

# Node Js Ödevi

## 5<sup>th</sup> Week

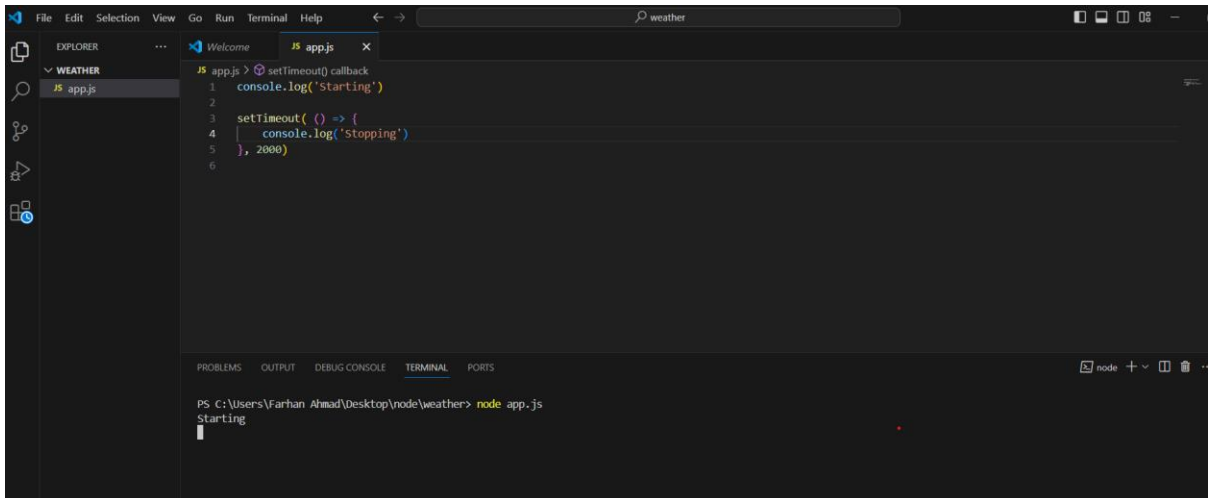
**Farhan Ahmad**

**20360859096**

Geçen hafta **Nodemon** ile **Arrow(OK) Modülünün** kullanışını ve **Debugger** gibi konuları incelemiştik. Bu hafta ise **Asynchronous HTTP Requests** ve **API** gibi konuları ele alacağız.

### Asynchronous HTTP Requests in Node.js

Node.js, geliştiricilere web tarayıcısı dışında JavaScript kodunu çalıştırma imkanı sunan popüler bir çalışma zamanı ortamıdır. Node.js'nin önemli özelliklerinden biri, asenkron programlamayı desteklemesidir, bu da geliştiricilerin engellemeyen HTTP istekleri yapmasını sağlar. Bu rapor, Node.js'in event loop mekanizması ve asenkron programlama yapısını ele alarak nasıl çalıştığını açıklamaktadır.



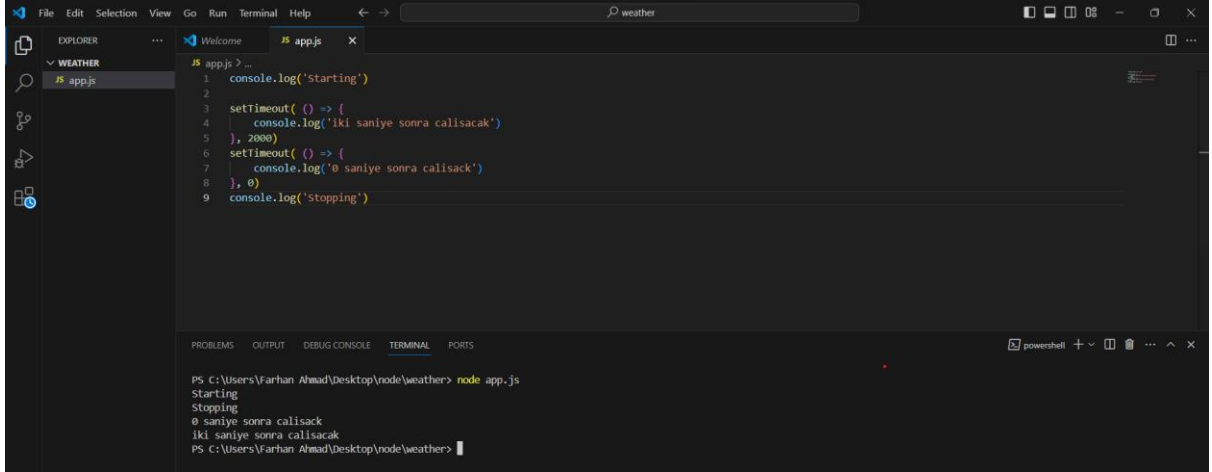
The screenshot shows the Visual Studio Code editor with a file named `app.js` open. The code in the editor is as follows:

```
1 console.log('Starting')
2
3 setTimeout(() => {
4   console.log('Stopping')
5 }, 2000)
6
```

The terminal at the bottom shows the command `node app.js` being executed, and the output is `Starting`.

## Event Loop ve Çağrı Yığını (Call Stack):

Node.js'in çalışma şekli, event loop ve çağrı yığını (call stack) arasındaki etkileşime dayanır. Event loop, Node.js'in tek iş parçacıklı doğasını korurken asenkron işlemleri yönetir. Herhangi bir işlem çağrıldığında, işlem çağrı yığınına eklenir. İşlem tamamlandığında ise yığından çıkarılır. Bu şekilde, programın çalışması sağlanır.



```
1 console.log('Starting')
2
3 setTimeout(() => {
4   console.log('İki saniye sonra çalışacak')
5 }, 2000)
6
7 setTimeout(() => {
8   console.log('0 saniye sonra çalışacak')
9 }, 0)
10 console.log('Stopping')
```

```
PS C:\Users\Farhan Ahmad\Desktop\node\weather> node app.js
Starting
Stopping
0 saniye sonra çalışacak
İki saniye sonra çalışacak
PS C:\Users\Farhan Ahmad\Desktop\node\weather>
```

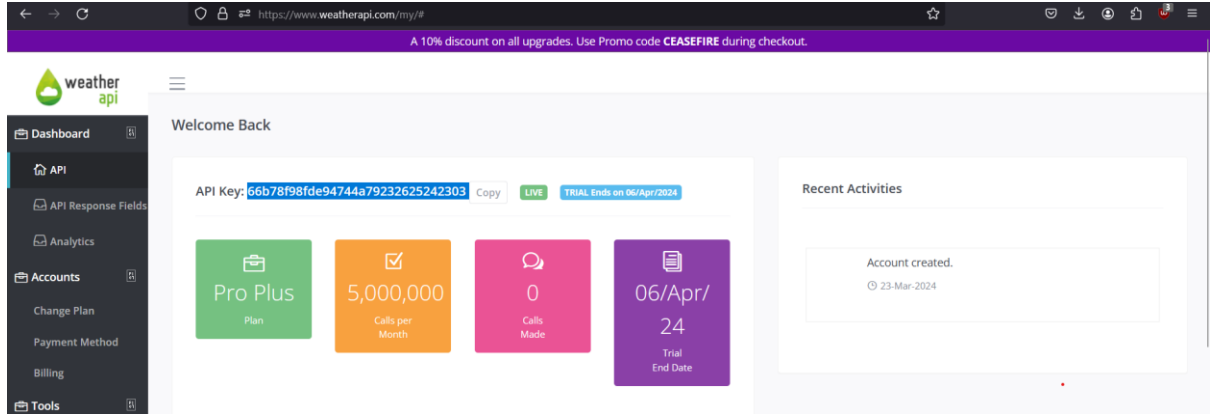
Yukarıdaki kodu çalıştırdığımızda ilk olarak “Starting”, sonra “Stopping”, sonra “0 saniye sonra çalışacak” sonra “iki saniye sonra çalışacak” olarak çalışmaktadır. Bu sıralama, Node.js'in asenkron doğasından kaynaklanmaktadır. Node.js, asenkron işlemleri yönetmek için event loop kullanır. Kodunuz çalıştırıldığında, console.log('Starting') işlemi ilk olarak çağrı yığınına eklenir ve hemen çalıştırılır. Sonra setTimeout fonksiyonları çağrı yığınına eklenir, ancak bunlar asenkron olarak çalıştırılacağı için event loop tarafından yönetilecektir.

## NodeJs'te API Kullanımı:

Weatherstack API'sini kullanarak hava durumu verilerini almak için Node.js uygulamasında nasıl bir istek yapabileceğinizi anlatamaktıyız. Aşağıda adım adım nasıl yapılacağımızdan bahsedilmiştir:

## Weatherapi API Kullanımı:

Hava durumu verilerine erişmek için Weatherapi API'sini kullanacağız. Bu API'ye erişmek için öncelikle <https://www.weatherapi.com> adresinden ücretsiz bir hesap oluşturmamız gerekir. Hesap oluşturulduktan sonra API erişim anahtarımızı alabiliriz. Örneğin, API erişim anahtarımız şu şekilde olabilir: 66b78f98fde94744a79232625242303.



## Node.js Uygulaması

Node.js uygulamamızı başlatmak için ilk adım olarak bir **npm** projesi olarak başlatmamız gerekmektedir. Bunun için klasörümüzde terminali açıp **npm init -y** komutunu kullanabiliriz. Daha sonra **request npm** modülünü yüklememiz gerekmektedir.

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

const apiKey = '66b78f98fde94744a79232625242303 ';
const location = 'London'; // Hedeflediğiniz şehir adını buraya yazın

const url =
`http://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${location}`;

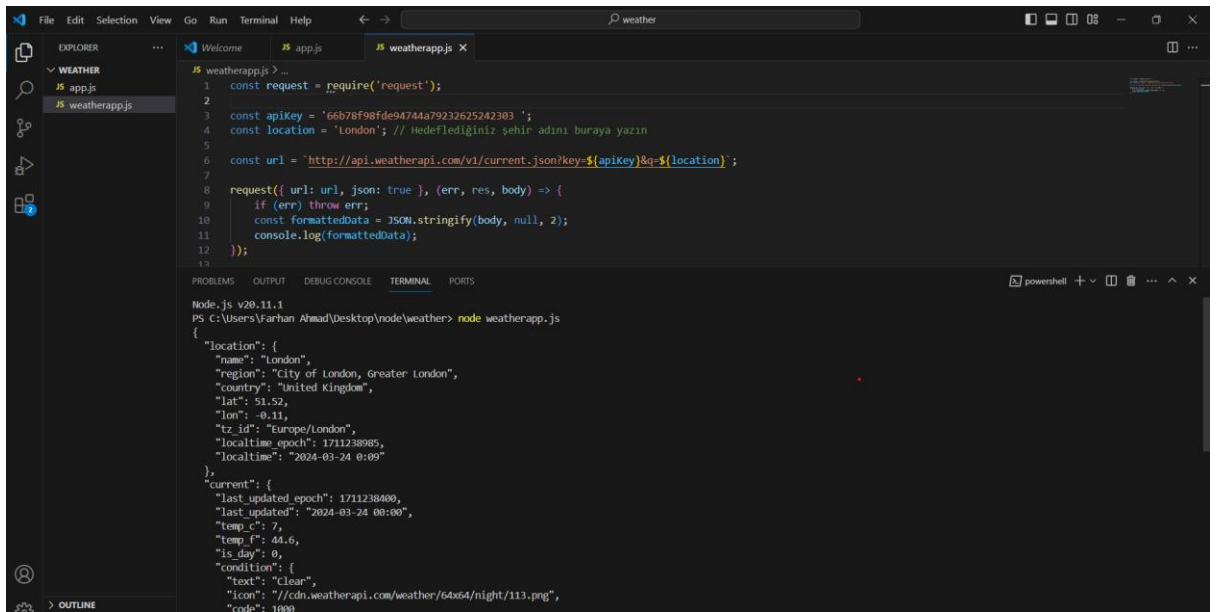
request({ url: url, json: true }, (err, res, body) => {
  if (err) throw err;
  console.log(body); // Hava durumu verileri burada
});
```

```
PS C:\Users\Farhan Ahmad\Desktop\node\weather> node weatherapp.js
{
  location: {
    name: 'London',
    region: 'City of London, Greater London',
    country: 'United Kingdom',
    lat: 51.52,
    lon: -0.11,
    tz_id: 'Europe/London',
    localtime_epoch: 1711237827,
    localtime: '2024-03-23 23:50'
  },
  current: {
    last_updated_epoch: 1711237500,
    last_updated: '2024-03-23 23:45',
    temp_c: 6,
    temp_f: 42.8,
    is_day: 0,
    condition: {
      text: 'Partly cloudy',
      icon: '//cdn.weatherapi.com/weather/64x64/night/116.png',
      code: 1003
    },
    wind_mph: 13.6,
    wind_kph: 22,
    wind_degree: 260,
    wind_dir: 'W',
    pressure_mb: 1009,
    pressure_in: 29.8,
    precip_mm: 0,
    precip_in: 0,
    humidity: 70,

```

## Node.js ile Hava Durumu API'sini Kullanarak Özelleştirilmiş İstekler ve Verilerin Biçimlendirilmesi:

- İstek Özelliklerini Özelleştirme



```
1 const request = require('request');
2
3 const apiKey = '66b78f98fde94744a79232625242303';
4 const location = 'London'; // Hedeflediğiniz şehir adını buraya yazın
5
6 const url = 'http://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${location}';
7
8 request({ url: url, json: true }, (err, res, body) => {
9   if (err) throw err;
10  const formattedData = JSON.stringify(body, null, 2);
11  console.log(formattedData);
12 });
```

```
Node.js v20.11.1
PS C:\Users\Farhan Ahmad\Desktop\node\weather> node weatherapp.js
{
  "location": {
    "name": "London",
    "region": "City of London, Greater London",
    "country": "United Kingdom",
    "lat": 51.52,
    "lon": -0.11,
    "tz_id": "Europe/London",
    "localtime_epoch": 1711238985,
    "localtime": "2024-03-24 0:09"
  },
  "current": {
    "last_updated_epoch": 1711238480,
    "last_updated": "2024-03-24 00:00",
    "temp_c": 7,
    "temp_f": 44.6,
    "is_day": 0,
    "condition": {
      "text": "Clear",
      "icon": "//cdn.weatherapi.com/weather/64x64/night/113.png",
      "code": 1000
    },

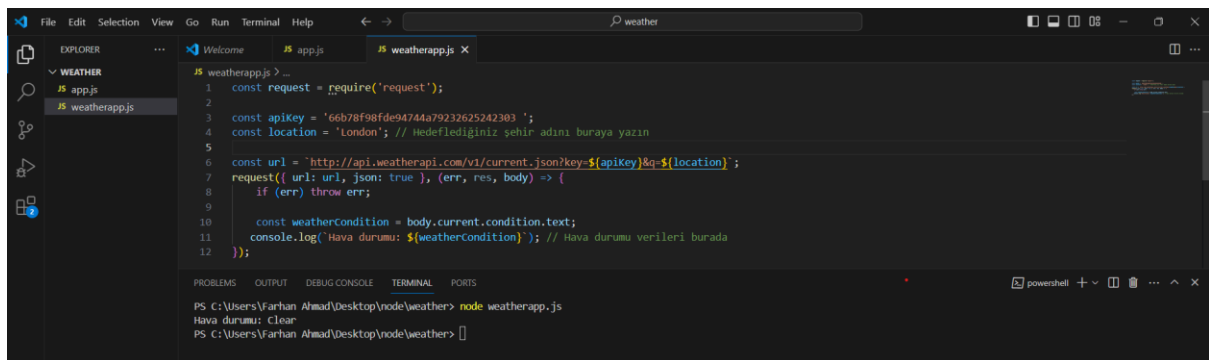
```

- Hava Durumu Tahmini Bilgilerini Almak:

API'den alınan verileri kullanarak hava durumu tahmini bilgilerini alacağız.

```
const weatherCondition = body.current.condition.text;
console.log(`Hava durumu: ${weatherCondition}`); // Hava durumu verileri burada
```

**body.current.condition.text;** : burada noktalar(.) ayırmak için kullanılıyor mesela body içinde **current** bölümünden condition bölgesindeki yazıyı yansıt demek istenmiş.

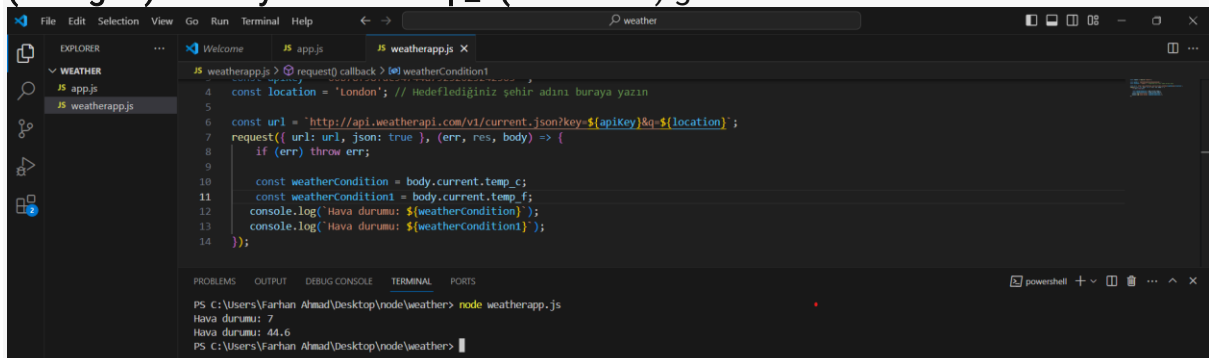


```
1 const request = require('request');
2
3 const apiKey = '66b78f98fde94744a79232625242303';
4 const location = 'London'; // Hedeflediğiniz şehir adını buraya yazın
5
6 const url = 'http://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${location}';
7 request({ url: url, json: true }, (err, res, body) => {
8   if (err) throw err;
9
10  const weatherCondition = body.current.condition.text;
11  console.log(`Hava durumu: ${weatherCondition}`); // Hava durumu verileri burada
12 });
```

PS C:\Users\Farhan Ahmad\Desktop\node\weather> node weatherapp.js  
Hava durumu: Clear  
PS C:\Users\Farhan Ahmad\Desktop\node\weather>

- Ölçü Birimlerini Değiştirme:

API'den alınan verileri farklı ölçü birimlerine çevirmek için **body.current.temp\_c** (santigrat) ve **body.current.temp\_f** (fahrenheit) gibi alanları kullanabilirsiniz.



```
1 const request = require('request');
2
3 const apiKey = '66b78f98fde94744a79232625242303';
4 const location = 'London'; // Hedeflediğiniz şehir adını buraya yazın
5
6 const url = 'http://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${location}';
7 request({ url: url, json: true }, (err, res, body) => {
8   if (err) throw err;
9
10  const weatherCondition = body.current.temp_c;
11  const weatherCondition1 = body.current.temp_f;
12  console.log(`Hava durumu: ${weatherCondition}`);
13  console.log(`Hava durumu: ${weatherCondition1}`);
14 });
```

PS C:\Users\Farhan Ahmad\Desktop\node\weather> node weatherapp.js  
Hava durumu: 7  
Hava durumu: 44.6  
PS C:\Users\Farhan Ahmad\Desktop\node\weather>

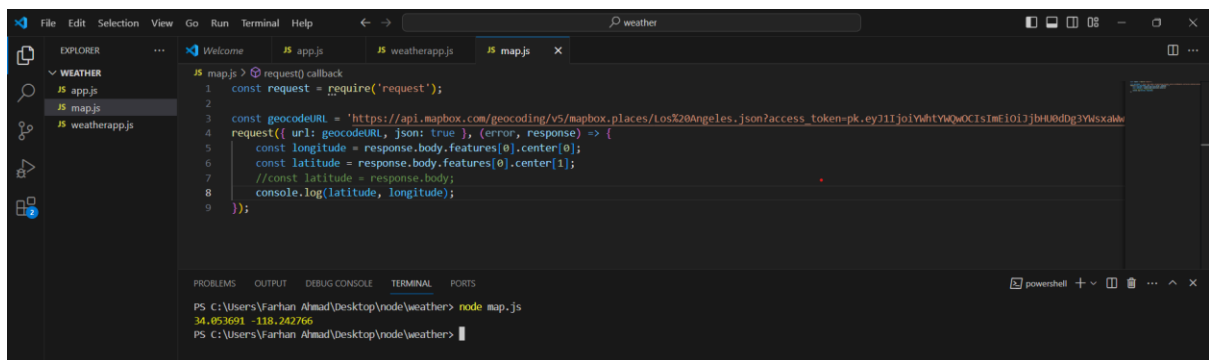
## Geocoding API Kullanımı:

Kullanıcının sağladığı adresi latitude ve longitude'a dönüştüren bir Geocoding servisini ele alacağız. İşlem adımlarını şu şekilde özetlenmektedir:

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

const geocodeURL =
'https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?access_token=pk.eyJ1IjoiYWhtYWQwOCIsImEiOiIjbjHU0dDg3YWsxZWw5Mm1ueHhlenNkempzIn0.UdvbLgMJdWENZGpG1cSsog&limit=1';

request({ url: geocodeURL, json: true }, (error, response) => {
  const longitude = response.body.features[0].center[0];
  const latitude = response.body.features[0].center[1];
  //const latitude = response.body;
  console.log(latitude, longitude);
});
```



Böylece, kullanıcının sağladığı adresi lat ve lon değerlerine dönüştürebilir ve bu bilgileri kullanarak harita entegrasyonu gibi işlemler gerçekleştirebilirsiniz.