ÖZLEM ARSLAN 19360859071

BURSA TEKNİK ÜNİVERSİTESİ NODE.JS ile WEB PROGLAMA DERSİ 5.HAFTA

TEORİ DERSİ RAPORU

Bu hafta hava durumu uygulaması geliştireceğiz. Asenkron web serverlar nasıl yapılır bunları inceleyeceğiz. İki farklı web sitenin apisini kullanacağız. Birincisi hava durumu apisi bu api verilen enlem ve boylam bilgilerine göre hava durumu bilgisini verir diğeri ise verilen adres için enlem ve boylam bilgisini vermektedir.

Önce yeni bir klasör oluşturalım adını WeatherApp verelim. Daha sonra app.js dosyası oluşturalım. Bu zamana kadar hep senkron kodlar ile uğraştık. Daha önceki yazdığımız kodlarda dosyaya yazdırma işlemi bulunmaktaydı bunu senkron olarak yapmıştık çünkü hard diske erişim gerektiren bir olaydı dolaysıyla zaman almaktadır. Asenkron ve senkron ile arasındaki farkı öğrenmeye çalışalım. Asenkron fonksiyon ekleyerek yazı yazdıralım.

```
console.log("başla")
//arrow fonksiyon tipinde fonksiyon tanımlandı. 2000 milisayni bekleyecektir.
setTimeout(() => {
   console.log("2 saniye beklemek için");
 }, 2000);

 setTimeout(() => {
   console.log("0 saniye beklemek için");
 }, 0); // 0 milisaniye bekler

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

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
başla
bitir
0 saniye beklemek için
2 saniye beklemek için
PS C:\Users\user\Desktop\WeatherApp>
```

Normalde yazdığımız bu kodun yukarıdan aşağıya çalışmasını bekleriz. Eğer senkron olarak çalışsaydı:

Başla

2 saniye beklemek için

0 saniye beklemek için

Bitir

Şeklinde çıktı vermesini beklerdik ama biz asenkron olarak ifadeleri düzenlediğimiz için çıktı beklenen gibi olmadı. setTimeout bir asenkron fonksiyondur bundan dolayı bu asenkron fonksiyon çalışabilmek için kodun geri kalan kısmını bekletmez evet oluşturur ve 2000 milisaniye sonra callback ile ekrana yazdırır. Bekleme işlemi kodun akmasına engel olamaz.

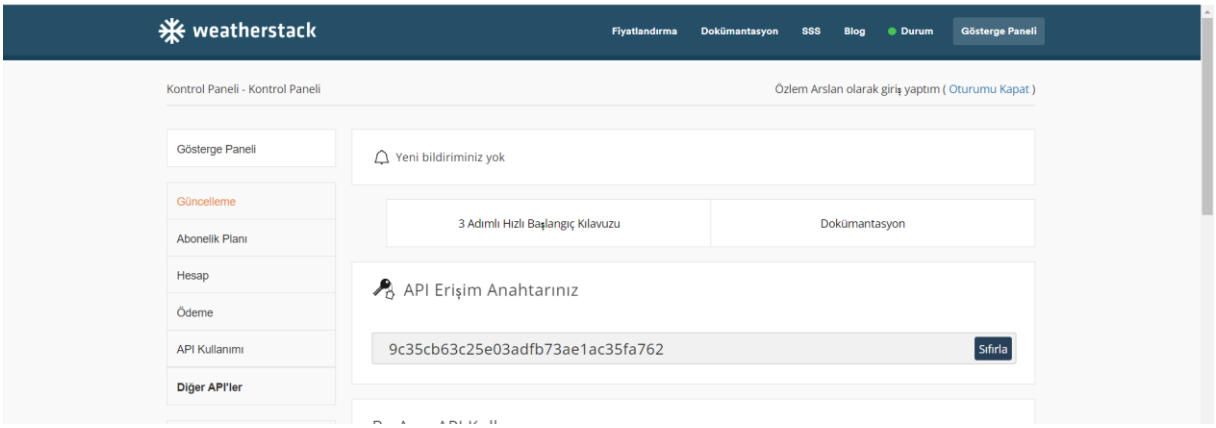
İlk olarak 0 saniye beklemek için yazar 2 saniye bekler daha sonra 2 saniye beklemek için yazısını basar. Nodejs de fonksiyonlar çalışırken stack mekanizması ile çalışır. Main fonksiyonu stacğin en altında yer alır daha sonra setTimeout() atılır. Önce bu fonksiyon çalışır tamamlanır daha sonra diğer fonksiyonlar çalışır. Main tamamlanmadan asenkron fonksiyonlara geçmez. Main fonksiyonu içindeki işlemler ne kadar uzun sürerse sürsün bu işlem bir dosyaya yazdırma işlemi olsa dahi tamamlanmadan asenkron fonksiyonlar çalıştırılamaz bu yüzden önce başla daha sonra bitir yazmıştır main kısmı tamamlanıp asenkronlara geçiş yapmıştır 0 saniye beklemek için ifadesi bitir ifadesinden daha önce yazılmış olmasına rağmen önce ekrana bitir basılmıştır.

Nodejs chormeda da kullanılan V8 motorunun üstüne implemant edilmiştir ama setTimeout V8 motorun bir parçası değildir V8 de yok Nodejs tarafından eklenmiştir.

Bu kısımda request isimli modülü kullanarak web sunucularına sorgu gönderip sunuculardan gelen yanıtı bekleyip cevabı alıp formatlayıp ekrana basmaya çalışacağız. Böylece uygulmamamız dış dünya ile iletişim kurabilcektir. Bu yüzden http isteklerinin nasıl yapıldığını öğrenelim. Kullanacağımız apiler weatherstack.com ve mapbox.com dır.

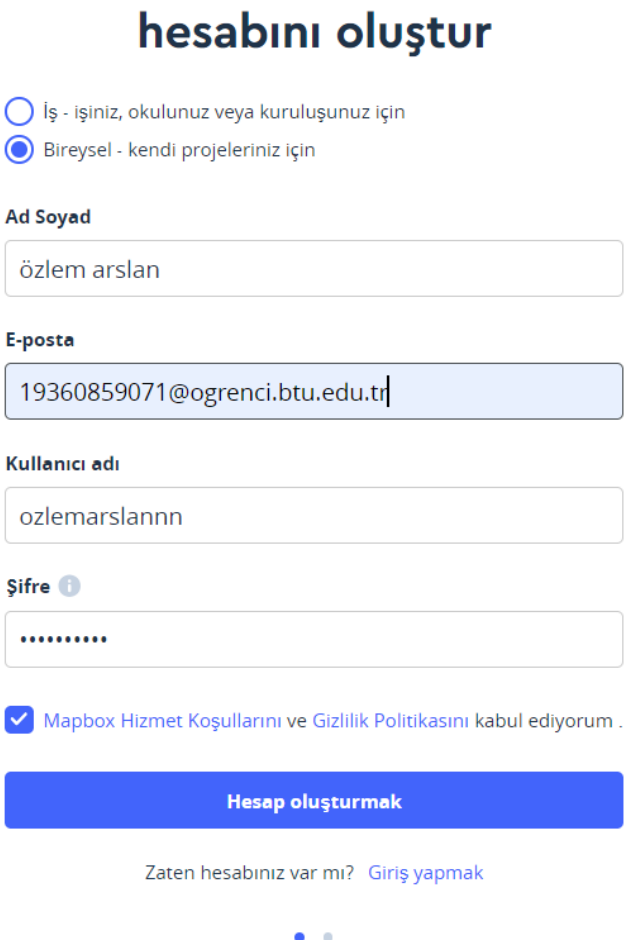
Şimdi weatherstack.com adresine girin ve sağ üst köşede bulunan ücretsiz kaydolun butonuna tıklayın. Free acount a tıklayın. Gerekli bilgileri girin bu site bize enlem boylam bilgisini verip hava durumunu alacağımız sitedir.

Gerekli bilgileri girdikten sonra karşımıza şu şekilde bir ekran çıkmaktadır:



Burada api erişim anahtar bilgimiz yer almaktadır. Bu aslında bir şifre denilebilir değerler rastgele oluşturulur. Bu bilgiyi kimlik doğrulaması yaparken kullanacağız.

Daha sonra mapbox.com a girin sign up a tıklayın ve bilgilerinizi girip hesap oluşturun bu site ise adres bilgisini girip enlem ve boylam bilgisini alacağımız sitedir.

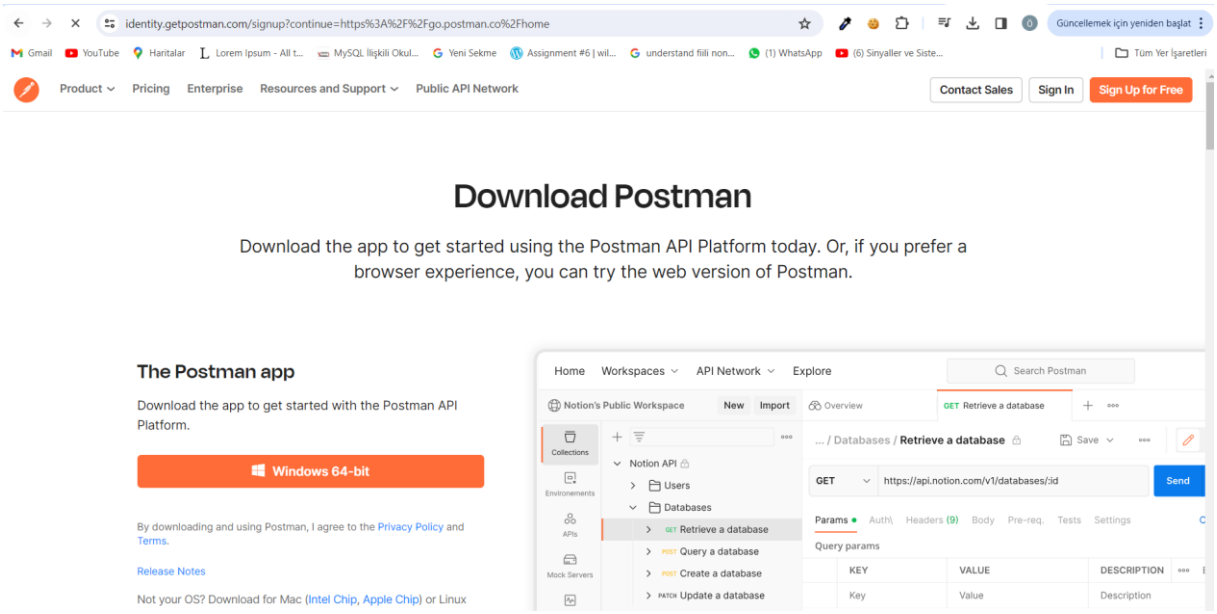


Normalde doğrudan adres bilgisini verip hava durumunu alabileceğimiz apiler de bulunmakta ama biz daha iyi öğrenmek için bunu iki aşamalı olarak yapalım.

Az önce yazdığımız kodları silebiliriz gerek kalmadı.

Postman: kendi bilgisayarımızda api oluşturduğumuzdan kendi bilgisayarımızdan apilere sorgu gönderip cevaplarını formatlı bir şekilde almamızı sağlayan bir uygulamadır.

Postman'ı indirmek için arma çubuğuna postman dowland yazın ve size uygun olan işletim sistemini seçtikten sonra karşınıza şu şekilde bir ekran çıkacaktır.



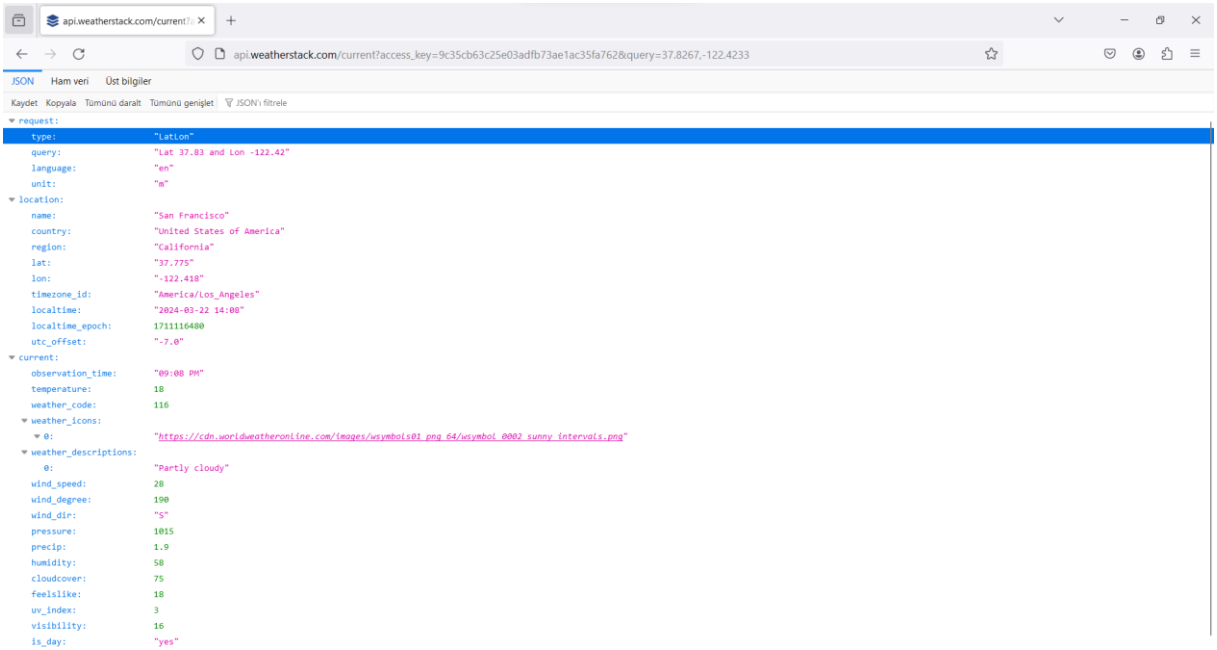
Windows 64-bit e tıklayın ve kurulumu tamamlayın.

Daha sonra firefox'un arama çubuğuna http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&query=37.8267,-122.4233 bu ifadeyi yazalım.

Chrome otomatik olarak http yi https e çeviriyor ama ücretsiz hesap https i desteklemediği için bunu Chrome da görüntüleyemedik.

Herhangi bir apiye sorgu yapılacağı zaman o apinin nasıl kullanılacağına dair dokümantasyondan bazı bilgilere bakılır örnek olarak temel adres bilgisi. Bu proje için temel adres: api.weatherstack.com daha sonra sorgu gönderirken hangi paramaetreleri göndermemiz lazım buna göre current? Dan sonra konsoldan parametre geçişi yapar gibi yapılır. İki tane parametre geçilecek bu parametreler ? den sonra gelir birden fazla parametreler & ile ayrılır. Her parametre arasında key value olacak şekilde aralarında = olarak yazılmalıdır. Anahtar= anahtarın kensidi& enlem boylam bilgisi.

Hava durumu verilerine erişmek için http://api.weatherstack.com dan sonra / konur ve şu anki hava durumu bilgilerini almak istediğimiz için current ifadesini yazarız daha sonra? bir sorgu geçekleştireceğimiz anlamına gelir ve ? den sonra wheatherstack.com dan aldığımız anahtar değerini ve konum bilgisini yani query i parametre olarak vermemiz gereklidir. Parametreler arasına & işareti koyarak ayırırız. Konum bilgisi olarak San Francisco'nun konum bilgisini kullanalım. Bu şekilde sorgumuzu hazırlamış olduk. Enter tuşuna basalım ve karşımıza çıkan sayfayı inceleyelim.



Talep işlemi gerçekleşti istediğimiz konum bilgisine göre hava durumu bilgileri verildi.

Ham veri kısmına tıklarsak karşımıza şu şekilde bir ekran çıkar.

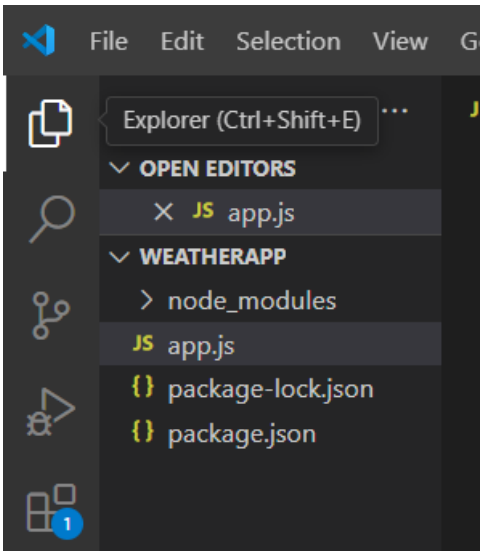
Şimdi kendi uygulamamıza geri dönemlim ve http isteğini uygulayalım. App.js içindeyken terminale npm install postman-request yazalım.

```
PS C:\Users\user\Desktop\WeatherApp> npm install postman-request

up to date, audited 56 packages in 930ms

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\user\Desktop\WeatherApp> npm install --save-dev @types/postman-request
npm ERR! code E404
npm ERR! 404 Not Found - GET https://registry.npmjs.org/@types%2fpostman-request - Not found
npm ERR! 404
npm ERR! 404  '@types/postman-request@*' is not in this registry.
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.

npm ERR! A complete log of this run can be found in: C:\Users\user\AppData\Local\npm-cache\_logs\2024-03-22T21_17_12_406Z-debug-0.log
PS C:\Users\user\Desktop\WeatherApp> node app.js
```

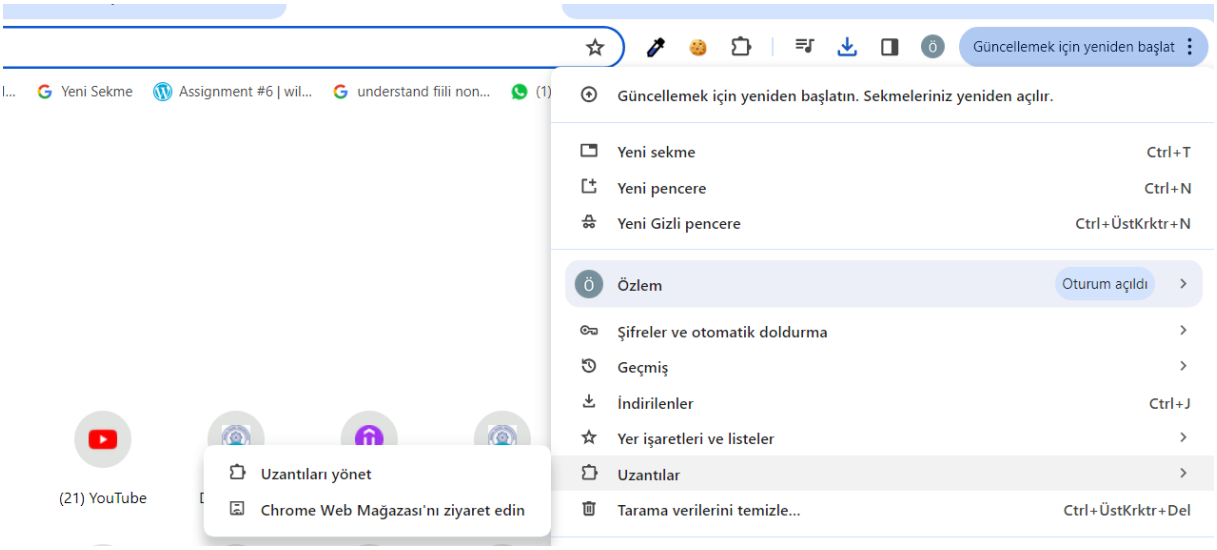
Daha sonra terminale npm init -y yazalım. -y ile soruları default olarak kendisi doldurur.

```
PS C:\Users\user\Desktop\WeatherApp> npm init -y
Wrote to C:\Users\user\Desktop\WeatherApp\package.json:

{
  "dependencies": {
    "postman-request": "^2.88.1-postman.33"
  },
  "name": "weatherapp",
  "version": "1.0.0",
  "main": "app.js",
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

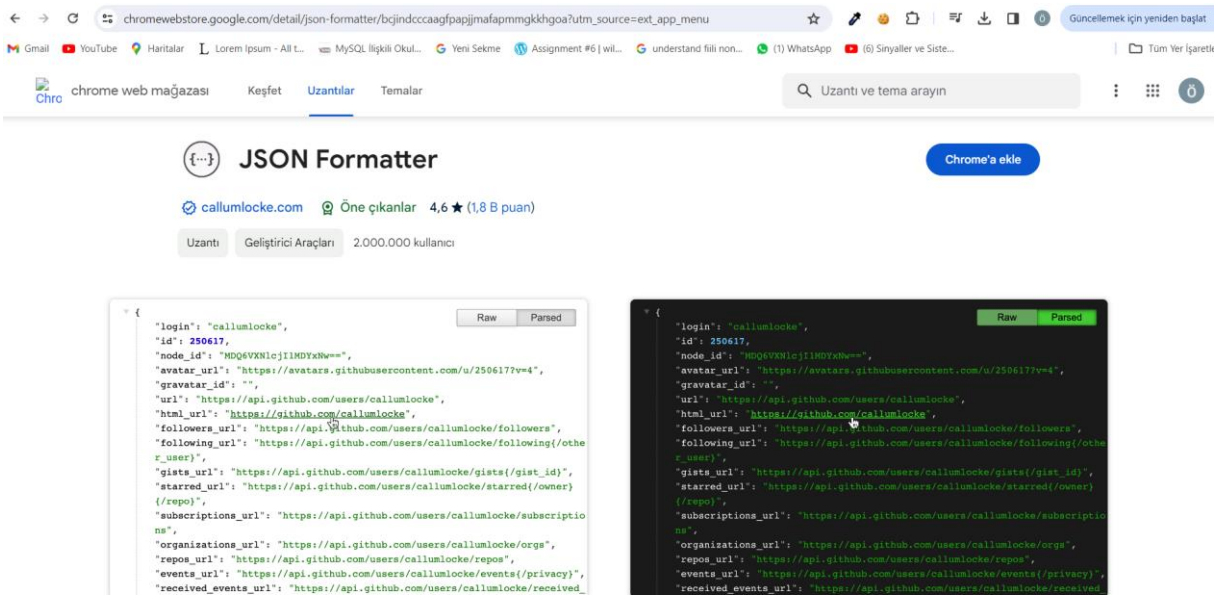
Kendisi otomatik olarak package.json oluşturur.



Daha sonra ek bilgi olarak şu adımları takip edin.



Daha sonra arama kısmına json formatter yazın.



Eğer chormeda sorguyu çalıştırabilseydik sorgu sonucu txt şeklinde gelecekti ve kolay anlaşıymıyor ama bu eklentiyi yüklersek formatlı bir şekilde sorguyu gösterecektir.Kullanacağımız hava durumu api: wheatherstack.com

App.js ye geri gelelim ve import işlemleri ile başlayalım.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&que
ry=37.8267,-122.4233";
//Requesti göndermek için kullanacağımız fonksiyon eğer hata varsa bu
// error ile belirlenecek ve sorgudan gelen cevap response ile tutulacak
request ({url:url},(error,response)=>{
   console.log(response)
})
```

url kısmına wheatherstack e gönderdiğimiz sorgu ifadesini yazalım daha sonra kodu terminalde çalıştıralım.

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
<ref *2> IncomingMessage {
  _events: {
    close: [ [Function], [Function (anonymous)] ],
    error: [Function (anonymous)],
    data: [Function (anonymous)],
    end: [ [Function: responseOnEnd], [Function (anonymous)] ],
    readable: undefined
  },
  _readableState: ReadableState {
    highWaterMark: 16384,
    buffer: [],
    bufferIndex: 0,
    length: 0,
    pipes: [],
    awaitDrainWriters: null,
    [Symbol(kState)]: 194778892
  },
  _maxListeners: undefined,
  socket: null,
  httpVersionMajor: 1,
  httpVersionMinor: 1,
  httpVersion: '1.1',
  complete: true,
  rawHeaders: [
    'Date',
    'Fri, 22 Mar 2024 21:17:33 GMT',
    'Content-Type',
    'application/json; Charset=UTF-8',
    'Transfer-Encoding',
    'chunked',
    'Connection',
    'keep-alive',
    'x-apilayer-transaction-id',
    '0e571c25-5c8d-4737-a771-dd611e7f935a',
    'access-control-allow-methods',
    'GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS',
```

Bu çıktının daha devamı var çok uzun bir çıktı vermektedir.

Response içinde body kısmında bazı önemli bilgiler bulunur Firefox içinde görülen hava durumu bilgileri body içinde yer alır. Sorgu yaparken http ile yaptığımız için gelen cevap da http ile gelir bu yüzden response içinde oldukça fazla bilgi içermektedir ama biz sadece response içindeki body kısmı ile ilgileneceğiz. Firefox içindeki response un bodysinde 3 kısım var request,location ve current bilgisi var ama bizim ihtiyacımız olan bilgi current içindedir. Gelen bilgi json formatında değil ama response.body kısmı json formatında bu yüzden json.parse işlemini yapmalıyız.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&que
ry=37.8267,-122.4233";
//Requesti göndermek için kullanacağımız fonksiyon
// error ile belirlenecek ve sorgudan gelen cevap response ile tutulacak
request ({url:url},(error,response)=>{
   //console.log(response) ihtiyacımız olan bilgi current olduğu için gelen bilgiyi azaltalım
```

```
  const data = JSON.parse(response.body) // response.body bilgisi json formatında olduğu için parse edilir
  console.log(data.current)
})
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
{
  observation_time: '11:06 PM',
  temperature: 17,
  weather_code: 296,
  weather_icons: [
    'https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0017_cloudy_with_light_rain.png
  ],
  weather_descriptions: [ 'Light Rain' ],
  wind_speed: 28,
  wind_degree: 210,
  wind_dir: 'SSW',
  pressure: 1014,
  precip: 0.1,
  humidity: 65,
  cloudcover: 100,
  feelslike: 17,
  uv_index: 3,
  visibility: 16,
  is_day: 'yes'
}
PS C:\Users\user\Desktop\WeatherApp>
```

Böylece daha az bir bilgi elde etmiş olduk.

Apiler ile ilgilenirken apiye nasıl bir sorgu göndermemiz lazım bunu dokümantasyondan öğrenebiliriz ve response kısmı önemlidir. Bu response kısmı hakkına bilgi edinmek için gelen bilgiyi yorumlayıp akışı anlaman gereklidir.

Bu kısımda gönderdiğimiz requestin özelliklerini ayarlayabiliriz.

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

Demek yerine şu şekilde kullanırsak:

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

Bu değişikliği yaparak gelen bilgi json formatında olacaktır. Doğrudan ekrana sadece derece bilgisini yazdırabiliriz.

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
17
PS C:\Users\user\Desktop\WeatherApp>
```

Gelen bilgi içindeki hangi bilgiyi kullanmak istediğimiz tamamen bize kalmıştır ister basıncı ister rüzgarı yazdır.

Hissedilen sıcaklığı da yazdıralım.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&que
ry=37.8267,-122.4233";
//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
    );

   })
```

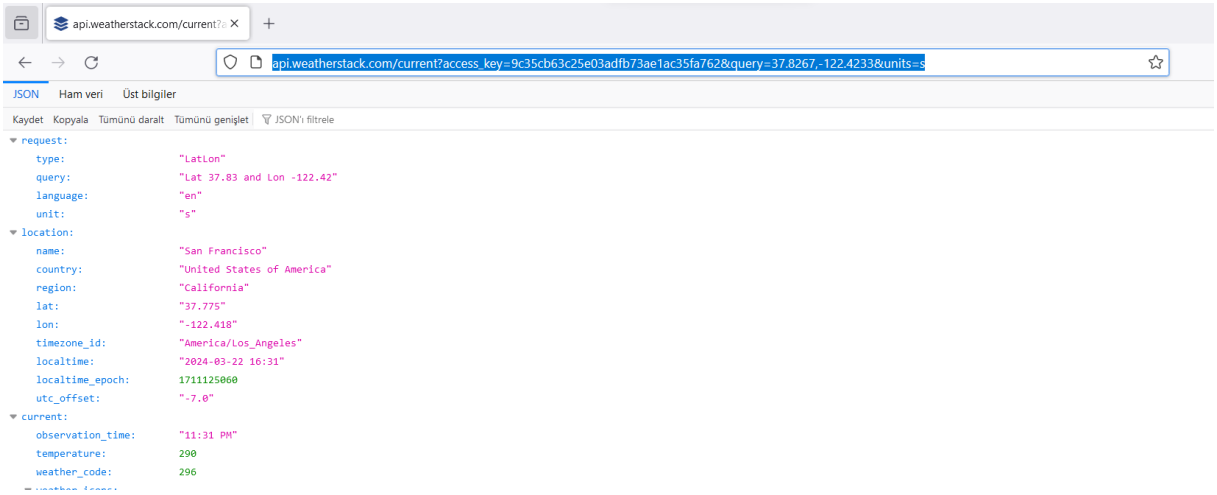
Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava sıcaklığı:17 Hissedilen:17
PS C:\Users\user\Desktop\WeatherApp>
```

Bu kısımda ise sorgudan aldığımız birimler default olarak santigrat şeklinde verilmiş eğer sorguyu gönderirken units=s dersek kelvin biçiminde units=f dersek fahrenayt şeklinde units=m dersek ise veya hiçbir şey yazmaksak default olarak santigrat birimden verileri döndürür.

Örnek olarak kelvin yapalım.

http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&query=37.8267,-122.4233&units=s



Kodumuzda sorgu yaparken kullandığımız url kısmını sorgunun cevabındaki verileri birimi kelvin olacak şekilde değiştirelim.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae1ac35fa762&que
ry=37.8267,-122.4233&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
     );

   })
```

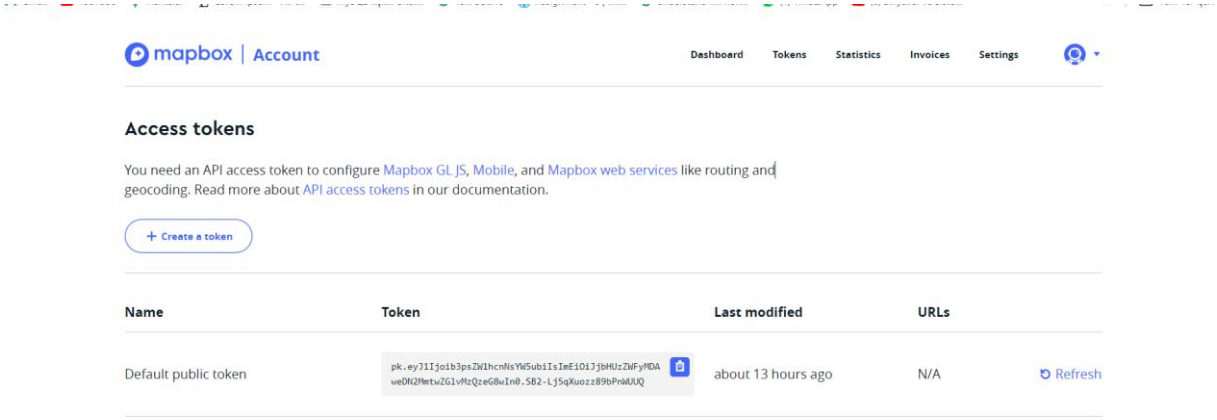
Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava sıcaklığı:290 Hissedilen:290
PS C:\Users\user\Desktop\WeatherApp>
```

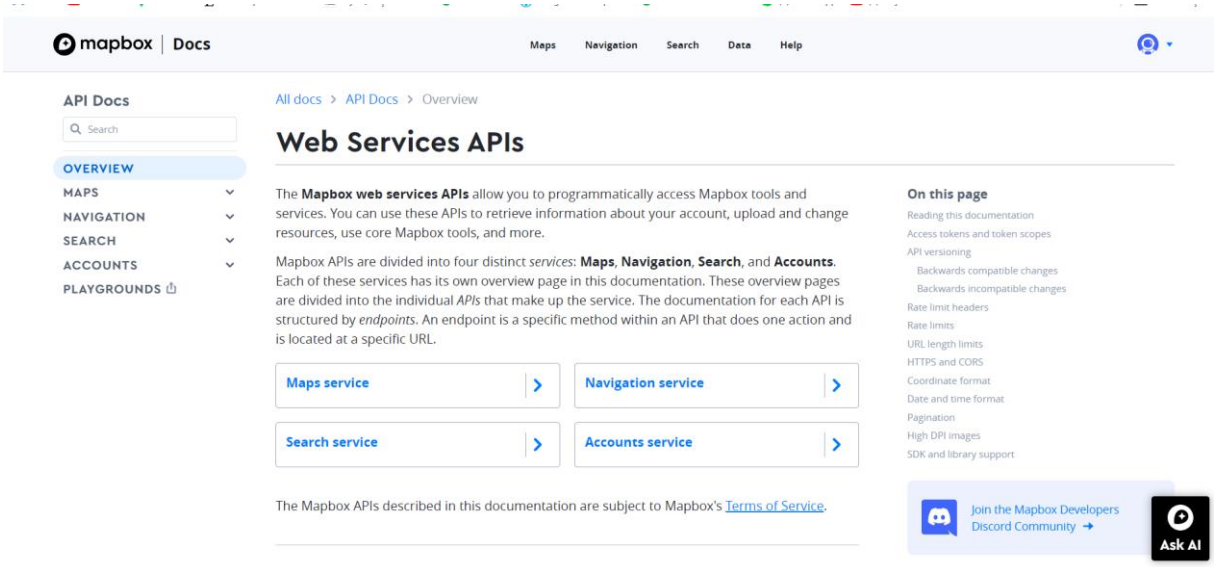
Sorgunun döndürdüğü cevaplar arasında Weather_descriptions yani hava bilgiside bulunur ama bu cevabı İngilizce olarak verir. Mesela clear yani açık bunu Türkçeye çevirmemiz gerekir bunu ek bir api kullanarak yapabiliriz.

Bu kısımda diğer servisimiz olan mapbox a bakalım zaten hesabımızı oluşturmuştuk. Mapbox apisi bize coğrafi kodlama imkânı sunacaktır. Bir adresin enlem ve boylam bilgisini elde etmemizi sağlayacaktır.

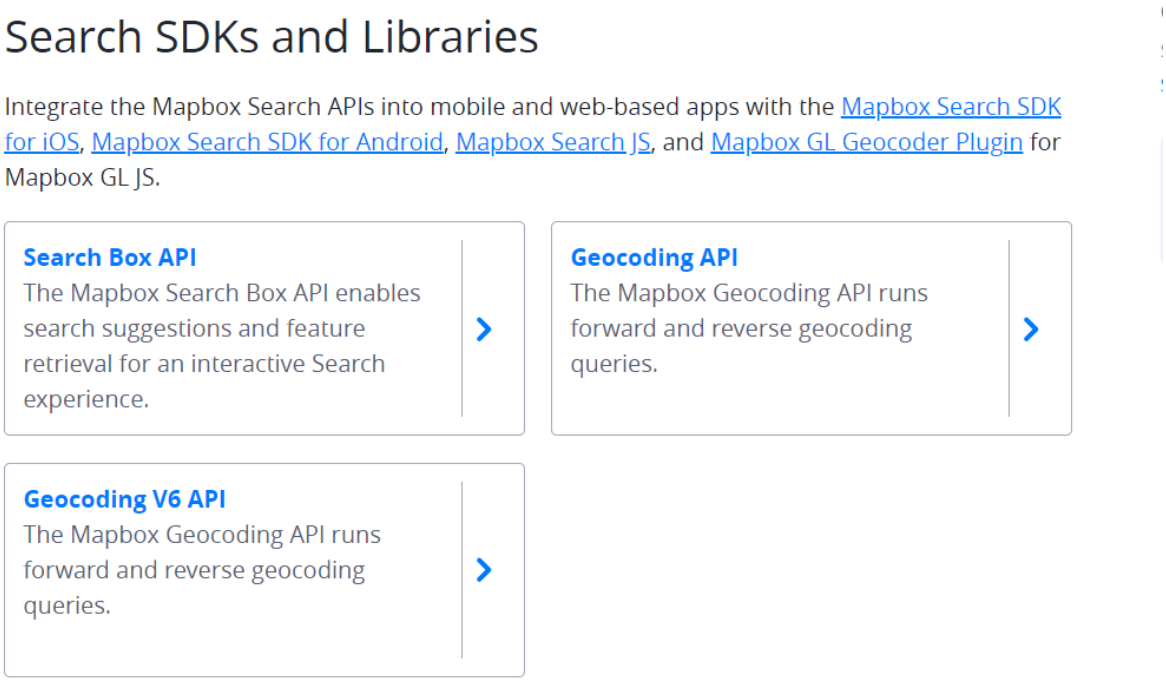
Mapbox.com' a girelim ve sağ üst tarafta bulunan token kısmına tıklayalım.



Daha sonra Mapbox web services linkine tıklayın.



Search service kısmına tıklayın. Bu kısım bizi coğrafi kodlama hizmetine götürecektir. Adresleri alıp koordinat çiftlerine dönüştürmemizi sağlayacaktır.



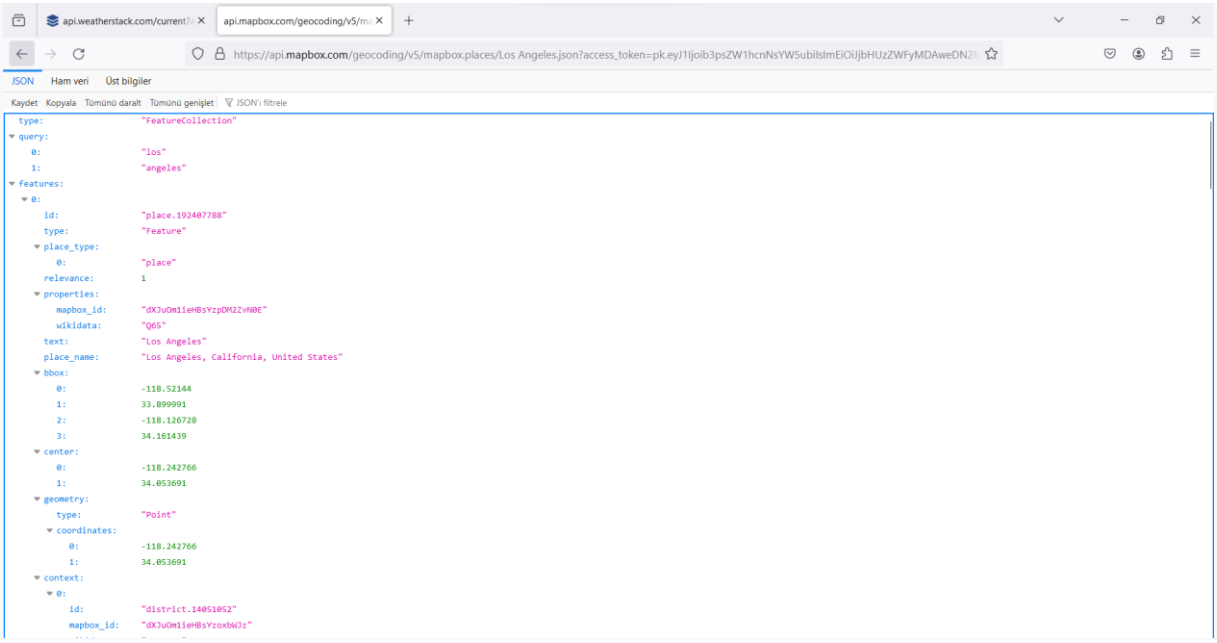
Buradan geocoding api kısmına tıklayalım.

Örnek bir api sorgusunun mapbox için nasıl gerçekleştiğini kontrol edelim.

Örnekte verilen url'i kopyalayalım ve arama çubuğunda aratalım.

```
https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?access_token=pk.e
yJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJjbHUzZWFyMDAweDN2MmtwZGlvMzQzeG8wIn0.SB2-
Lj5qXuozz89bPnWUUQ
```

şu şekilde bir çıktı alırız. Json formatında sorgu cevabını alırız



Feature kısmı 5 elemanlı bir dizidir. Konumla ilgili 5 bölge bulmuştur. Doğruya yakın yerler ilk sırada verilir. Los Angeles için etiketlenen 5 bölge getirir.

Bbox Los Angelesin bir kutu içine konulduğunda bu kutunun sınırlarını belirtir.

Geocoding işleminin tersi de bulunmaktadır reverse geocoding denir. Bu işlem ise enlem ve boylam bilgisi verilip o koordinatların yerin ismini vermesidir.

App.js ye geri gelelim ve kodlarımızı ekleyelim.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
1ac35fa762&query=37.8267,-122.4233&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
      );

    })
```

```
    const geocodeUrl =
"https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?a
ccess_token=pk.eyJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJjbHUzZWFyMDAweDN2Mmtw
ZGlvMzQzeG8wIn0.SB2-Lj5qXuozz89bPnWUUQ";

    request({ url: geocodeUrl, json: true }, (error, response) => { //
hata ve sonuc gelebilir bu yüzden error ve responsone kullandık
      const longitude = response.body.features[0].center[0]; // boylam
bilgisi features bir arry döndürelen 5 bölgeden ilkini aldık
      const latitude = response.body.features[0].center[1]; // enlem
bilgisi center da bir arry  enlem ve boylam bilgisini alırız
      console.log("Enlem : " + latitude + "Boylam : " + longitude);
    });
```

Hangi api daha hızlı çalışırsa stacktan ilk o alınır ve çıktısında da ilk o yazılır.

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Enlem : 34.053691Boylam : -118.242766
Hava sıcaklığı:284 Hissedilen:282
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava sıcaklığı:284 Hissedilen:282
Enlem : 34.053691Boylam : -118.242766
```

Şimdi hata handler etmeye odaklanalım. Uzaktaki bir sunucuya bağlanıp bir sorgu almaya çalışıyoruz bu yüzden birçok hata ile karşılaşabiliriz. Network ü disable edebiliriz, yanlış web sitesi adresi girilebilir, erişim anahtarını yanlış girebiliriz bu mantıksal bir hata olur ve wheatherstack içinde hata handler edilir.

Biz hataları error içinde yakalamaya çalışalım. Kodumuzda mapbox için yazdığımız kısmı yorum satırına alalım ve sadece console.log(error) ifadesini yazdıralım.

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
1ac35fa762&query=37.8267,-122.4233&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
    //   );
    console.log(error)

    })
    // const geocodeUrl =
"https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?a
ccess_token=pk.eyJ1Ijoib3psZW1hcnNsYW5ubiIsImEiOiJjbHUzZWFyMDAweDN2Mmtw
ZGlvMzQzeG8wIn0.SB2-Lj5qXuozz89bPnWUUQ";

    // request({ url: geocodeUrl, json: true }, (error, response) => {
// hata ve sonuc gelebilir bu yüzden error ve responsone kullandık
    //   const longitude = response.body.features[0].center[0]; //
boylam bilgisi features bir arry döndürelen 5 bölgeden ilkini aldık
    //   const latitude = response.body.features[0].center[1]; // enlem
bilgisi center da bir arry  enlem ve boylam bilgisini alırız
    //   console.log("Enlem : " + latitude + "Boylam : " + longitude);
    // });
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
null
PS C:\Users\user\Desktop\WeatherApp>
```

Şimdi hata durumu oluşturalım etherneti devre dışı bırakalım. Request geriye ya başarılıdır ya da başarısızdır. Response içi dolu olursa error içi boş olmalıdır bu durum tam terside olabilir.

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Error: getaddrinfo ENOTFOUND api.weatherstack.com
    at GetAddrInfoReqWrap.onlookupall [as oncomplete] (node:dns:118:26) {
  errno: -3008,
  code: 'ENOTFOUND',
  syscall: 'getaddrinfo',
  hostname: 'api.weatherstack.com'
}
PS C:\Users\user\Desktop\WeatherApp>
```

```
// önce import işlemini kullanalım.
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
1ac35fa762&query=37.8267,-122.4233&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) { // hata var mı
        console.log("Hava durumu servisine bağlantı kurulamadı.");
      } else {
        console.log(
          "Hava sıcaklığı:" +
            response.body.current.temperature +
            " Hissedilen:" +
            response.body.current.feelslike
        );
      }
    })
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava sıcaklığı:284 Hissedilen:282
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava durumu servisine bağlantı kurulamadı.
PS C:\Users\user\Desktop\WeatherApp>
```

İlk çıktıda ethernet bağlantısı kesilmemiş ve her şey yolundadır ama ikinci çıktıda ethernet bağlantısı kesilmiştir ve hata mesajını ekrana bastı.

Peki sorgu gönderdiğimiz linkte enlem ve boylam bilgisini belirtmesek ne olur?

http açısından sorgu gönderme başarılı ama cevapta yanlışlık olur bu tür hatayı kodun içinde handler etmek için error da düzenleme yapmamız mantıklı olmaz çünkü hata error içinde değil response içindedir. // önce import işlemini kullanalım.

```
const request =  require('postman-request');
// Chrome a ya da Firefox a yazdığımız url bilgisini girelim
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
1ac35fa762&query=37.8267,-122.4233&units=s";
//Requesti göndermek için kullanacağımız fonksiyon
// error ile belirlenecek ve sorgudan gelen cevap response ile
tutulacak
request({ url: url, json:true }, (error, response) => {
    if (error) { // direkt bir hata olduğu durumda mesela ethernet
bağlantısı kesilirse
        console.log("Hava durumu servisine bağlantı kurulamadı.");
      }
      else if(response.body.error){ // sorgu başarılı ama  gelen
cevapta bir hata varsa mesela enlem ve boylam bilgisi sorguda yoksa
        console.log("Girilen konum bilgisi bulunamadı.")
      }
      else { //hiçbir hata durumu oluşmamışsa
        console.log(
          "Hava sıcaklığı:" +
            response.body.current.temperature +
            "Hissedilen:" +
            response.body.current.feelslike
        );
      }

    })
```

Çıktısı:

```
PS C:\Users\user\Desktop\WeatherApp> node app.js
Hava sıcaklığı:284Hissedilen:282
PS C:\Users\user\Desktop\WeatherApp> node app.js
Girilen konum bilgisi bulunamadı.
```

İlk çıktıda herhangi bir hata yok ama ikinci çıktı için eğer const url kısmında enlem ve boylam bilgisini silersen yani

```
const url
="http://api.weatherstack.com/current?access_key=9c35cb63c25e03adfb73ae
1ac35fa762&units=s";
```

bu şekilde tanımlama yaparsan çıkan sonuçtur.