Node.JS ile Web Programlama
Rapor - 5

Mahire Zühal Özdemir

İÇİNDEKİLER

ŞEKİLLER DİZİN

setTimeOut Kod Örneği:

Bu JavaScript kodu, asenkron işlemleri göstermek için kullanılıyor.İlk setTimeout fonksiyonu, 2 saniyelik bir gecikme belirtir ve içindeki işlev, 2 saniye sonra '2 saniye bekleme' mesajını konsola yazdırır.

İkinci setTimeout fonksiyonu ise, 0 saniyelik bir gecikme belirtir ve içindeki işlev, diğerleri tamamlandıktan hemen sonra çalışır. Ancak, JavaScript'in asenkron doğası nedeniyle, bu işlevin önce mi sonra mı çalışacağı belirsizdir. Bu nedenle, '0 saniye bekleme' mesajı 'Bitir' mesajından önce veya sonra konsola yazılabilir.

setTimeout, aslında V8 motorunun bir parçası değil; Node.js tarafından sonradan eklenmiş bir fonksiyondur.

```
console.log('Starting')

setTimeout(() => {
    console.log('2 saniye bekleme')
},2000)


setTimeout(() => {
    console.log('0 saniye bekleme')
},0)

console.log('Bitir')
```

Şekil 1: settimeout kod örneği

```
C:\Users\zuhal\Desktop\Node.Js\Hafta-5>node app.js
Starting
Bitir
0 saniye bekleme
2 saniye bekleme
```

Kod Çıktısı

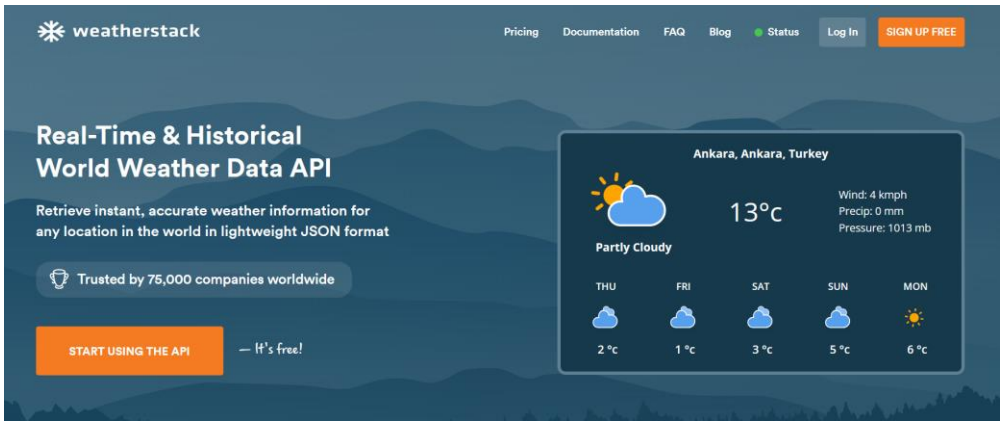
Weather-App

Bu hafta Node.JS ile hava durumu uygulaması yapacağız. Uygulamamız, çeşitli API sağlayıcılarından hava durumu bilgilerini alacak ve kullanıcıya sunacak. İlk adım olarak, kullanabileceğimiz API sağlayıcılarını gözden geçirelim:

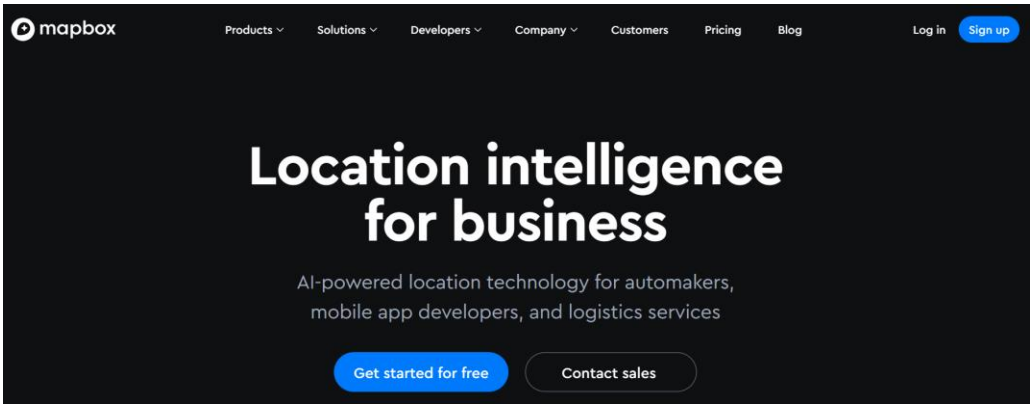
- Weather Stack
- Mapbox

Bu sağlayıcıların sunduğu API'leri kullanabilmek için öncelikle uygulamalara kaydolmamız gerekiyor. Kaydolduktan sonra, API anahtarlarını alıp uygulamamızda kullanabiliriz. Daha sonra, bu API'leri entegre ederek hava durumu verilerini çekecek ve kullanıcı arayüzünde göstereceğiz. Hava durumu

uygulamamızın kullanıcıya güncel ve doğru bilgiler sunması için geliştirme sürecini adım adım ilerleteceğiz.



Şekil 2: WeatherStack App



Şekil 3: Mapbox App

Request Modülü

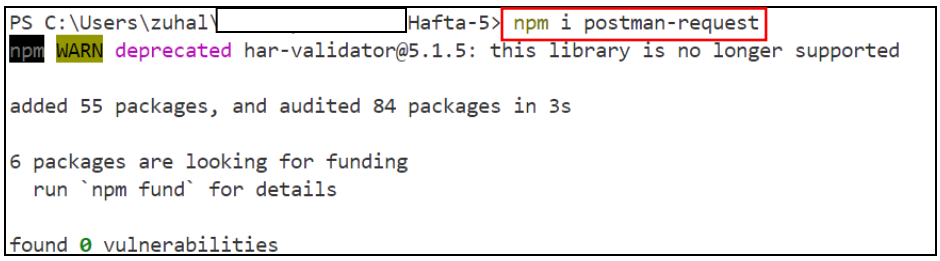
request, Node.js ortamında HTTP isteklerini yapmak için kullanılan bir modüldür. Bu modül, HTTP GET, POST, PUT, DELETE gibi istekleri kolayca oluşturmanıza ve sunucudan veri almanıza olanak tanır. Özellikle web tarayıcısı dışında, sunucu taraflı uygulamalar geliştirirken veya bir HTTP isteğinin sonucuna göre işlem yapmanız gereken durumlarda sıkça kullanılır. Örneğin, API'lerle iletişim kurmak, web sayfalarını indirmek veya veri çekmek gibi işlemler için request modülü oldukça yaygın olarak kullanılır. request

Postman-Request Modülü

postman-request, Postman tarafından geliştirilen ve HTTP isteklerini yapmak için kullanılan bir Node.js modülüdür. Bu modül, Postman'ın gelişmiş özelliklerini ve güvenilirliğini Node.js ortamına taşır ve HTTP isteklerini oluşturmak ve yönetmek için kullanıcı dostu bir arabirim sunar.

postman-request, genellikle Node.js uygulamalarında dış kaynaklardan veri almak veya API'lerle iletişim kurmak gibi durumlarda kullanılır. İstekleri göndermek, yanıtları işlemek ve çeşitli HTTP yöntemlerini (GET, POST, PUT, DELETE vb.) kullanmak için kullanılabilir. postman-request

postman-request indirip kullanmak için '***npm i postman-request***' komutu kullanılır.



Şekil 4: Postman-request Kurulum

Api'lerden Sorgu İşlemleri

Weather Stack

Weather stack uygulamasına kaydolduktan sonra tarafımıza sağlanan anahtarı kullanarak basit bir url ile sorgu verilerine erişebiliriz.

http://api.weatherstack.com/current?access_key={kullanıcı anahtarı}&query={sorgu için enlem ve boylam}

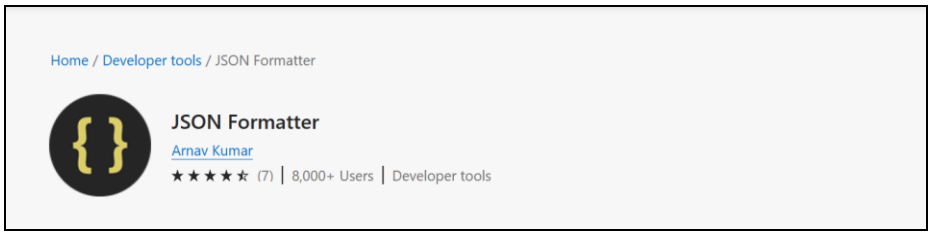
Enlem ve boylam verileri query=37.8267,-122.4233 şeklinde yazılarak sonuçlara erişilebilir.

Anahtar ve girilen enlem boylam değerleri doğru olduğu sürece json çıktısı döndürecektir.

Anahtar değeri yanlış veya eksikse sorgu sonucu aşağıdaki gibi Json formatında bir veri dönecektir.

Şekil 5: Hatalı anahtar sorgusu

Çıktımız Json formatında olacağını söylemiştik. Düz metin yerine daha düzgün bir şekilde çıktıyı görüntülemek için tarayıcının ***'extensions'*** kısmından Json formatter indirilebilir.



Şekil 6: Json Formatter

Weather Stack Uygulaması

Aşağıdaki kodu kullanarak, Node.js ortamında Weatherstack API'si ile hava durumu bilgilerini alan bir HTTP isteği oluşturabiliriz.

İlk satırda postman-request modülü request değişkenine atanır. Bu, HTTP istekleri yapmak için kullanılacak bir modülü temsil eder.

İkinci satırda, isteğin gönderileceği API'nin URL'si olan url değişkeni oluşturulur. Bu URL, Weatherstack API'sine bir istek göndermek için kullanılacaktır. URL'de aşağıdaki bileşenler bulunur:

Temel URL: https://api.weatherstack.com/current

Parametreler:

access_key: Weatherstack API'sini kullanmak için gerekli olan erişim anahtarı.

query: Hava durumu bilgisini almak istediğiniz konumu belirtir.(Enlem,Boylam)

```
const request = require('postman-request')
const url = 'https://api.weatherstack.com/current?access_key=5c3376fd0dd61382d80dc06521e4bf2a&query=37.8267,-122.4233'
```

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

})
```

Gelen hava durumu bilgilerinin belirli bir birimde gelmesini istiyorsak url sonuna ***units*** değişkeni atanır.

```
const url = 'https://api.weatherstack.com/current?access_key=5c3376fd0dd61382d80dc06521e4bf2a&query=37.8267,-122.4233&units=f'
```

`units`	• `m` for Metric • `s` for Scientific • `f` for Fahrenheit

```
request({url:url, json:true}, (error, response) =>{
    //console.log(response.body.current)
    console.log('Hava Sıcaklığı : ',response.body.current.temperature ,
    'Hissedilen: ',response.body.current.feelslike)
})
```

MapBox

Mapbox uygulamasına kaydolduktan sonra tarafımıza sağlanan anahtarı kullanarak basit bir url ile sorgu verilerine erişebiliriz.

https://api.mapbox.com/geocoding/v5/mapbox.places/{Girdi Konum}.json?access_token=pk.{kullanıcı anahtarı}

MapBox Uygulaması

```
const geocodeUrl = 'https://api.mapbox.com/geocoding/v5/mapbox.places/Bursa.json?access_token=pk.(kisiye verilen api adresi)'
```

```
request({url:geocodeUrl, json:true},(error,response)=>{
    const boylam = response.body.features[0].center[0]
    const enlem = response.body.features[0].center[1]
    console.log('enlem: ' + enlem + 'boylam: '+ boylam)
})
```

Sorgu başarılı ise response dolu error boş sorgu başarılı değilse response boş error dolu olur.

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

Aşağıdaki kod kullanılarak;

- request fonksiyonu, geocodeUrl adlı bir URL'ye bir HTTP isteği gönderir. İsteğin yanıtı, bir JSON nesnesi olarak alınır (json: true seçeneğiyle).
- İsteğin tamamlanmasıyla birlikte, bir geri çağrı işlevi başlatılır. Bu işlev, error ve response parametrelerini alır.
- İlk olarak, error kontrol edilir. Eğer bir hata varsa, "hava durumu servisine bağlantı kurulamadı" mesajı konsola yazdırılır.
- Eğer bir hata yoksa, response nesnesinin body özelliği içinde hava durumu bilgileri bulunur. Bu bilgiler konsola yazdırılır. ***response.body.current.temperature*** ile mevcut sıcaklık, ***response.body.current.feelslike*** ile hissedilen sıcaklık bilgileri alınır ve konsola yazdırılır.

```
request({url:geocodeUrl, 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)
    }
})
```

Aşağıdaki kod yapısını kullanarak;

- request fonksiyonu, geocodeUrl adlı bir URL'ye bir HTTP isteği gönderir. İsteğin yanıtı, bir JSON nesnesi olarak alınır (json: true seçeneğiyle).
- İsteğin tamamlanmasıyla birlikte, bir geri çağrı işlevi başlatılır. Bu işlev, error ve response parametrelerini alır.
- İlk olarak, error kontrol edilir. Eğer bir hata varsa, "hava durumu servisine bağlantı kurulamadı" mesajı konsola yazdırılır.
- Ardından, yanıt kontrol edilir. Eğer yanıtın body özelliği bir hata içeriyorsa, "girilen konum bilgisi bulunamadı" mesajı konsola yazdırılır.
- Yukarıdaki durumlar sağlanmıyorsa, yanıt içinde hava durumu bilgileri bulunmuştur ve bu bilgiler konsola yazdırılır. **response.body.current.temperature** ile mevcut sıcaklık, **response.body.current.feelslike** ile hissedilen sıcaklık bilgileri alınır ve konsola yazdırılır.
- Son olarak, error nesnesi konsola yazdırılır. Bu, herhangi bir hata olup olmadığını kontrol etmek için bir tür hata işleme mekanizması sağlar.

```
request({url:geocodeUrl, 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)
    }
})
```