

Node.js Section 5

Merhabalar bugün node.js serimize devam ediyoruz ve 5.bölümdeyiz.

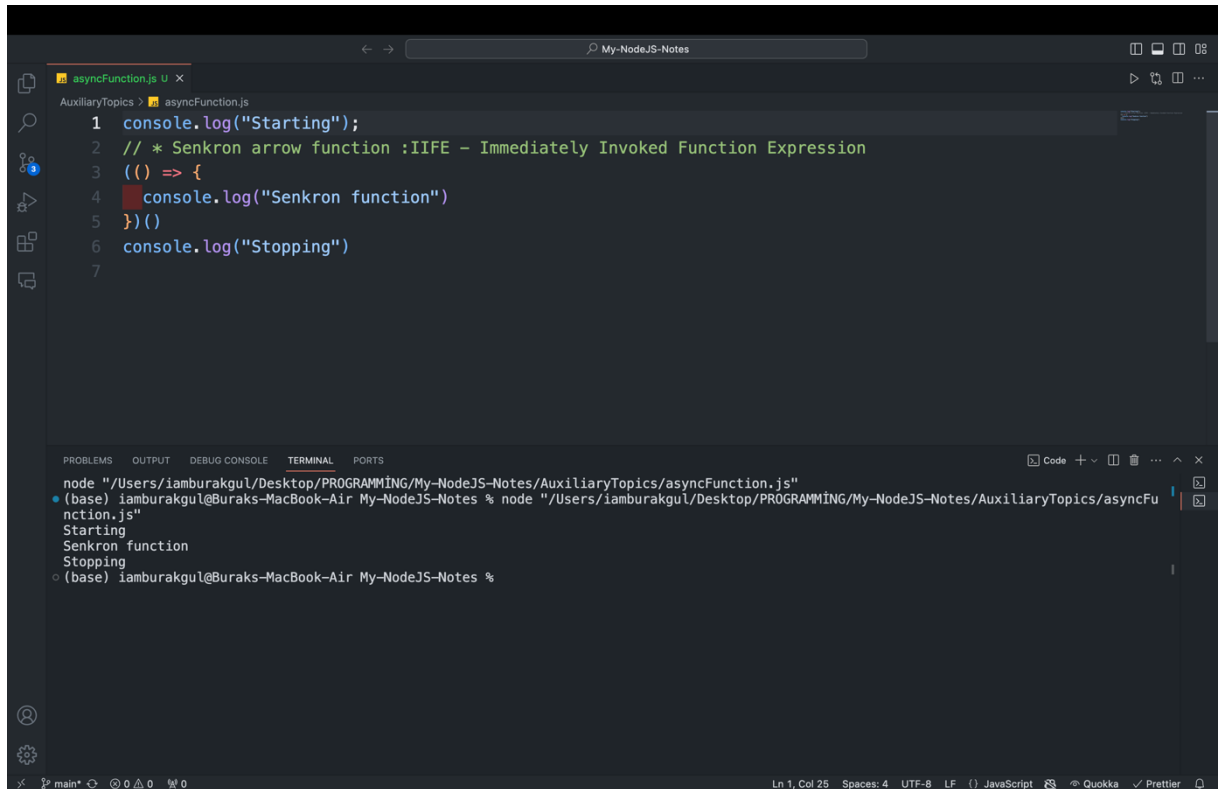
Öncelikle auxiliary topics klasöründe asyncFunction adında bir dosya oluşturup. Asenkron fonksiyonlara bakalım.

<https://developer.mozilla.org/en-US/docs/Web/API/setTimeout>

Buradan kullanacağımız setTimeout(code,delay) fonksiyonuna bakabilirsiniz.

Bu fonksiyon delay parametresi olarak verilen süre(ms) sonunda code kısmında yazdığımız kodu çalıştırır.

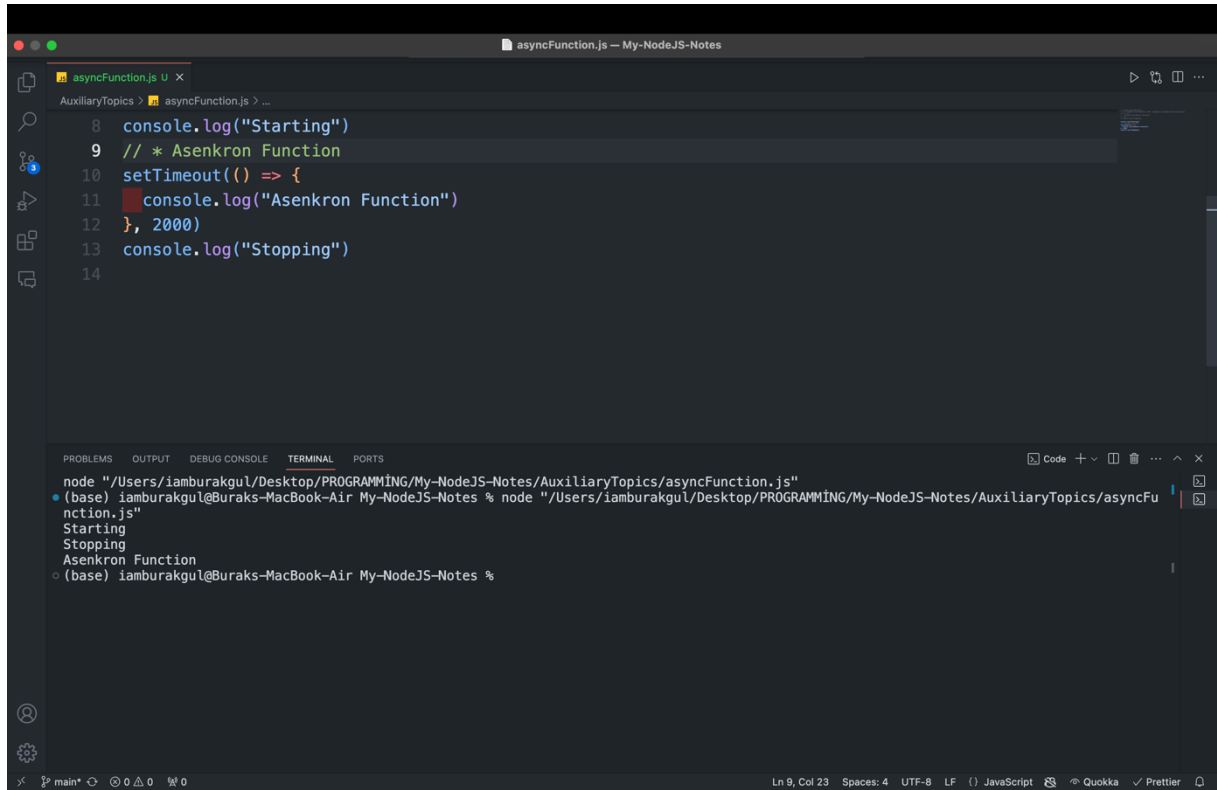
delay parametresini vermezsek 0 olarak varsayar ve hemen kodu çalıştırır.



```
1 console.log("Starting");
2 // * Senkron arrow function :IIFE - Immediately Invoked Function Expression
3 (() => {
4   console.log("Senkron function")
5 })()
6 console.log("Stopping")
7
```

```
node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
Starting
Senkron function
Stopping
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes %
```

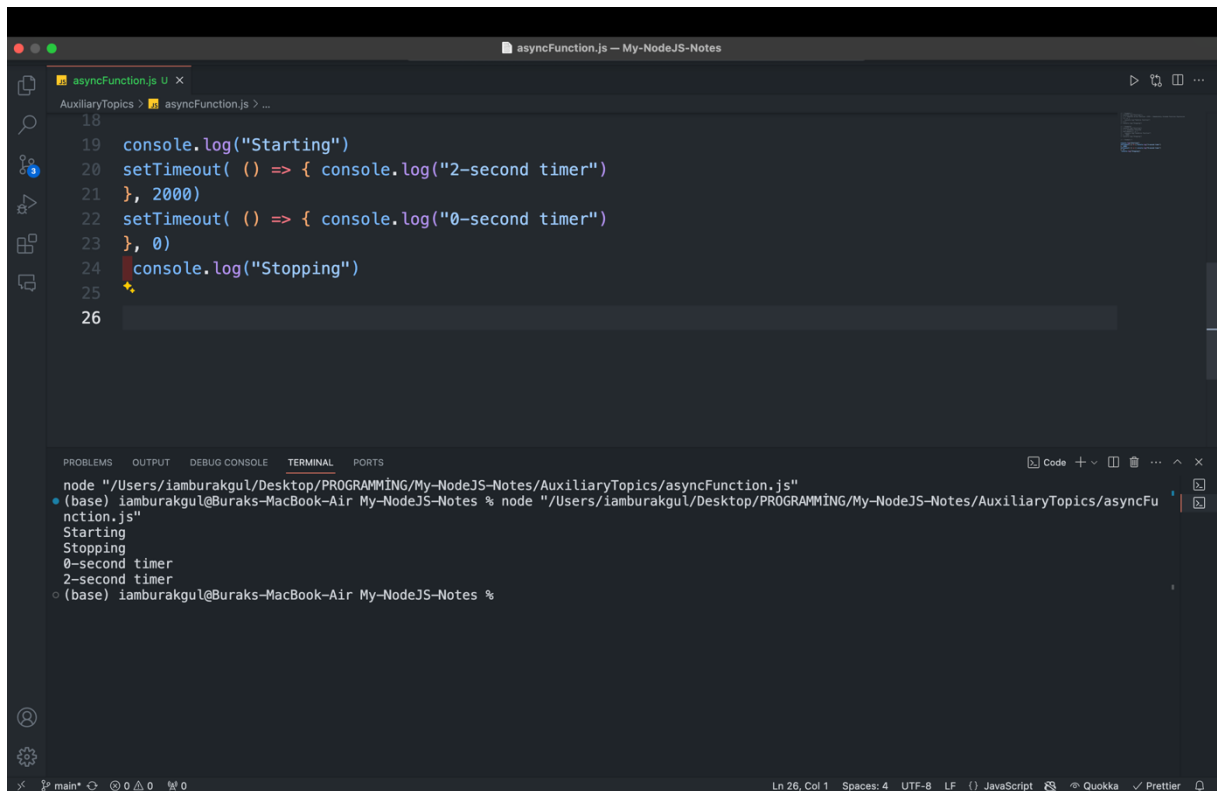
Buradaki örnek senkron programlama fonksiyon bitene kadar bekler ve bittiği zaman diğer fonksiyonlara geçer.



```
8 console.log("Starting")
9 // * Asenkron Function
10 setTimeout(() => {
11   console.log("Asenkron Function")
12 }, 2000)
13 console.log("Stopping")
14
```

```
node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
Starting
Stopping
Asenkron Function
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes %
```

Buradaki örnekte ise önce starting sonra stopping ve en sona Asenkron Function yazıyor sebebi ise setTimeout fonksiyonunun Asenkron olması ve 2 sn bekletmemizden ötürü.



```
18 console.log("Starting")
19 setTimeout( () => { console.log("2-second timer")
20 }, 2000)
21 setTimeout( () => { console.log("0-second timer")
22 }, 0)
23 console.log("Stopping")
24
25
26
```

```
node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/AuxiliaryTopics/asyncFunction.js"
Starting
Stopping
0-second timer
2-second timer
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes %
```

Burada ise o second timer Stopping mesajından sonra çalıştı neden ? Buna bakalım şimdi.

Biz ne her ne zaman bir fonksiyon çağırırsak/call bu fonksiyon call stack'e eklenir. Main fonksiyonu ise bu call stack'in en altında bulunur yani Main fonksiyonu call stack'e ilk eklenir.

Fonksiyon bittiğinde görevini tamamladığında call stack'den kaldırılır. Main fonksiyonu da kaldırıldığında ise program biter.

Şimdi buradaki örneğimizde neler olduğunu adım adım açıklayalım :

- 1- Main fonksiyonu call stack'e eklenir.
- 2- Console.log("Starting") call stack'e eklenir ve en üstte bu fonksiyon vardır.
- 3- 2.adımdaki fonksiyon bittiğinde call stack'den kaldırılır.(Şu anda Main fonksiyonu var hala stackde)

Ek Bilgi :

setTimeout fonksiyonu V8 script motorunun bir partı değildir. Bu fonksiyon node.js tarafından tanımlanmıştır ve bu fonksiyonun API'si c++ tarafından implemente edilmiştir.

Bu fonksiyon event-callback çiftidir.Yani bir olay olduğunda çalışacak fonksiyonla beraber yazılır.O olay tamamladığında callback fonksiyonu çağrılır.

Call stack single threadlidir.

- 4- setTimeout 2 second call stack'e eklenir ve node API'sine gönderilir
- 5- setTimeout 0 second call stack'e eklenir ve node API'sine gönderilir
- 6- Biz olayların tamamlanmasını bekleriz. setTimeout 0 hatta ilk tamamlanmıştır ve callback fonksiyonu callback queue yapısına eklenir.
- 7- Callback queue yapısı call stack yapısı boş mu değil mi diye kontrol eder.
 - a. Eğer boş ise callback fonksiyonunu direkt çalıştırır.
 - b. Eğer boş değilse call stack'deki fonksiyonları bekler
- 8- Şuan call stackde Main fonksiyonu vardır.
- 9- Console.log("Finishing") fonksiyonu call stack'e eklenir.(üstten bir bakış atarsak log finishing fonksiyonu ve altında da main fonksiyonu olur).Mesaj yazılır ve bu fonksiyon call stack'den kaldırılır.
- 10-Daha sonra main fonksiyonu da call stack'den kaldırılır.
- 11-Callback queue call stack i kontrol eder ve boş olduğunu görür.callback fonksiyonunu callback queue den kaldırır ve setTimeout 0 ı call stacke ekler.Call stackde üstte olduğu için bu fonksiyon ,çalıştırılır ve call stackden kaldırılır bu fonksiyon.

12-2 saniye olayı tamamlandığında , bu eventin/olayın callback fonksiyonu event loop'a eklenir.callback i callback queue den call stack' e alır.Console.log() 2 seconds call stack'e eklenir ve geri çekilir.

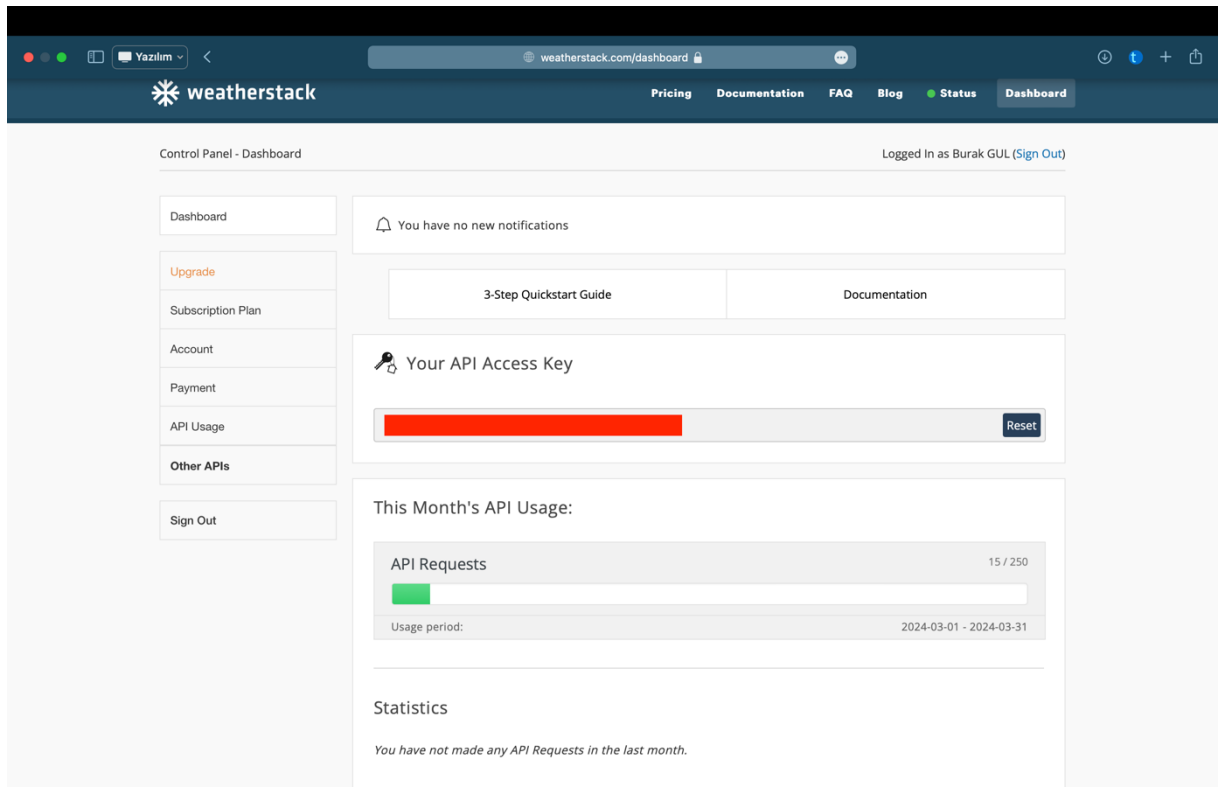
Şimdi gerçek hayat projelerine geçelim.

Hava Durumu uygulaması yapacağız.Section 5 adında bir klasör oluşturalım.

Bu siteye gidip üye olalım ve buradan hava durumları için sorgu atacağız.Bize bir apiKey verecek bu önemli bizim için.

Npm init ile projemizi kuralım ve <https://www.npmjs.com/package/postman-request> bu npm i de ,npm i postman-request yazarak indirelim.

Şimdi ise <https://weatherstack.com> bu siteye gidelim ve üyelik oluşturalım



Burada api Access key yazan bizim api anahtarımız.

Yazılım


weatherstack.com/documentation

🔍


🌐

+

🔖


 weatherstack

PricingDocumentationFAQBlogStatusDashboard

 3-Step Quickstart Guide

Feel like jumping right into making API calls? Get a free weatherstack account and simply use our 3-Step Quickstart Guide.

GO TO QUICKSTART GUIDE

 Code Examples

Our API supports all major programming languages. Click below to browse through examples in PHP, Python, Node, jQuery and Ruby.

CODE EXAMPLES

Getting Started

API Authentication

HTTPS Encryption

API Error Codes

API Features

Current Weather

Historical Weather

Historical Time-Series

Weather Forecast

Location Lookup

Options

Query Parameter

Units Parameter

Language Parameter

JSONP Callbacks

API Documentation

Welcome to the weatherstack API documentation. Using the instructions and interactive code examples below you will be able to start making API requests in a matter of minutes. If you have an account already and prefer to skip our detailed documentation, you can also jump to our [3-Step Quickstart Guide](#) right away.

The weatherstack API was built to deliver accurate weather data for any application and use case, from real-time and historical weather information all the way to 14-day weather forecasts, supporting all major programming languages. Our straightforward API design will make it easy to use the API — continue reading below to get started.

Getting Started

API Authentication

The first step to using the API is to authenticate with your weatherstack account's unique API access key, which can be found in your account dashboard after registration. To authenticate with the API, simply use the base URL below and pass your API access key to the API's `access_key` parameter.

Example API Request:

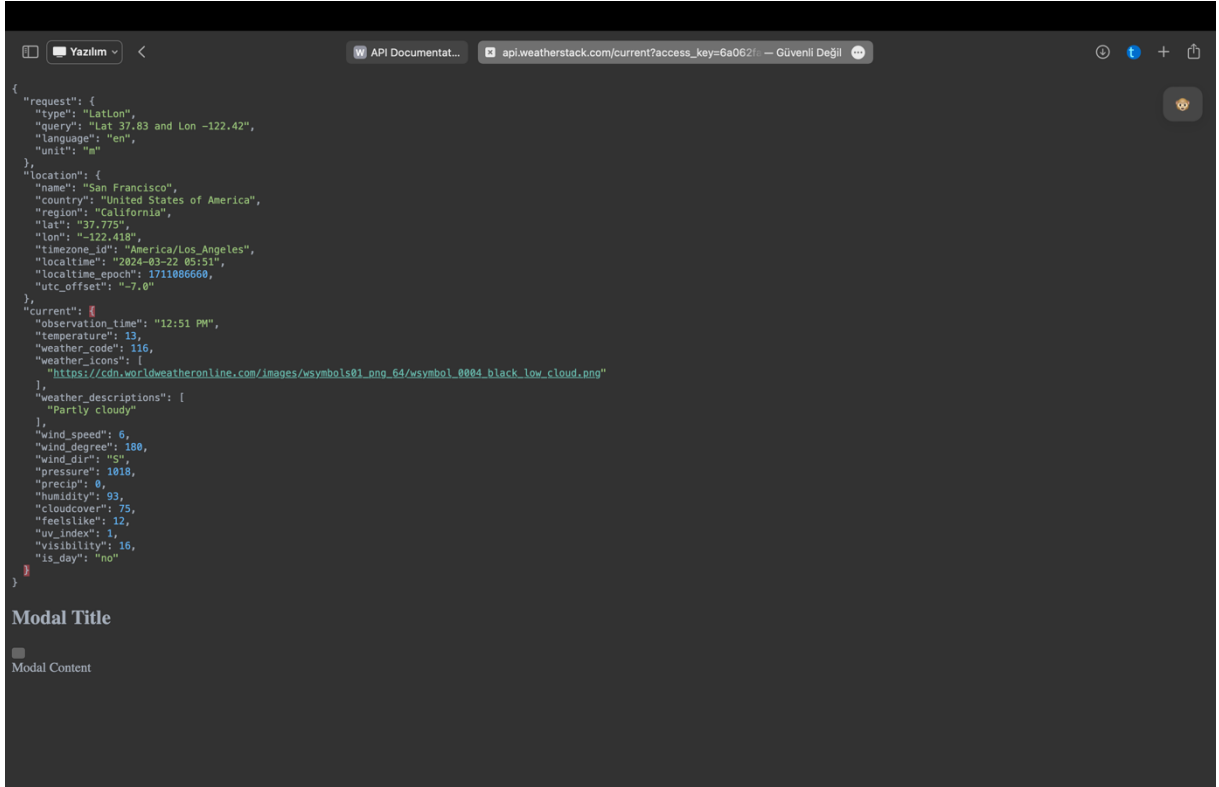
```
http://api.weatherstack.com/current
? access_key = 6a062fa08038e848eb2f9ad4b2ce7c10
& query = New York
```

Run API Request

Documentation kısmından ,optionsda yazan query parameter kısmını okursanız anlayacağınız üzere sorgumuzda hangi parametreleri vererek neler elde edeceğimizi gösteriyor.

http://api.weatherstack.com/current?access_key=burayaAPIKeyiniYaz&query=37.8267,-122.4233

bunu alıp tarayıcımızda url kısmına yapıştırırsak aslında bir weatherstack sitesine ir sorgu atıyoruz ve bize cevap dönüyor.

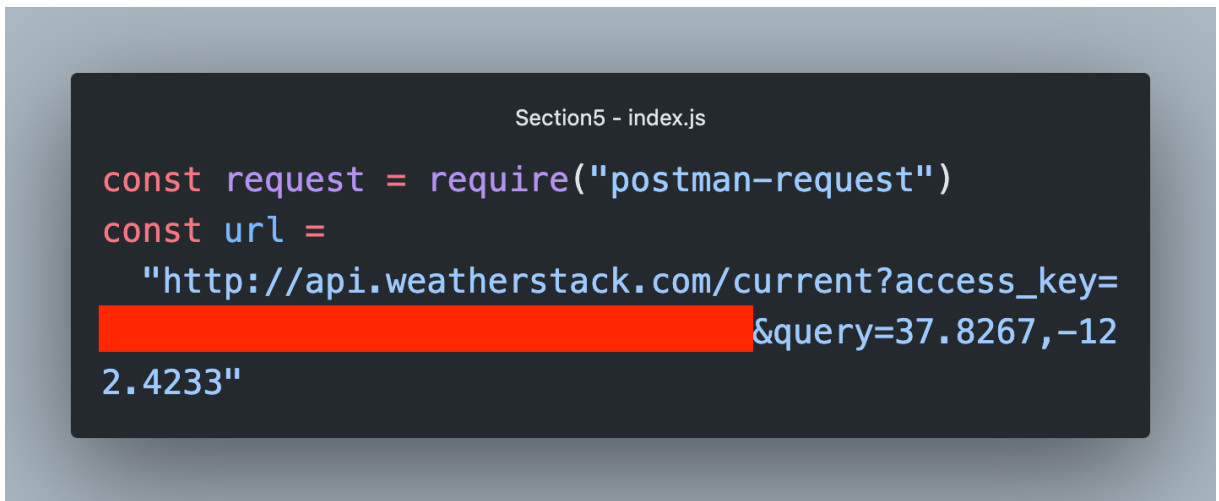


Böyle bir JSON yapısı dönüyor bize.

Taraycımızda böyle gözükmesi için JSON Viewer eklentilerinden birini indirebiliriz tarayıcımıza.

Şimdi bu elle yaptıklarımızı kod ile yapalım.

Request modelini index.js dosyamıza ekleyelim ve url adında bir değişken oluşturup tarayıcımıza girdiğimiz url i string şeklinde atayalım bu değişkene.



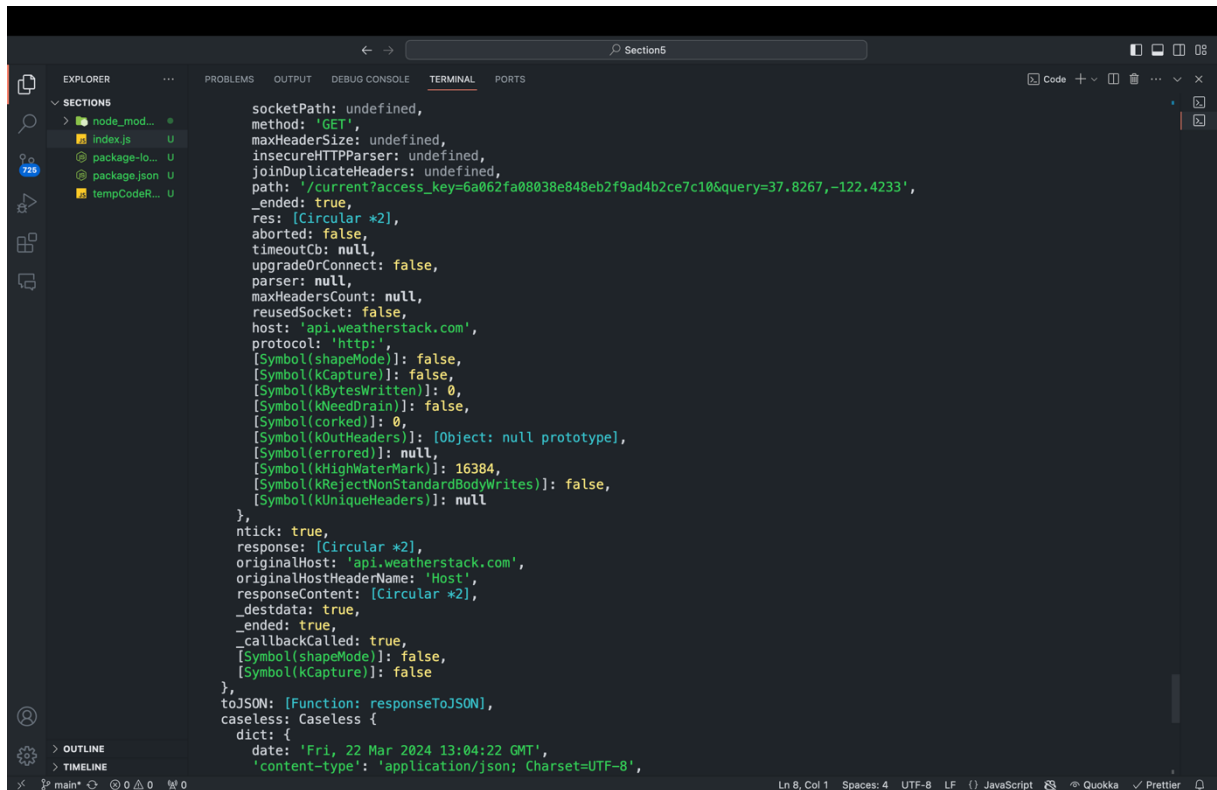
Kırmızı kısma api anahtarımızı atayalım.Bu kısımları ilerideki kodlarımızı buraya eklerken göstermeyeceğim tek tek api key'i gizlemek için.

```
Section5 - index.js

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

Bu fonksiyon nedir ne iş yapar ? Belirttiğimiz url e istek atar ve response dönerse response ile error dönerse error ile neler yapmak istediğimiz bir fonksiyonu parametre olarak almaktadır aslında.

Bu kodu da ekleyip çalıştırdığımızda bize böyle şeyler gözüküyor aslında burada response u yazdırdık. Bize response un body kısmı lazım. Ek olarak verdiğimiz JSON türünde olduğu için bunu parse etmemiz gerek.



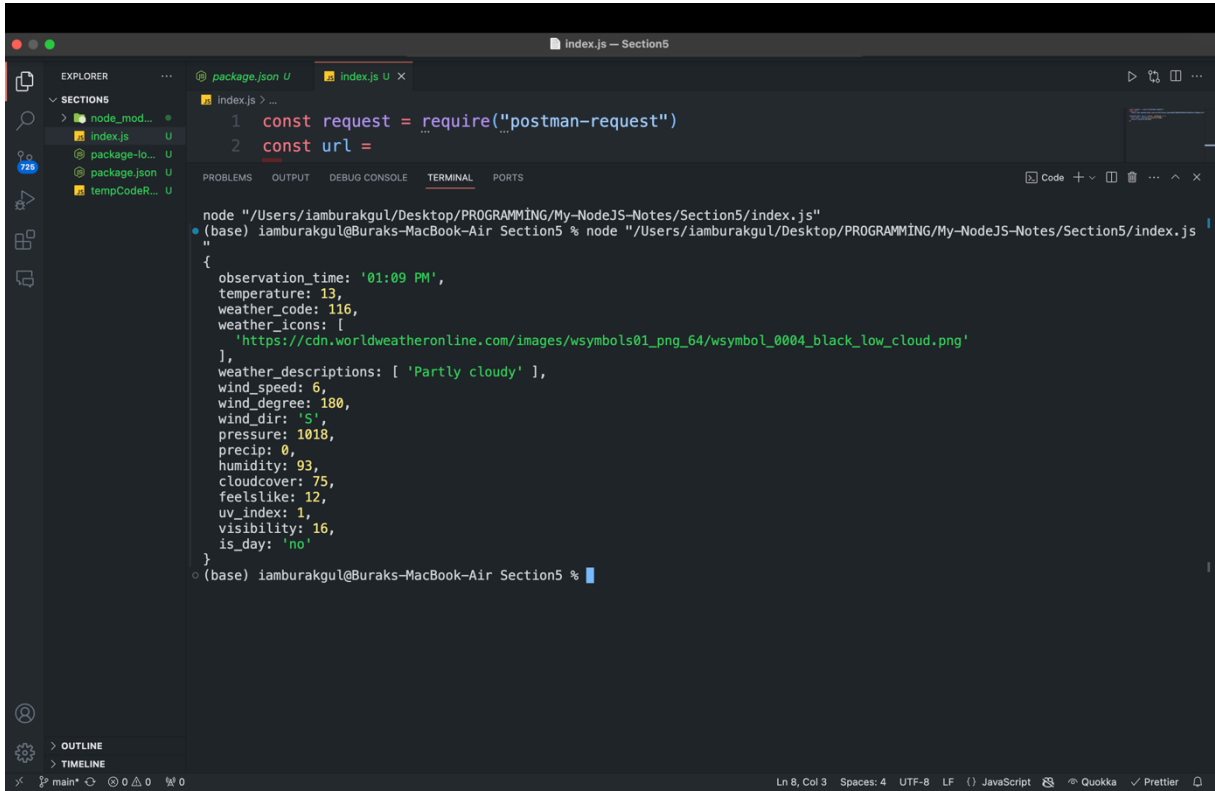
```
socketPath: undefined,
method: 'GET',
maxHeaderSize: undefined,
insecureHTTPParser: undefined,
joinDuplicateHeaders: undefined,
path: '/current?access_key=6a062fa08038e848eb2f9ad4b2ce7c10&query=37.8267,-122.4233',
ended: true,
res: [Circular *2],
aborted: false,
timeoutCb: null,
upgradeOrConnect: false,
parser: null,
maxHeadersCount: null,
reusedSocket: false,
host: 'api.weatherstack.com',
protocol: 'http:',
[Symbol(shapeMode)]: false,
[Symbol(kCapture)]: false,
[Symbol(kBytesWritten)]: 0,
[Symbol(kNeedDrain)]: false,
[Symbol(corked)]: 0,
[Symbol(kOutHeaders)]: [Object: null prototype],
[Symbol(error)]: null,
[Symbol(kHighWaterMark)]: 16384,
[Symbol(kRejectNonStandardBodyWrites)]: false,
[Symbol(kUniqueHeaders)]: null
},
ntick: true,
response: [Circular *2],
originalHost: 'api.weatherstack.com',
originalHostHeaderName: 'Host',
responseContent: [Circular *2],
_destdata: true,
ended: true,
_callbackCalled: true,
[Symbol(shapeMode)]: false,
[Symbol(kCapture)]: false
},
toJSON: [Function: responseToJSON],
caseless: Caseless {
  dict: {
    date: 'Fri, 22 Mar 2024 13:04:22 GMT',
    'content-type': 'application/json; Charset=UTF-8',
```

Şimdi değişikliklerimizi yapalım.

```
Section5 - index.js

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

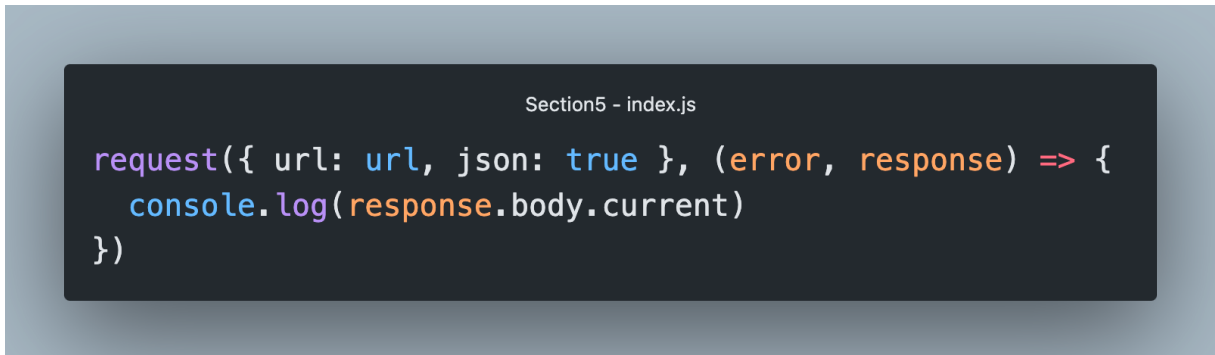
Tekrar çalıştıralım



```
index.js U X
1 const request = require("postman-request")
2 const url =

node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5/index.js"
(base) iamburakgul@Buraks-MacBook-Air Section5 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5/index.js"
{
  observation_time: '01:09 PM',
  temperature: 13,
  weather_code: 116,
  weather_icons: [
    'https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymb0l_0004_black_low_cloud.png'
  ],
  weather_descriptions: [ 'Partly cloudy' ],
  wind_speed: 6,
  wind_degree: 180,
  wind_dir: 'S',
  pressure: 1018,
  precip: 0,
  humidity: 93,
  cloudcover: 75,
  feelslike: 12,
  uv_index: 1,
  visibility: 16,
  is_day: 'no'
}
(base) iamburakgul@Buraks-MacBook-Air Section5 %
```

Gördüğümüz gibi veriler geliyor ama istek atarken bana verileri json türünde göndermesini söyleyip aynı sonucu alabilirim.



```
Section5 - index.js

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

Gelen sonuçtan istediklerimizi alabilir ve istediklerimizi yazdırabiliriz.


```
index.js — Section5
4
5 request({ url: url, json: true }, (error, response) => {
6   console.log(
7     "Hava Sıcaklığı : " +
8     response.body.current.temperature +
9     " Hissedilen : " +
10    response.body.current.feelslike
11  )
12 })
13
```

```
node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5/index.js"
(base) iamburakgul@Buraks-MacBook-Air Section5 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5/index.js"
Hava Sıcaklığı : 12 Hissedilen : 11
(base) iamburakgul@Buraks-MacBook-Air Section5 %
```

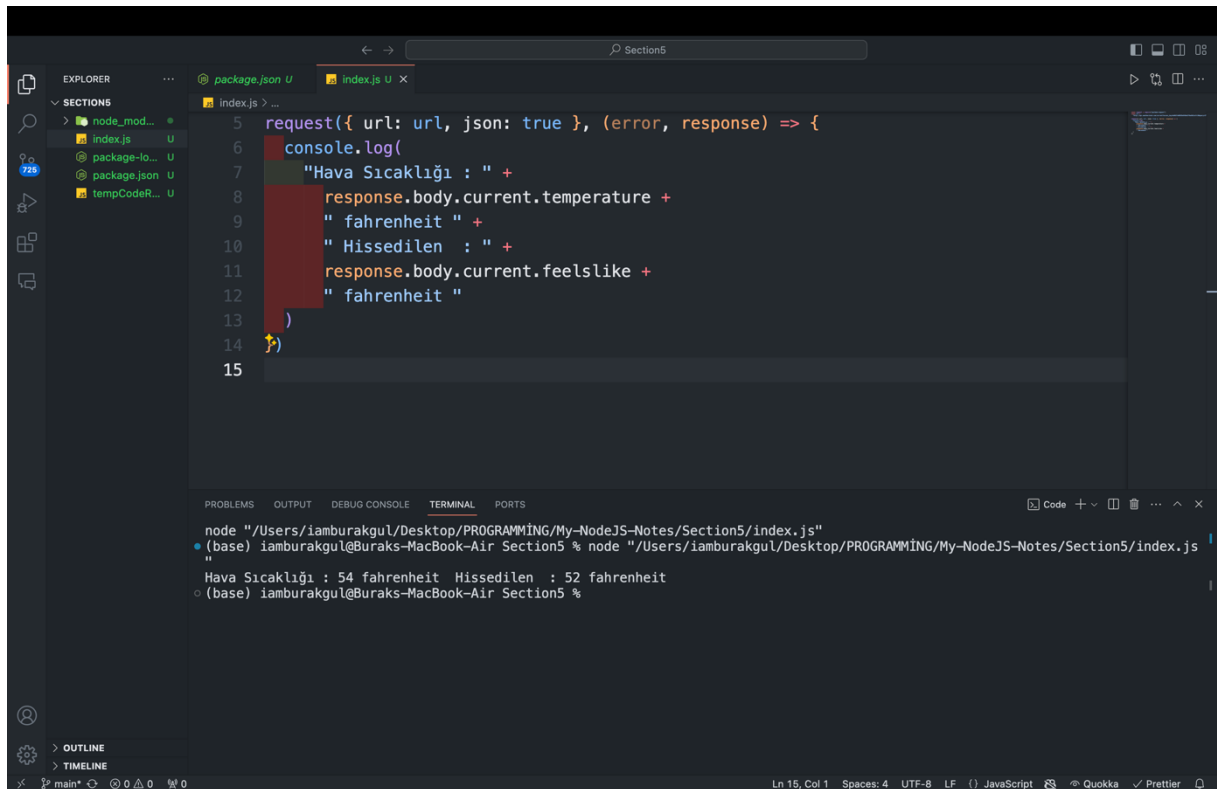
Şimdi biz sıcaklıkları yazdırdık ama birimi neydi ? Bunu da weatherstack dökümanından okuyup anlayacağız ve hatta fahrenheit olarak değiştirelim bunu.

<https://weatherstack.com/documentation> buradan options'ın altındaki Units Parameter kısmına tıklayarak anlayabilirsiniz.

url değişkenimizi değiştirelim ve metriklerden fahrenheit için olanı ekleyelim.

```
Section5 - index.js
const request = require("postman-request")
const url =
  "http://api.weatherstack.com/current?access_key=
  [REDACTED]=37.8267,-12
  2.4233&units=f"
```

Ve çalıştıralım.



The screenshot shows a VS Code editor with a file named `index.js` open. The code in the file is as follows:

```
5 request({ url: url, json: true }, (error, response) => {
6   console.log(
7     "Hava Sıcaklığı : " +
8     response.body.current.temperature +
9     " fahrenheit " +
10    " Hissedilen : " +
11    response.body.current.feelslike +
12    " fahrenheit "
13  )
14 })
15
```

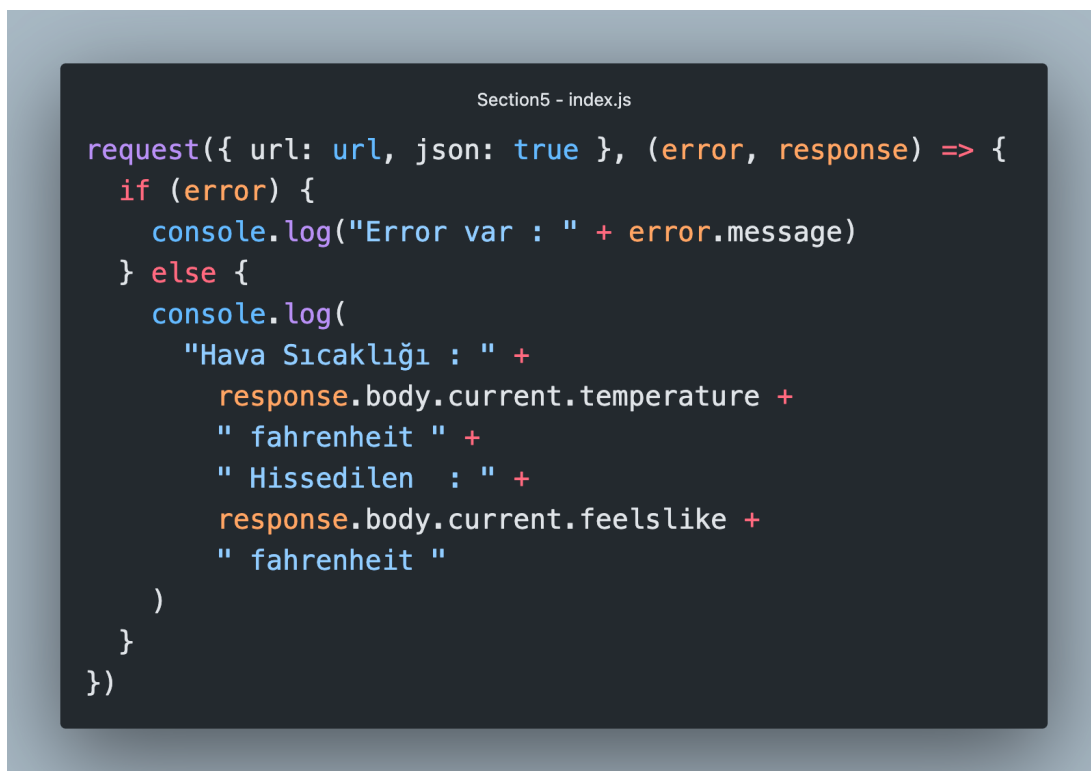
The terminal at the bottom shows the command `node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5/index.js"` being executed, and the output is:

```
Hava Sıcaklığı : 54 fahrenheit Hissedilen : 52 fahrenheit
(base) iamburakgul@Buraks-MacBook-Air Section5 %
```

Gördüğümüz gibi sayıları ve yazıları değiştirdik metric ayarları ile oynayarak.

Şimdi direkt response üzerinden gittik ama belki response değil de error dönecek bilemeyiz. Error varsa erroru yazdırırız yoksa response dönmüş demektir.

Bu yüzden kodumuzu şu şekilde düzeltebiliriz.



```
Section5 - index.js

request({ url: url, json: true }, (error, response) => {
  if (error) {
    console.log("Error var : " + error.message)
  } else {
    console.log(
      "Hava Sıcaklığı : " +
      response.body.current.temperature +
      " fahrenheit " +
      " Hissedilen : " +
      response.body.current.feelslike +
      " fahrenheit "
    )
  }
})
```

Şimdi ise başka bir siteye üye olacağız ve oradan veri çekeceğiz.Şehir adresini girince bize lokasyon veren kısmını kullanalım ve aldığımız lokasyon ile de diğer apiden hava durumu verilerini çekelim.

<https://www.mapbox.com> sitemiz budur.

Buradan da bir API Key verecek bize.

<https://docs.mapbox.com/api/search/geocoding/> buradan da nasıl sorgu atacağımıza bakabilirsiniz.

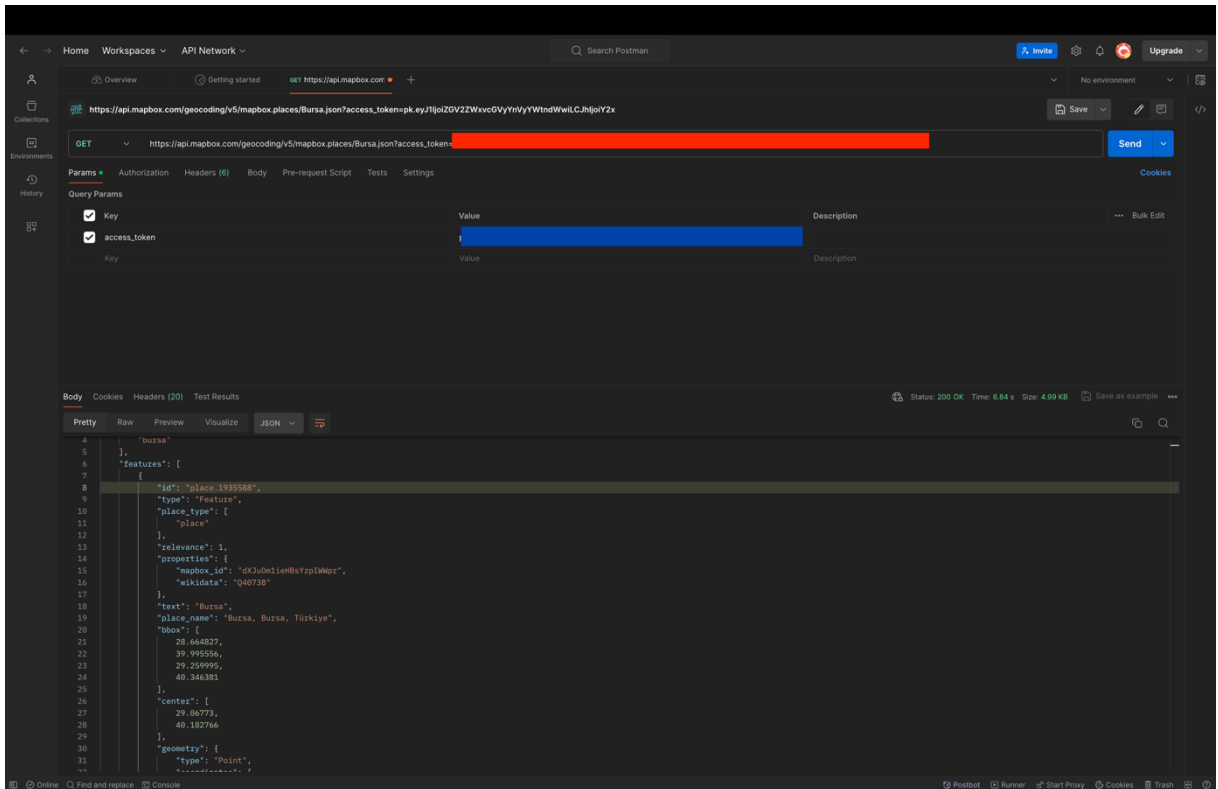
Şimdi bir daha tarayıcıdan istek atmak yerine postman uygulamasını kullanalım istek atmak için.

Tarayıcıdan request atmak istersek de

https://api.mapbox.com/geocoding/v5/{endpoint}/{search_text}.json bu yapıyı kullanacağız.

https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?access_token=pk.eyJ1IjoizGV2ZWxvcGVyYnVhcnVtndWwILCJhIjoieY2x0emhia29pMDAxZzJrbzlyNmldjNHNiJ9.9RWqetLhXZXeeFD7WVLORQ

Bu tarzda olacaktır url kısmımız.



Mavi kısma API Keyimizi girince kırmızı kısma kendisi otomatik olarak eklemektedir. Ve sorgu sonucumuz aşağıda yazmaktadır gördüğümüz gibi.

Bize body kısmından feature dizisinin ilk elemanından center dizisi lazım. Bunları elde ettiğimiz zaman tamamdır.

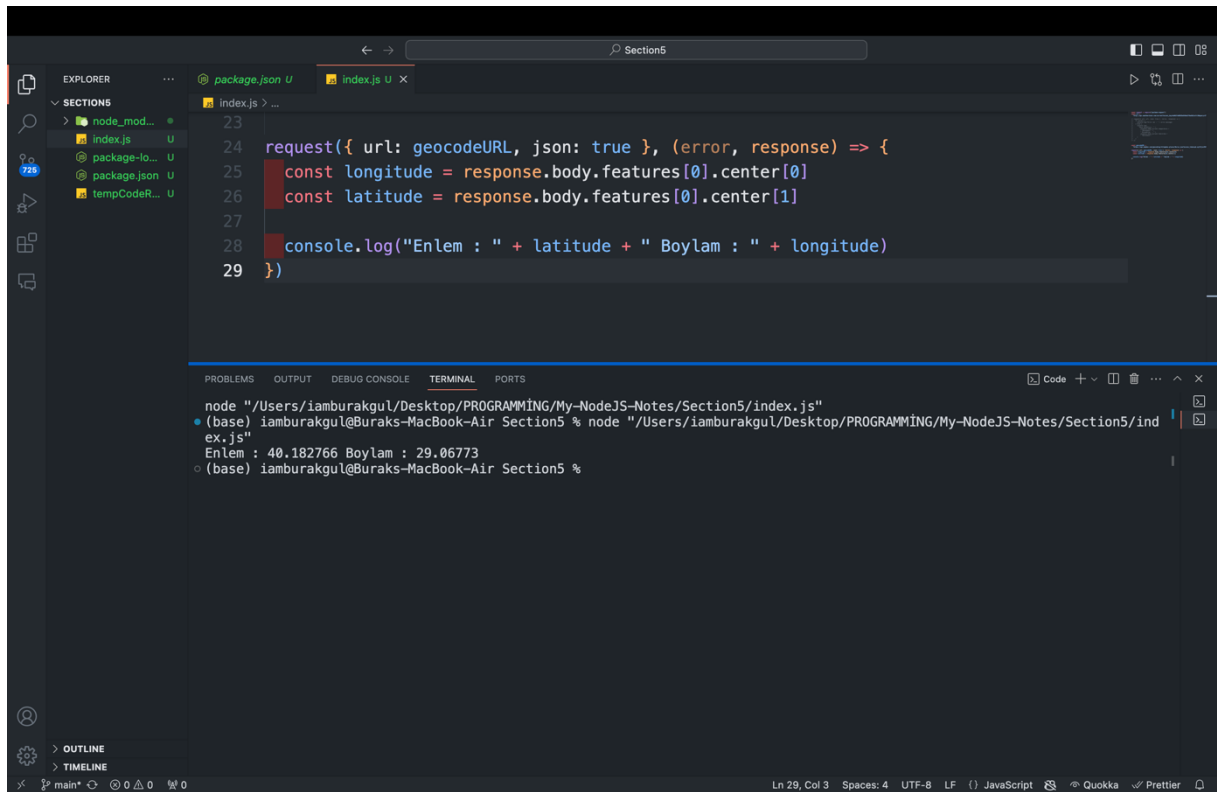
url kısmını geocodeURL adlı değişkene atayalım ve şu şekilde kullanalım.

```
Section5 - index.js

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 + " Boylam : " + longitude)
})
```

Çalıştırınca ise sonucu aşağıdadır. URL kısmında Bursa için istek attık ve çalıştırdığımızda çıkan sonuç aşağıdadır.



Bu haftalık burada bırakıp sonra ki hafta devam edelim işlemlerimize.