Gizem AVCI 21360859071

NodeJs Web Programlama Hafta-5 Raporu

Bu hafta yeni bir projeye başladık. Hava durumu uygulaması işlevinde bir uygulama gelişmeye başladık.

Bugün asenkron web server ları nasıl oluyor ona baktık. İki farklı web sitesinin API sini kullandık. Bunlardan biri hava durumu API. Bu API , verilen enlem boylam bilgileri için hava durumu bilgisini getiriyor. Diğer API ise verilen bir adres için enlem ve boylam bilgisini getiriyor. Bu iki API birlikte entegre kulandık. İstedğimiz bir yerin enlem ve boylamını bulduk ve bununla beraber enlem boylamıyla da hava durumu bilgisini aldık.

İki klasör oluşturduk : WeatherApp,NodeAPP.

İlk olarak WeatherApp e bakacağız. app.js dosyası açtık.

Bu zamana kadar hep senkron işlemler yaptık. Şimdi asenkron işlemler yaptık ve bunu da daha iyi kavramak için küçük bir kod yazdık.

İki yazı arasına setTimeout() fonksiyonu kullandık. Parametre olarak arrow formatı ve int sayı aldı bu da kaç milisaniye beklediğini ifade ediyor.

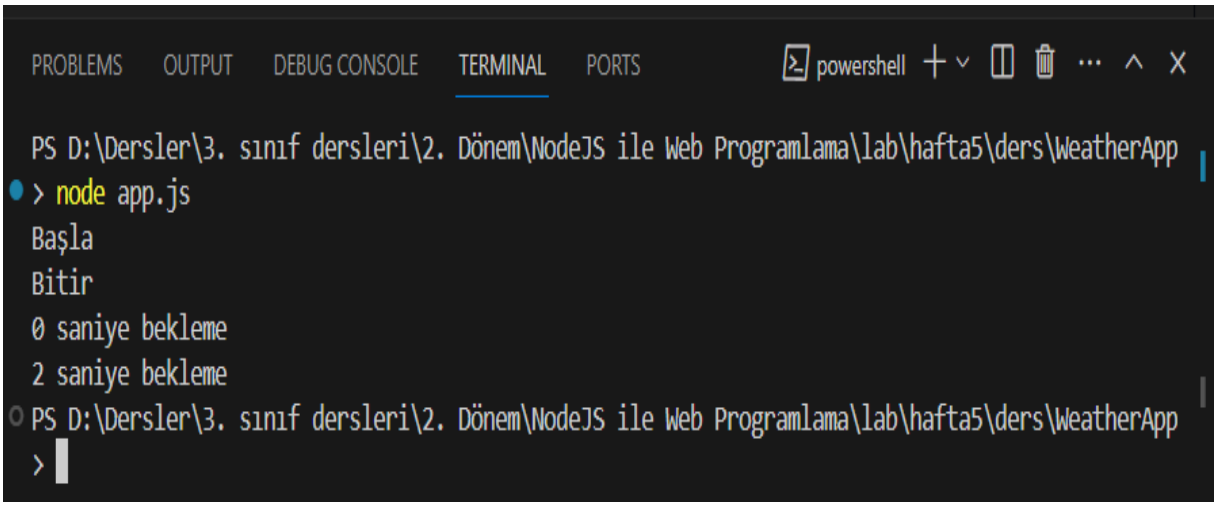
```
console.log("Başla");

//asenkron fonksiyon
setTimeout(() => {
  console.log("2 saniye bekleme");
}, 2000); //2 saniye bekler

setTimeout(() => {
  console.log("0 saniye bekleme");
}, 0); //2 saniye bekler

console.log("Bitir");
```

Çıktı olarak şu geldi :



Görüldüğü üzere çıktıda ilk 0 saniye bekleme yazıldı fakat biz ilk önce 2 saniye bekleme bloğunu yazdırmıştık.

Yani kod yukarıdan aşağı doğru çalışmadı bu da asenkron olduğunu gösteriyor.

setTimeout() asenkron bir fonksiyondur. Bu fonksiyonlar kodun geri kalan kısmını bekletmez. 2 saniye bekleme esnasında kod devam ettiğinden kayıp uğramadan bir sonraki aşamaya gider. Ardından 0 saniye bekleme bloğuna gider. Fakat mantıken bakınca ilk başla, ardından 2 saniye beklerken 0 saniyeyi yazdırma, ardından bitiri yazdırma ve son olarak 2 saniye beklemeyi yazdırma çalışmasını bekleriz. Ama koda bakınca pek de aynı gözükmüyor. İlk bitir oldu çünkü nodeJS de fonksiyon çalışırken stack mekanizması var ve bu stack üzerine eklenerek çalıştırılır. Bu stack'te esas ve o stack in en altında olan main fonksiyonudur. Yani bu koda göre bakarsak asenkron olmayan kısımlar main fonksiyondur. Bu fonksiyon tamamlanmadan stackteki diğer fonksiyonlar çalıştırılmaz.

Bu yüzden 0 saniyeden önce bitir satırı çalışır ardından 2 saniye bekleme kısmı ve son olarak 0 saniye bekleme kısmı çalışır. Fakat 0 saniye daha önceden bittiğinden ilk onun çıktısını alırız.

Sonuç olarak özetlemek gerekirse kod tarama esnasında ilk olarak <Başla> yazar ardından yine main fonksiyonu içerisindeki <Bitir> yazar. Ardından stack in içerisindeki diğer fonksiyonlar yazdırılır. Bu diğer fonksiyonlar ise setTimeout() fonksiyonudur. Tanımlanmanın ardından stack in içerisine gönderilir ve burada çalıştırılır. 2000 mili saniyeden geriye doğru sayar. Diğeri ise 0 dan geriye doğru sayar ve önce 0 saniye bittiği için ilk olarak <0 saniye bekletme> ardından da <2 saniye bekletme> satırları çıktı verir.

setTimeout(),NodeJS ile birlikte gelen özelliktir. V8 motorunun bir parçası değil ekstradan eklenmiş nodejs ile birlikte gelen özelliktir.

Asenkron mantığını anladıktan sonra request adında modülü kullanarak web sunucularına sorgu gönderdik ve o sunuculardan gelen yanıtı bekleyip sonucu aldıktan sonra da formatlayıp ekrana yazdırdık.

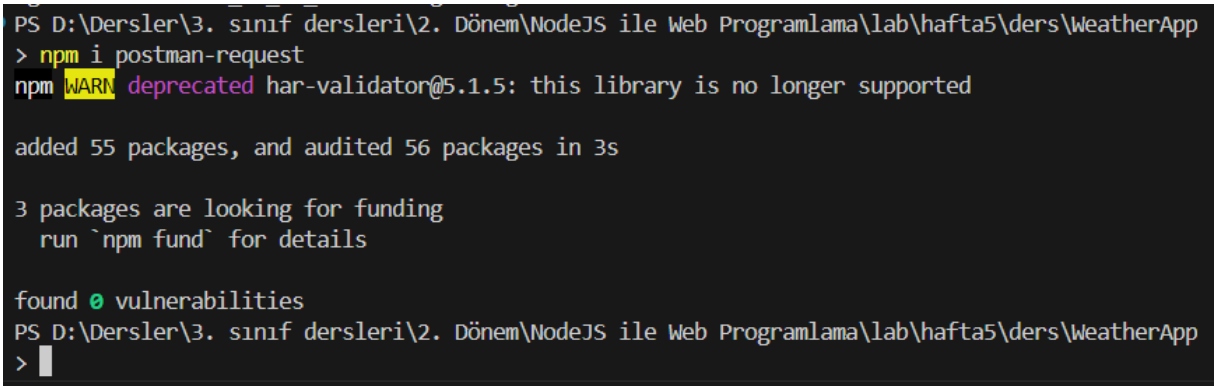
Bu aşamada kullanılacak API ler <weatherstack.com> ve <mapbox.com> dan gelecektir. İlk olarak weatherstack.com a baktık. Ücretsiz bir şekilde kaydolduk.

Ardından verilen API erişim anahtarımızı aldık. Bu anahtarı kullanarak sorgulama yaptık.

Mapbox da ise yine giriş yaptık. Ondan da üye olduk.

Ardından app.js dosyamıza girip < npm init -yes> diyerek sorulara otomatik default yanıt oluşturarak package.json dosyamızı oluşturduk.

<npm i request> ve < npm i postman-request> modüllerini terminalde kurduk. Postman request i alıyor ve geliştirmeye devam ediyor.



weather-stack e sorgu gönderdik.

api.weatherstack.com/current?access_key=fe043c506076165831ff246584cedde3&query=37.8267,-122.4233

<api.weatherstack.com> bu bizim temel adres bilgimiz. API sorgusu yaparken buna sorgu gönderiyoruz. Ardından soru işaretinden sonra aynı konsolda parametre geçişi yapar gibi burada da parametre geçişi yapıyoruz. İki tane parametre geçirdik ve bu parametreler soru işaretinden sonra geliyor ve birden fazla parametre olma durumunda ampersand yani (&) işareti kullanarak bunları birbirinden ayırıyoruz. Her bir parametre key-value yani anahtar değer ilişkisi ile arasında eşittir olacak şekilde yazılması gerekiyor. İlk parametre olarak anahtar ve eşittir yazıp weatherstack.com dan aldığımız API anahtar erişim değerini yazdık ve ikinci parametre olarak ise query ile enlem ve boylam bilgisini aldık.

Bu link chrome de görüntüleme sağlamadı çünkü Chrome otomatik olarak http den https e çevirdi ve bu şekide ücretsiz hesaptan erişim sağlanamadı.

Bir uzantı ekledik. Bunu da şu yol ile yaptık :

Uzantılar->Chrome web sayfasını ziyaret edin->json formatter aramasını yaptır -> çıkan eklentiyi kur

Bu sayede json formatlı şekilde çıktı verir.

Bunu da bitirdikten sonra tekrar konumuza dönersek FireFox gibi başka tarayıcılardan linkimizi girdirip çıktı alabildik fakat ben yine Chrome üzerinden https yazan yerde <s> kısmını silerek de sonuç elde edebildim. Şimdi çıkan sonuca bakalım:


```
{
  "request": {
      "type": "LatLon",
      "query": "Lat 37.83 and Lon -122.42",
      "language": "en",
      "unit": "m"
  },
  "location": {
      "name": "San Francisco",
      "country": "United States of America",
      "region": "California",
      "lat": "37.775",
      "lon": "-122.418",
      "timezone_id": "America/Los_Angeles",
      "localtime": "2024-03-23 06:00",
      "localtime_epoch": 1711173600,
      "utc_offset": "-7.0"
  },
  "current": {
      "observation_time": "01:00 PM",
      "temperature": 11,
      "weather_code": 296,
      "weather_icons": [
          "https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0033_cloudy_with_light_rain_night.png"
      ],
      "weather_descriptions": [
          "Light Rain"
      ],
      "wind_speed": 17,
      "wind_degree": 240,
      "wind_dir": "WSW",
      "pressure": 1012,
      "precip": 0.6,
      "humidity": 89,
      "cloudcover": 75,
      "feelslike": 9,
      "uv_index": 1,
      "visibility": 11,
      "is_day": "no"
  }
}
```

Location bilgisi ; şehir, bölge, enlem, boylam ve artı olarak saat sıcaklık … gibi bilgileri buradan alabiliriz.

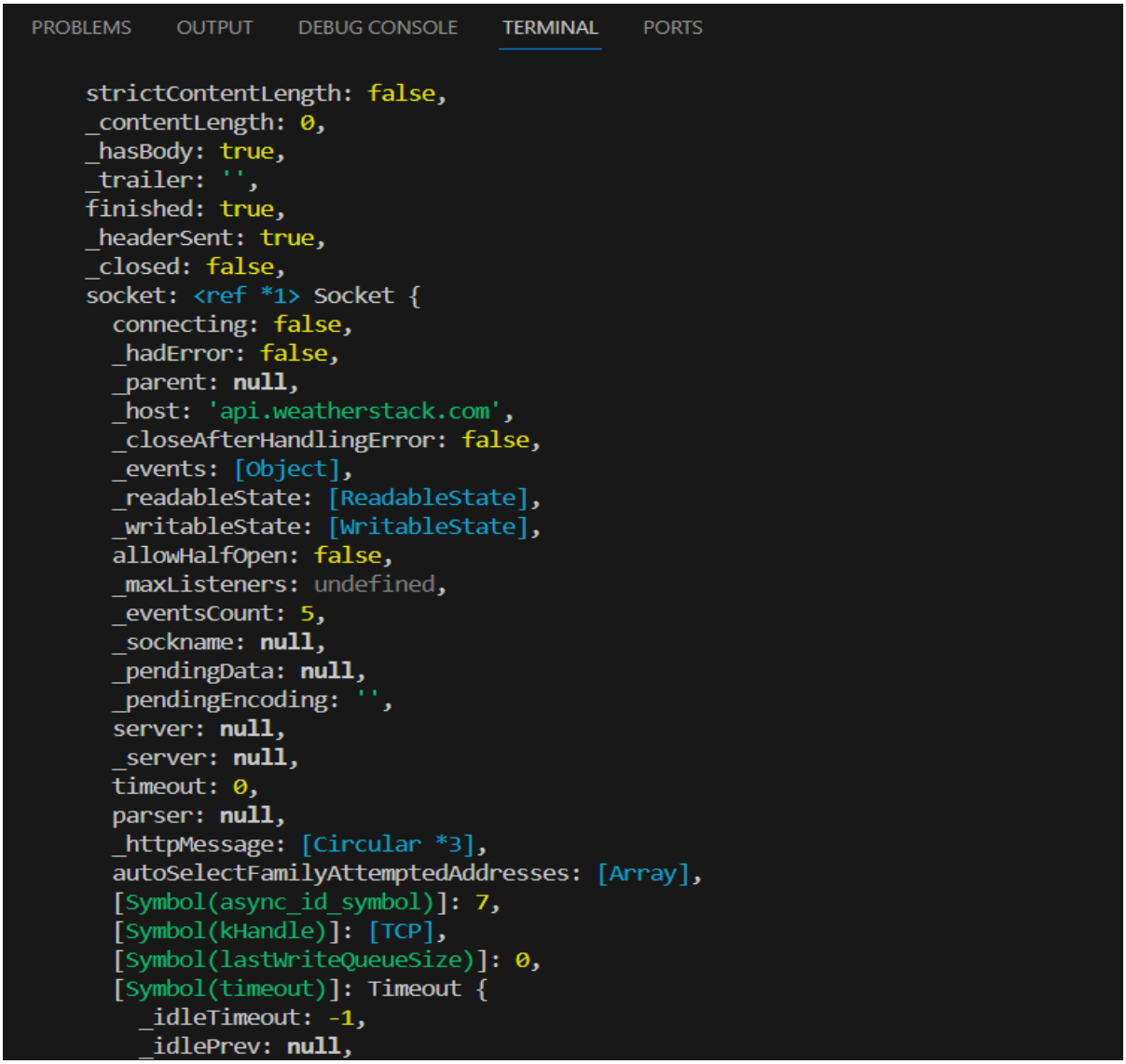
app.js dosyasına döndük.

Postman.com dan postman indirip API ye sorgu gönderebiliriz.

```
const request = require("request");
url="http://api.weatherstack.com/current?access_key=fe043c506076165831ff246584cedde3&query=37.8267,-122.4233";

request({ url: url }, (error, response) => {
  console.log(response);
});
```

Request'e url göndeririz. url : url diyerek benim kullanacağım url kodda yazdığım url anlamına getiriyoruz. Error hata olma durumunda hatayı tespit ederken responce , hata olmama durumunda karşıdan gelen yanıtı içine ekler. Terminalde kodumuzu çalıştırırsak şu çıktıyı alırız :



Bunun gibi uzunca bir çıktı alırız ve bizim için önemli olan body kısmıdır.

Linke gittiğimizde içinde genel üç bilgi var : request , location ve current. Bizim ihtiyacımız olan da current içerisinde bilgiler.

Şimdi yazdığımız koda bakarsak :

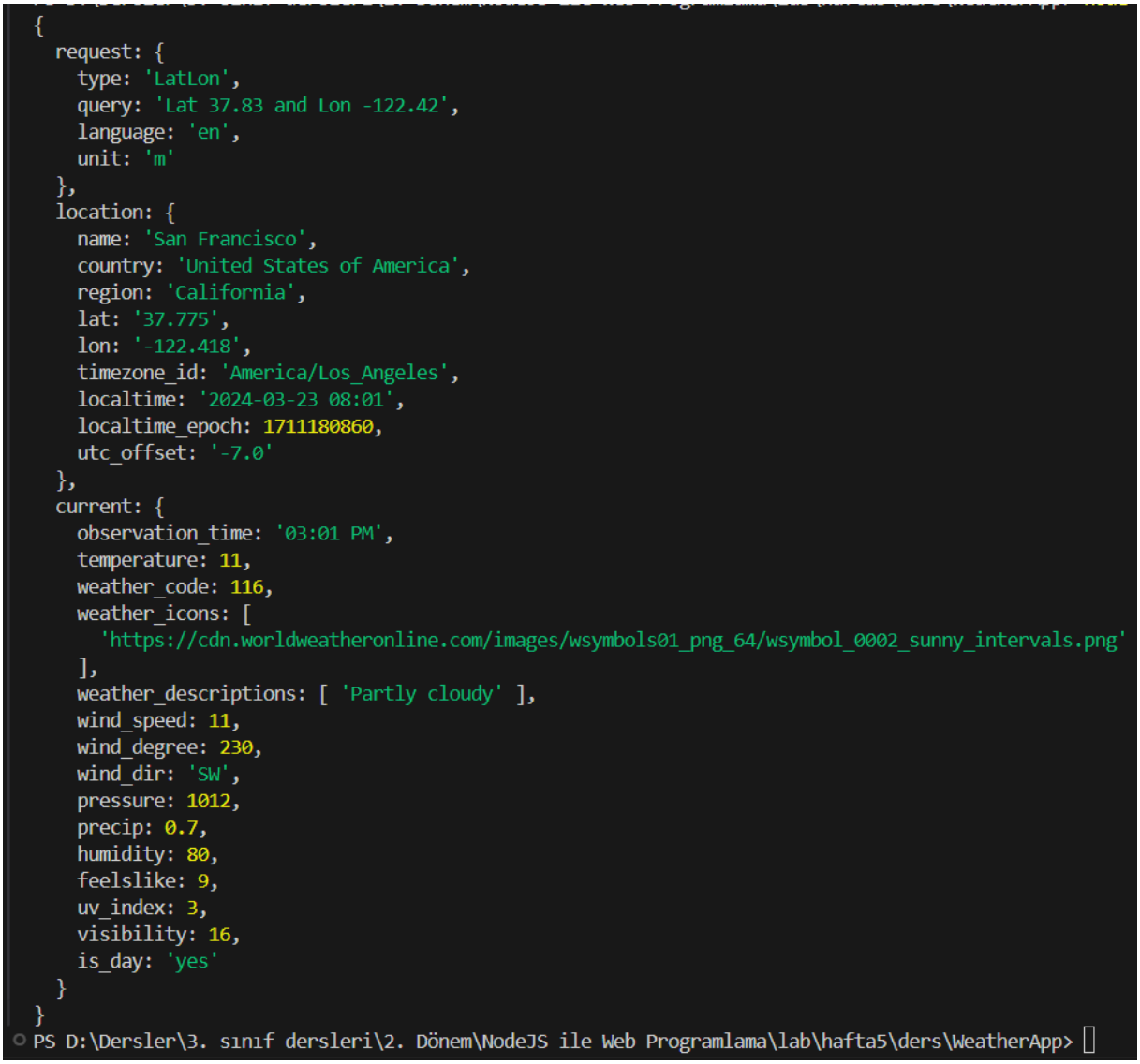
```
const request = require("request");

url =

"http://api.weatherstack.com/current?access_key=fe043c506076165831ff
246584cedde3&query=37.8267,-122.4233";

request({ url: url }, (error, response) => {
  const data = JSON.parse(response.body);
  console.log(data);
});
```

Body kısmını <JSON.parse()> json formatını javascript nesnesine dönüştürmeye yarar. Biz de body kısmını data değişkeni içerisine atık ve console.log ile yazma işlemini gerçekleştirdik. Çıktımız :



Ardından ihtiyacımız olan current i yazdırdık :

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

url =
"http://api.weatherstack.com/current?access_key=fe043c506076165831ff
246584cedde3&query=37.8267,-122.4233";

request({ url: url }, (error, response) => {
  const data = JSON.parse(response.body);
  console.log(data.current);
});
```

Bunun çıktısı ise :

```
  node app.js
{
  observation_time: '03:35 PM',
  temperature: 11,
  weather_code: 116,
  weather_icons: [
    'https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0002_sunny_intervals.
png'
  ],
  weather_descriptions: [ 'Partly cloudy' ],
  wind_speed: 9,
  wind_degree: 160,
  wind_dir: 'SSE',
  pressure: 1012,
  precip: 0.7,
  humidity: 80,
  cloudcover: 75,
  feelslike: 9,
  uv_index: 3,
  visibility: 16,
  is_day: 'yes'
}
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta5\ders\WeatherApp
>
```

url dışında json:true de gönderebiliriz burada requestin özelliğni ayarlıyoruz. Yani url kısmı url , json kısmı ise true. Artık doğrudan parse etmeden yazdırabiliriz. O da şu şekilde olacaktır :

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

url =

"http://api.weatherstack.com/current?access_key=fe043c506076165831ff
246584cedde3&query=37.8267,-122.4233";

request({ url: url, json: true }, (error, response) => {
  console.log(response.body.current);
});
```

Şimdi de bunun üzerinden istenilen bilgileri elde edebiliriz.

Mesela sıcaklık, rüzgar yönü vs. gibi bilgiler. Bunu da response içerisindeki bodyden ondan da current bloğundan ulaşırız.

Mesela bir koda bakmak gerekirse :

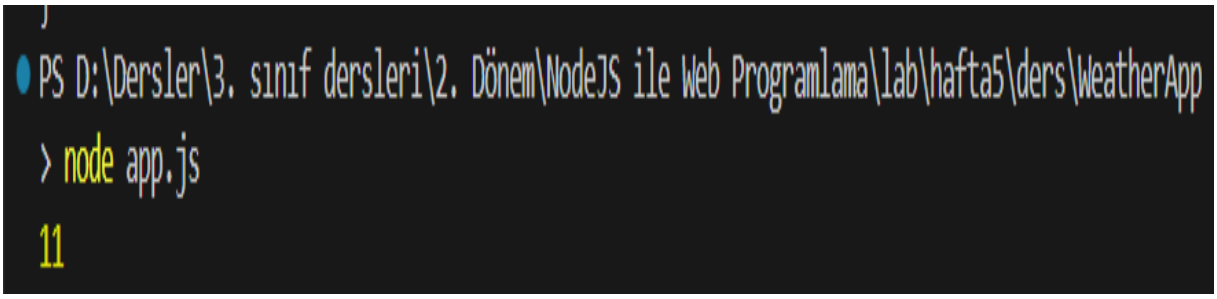
```
const request = require("request");

url =
```

```
"http://api.weatherstack.com/current?access_key=fe043c506076165831ff
246584cedde3&query=37.8267,-122.4233";

request({ url: url, json: true }, (error, response) => {
  console.log(response.body.current.temperature);
});
```

Çıktı olarak da sadece sıcaklık bilgisini verdi :



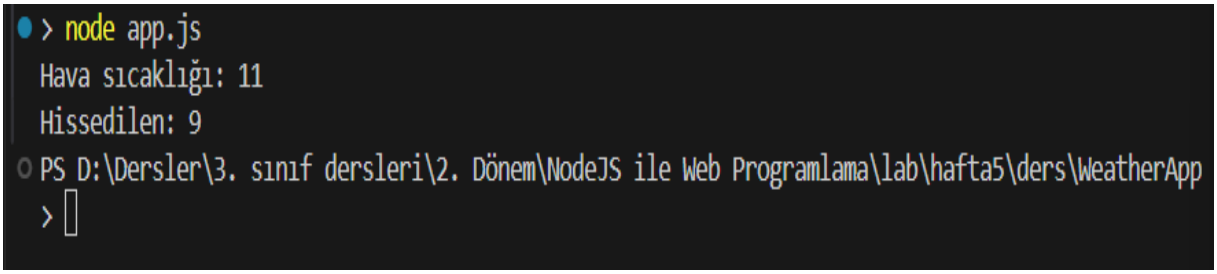
Ardından başka özelliğe bakmak gerekirse :

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

url="http://api.weatherstack.com/current?access_key=fe043c5060761658
31ff246584cedde3&query=37.8267,-122.4233";

request({ url: url, json: true }, (error, response) => {
  console.log(
    "Hava sıcaklığı: " +
      response.body.current.temperature +
      "\nHissedilen: " +
      response.body.current.feelslike
  );
});
```

Buranın çıktısı da :



Şimdi de sıcaklık üzerinde durursak ; sıcaklık derece fahrenheit , celcius , kelvin olarak ölçülebilir. Çıktımızda istediğimiz yanıt içinse

units = m → santigrat

units = s → kelvin

units = f → fahrenheit

Hiçbir şey yazılmazsa ise default olarak santigrat derece olarak gelecektir.

Koda bakmak gerekirse url sonuna yine ampersand ile ayırıp yazdırmak gerekirse :

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

url =

"http://api.weatherstack.com/current?access_key=fe043c506076165831ff
246584cedde3&query=37.8267,-122.4233&units=s";

request({ url: url, json: true }, (error, response) => {
  console.log(
    "Hava sıcaklığı: " +
      response.body.current.temperature +
      "\nHissedilen: " +
      response.body.current.feelslike
  );
});
```

Çıktı olarak da :

```
> node app.js
Hava sıcaklığı: 284
Hissedilen: 282
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta5\ders\WeatherApp
>
```

weather_descriptions , hava bilgisi verir. Ama ingilizce olarak. Mesela “clear”. Yani sözel hava durumu bilgisi de diyebiliriz.

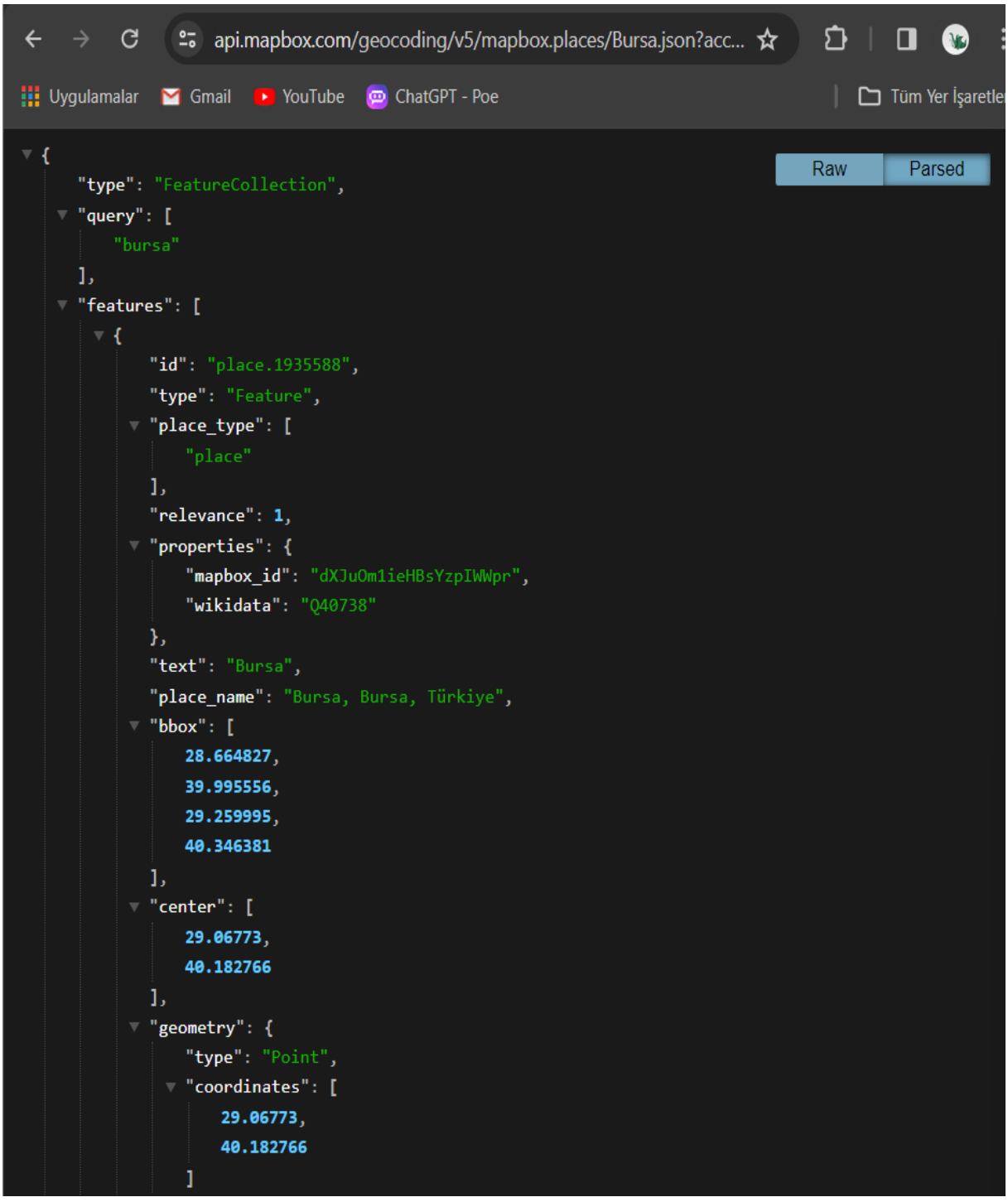
Diğer servise geçtik. Mapboz.com üzerinden hesap oluşturmuştuk. Bu kısımda İlkinde key bilgisini access olarak vermiştik şimdi ise de token olarak gireriz. Link olarak bu kez :

api.mapbox.com/geocoding/v5/mapbox.places/Bursa.json?access_token=pk.eyJ1IjoiZ3ptYXZjMCIsImEiOiJjbHU0NDh3M3UxZWRuMmpuMHZ3eWRzaHdhIn0.TOJnU1NyRoRbIepClJMs7Q

yazarız. Linkte konum bilgisi ve mapbox.com dan aldığımız token bilgisini girdik.

Enlem ve boylam bilgisi almaya çalışıyor onu da center ‘den alacak.

Linke baktığımız zaman uzunca bilgiler içeriyor.id bilgisi , tip, mekan tipi, koordinatlar , context bloğu gibi bir çok çıktı veriyor. Yine Chrome den çalıştırdığım linke bakmak gerekirse:



Ardından enlem ve boylam bilgilerini almaya çalıştık. Linke baktığımızda feature nin ilk indeksi ve onun içinden de center in ilk ve ikinci indeksinden bilgiler çekilebilir.

Kod üzerinden bakmak gerekirse :

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

url="http://api.weatherstack.com/current?access_key=fe043c506076165831ff246584cedde3&query=37.8267,-122.4233&units=s";
```

```
request({ url: url, json: true }, (error, response) => {
  console.log(
    "Hava sıcaklığı: " +
      response.body.current.temperature +
      "\nHissedilen: " +
      response.body.current.feelslike
  );
});

const
geocodeUrl="https://api.mapbox.com/geocoding/v5/mapbox.places/Bursa.
json?access_token=pk.eyJ1IjoiZ3ptYXZjMCIsImEiOiJjbHU0NDh3M3UxZWRuMmp
uMHZ3eWRzaHdhIn0.TOJnU1NyRoRbIepClJMs7Q";

request({ url: geocodeUrl, json: true }, (error, response) => {
  const longitude = response.body.features[0].center[0];
  const latitude = response.body.features[0].center[1];
  console.log("Enlem : " + latitude + "\nBoylam : " + longitude);
});
```

Çıktı olarak da şunu aldık :

```
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta5\ders\WeatherApp
> node app.js
Enlem : 40.182766
Boylam : 29.06773
Hava sıcaklığı: 285
Hissedilen: 283
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta5\ders\WeatherApp
```

Çıktı olarak bize responce geldi ama hata da alabilirdik.

Hata nasıl handler edilir ona baktık. Uzaktaki sunucuya bağlayıp bilgi almaya çalışıyoruz tabii ki de burada hatalar alınabilir.

Networkluk hata oluşabilir. İnternet bağlantısı kesilebilir. Yanlış web sitesi girilebilir.

Mantıksal hata da alabiliriz onu da weathersteak içerisinde handler adilir ve çıktı olarak bize verilir.

Bu hatalar error içerisine girecektir.

```
request({ url: url, json: ture }, (error, response) => {
  console.log(error)
});
```

Bunu yazdırma durumunda hata durumu : null değerini alır olur.

Etherneti devre dışı bırakıp çalıştırsak:

Hata alırız response yi yazdırmaya çalışırsak <undefined> çıktısını yazdırır.

Sorgu ya başarılı ya başarısızdır. Başarılı ise responce aksi halde error un içi dolar.

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

"api.weatherstack.com/current?access_key=%&5c3376fd0dd61328d80dc0652
1e4bf2a&query=37.8267,-122.4233&units=s";
request({ url: url, json: true }, (error, response) => {
  if (error) {
    console.log("Hava durumu servisine bağlantı kurulamadı.");
  } else {
    console.log(
      "Hava sıcaklığı:" +
        response.body.current.temperature +
        "Hissedilen:" +
        response.body.current.feelslike
    );
  }
});
```

İlk linkte enlem boylam bilgisini yazmazsak ne olur?

Sorgu gönderme başarılı http request başarılı olur. Fakat cevabın içeriği hatalıdır. Ve istenilen değer elde edilemez. Bu tür hatayı handler edilebilmesi için else if bloğu açarız ve içerisine gereken çıktıyı atarız.

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

"api.weatherstack.com/current?access_key=%&5c3376fd0dd61328d80dc0652
1e4bf2a&query=37.8267,-122.4233&units=s";
request({ url: url, json: true }, (error, response) => {
  if (error) {
    console.log("Hava durumu servisine bağlantı kurulamadı.");
  }
  else if(response.body.error){
    console.log("Girilen konum bilgisi bulunamadı.")
  }
  else {
    console.log(
      "Hava sıcaklığı:" +
        response.body.current.temperature +
```

```
        "Hissedilen:" +
        response.body.current.feelslike
    );
  }
});
```

Girilen url de enlem boylamı kaldırıp kodu çalıştırma durumunda yazdığımız else if bloğuna düşer.