

Furkan Kazım Çam 21360859036

5.Hafta

Yeni bir klasör oluşturarak işleme başlayınız. İsmine weatherApp veriniz. Oluşturduğunuz dosyaya app.js ismini veriniz.

Aşağıdaki kodları yazın.

```
console.log("Başla");

setTimeout(()=>{

    console.log("2 saniye bekle");

},2000)

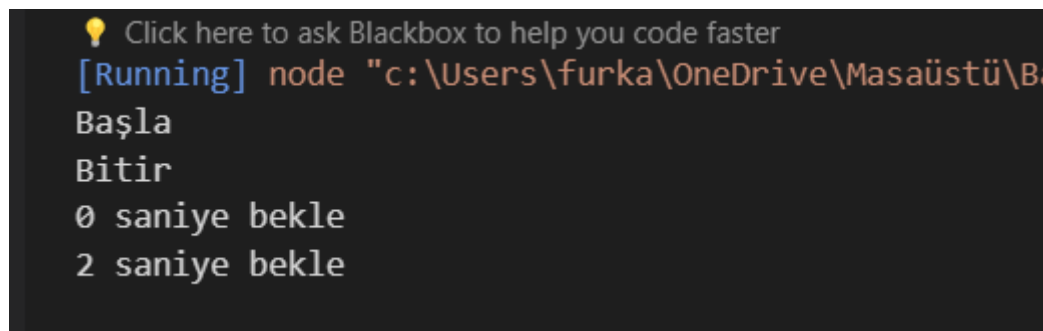
setTimeout(()=>{

    console.log("0 saniye bekle");

},0)

console.log("Bitir");

ve çalıştırın.
```



```
Click here to ask Blackbox to help you code faster
[Running] node "c:\Users\furka\OneDrive\Masaüstü\Ba
Başla
Bitir
0 saniye bekle
2 saniye bekle
```

Neden bu sıra ile çalıştı?

Çünkü 2 saniyelik beklemeye neden olan fonksiyon kodun geri kalanını geciktirmez.

Peki neden 0 saniye bekle önce çalıştı?

Fonksiyon çağrıldığında stacke atılır. Mainden sonra ilk olarak 2 saniye beklemeye neden olan fonksiyon stacktedir. Onun üstüne ise beklememe neden olacak olan fonksiyon gelir. Stack mantığından dolayı ilk olarak “0 saniye bekle” sonra ise “2 saniye bekle” ekrana yazdırılır.

SetTimeout fonksiyonunun Node.js'in API'sinin bir parçası olduğunu belirtmek önemlidir. Bu, JavaScript'in asenkron doğasını anlamak için temel bir noktadır. Bir işlem, özellikle de zamanlayıcı işlevlerle, çağrı yığına eklenir ve ardından Node API'sine iletilir. Bu, işlemin bekletilmesini ve diğer işlemlerin çalıştırılmasını sağlar.

Bir işlemin çağrı yığından çıkarılması ve callback metodunun tetiklenmesi, Node.js'in singlethreaded yapısının etkili bir şekilde kullanılmasını sağlar. Örneğin, setTimeout(0) çağrısı, diğer işlemlerle birlikte sırayla çalışır ve Node API'sine iletdikten sonra callback queue'ye eklenir. Bu, JavaScript'in asenkron doğasının temel bir örneğidir.

Event loop callback queue'yu kontrol eder ve çağrı yığınının boş olup olmadığını düzenler. Eğer çağrı yığını boş ise, kuyruktaki işlevleri sırayla çağırır. Bu şekilde, programın asenkron olarak çalışması ve performansın artması sağlanır.

Weatherstack

weatherstack.com adlı siteye gidin ve ücretsiz hesap açın. Bu site size rastgele API access key verecektir.

Şuanda ki hava durumu verilerini ise <http://api.weatherstack.com/> URL'ye sahip siteden alacağız.

Keyi aldıktan sonra tarayıcıyı açıp aşağıdaki URL'yi yazın.

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

Karşımıza JSON formatında bilgiler görüyoruz. Yapacağımız uygulama ile burdan bilgileri çekmeye çalışacağız.

App.js'in içeriğini temizleyin.

Request modülünü kullanacağız. Modül eski fakat hala kullanılmaktadır. Postman yerine postma-request kullanıcaz.

İlk olarak npm init ile başlayalım. Soruları hızlı geçmek istiyorsak npm init -y yazmalıyız. Herhangi bir soru sormadan npm başlatılacaktır.

Request modülünü indirerek devam edelim.

npm i request yazarak indirebilirsiniz.

App.js dosyasına request modülünü dahil edelim.

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

tarayıcınızdan URL'yi kopyalayın.

```
Const url=http://api.weatherstack.com/current?access_key=  
sizinkeyiniz&query=37.8267,-122.4233
```

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

Yazın ve çalıştırın. Birden fazla özellik bulunmaktadır. Biz ise burda body özelliğini kullanacağız. Bunun içi aşağıdaki kodu yazalım.

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

Tekrar çalıştırın.

Bilgi çekiyoruz, birde çekilen bilgiyi json türünde çekelim.

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

Json:true bize bu özelliği sunar

Ayarları değiştirerek devam edelim.

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

Aynı sonucu almalısınız. Farklılaştırmak için bir uzantıya ihtiyacınız var. Json formatter adlı uzantıyı tarayıcınıza ekleyin. Ekledikten sonra sayfayı yenileyin. Bu işlem ekranımızın ön yüzünü değiştirdi.

Peki diğer bilgileri nasıl çekeriz?

```
request({ url: url, json:true }, (error, response) => {  
  console.log('It is currently ' + response.body.current.temperature + ' degrees and it feels  
  like ' +  
  response.body.current.feelslike + ' degrees.')
```

Çalıştırın. Birimler karışık gelmektedir. Bu durumu düzeltelim. weatherstack documentation sayfasına gidin ve birimler kısmını arayın ve aşağıdaki düzenlemeleri yapın.

units = m for celcius, default

units = s for scientific, in kelvin, f for Fahrenheit.

Şimdi ise URL'yi güncellememiz gerekmektedir.

```
url=http://api.weatherstack.com/current?access_key=  
sizinkeyiniz&query=37.8267,-122.4233&units=f'
```

Yukardaki işlemler tamamlandıysa programı tekrar çalıştırabilirsiniz. Text'i güzelleştirerek daha okunaklı bir metin elde edebiliriz.

```
request({ url: url, json:true }, (error, response) => {  
  
  console.log(response.body.current.weather_descriptions[0] + ' Şu anda ki derece:' +  
  response.body.current.temperature + 'Hissedilen derece ise:' +  
  response.body.current.feelslike + '  
degrees.')})
```

Tekrar çalıştırarak konsolda sonucu elde edebiliriz.

GEOCODING SERVICE

İlk olarak, erişim anahtarlarını kontrol edip varsayılan genel anahtarı kopyalamakla başlanır. Ardından, Mapbox API dökümantasyonuna gidilerek ileri jeokodlama servisi aranır. Burada, örnek bir istek verilir ve Los Angeles için bir sorgu yapılabilmesi için gerekli olan erişim anahtarı sağlanır.

İsteğe bağlı parametreler araştırılarak, isteğin özelleştirilmesi için dil, limit gibi parametrelerin nasıl kullanılabileceği incelenir. Sorgu dizesi değiştirilerek, isteğin sonuçlarında yalnızca bir öge olacak şekilde limit parametresi belirlenir. Bu adımdan sonra, app.js dosyasına geçilir ve belirtilen URL kopyalanarak kullanılır.

Const geocodeURL =

'https://api.mapbox.com/geocoding/v5/mapbox.places/Los%20Angeles.json?access_token=pk.eyJ1IjoiaWZ

zNSIsImEiOiJa215YnU1NHlwMmU4MnVvMnI0dnI2YndjIn0.ilitdw0flCDPjlnjP0Kl7g&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]
```

```
  console.log(latitude, longitude)
```

```
})
```

Son olarak, program yeniden alıřtırılır. Bu ařamada, JavaScript kodu kullanılarak Mapbox API'ye yapılan iřteęin sonucunda elde edilen konum bilgileri alınır ve iřlenir. Bylece, adım adım izlenerek bir uygulamanın Mapbox API'sini kullanarak konum arama iřlemlerini gerekleřtirmesi saęlanır. Bu sre, uygulamanın kullanıcılarına doęru ve gncel konum bilgilerini sunmasını saęlar.