Kadir ÇAPKIN 21360859023

NodeJS ile Web programlama Raporu Hafta – 6

Weather app – 2

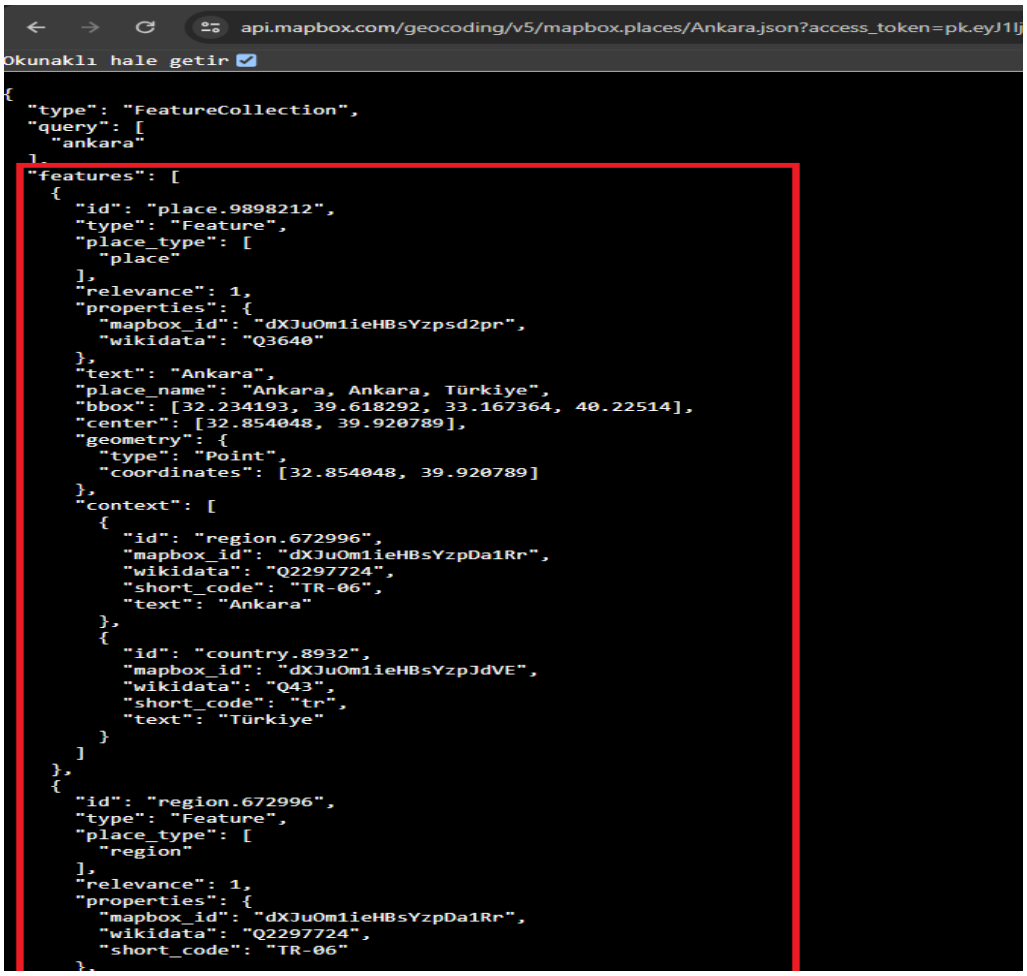
Bu hafta öncelikli olarak geçen öğrendiğimiz api verisini geocode api'sine tekrar uygulayarak başladık. API kullanımını hatırlamak gerekirse öncelikle bir değişkende API url'imizi tutuyorduk ardından request modülünü projemize dahil ederek , bu request modülü sayesinde api url'ine bir talep atıyorduk. Bu talepte dikkat etmemiz gereken şeyler ilk parametre olarak bir nesne alır. Bu nesne attirbute url, ikincisi ise dosya formatı olan json:true olarak belirlenir. İkinci parametre ise bir callback fonksiyon alır ilk parametresi error döndürür ikinci parametresi ise gönderilen talepe karşılık geri dönüş değeri olan response döndürür.

```
const geocodeURL =
"https://api.mapbox.com/geocoding/v5/mapbox.places/Bursa.json?access_token=pk.eyJ
1Ijoia2FkaXIzMjMzIiwiYSI6ImNsdTM5cno3NjA4NnAybW5kZTN3dG9nbWMifQ._hU69OTmxDX6TQIiT
d41-w"

request({url:geocodeURL,json:true},(error,response)=>{
    if(error){
        console.log("Servise baglanamadi.")
    }else if(response.body.features.length == 0){
        console.log("istenilen bilgi bulunamadi.")
    }else{
        const x = response.body.features[0].center[0]; // boylam
        const y = response.body.features[0].center[1]; // enlem
        const name = response.body.features[0].text; // isim

        console.log(x);
        console.log(y);
        console.log(name);
    }
})
```

Yukarıda gördüğümüz kodda amacımız şu oldu: Öncelikli olarak ilk if koşulu istenilen sunucuya gönderilen talebin ulaşıp ulaşmadığını kontrol eder örneğin bir ağ kesintisi yaparsak Servise bağlanamadı çıktısını alırız. İkinci koşul ise talep ettiğimiz şey ile alakalıdır eğer talep ettiğimiz şey bulunmuyorsa örneğin: Şu an url içerisinde Bursa girildi fakat 1x23c1sx gibi bir değer girilmiş olsaydı dünya üzerinde böyle bir konum olmadığı için ikinci koşula girecek ve ekrana istenilen bilgi bulunamadı bastıracaktı. Bunun sebebi bir sorgu yaptığımızda talep edilen konum ile ilgili bilgilerin features adı altında bir array ile gelmesidir. Eğer bu array'in length (uzunluk) değeri 0 ise böyle bir konum bulunmamaktadır.



Bunu yaptıktan sonra ise callback fonksiyonlar ile ilgili testler yaptık bunun için callback-test.js adlı bir dosya oluşturduk. Callback fonksiyonu daha iyi görebilmek adına öncelikli olarak setTimeout fonksiyonunu yazdık.

```
setTimeout(() => {
    console.log("İki saniye bekleniyor...");
}, 2000);
```

Bu fonksiyon ilk parametresine bir fonksiyon alan ikinci parametresine ise bir bekleme süresi alan bir fonksiyondur. İlk parametre olarak aldığı bir fonksiyonu iki saniye beklettikten sonra ekrana bastırır. Bu bir callback fonksiyondur aynı zamanda asenkron olarak çalışır. Her callback fonksiyon asenkron olarak çalışmak zorunda değildir. Yani demek istediğim bir bekletme yapısında kalmak zorunda değildir senkron bir şekilde doğrudan da işlem yapabilir. Bunun örneğini vermek istersek find veya filter metodu verilebilir. Örneğin find metoduna bakarsak : F

```
const names = ["Kadir","Ayse","Bayram","Halil"];

const findName = names.find((name)=>{
    return name.endsWith('r');
})
```

Burada find metodu şu şekilde çalışır bir names arrayi içerisinde find metodunun çağırırsak parametre olarak bir callback fonksiyon alır bu callback fonksiyonun parametresi her bir array elemanının kendisidir.. Yani demek istediğim names array'inin bütün elemanları find metodu sayesinde incelenir ve eğer bu elemanlardan her hangi birisi 'r' harfi ile bitiyorsa findName değişkenine return edilir. Buradaki find metodunun parametre olarak aldığı callback fonksiyonu, settimeout fonksiyonuna nazaran asenkron değil senkron çalışır. Kod çıktısı ise şu şekildedir.

```
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node callback-test.js
Kadir
```

Callback fonksiyonlar ile ilgili başka bir örnek yapmak istersek:

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

}

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

Bu şekilde bir fonksiyon yazdığımızı varsayalım bu fonksiyonun normalde çıktısının data nesnesi olmasını bekleriz ama kod çıktımız şu
şekildedir.

```
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node callback-test.js
Kadir
```

Bunun sebebi şudur biz fonksiyonu çağırıyoruz ve burada sanki fonksiyon hemen tamamlanmış gibi düşünüyoruz. Sonuç olarak bastırdığımız şey gözükmüyor bunun sebebi return 2 saniye bekleme yapan bir asenkron bir fonksiyondur kod çalıştırıldığı anda zaten const data'ya geocode aktarılmaya çalışılıyor fakat 2 saniye beklemeli olarak ardından bir

satır altındaki kod bu 2 saniyelik beklemeyi yapmadan anında çalıştırılıyor bu yüzden ekranda undefined yazısını görüyoruz. Bu sorunu çözebilmek içinse 2.parametre olan callback fonksiyonu kullanarak mevcut fonksiyonu şu şekilde düzenlersek :

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

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

Buradaki beklentimiz şu şekildedir : Asenkron fonksiyonun içerisine callback fonksiyonu yardımıyla console.log fonksiyonunu attık. Bu sayede olan şey fonksiyonumuz(geocode) doğrudan cagirilicak ardindan settimeout fonksiyonu yardımıyla 2 saniye bekletiliyor ardindan parametre olarak aldığı callback fonksiyon cagirildi çıktı olarak ise şunu alıyoruz:

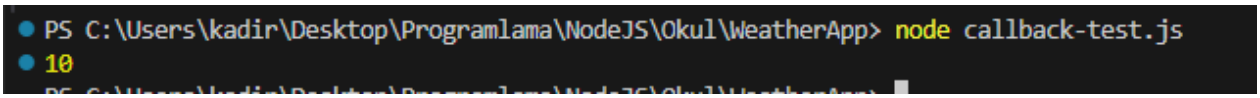
```
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node callback-test.js
{ latitude: 0, longitude: 0 }
```

Şimdi başka bir örnek yapalım parametre olarak 3 sayı alsın ve bunları asenkron bir şekilde toplasın

```
const addNumbers = (a,b,c,callback) => {
    setTimeout(() => {
        callback(a+b+c);
    },3000);
}

addNumbers(2,5,3,(result)=>{
    console.log(result);
})
```

Kodumun şu şekilde olucak ilk 3 parametre toplanıcak sayıları alır. Ardından setTimeout fonksiyonu içerisinde callback fonksiyon çağırılır bu callback fonksiyon ise parametre olarak aldığı şeyi ekrana bastırır. Burada callback fonksiyonuna parametre olarak addNumbers fonksiyonun parametrelerinin toplamı veriliyor bu yüzden kod çıktımız şu şekilde oluyor:



Şimdi yaptığımız örnekleri bir sonuca bağlamak adına weatherapp uygulamamıza entegre edelim. Öncelikli olarak istediğimiz şey şu kod'u çalıştırdığımızda node app.js //3.parametre 3. parametre olarak gelen kısma bir şehir ismi yazıp o şehir ile ilgili belirli bilgileri listelemek. Bunu yapabilmek için yazdığımız API isteklerini farklı .js uzantılı dosyalara taşıyıp util klasörü altında export etmemiz gerekiyor. Önce weatherApi'den başlarsak:

```
const request = require("request")
const weatherapi = (enlem, boylam, callback) => {
  const url =
    "http://api.weatherstack.com/current?access_key=2a7c2b083d9ceb97b1c8f7dbe738d
e81&query=" + enlem + "," + boylam

  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,
        "Konum Bilgisi:" + response.body.location.name + " Hava sicakliği : " +
response.body.current.temperature + " Hissedilen : " +
response.body.current.feelslike
      )
    }
  })
}

module.exports = weatherapi
```

Öncelikli olarak yaptığımız bu talep işlemini bir fonksiyon içerisinde taşıyoruz. Bu fonksiyon parametre olarak bir enlem,boylam ve callback fonksiyon alıcak. Bunu bir fonksiyona taşımamızdaki sebep girmek istediğimiz coğrafi konumu elle değiştirmek ile uğraşmayıp geocode'a parametre olarak yolladığımız şehrin koordinatlarını alıp direkt bu fonksiyon parametrelerine(enlem ve boylam) aktararak hava durumu bilgisini daha kolay bir şekilde elde etmektir. Buradaki callback fonksiyonunu incelemek istersek bu callback fonksiyon bize bir error ve bir data döndürecek şekilde ayarladık birinci parametre error ikinci parametre ise datayı döndürecek o yüzden koşul şartlarına baktığımızda , ilk iki şartta 2. Parametre olarak define olmasının sebebi bir hatanın ortaya çıkması ve bize bir data döndürmemesini belirtmek adına konuldu. Ardından 3.koşula baktığımızda ise ilk parametre olarak undefined ikinci parametre olarak ise bize bilgi mesajı döndürüyor. İlk

parametrenin undefined olmasının sebebi bir hata döndürmemesidir. Daha sonrasında ise mevcut modüle export edip paylaşmış oluyoruz. 2.API talebi bulundurduğumuz geocode'u da aynı şekilde utils klasörü altında bir .js uzantılı dosyaya taşırsak:

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

const geocode = (address, callback) => {
  // function body goes here
  const geocodeURL =
    "https://api.mapbox.com/geocoding/v5/mapbox.places/" + encodeURI(address) +
".json?access_token=pk.eyJ1Ijoia2FkaXIzMjMzIiwiYSI6ImNsdTM5cno3NjA4NnAybW5kZTN3dG
9nbWMifQ._hU69OTmxDX6TQIiTd41-w"

  request({ url: geocodeURL, json: true }, (error, response) => {
    if (error) {
      callback("Geocoding servisine bağlanamadı", undefined)
    } else if (response.body.features.length == 0) {
      callback("Belirttiğiniz konum bulunamadı", undefined)
    } else {
      const boylam = response.body.features[0].center[0] // boylam bilgisini verir
      const enlem = response.body.features[0].center[1] // enlem bilgisini verir
      const konum = response.body.features[0].place_name

      callback(undefined, {
        boylam: boylam,
        enlem: enlem,
        konum: konum
      })
    }
  })
}


module.exports = geocode
```

Burada yaptığımız işlemde weatherapi'ye benzerdir. Olay şu şekilde ilerliyor geocode fonksiyonu ilk parametre olarak konum bilgisini , ikinci parametre olarak ise bir callback fonksiyonu bulunduruyor. Birinci parametre olan konum bilgisini biz kodu çalıştırırken vereceğiz ikinci callback fonksiyonun işlevi ise şu şekildedir. İstenilen şehre göre api'ye bir talep atılır. Bu talep sonucu ise callback fonksiyonunda tutulur. Yani demek istediğim callback fonksiyonunun ilk parametresi error tutarken ikinci parametre ise api'den gelen veriyi tutacak şekilde ayarladık. Eğer istenilen bilgili ulaşılamadıysa ilk iki koşul bloğunda olduğu gibi ilk parametreye bir hata mesajı , 2.parametreye ise undefined verilir. Sonuç olarak hata bulundurmuyorsa ilk parametreye undefined , ikinci parametreye ise talep

sonucunda elde edilen bilgiler bir nesne şeklinde döndürülür. Bu nesneler ise boylam, enlem ve konum bilgileridir. Ardından bu fonksiyonu da export ediyoruz. Şimdi bu iki dosyayı index.js içerisinde birlikte kullanmamız gerekiyor. Yapmak istediğimiz şey yukarıda yazdığım gibi uygulama çalıştırırken bir şehir ismi vermek , ve bu şehir ismine göre geocode.js içerisinden konum bilgisinin verilmesi, ardından bu konum bilgisinin weatherapi fonksiyonuna parametre olarak verilmesi ve sonuçlarının ekrana bastırılmasıdır. Kodumuza bakacak olursak :

```
const request = require('request');
const geocode = require('./utils/geocode');
const weatherapi = require('./utils/weatherapi');

const address = process.argv[2]; // node app.js KonumBilgisi //Bu parametlere array şeklinde döndürür konum bilgisi bu yüzden 2.indexte bulunur.

geocode(address,(errGeoAPI,data)=>{
    if(errGeoAPI){
        console.log("Hata bulundu:", errGeoAPI);
        return
    }

    weatherapi(data.enlem,data.boylam,(errWeatherAPI,data)=>{
        if(errWeatherAPI){
            console.log("Hata bulundu:",errWeatherAPI);
            return
        }
        console.log(data);
    })

})
```

Öncelikli olarak utils içerisine yazdığımız .js uzantılı dosyaları app.js dosyamıza dahil ediyoruz. Ardından kodu çalıştırırken girdiğimiz adres(node app.js Antalya) bilgimizi address (Antalya) adlı değişkende tutuyoruz. Daha sonrasında geocode fonksiyonumu çağırıyoruz bu fonksiyon hatırlarsak ilk parametre olarak bir konum , ikinci parametre olarak ise bir callback fonksiyonu alıyordu. Bu callback fonksiyonu ise iki parametre alıyordu birinci parametre bir error ikinci parametre ise :

```
      callback(undefined, {
        boylam: boylam,
        enlem: enlem,
        konum: konum
      })
```

Bilgilerini tutan bir nesneydi. Ardından weatherapi fonksiyonumuzu çağırıyoruz. Bu fonksiyonda üç parametre alıyordu: Birinci parametre enlem, ikinci parametre boylam , üçüncü parametre ise ilk parametresi error ikinci parametresi data olan bir callback fonksiyondu. İşte tam burada geocode datasından dönen , istediğimiz şehrin enlem ve boylam bilgilerini weatherapi fonksiyonunun ilk iki parametresine aktarıyoruz. Bunu yapabiliyoruz çünkü zaten weatherapi fonksiyonu, geocode fonksiyonunun callback fonksiyonunda çağırmıştık bu sayede konum datasına doğrudan ulaşabiliyoruz.

```
(errGeoAPI,data)=>{
    if(errGeoAPI){
        console.log("Hata bulundu:",errGeoAPI);
        return
    }

    weatherapi(data.enlem,data.boylam,(errWeatherAPI,data)=>{
        if(errWeatherAPI){
            console.log("Hata bulundu:",errWeatherAPI);
            return
        }
```

Bu şekilde weatherapi fonksiyonunu çağırdıktan sonra ilk önce her hangi bir hata aldık mı diye bir koşul bloğu ekliyoruz eğer geliyorsa return diyerek fonksiyonu sonlandırıyoruz eğer gelmediyse mevcut data bilgisini konsola yazdırıyoruz. Şimdi kodumuzu çalıştırırsak ekran çıktısı olarak şöyle gözüküyor.

```
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Bursa
Konum Bilgisi:Bursa Hava sicakliği : 16 Hissedilen : 16
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Balıkesir
Konum Bilgisi:Karassi Hava sicakliği : 17 Hissedilen : 17
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Ankara
Konum Bilgisi:Ankara Hava sicakliği : 15 Hissedilen : 16
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Edirne
Konum Bilgisi:Edirne Hava sicakliği : 13 Hissedilen : 13
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Antalya
Konum Bilgisi:Antalya Hava sicakliği : 19 Hissedilen : 19
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Paris
Konum Bilgisi:Paris Hava sicakliği : 10 Hissedilen : 9
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Yakutsk
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp> node app.js Yakutsk
Konum Bilgisi:Якутск Hava sicakliği : -29 Hissedilen : -35
PS C:\Users\kadir\Desktop\Programlama\NodeJS\Okul\WeatherApp>
```