. Bu sayede gereken dosyalarda kodumuz yazacak ve biz app.js dosyası içerisinde kodun nasıl çalıştığını görmeden, uğraşmadan yalnızca bir yer ismi girerek çıktılara erişebileceğiz.

Bunun içinse ilk olarak utils adında bir klasör oluşturduk. Ardından içerisine geocode.js ve forecast.js adında iki dosya oluşturduk. Sırasıyla içlerini doldurduk. İlk olarak geocode.js dosyasına baktık ve geocode ye ait kodları geocode.js dosyasına attık. Artı olarak da şunları yazdık:

```
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.eyJ1IjoiZ3ptYXZjMCI6ImEiOiJjbHU0NDh3M3UxZWZlMmMpuMHZ3eWRzaHdhIn0.TOJnU1NyRoRbIepClJMs7Q";

  request({ url: url, json: true }, (error, response) => {
    if (error) {
      //console.log("Geocoding servisine bağlanamadı.");
      //error ve response dondurması yapıyoruz
      callback("Geocoding servisine bağlanamadı.", undefined);
    } else if (response.body.features.length === 0) {
      //console.log("Belirtilen konum bulunamadı.");
      callback("Belirtilen konum bulunamadı.", undefined);
    } else {
      const longitude = response.body.features[0].center[0];
      const latitude = response.body.features[0].center[1];
      //console.log("Enlem : " + latitude + "\nBoylam : " + longitude);
      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;
```

- İlk olarak request modülünü dahil ettik.
- Ardından bir fonksiyon tanımladık. Bu fonksiyonu da callback-test.js dosyasında oluşturduğumuz fonksiyonların mantığı ile yaptık. İlk olarak gerekli bir parametre ardından callback fonksiyonu. Oluşturacağımız fonksiyonda değişecek olan tek şey şehir bilgisi olduğundan parametre olarak yalnızca onu yazdık.
- Arrow fonksiyonu formunda oluşturup içerisine app.js dosyasından kestiğimiz kodları yapıştırdık. Daha önceden 2 url bilgisi olduğundan geocodeUrl yazdığımız değişkeni url olarak değiştirip request içerisinde de url olarak güncelledik.
- Ardından url kısmında şehir yazdırdığımız yeri değiştirdik. Artık gelen adres üzerinden orası güncellenecekti. Fakat adres yazma kısmında boşluk bırakarak yazılması dahilinde hata alacaktık bu yüzden de buna engel olmak adına encodeURIComponent() fonksiyonunu kullandık.
- Artık bir fonksiyon kullandığımızdan dolayı ekrana console.log ile yazdırmak yerine çıktı olarak döndürdük. Çünkü fonksiyonun çağırıldığı yerde ne yapacağına karar vermesi daha doğru bir yaklaşım olacaktı. Bu sebeple de console.log yerine callback kullandık. Callback kullanırken ise geriye error ve response gönderilecekti. Bunlardan yalnızca biri dolu olarak gönderilecek.
 - If bloğuna bakarsak bu blokta error döndürülecek. Bu yüzden callback içerisinde error yani ilk parametre console.log da yazdırılacak mesajı yazarken ikinci parametreye undefined yazdırdık.
 - Else-if bloğunda da yine error kısmı döndürülecek. İlk parametre olarak yine console.log da yazan mesajı yazarken, response kısmında da yine undefined yazdırdık.
 - Son olarak else bloğunda ise bu sefer de error kısmına yani callback fonksiyonunun ilk parametresine undefined yazdırdık ve ikinci parametre olan response kısmına ise json formatında enlem, boylam ve adres bilgisini döndürdük.
- Son olarak da yazdığımız geocode fonksiyonuna ulaşabilmek için exports ettik.

Şimdi app.js dosyası içerisine bakacak olursak da ilk olarak geocode.js dosyasını dosyamıza dahil ettik.

```
const geocode = require("../utils/geocode");
```

```
const request = require("postman-request");

const geocode = require("../utils/geocode");

geocode("Bursa", (error, data) => {
  console.log("Hata: ", error);
  console.log("Data: ", data);
});
```

Bu kod ile de app.js dosyasında çalıştırdık. Çıktı olarak da:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + - [ ] [ ] ... ^ X

• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
  Hata: Geocoding servisine bağlanamadı.
  Data: undefined
○ PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

Aldık çünkü henüz ethernet bağlantısını açmadık. Aynı işlemi açıp yaptığımızda ise şu sonucu aldık:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + - [ ] [ ] ... ^ X

• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
  Hata: undefined
  Data: {
    longitude: 29.06773,
    latitude: 40.182766,
    location: 'Bursa, Bursa, Türkiye'
  }
○ PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

Son olarak da geçersiz bir konum girersek:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + - [ ] [ ] ... ^ X

• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
  Hata: Belirtilen konum bulunamadı.
  Data: undefined
○ PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

Üç şart için de kodumuz başarılı bir şekilde çalıştı.

İlk web servisimizi oluşturmuş olduk.

Daha sonra ikinci servis için yine utils dosyası altında olan forecast.js dosyasına yazmaya başladık.

Yine bu dosyada da app.js dosyasından kestiğimiz kodlar ile yazdık.

İlk olarak kodlara bakmak gerekirse:


```

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

const forecast = (longitude, latitude, callback) => {
  const url =
    "http://api.weatherstack.com/current?access_key=fe043c506076165831ff246584" +
    "cedde3&query=" +
    longitude +
    ",-" +
    latitude +
    "&units=s";

  request({ url: url, json: true }, (error, response) => {
    if (error) {
      //console.log("Hava durumu servisine bağlantı kurulamadı.");
      callback("Hava durumu servisine bağlantı kurulamadı.", undefined);
    } else if (response.body.error) {
      //console.log("Girilen konum bilgisi bulunamadı.");
      callback("Girilen konum bilgisi bulunamadı.", undefined);
    } else {
      callback(
        undefined,
        "Hava şu anda: " +
          response.body.current.weather_descriptions[0] +
          "Hava sıcaklığı şu anda: " +
          response.body.current.temperature +
          " derece ve hissedilen sıcaklık: " +
          response.body.current.feelslike +
          " derece"
      );
    }
  });
};

module.exports = request;

```

- İlk işlem diğer sefer de olduğu gibi require() fonksiyonunu kullanarak request modülünü dahil etmek oldu.
- İkinci olaraksa forecast adında olan fonksiyonu tanımladık. Bu fonksiyona parametre olarak enlem, boylam ve callback fonksiyonunu atadık. Ve kestiğimiz kodu yapıştırdık.
- Url konusunda da sırasıyla enlem ve boylam olmak üzere değişkenleri girdim.
- Console.log yapmak yerine ise callback döndürmesi yaptık. İlk ve ikinci durumda hata durumu olduğundan response undefined oldu.

- Son durumda ise error undefined olup response kısmına enlem, boylam ve sıcaklık bilgisini girdik.
- Yine son olarak bu fonksiyona ulaşmak için forecast fonksiyonunu exports ettik.

Bu kod da bitmiş oldu. Ardından app.js dosyasına dönüp gerekli kontrolleri yaptık. Ondan önce de forecast.js dosyasını kodumuza dahil ettik.

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

Ardından da fonksiyonu çağırma işlemini yazdık.

```
const request = require("postman-request");

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

forecast(-75.7088, 44.15556, () => {
  console.log("Hata: ", error);
  console.log("Data: ", data);
});
```

Çıktı olarak da şunu verdi:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
Hata: undefined
Data: {
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
• Hata: Girilen konum bilgisi bulunamadı.
Data: undefined
○ PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

Doğru bir konumda da şu çıktıyı verir:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
Hata: undefined
Data:
Hava şu anda: Partly cloudy
Hava sıcaklığı şu anda: 16 derece
Hissedilen sıcaklık: 16 derece
Hata: undefined
Data: {
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
• }
○ PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

İki web servisi de başarılı bir şekilde çalıştı. Ardından bu iki kodu birleştik. İlk olarak geocode fonksiyonuna parametre olarak şehir bilgisi girerek çalıştırdık. Ardından fonksiyon içerisinde gelen response bilgilerinden enlem ve boylam bilgisini forecast fonksiyonuna göndererek girdiğimiz şehrin hava durum bilgilerini almış olduk. Bunu da kod üzerinde şu şekilde yaptık:

```
const request = require("request");

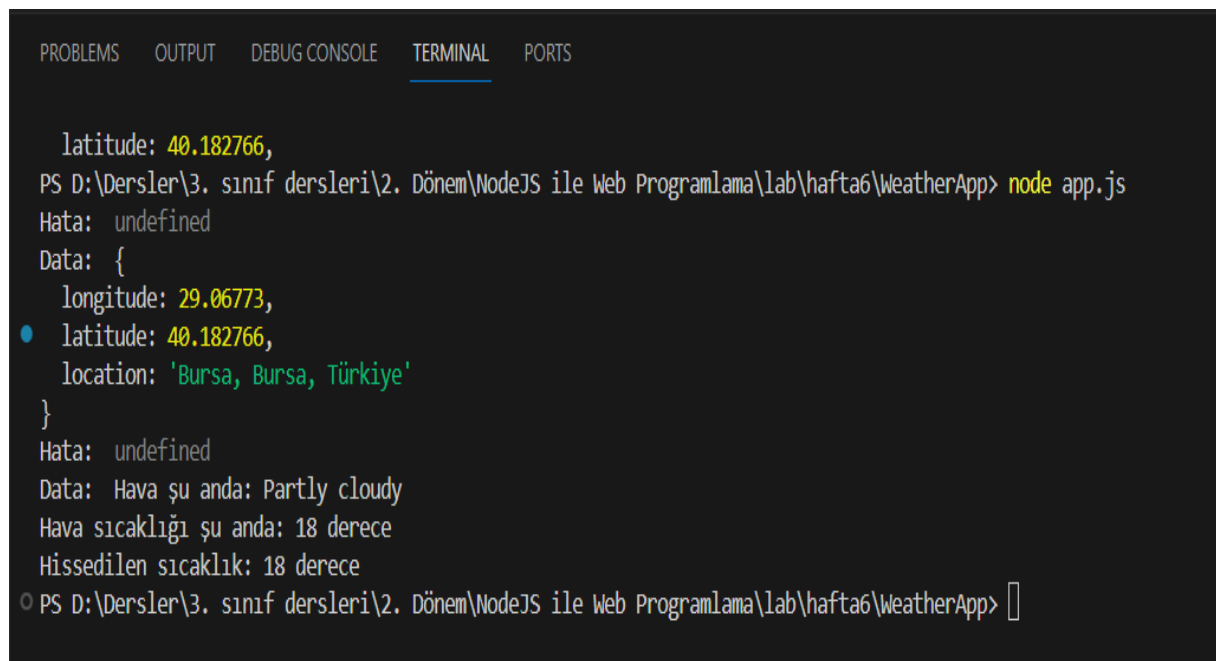
const geocode = require("./utils/geocode");

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

geocode("Bursa", (error, data) => {
  console.log("Hata: ", error);
  console.log("Data: ", data);

  forecast(data.latitude, data.longitude, (error, data) => {
    console.log("Hata: ", error);
    console.log("Data: ", data);
  });
});
```

Çıktı olarak da aşağıdaki sonucu aldık:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

latitude: 40.182766,
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
Hata: undefined
Data: {
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
Hata: undefined
Data: Hava şu anda: Partly cloudy
Hava sıcaklığı şu anda: 18 derece
Hissedilen sıcaklık: 18 derece
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> |
```

Kodumuz başarılı biçimde çalışıyor. Ardından kod üzerinde birkaç değişiklik yaptık. Mesela çıktıda enlem, boylam bilgisini düzeltmek gibi. Enlem boylam bilgisini çıktı üzerinde göstermedik. Yalnızca hata olma durumunda hata bilgisini yazdırdık. Aynı şeyi forecast için de tekrarladık.

Bu değişikliğin ardından güncel kodumuz şu hali aldı:

```
geocode("Bursa", (error, data) => {
  if (error) {
    return console.log(error);
  } else {
    forecast(data.latitude, data.longitude, (error, forecastData) => {
      if (error) {
        return console.log(error);
      }
      console.log(data.location);
      console.log(forecastData);
    });
  }
});
```

Çıktı olarak da şu sonucu aldık.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
Bursa, Bursa, Türkiye
Hava şu anda: Partly cloudy
Hava sıcaklığı şu anda: 18 derece
Hissedilen sıcaklık: 18 derece
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp>
```

Geocode parametresi için Bursa yerine Yıldırım yazarsak yani şehir değil de başka bir konum bilgisi girersek şu çıktıyı alırız:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
Yıldırım Bayezid Camii, Yıldırım Mh. 3. Amaç Sk., Bursa, Bursa 16350, Türkiye
Hava şu anda: Sunny
Hava sıcaklığı şu anda: 19 derece
Hissedilen sıcaklık: 19 derece
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp>
```

Ardından adres bilgisini elden kod üzerinden değil de argüman olarak almayı denedik.

Address adındaki bir değişkene process komutu ile gönderdiğimiz adresi atadık. Daha önce de öğrendiğimiz üzere argüman kullandığımızda ilk iki çıktı işimize yaramıyor ve argüman olarak gönderdiğimiz bilgi üçüncü eleman yani iki numaralı indiste bulunuyor. Bu yüzden döndürülen array içerisindeki ikinci elemanı adrese atadık. Ardından parametre olarak aldık o da şu şekilde oldu:

```
const address = process.argv[2];
geocode(address, (error, data) => {
  if (error) {
    return console.log(error);
  } else {
    forecast(data.latitude, data.longitude, (error, forecastData) => {
      if (error) {
        return console.log(error);
      }
      console.log(data.location);
      console.log(forecastData);
    });
  }
});
});
```

Çıktı olarak da şunu aldık:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + v []
• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js Boston
Hava şu anda: Clear
Hava sıcaklığı şu anda: 3 derece
Hissedilen sıcaklık: 0 derece
• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> []
•
```

Başka bir deneme ile de şu oldu:

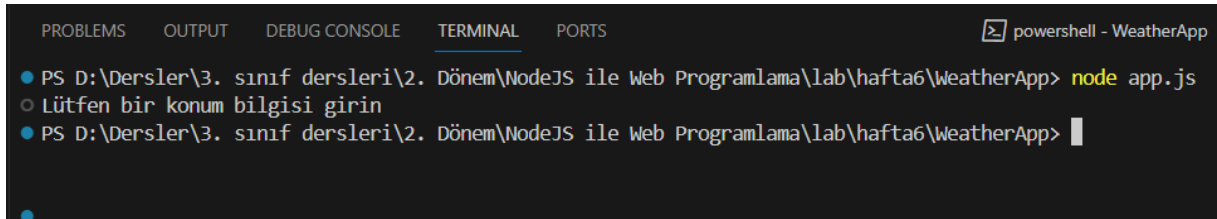
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp + v []
• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js Alsancak
Girilen konum bilgisi bulunamadı.
• PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> []
•
```

Son olarak da girilen adresin olup olmama durumunu kontrol etmek gerekirse:

```
const address = process.argv[2];

if (!address) {
  console.log("Lütfen bir konum bilgisi girin");
} else {
  geocode(address, (error, data) => {
    if (error) {
      return console.log(error);
    } else {
      forecast(data.latitude, data.longitude, (error, data) => {
        if (error) {
          return console.log(error);
        }
        console.log(data);
      });
    }
  });
}
```

Bu sefer de argüman olarak göndermeme bir bilgi göndermeme durumunda uyarı mesajı alırız. Son olarak onun da çıktısına bakmak gerekirse:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - WeatherApp
● PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> node app.js
○ Lütfen bir konum bilgisi girin
● PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta6\WeatherApp> 
```