

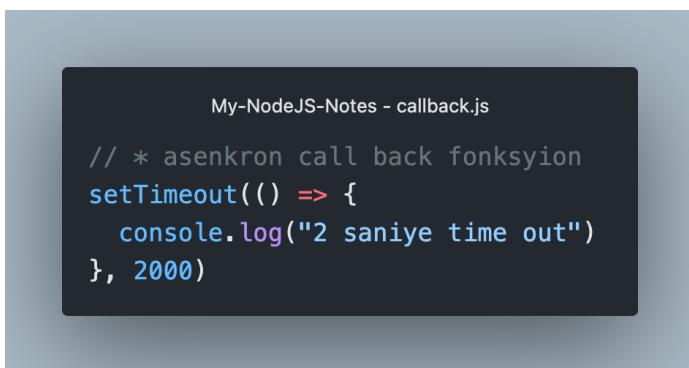
# Node.js Section 6

Merhabalar bugün node.js serimize devam ediyoruz ve 6.bölümdeyiz.

Oncelikle auxiliary topics klasöründe callback.js adında bir dosya oluşturup. callback fonksiyonlara bakalım.

Bir fonksiyonun parametre olarak başka bir fonksiyonu alması ve onu kendi içinde çağırması olayına callback denir. Parametre olarak alıp çağrıdığımız fonksiyona ise callback fonksiyon deriz.

Callback fonksiyonlar senkron asenkron olabilirler. Genellikle asenkron olurlar.



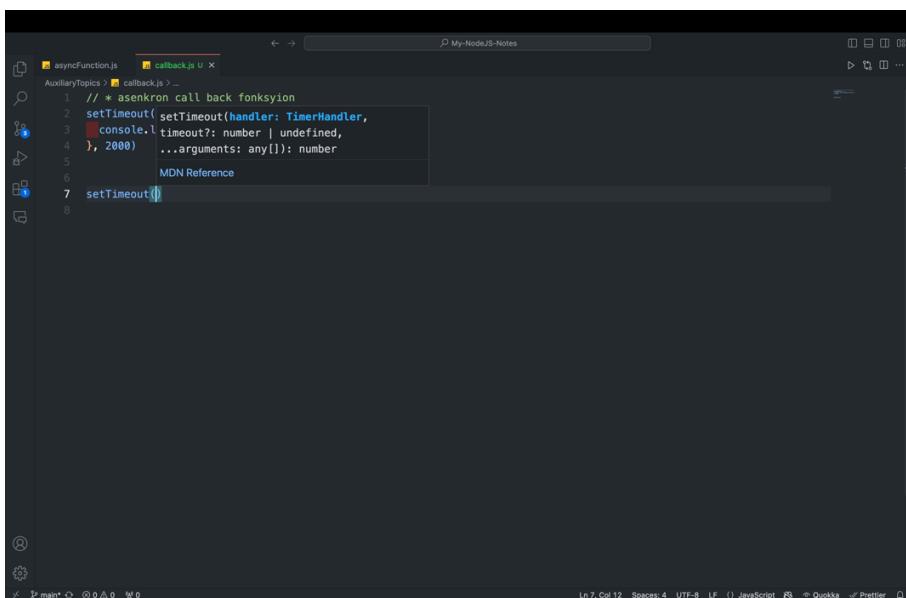
```
My-NodeJS-Notes - callback.js

// * asenkron call back fonksyon
setTimeout(() => {
  console.log("2 saniye time out")
}, 2000)
```

Burada setTimeout fonksiyonuna parametre olarak iletilen fonksiyon

```
() => {
  console.log("2 saniye time out")
}
```

Callback fonksiyondur.



```
// * asenkron call back fonksyon
setTimeout(setTimeout(handler: TimerHandler,
  timeout?: number | undefined,
  ...arguments: any[]): number
  MDN Reference
) =>
```

Handler dediği kısım bizim callback fonksiyonumuz. Callback fonksiyona completion handler dendiği zaman şaşırmamak lazım aynı şeyi ifade ediyorlar.

```
My-NodeJS-Notes - callback.js

// ! callback fonksiyonlarının hepsi asenkron değildir senkron olan da vardır

// * senkron callback fonksiyon
const names = ["Ali", "Berk", "Canan"]
const shortNames = names.filter((name) => {
    return name.length <= 4
})
```

Mesela buradada filter fonksiyonu bir parametre olarak fonksiyon almıştır. Bu parametre olarak aldığı fonksiyona names dizisinin bir elemanını verir , fonksiyon kendi içinde değerlendirme yapar ve o elemanı kabul edip etmeyeceğini(true/false) olarak söyler.

Buradaki şartımız harf sayısının 4 den küçük eşit olması gerektidir.

Şimdi bizim uygulamamız hava durumu uygulamasıydı o yüzden bir fonksiyon yazalım setTimeout gibi. Bir parametresi callback fonksiyon olsun diğer parametresi de string veri olsun(adresimiz).

Fonksiyonumuzun adı geocode olsun.

```
My-NodeJS-Notes - callback.js

// senkron fonksiyon
const geocode = (address, callback) => {
    const data = {
        latitude: 0,
        longitude: 0
    }
    return data
}
const data = geocode("Bursa")
console.log(data)
```

```
{ latitude: 0, longitude: 0 }
```

Çıktımız bu olacaktır.

Bu bir asenkron fonksiyondur.

Asenkron hale getirmek için içinde bir asenkron fonksiyon kullanalım.

```
My-NodeJS-Notes - callback.js

// asenkron fonksiyon
const geocode = (address, callback) => {
  setTimeout(() => {
    const data = {
      latitude: 0,
      longitude: 0
    }
    return data
  }, 2000)
}
const data = geocode("Bursa")
console.log(data)
```

Bu kodu çalıştırıralım ve undefined yazımızı görelim.

The screenshot shows the VS Code interface with the 'callback.js' file open in the editor. The code is identical to the one shown in the previous image. In the bottom right corner, the terminal window displays the following output:

```
(base) iamburakgul@Buraks-MacBook-Air AuxiliaryTopics % node callback.js
undefined
(base) iamburakgul@Buraks-MacBook-Air AuxiliaryTopics %
```

Neden undefined yazdı peki ?

Kodumuzda undefined yazdırılmasının sebebi, geocode fonksiyonunun bir değer döndürmemesidir. JavaScript'te, bir fonksiyonun dönüş değeri olmadığından (yani fonksiyon içinde return ifadesi kullanılmadığında veya return ifadesi olmadan sona erdiğinde), fonksiyonun dönüş değeri undefined olur. geocode fonksiyonunuz asenkron bir işlem (setTimeout) içeriyor ve bu işlem tamamlandığında bir callback fonksiyonu çağrırmak yerine doğrudan bir nesne döndürmeye çalışıyor, ancak bu nesne setTimeout fonksiyonunun kendi kapsamında kalıyor ve geocode fonksiyonunun dışına bir değer döndürülmüyor.

Asenkron bir işlem gerçekleştirdiğimizde, işlemin tamamlanmasını beklemek ve sonucu dışarıya aktarmak için bir callback fonksiyonu kullanmamız gereklidir. geocode fonksiyonunu, bir callback fonksiyonu alacak ve bu callback fonksiyonunu, setTimeout içindeki asenkron işlem tamamlandığında çağıracak şekilde yazmalıyız.



```
My-NodeJS-Notes - callback.js

// asenkron fonksiyon
const geocode = (address, callback) => {
  setTimeout(() => {
    const data = {
      latitude: 40.182846,
      longitude: 29.066834
    }
    callback(data)
  }, 2000
}

geocode("Bursa", (data) => {
  console.log(data) // Bu callback fonksiyonu `setTimeout` tamamlandığında çağrılır
})
```

Bu şekilde yaptığımızda bir sorun kalmayacaktır.

Bu kodda, geocode fonksiyonu bir address ve bir callback fonksiyonu alır. Asenkron işlem (setTimeout) tamamlandığında, callback fonksiyonu çağrılır ve data nesnesi bu fonksiyona argüman olarak geçirilir. Sonuç olarak, data nesnesi console.log(data); satırı ile yazdırılır.

geocode fonksiyonunun dışında bir değişkene atama yaparak bu değeri yakalamaya çalışmak işe yaramaz, çünkü asenkron işlem tamamlandığında, senkron olarak çalışan kodun geri kalanı çoktan çalışmış ve bitmiş olur.

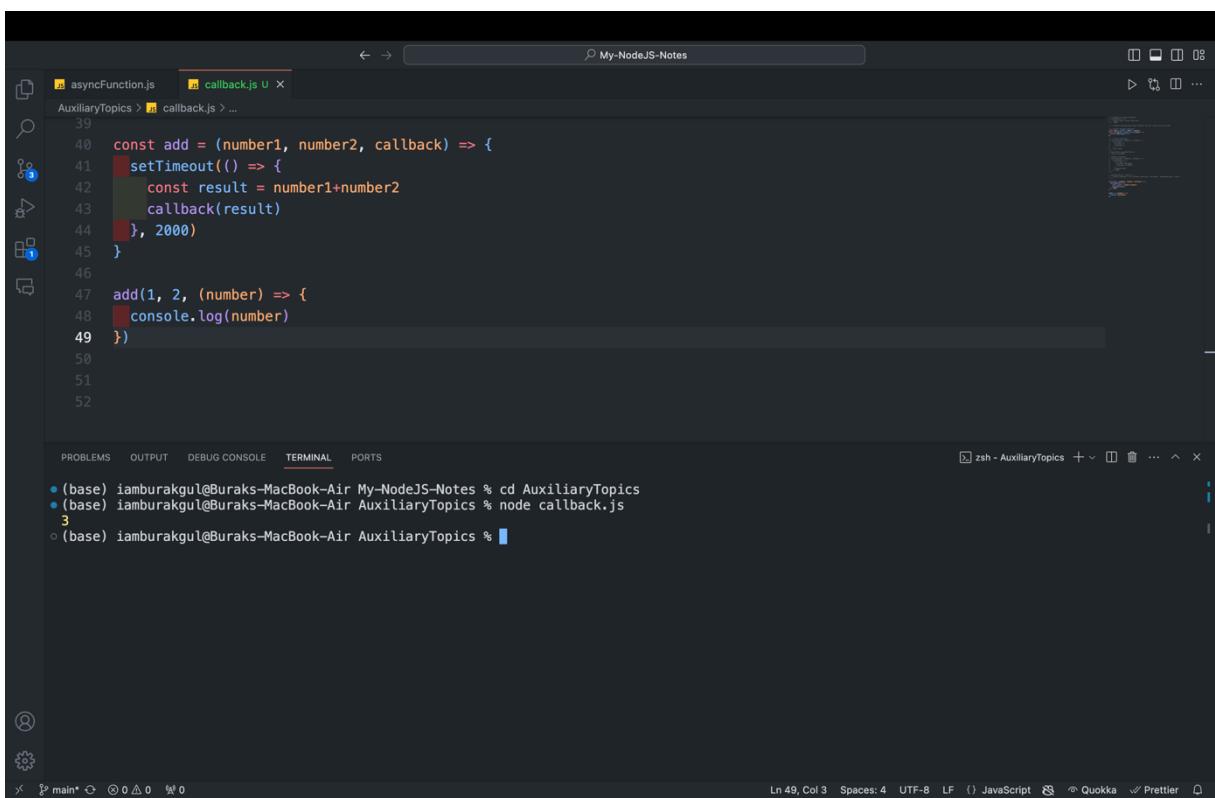
Şimdi ise başka bir callback fonksiyonu tanımlayalım. Parametre olarak aldığı sayıları 2 sn geçtikten sonra toplasın ve callback fonksiyonuna parametre olarak geçsin.

```
My-NodeJS-Notes - callback.js

const add = (number1, number2, callback) => {
  setTimeout(() => {
    const result = number1+number2
    callback(result)
  }, 2000)
}

add(1, 2, (number) => {
  console.log(number)
})
```

Ve çalıştırıralım. 2 sn sonra 3 yazısını göreceğiz.



```
callback.js

const add = (number1, number2, callback) => {
  setTimeout(() => {
    const result = number1+number2
    callback(result)
  }, 2000)
}

add(1, 2, (number) => {
  console.log(number)
})
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) iamburakgul@Buraks-MacBook-Air My-NodeJS-Notes % cd AuxiliaryTopics
(base) iamburakgul@Buraks-MacBook-Air AuxiliaryTopics % node callback.js
3
(base) iamburakgul@Buraks-MacBook-Air AuxiliaryTopics %
```

Şimdi callback fonksiyonu anladığımıza göre bunu weather app projemizde kullanalım.

Öncelikle ne yapacağımız ona bakalım.

Geocode apisinden şehir ismi girerek lokasyon(enlem/boylam) bilgilerini alacağız ve bu bilgileri weatherstack apisine verip hava sıcaklığı bilgilerini alacağız.

O zaman geocode için başlayalım. Section 5 klasörümüzü Section5-6 yapıp o klasöre girelim.

Utils adında bir klasör oluşturalım ve burada geocode adında dosyada geocode api işlemlerini yapalım.

Biz bu klasörlere ayırip farklı yerlerde neden yazmak istiyoruz ?

Her bir şehir için bu şekilde tek tek url oluşturmak fonksiyon yazmak mantıksız. N adet şehir ismi için çalışabilir hale getirmeliyiz.

```
Section5-6 - geocode.js

const geocode = (address,callback) => {
    // function body goes here
}
geocode("Bursa", (error,data) => {
    // callback function body goes here
})
```

Genel yapımız bu olacak. Address yerine Bursa,Sivas gibi verileri gireceğim ve callback ile çağrılmam gereken yerlerde çağrıracagız.

Callback fonksiyonumuz request sonucu dönen respondedan dönen error ve datayı alacak. Hata varsa data olmaz , data varsa hata olmaz bunu da hatırlamamız gerekiyor.

Request fonksiyonu asenkron olduğu için geocode fonksiyonu da asenkron olacaktır.

#### Section5-6 - index.js

```
const geocodeURL =
  "https://api.mapbox.com/geocoding/v5/mapbox.places/Bursa.json?ac-
cess_token=pk.eyJ1IjoiZGV2ZWxvcGVyYnVvYWtndWwiLCJhIjoiY2x0emlyNThq
MDBmczJ1cGRnNmw4YTRoaiJ9.TKyh0iv0a_FSkhRakSeAyA"

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)
})
```

Index.js içindeki bu kodumuzu geocode.js içinde yapmaya çalışalım.  
Fonksiyonumuzda url olacak ve url e değişken olarak Bursa,Sivas İstanbul gibi  
değişkenler için address parametresini kullanacağız.

Request kısmını tamamen kopyalayıp geocode içine koyacağz. Hata varsa hatayı yoksa  
gelen enlem ve boylam bilgilerine ek olarak şehir ismiyle beraber bir nesne olarak  
döndürelim.

```

Section5-6 - geocode.js

const geocode = (address, callback) => {
    // function body goes here
    const geocodeURL =
        "https://api.mapbox.com/geocoding/v5/mapbox.places/" +
        encodeURI(address) +
        ".json?access_token=pk.eyJ=eyJ1IjoiZGV2ZWxvcGVyYnVyYWtndWwiLCJhIjoiY2x0emlyNThqMDBmcz
J1cGRnNmw4YTRoaiJ9.TKyh0iv0a_FSkhRakSeAyA"

    request({ url: geocodeURL, json: true }, (error, response) => {
        // ! bursa yerine saçma sapan bir şeyler girersek response'un features dizisi boş
        // gelmektedir.
        if (error) {
            callback("Geocoding servisine bağlanamadı", undefined)
        } else if (response.body.features.length == 0) {
            callback("Belirttiğiniz konum bulunamadı", undefined)
        } else {
            const longitude = response.body.features[0].center[0]
            const latitude = response.body.features[0].center[1]
            const location = response.body.features[0].place_name

            callback(undefined, {
                longitude: longitude,
                latitude: latitude,
                location: location
            })
        }
    })
}

geocode("Bursa", (error, data) => {
    // callback function body goes here
    if (error) {
        console.log("Error : ", error)
        return
    }
    console.log("Data : ", data)
})

```

geocodeURL içinde encodeURL kullanmamızın sebebi bazen kullanıcı address kısmını yazarken boşluk da koyarsa vs diye onları ayırtırmaktır.

İf in ilk kısmında ki error servise bağlanamama errorudur.

İf else kısmında ki sorun Bursa yerine saçma sapan şeyler girersek bulamama sorunudur.

Else kısmı ise başarılı ve response dan gelen verileri alıp callback'e vermiş durumdayız.

Şimdi bu geocode.js dosyasını kullanabilmek için export etmeliyiz ve postman'ı da yüklemeliyiz bu dosyamıza.

Son hali bu şekildedir dosyamızın .

```

Section5-6 - geocode.js

const request = require("postman-request")

const geocode = (address, callback) => {
    // function body goes here
    const geocodeURL =
        "https://api.mapbox.com/geocoding/v5/mapbox.places/" +
        encodeURI(address) +
        ".json?access_token=pk.eyJ1IjoiZGV2ZWxvcG VyYnVYWTndWwiLCJhIjoiY2x0emlyNThqMDBmczJ1cGRnNm w4YTRoaiJ9.TKyh0iv0a_FSkhRakSeAyA"

    request({ url: geocodeURL, json: true }, (error, response) => {
        // ! bursa yerine saçıma sapan bir şeyle girersek response'un features dizisi boş gelmektedir.
        if (error) {
            callback("Geocoding servisine bağlanamadı", undefined)
        } else if (response.body.features.length == 0) {
            callback("Belirttiğiniz konum bulunamadı", undefined)
        } else {
            const longitude = response.body.features[0].center[0]
            const latitude = response.body.features[0].center[1]
            const location = response.body.features[0].place_name

            callback(undefined, {
                longitude: longitude,
                latitude: latitude,
                location: location
            })
        }
    })
}

geocode("Bursa", (error, data) => {
    // callback function body goes here
    if (error) {
        console.log("Error : ", error)
        return
    }
    console.log("Data : ", data)
})

module.exports = geocode

```

Deneyelim şimdi.

```

index.js  geocode.js U
utils > geocode.js > ...
  3  const geocode = (address, callback) => {
  10  request({ url: geocodeURL, json: true }, (error, response) => {
  11    if (error) {
  12      callback("Geocoding servisine bağlanamadı", undefined)
  13    } else if (response.body.features.length == 0) {
  14      callback("Belirttiğiniz konum bulunamadı", undefined)
  15    } else {
  16      const longitude = response.body.features[0].center[0]
  17      const latitude = response.body.features[0].center[1]
  18      const location = response.body.features[0].place_name
  19
  20      callback(undefined, {
  21        longitude: longitude,
  22        latitude: latitude,
  23        location: location
  24      })
  25    }
  26  })
  27
  28  geocode("Bursa", (error, data) => {
  29    // callback function body goes here
  30    if (error) {
  31      console.log("Error : ", error)
  32      return
  33    }
  34    console.log("Data : ", data)
  35  })
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/utils/geocode.js"
• (base) iamburakgul@Buraks-MacBook-Air Section5-6 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/utils/geocode.js"
Data :
{
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
• (base) iamburakgul@Buraks-MacBook-Air Section5-6 %

```

Ln 39, Col 1 Spaces: 4 UTF-8 LF () JavaScript Quokka Prettier

Verilerimiz geldi.

Şimdi de weatherstack apisi için forecast adında bir dosya oluşturup orada işlemlerimizi yapalım.

Enlem ve boylam bilgileri parametre olarak gelmeli fonksiyona.

```
Section5-6 - index.js

const url =
  "http://api.weatherstack.com/current?access_key=6a062fa08038e848eb2f9ad4b2ce7c10&query=37.8267,-122.4233&units=f"

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 "
    )
  }
})
```

index.js deki bu weatherstack apisine enlem boylam bilgilerini verip sonuç aldığımız işi forecast dosyasında yapacağız.

Genel yapımız bu olmalı.

#### Section5-6 - forecast.js

```
const request = require("postman-request")

const forecast = (longitude, latitude, callback) => {
    // function body goes here
}

forecast(37.8267, -122.4233, (error, data) => {
    // callback function body goes here
})

module.exports = forecast
```

Forecast fonksiyonuna boylam ve enlem bilgilerini göndereceğiz. Forecast fonksiyonu request atacak weatherstack api sine. Response dan error dönerse callbackin errorunu kendimiz vereceğiz ve data sı undefined olacak. Error dönmezse datayı callbacka vereceğiz ve error u undefined olarak vereceğiz.

url kısmında ise boylam ve enlemi ekleyeceğiz.

```
Section5-6 - forecast.js

const request = require("postman-request")
const forecast = (longitude, latitude, callback) => {
  const url =
    "http://api.weatherstack.com/current?access_key=6a062fa08038e848eb2f9ad4b2ce7c10&query=" +
    longitude +
    "," +
    latitude

  request({ url: url, json: true }, (error, response) => {
    // const data = JSON.parse(response.body)
    // console.log(data.current) // * json : true yazmadığımız kısımda böyle gelir ve biz veriyi
    JSON formatına çeviririz.
    if (error) {
      callback("Hava durumu servisine bağlantı kurulamadı", undefined)
    } else if (response.body.error) {
      callback("Girilen konum bilgisi bulunamadı", undefined)
    } else {
      callback(
        undefined,
        "Hava sıcaklığı : " +
        response.body.current.temperature +
        " Hissedilen : " +
        response.body.current.feelslike
      )
    }
  })
}

forecast(37.8267, -122.4233, (error, data) => {
  // callback function body goes here
  if (error) {
    console.log("Error : ", error)
    return
  }
  console.log("Data : ", data)
})

module.exports = forecast
```

Dosyamız bu hale gelecektir.

Şimdi çalıştıralım bu dosyayı ve sonucu görelim.

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'forecast.js' selected. The main editor window displays the following code:

```
utils > forecast.js > forecast > url
1 const request = require("postman-request")
2 const forecast = (longitude, latitude, callback) => {
3   const url =
4     "http://api.weatherstack.com/current?access_key=6a062fa08038e848eb2f9ad4b2ce7c10&query=" +
5     longitude +
6     "," +
7     latitude
8
9   request({ url: url, json: true }, (error, response) => {
10     // const data = JSON.parse(response.body)
11     // console.log(data.current) // * json : true yazmadığımız kısımda böyle gelir ve biz veriyi JSON formatına çeviri
12   })
13 }
14
15 module.exports = forecast
```

The terminal at the bottom shows the command 'node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/utils/forecast.js"' being run, followed by the output: 'Data : Hava sıcaklığı : 10 Hissedilen : 7'. The status bar indicates the file is a JavaScript file.

Şimdi bu da tamam. Bu dosyalarımızı index.js e ekleyip orada kullanalım.

Öncelikle geocode ve forecast içinde fonksiyonu kullandığımız yerleri silelim.

Önce geocode sonra forecast fonksiyonlarını kullanmamız lazım ama callback chaining yapmamız lazım. Sırası ile yazarsak önce hangi fonksiyonun çalışacağını bilemeyeziz.

Burada geocode fonksiyonunun çalışmış halini görüyoruz.

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'index.js' selected. The main editor window displays the following code:

```
const request = require("postman-request")
const geocode = require("./utils/geocode")
const forecast = require("./utils/forecast")

geocode("Bursa", (error, data) => {
  if (error) {
    console.log("Error :", error)
    return
  }
  console.log("Data :", data)
})
```

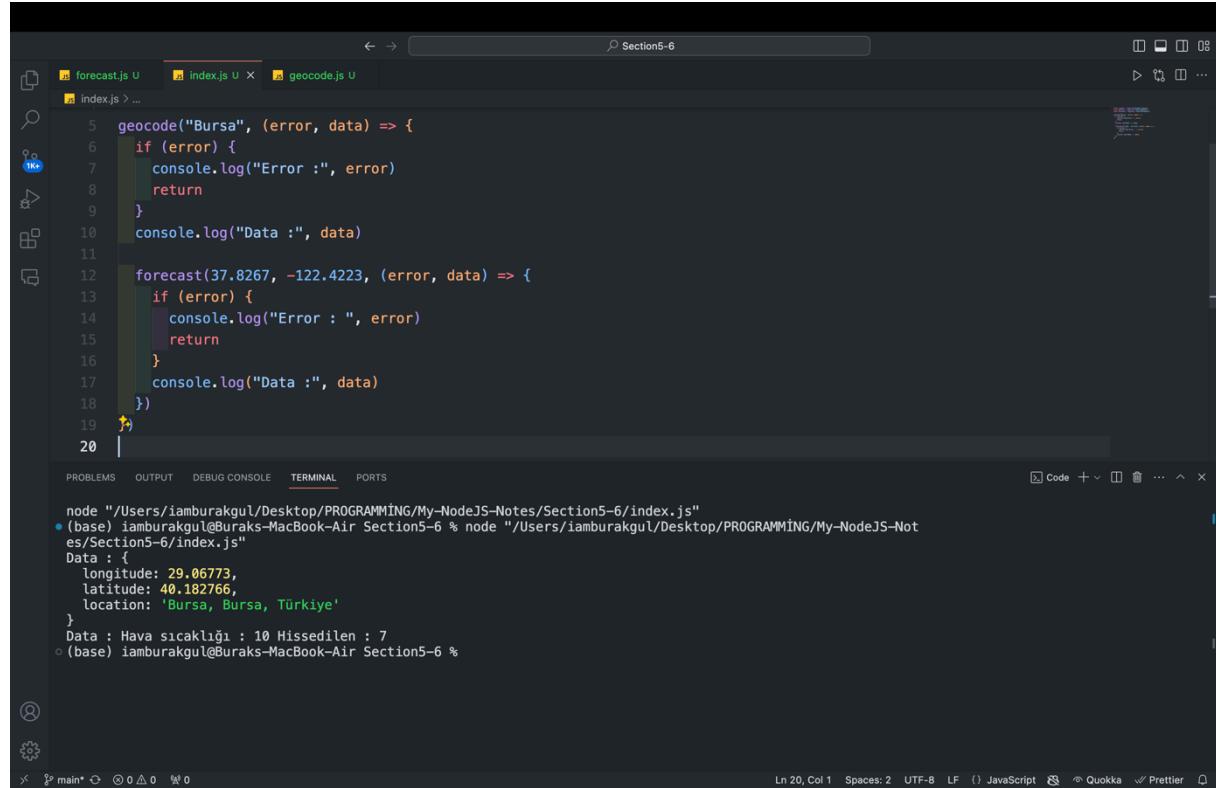
The terminal at the bottom shows the command 'node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/index.js"' being run, followed by the output: 'Data : { longitude: 29.06773, latitude: 40.182766, location: 'Bursa, Bursa, Türkiye' }'. The status bar indicates the file is a JavaScript file.

```
Section5-6 - index.js

geocode("Bursa", (error, data) => {
  if (error) {
    console.log("Error :", error)
    return
  }
  console.log("Data :", data)

  forecast(37.8267, -122.4223, (error, data) => {
    if (error) {
      console.log("Error : ", error)
      return
    }
    console.log("Data :", data)
  })
})
```

Şimdi burayı da çalıştıralım.



```
index.js > ...
5  geocode("Bursa", (error, data) => {
6    if (error) {
7      console.log("Error :", error)
8      return
9    }
10   console.log("Data :", data)
11
12   forecast(37.8267, -122.4223, (error, data) => {
13     if (error) {
14       console.log("Error : ", error)
15       return
16     }
17     console.log("Data :", data)
18   })
19
20 |
```

```
node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/index.js"
(base) iamburakgul@Buraks-MacBook-Air Section5-6 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section5-6/index.js"
Data : {
  longitude: 29.06773,
  latitude: 40.182766,
  location: 'Bursa, Bursa, Türkiye'
}
Data : Hava sıcaklığı : 10 Hissedilen : 7
(base) iamburakgul@Buraks-MacBook-Air Section5-6 %
```

Verilerimiz harika ama statik durumdan dinamik duruma getirelim hemen.Ek olarak da verimizi daha düzgün yazdıralım.

```

Section5-6 - index.js

const request = require("postman-request")
const geocode = require("./utils/geocode")
const forecast = require("./utils/forecast")
const chalk = require("chalk")

geocode("Bursa", (error, data) => {
  if (error) {
    console.log("Error :", error)
    return
  }

  forecast(data.longitude, data.latitude, (forecastError, forecastData) => {
    if (forecastError) {
      console.log(chalk.red.inverse("Error : " + forecastError))
      return
    }
    console.log(chalk.green.inverse(data.location + forecastData))
  })
})

```

Index.js dosyamız bu hale getirelim ve console a renkli yazmak için chalk kullanalım.

Şimdi de çalıştırıp sonucunu görelim.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• (base) iamburakgul@Buraks-MacBook-Air Section5-6 % node index.js
Bursa, Bursa, Türkiye Hava sıcaklığı : 15 Hissedilen : 18
• (base) iamburakgul@Buraks-MacBook-Air Section5-6 %

```

Bir sorunumuz var Şehir ismi statik halde. Onu da terminalden alalım.

```
Section5-6 - index.js

const geocode = require("./utils/geocode")
const forecast = require("./utils/forecast")
const chalk = require("chalk")

const address = process.argv[2]

geocode(address, (error, data) => {
  if (error) {
    console.log("Error :", error)
    return
  }

  forecast(data.longitude, data.latitude, (forecastError, forecastData) => {
    if (forecastError) {
      console.log(chalk.red.inverse("Error : " + forecastError))
      return
    }
    console.log(chalk.green.inverse(data.location + forecastData))
  })
})
```

Çalıştırıralım ve sonuç görelim.

The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. At the top, there's a navigation bar with icons for back, forward, and search, followed by the title "Section5-6". Below the title is a tab bar with three tabs: "forecast.js U", "index.js U", and "geocode.js U".

The main area displays the "index.js" file content:

```
const address = process.argv[2]

geocode(address, (error, data) => {
  if (error) {
    console.log("Error :", error)
    return
  }

  forecast(data.longitude, data.latitude, (forecastError, forecastData) => {
    if (forecastError) {
      console.log(chalk.red.inverse("Error : " + forecastError))
      return
    }
    console.log(chalk.green.inverse(data.location + forecastData))
  })
})


```

Below the code editor is a terminal window showing the output of running the script:

```
(base) iamburakgul@Buraks-MacBook-Air Section5-6 % node index.js Sivas
Sivas, Sivas, Türkiye Hava sıcaklığı : 17 Hissedilen : 17
(base) iamburakgul@Buraks-MacBook-Air Section5-6 %
```

The bottom status bar indicates the file is 16 lines long, has 25 columns, uses 2 spaces per tab, is in UTF-8 encoding, and is a JavaScript file. It also shows icons for Quokka and Prettier.