

Gizem Avcı 21360859071

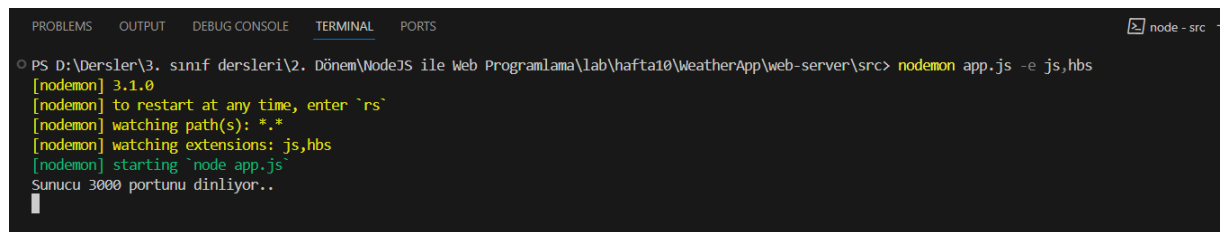
## NodeJs ile Web Programlama Hafta-10 Raporu

---

Bugünkü dersimizde hava durumu sayfamızı API alacak şekilde değiştirmeyi amaçladık. Öncelikle deneme yapmak için boş json dosyası gönderen bir url oluşturduk. Bunu da app.js dosyasına şu şekilde yazdık:

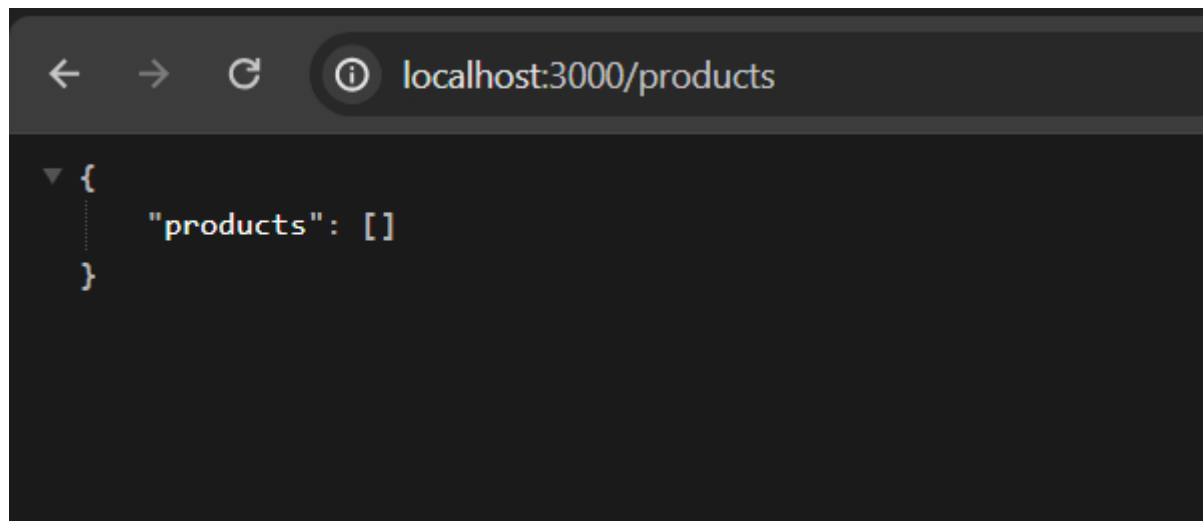
```
app.get('/products', (req, res) => {  
  res.send({  
    products: []  
  })  
})
```

Dosyayı kaydettikten sonra nodemon ile çalıştırdık.



```
PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src> nodemon app.js -e js,hbs  
[nodemon] 3.1.0  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,hbs  
[nodemon] starting `node app.js`  
Sunucu 3000 portunu dinliyor..  
|
```

Ardından sayfamıza girdik.



Çıktı olarak boş bir array aldık. Sonuç statikti ve hiçbir şekilde değişmiyordu. Belli bir sonuç döndürmek veya aramak istiyorsak sorgu dizisi kullanmalıyız. Sorgu dizisi, ?key=value şeklinde URL 'nin sonunda gelir, birden fazla key-value çifti geçirmek için &(ampersand) kullanırız.

Deneme olarak ilk query kısmını yazdırdık.

```
app.get('/products', (req, res) => {  
  console.log(req.query)  
  res.send({  
    products: []  
  })  
})
```

Çıktı olarak da şunu aldık:

```
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`  
Sunucu 3000 portunu dinliyor..  
{  
}  
[]
```

Sayfayı her güncellediğimizde konsolde boş bir array geldi. Hala bir şey değişmedi. Belirli bir sorgu dizesine erişmek için kodu şu şekilde güncelledik:

```
app.get('/products', (req, res) => {  
  console.log(req.query.search)  
  res.send({  
    products: []  
  })  
})
```

Kodu yine güncelledik ve sayfamıza girdik. Hala bir değişiklik olmamakla beraber konsolda bu sefer de şu çıktıyı aldık:

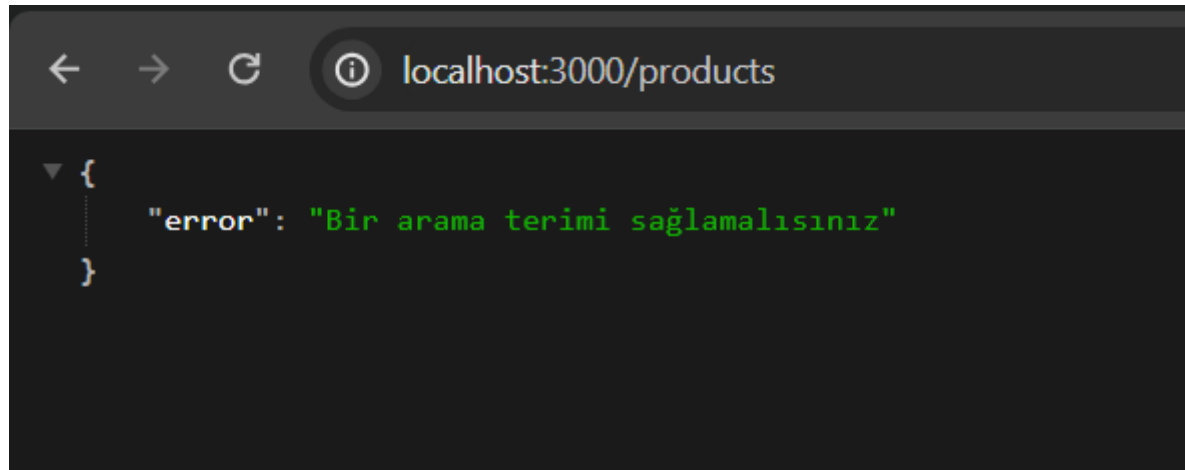
```
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`  
Sunucu 3000 portunu dinliyor..  
undefined  
undefined  
[]
```

Bu sefer de sonuç undefined çıktı.

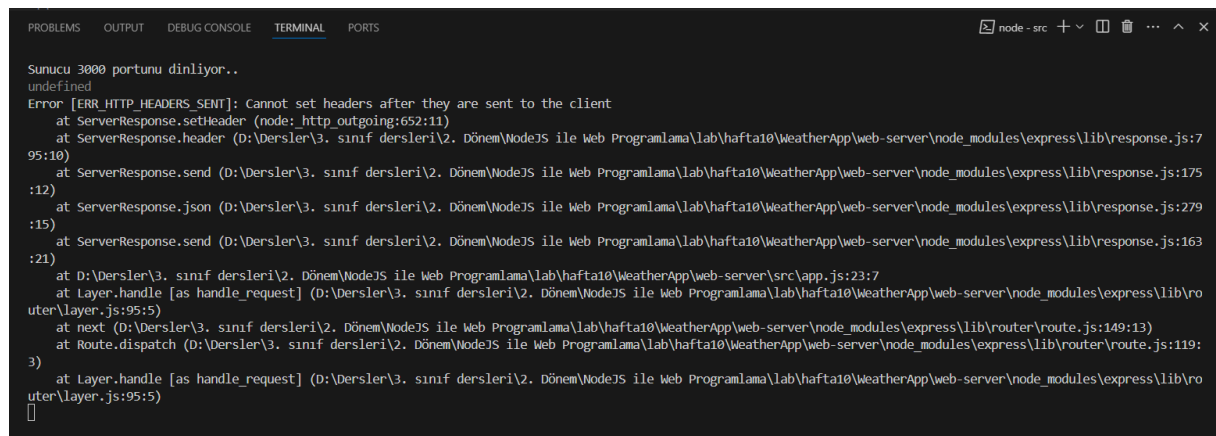
Sorgu dizilerini zorunlu olmasını sağladık. Aramayı zorunlu, derecelendirmeyi ise isteğe bağlı yaptık:

```
app.get('/products', (req, res) => {
  if (!req.query.search) {
    res.send({
      error: 'Bir arama terimi sağlamalısınız'
    })
  }
  console.log(req.query.search)
  res.send({
    products: []
  })
})
```

Bu şekilde hata girdiğimizde ise sayfamız:



Konsolda ise:



Hata aldık. Bu sayede arama yapmama durumunda sayfamızdan uyarı aldık.

Bu çıktıdan da görüldüğü üzere tek bir istek için iki kez yanıt gönderilmez. Bu nedenle geri dönmeli veya else koşulu kullanmalıyız. Fakat genellikle return kullanır.

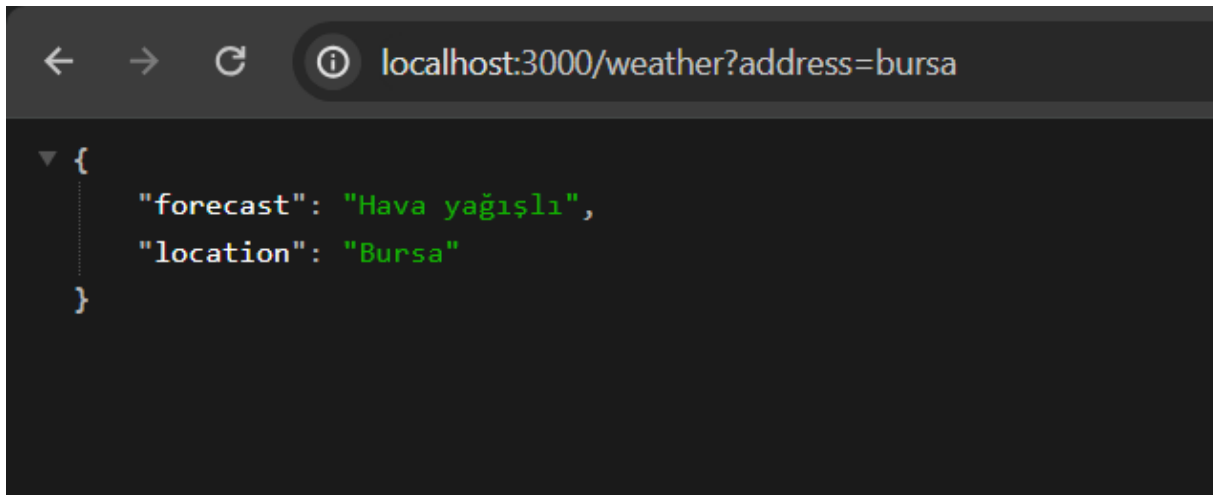
```
app.get('/products', (req, res) => {  
  if (!req.query.search) {  
    return res.send({  
      error: 'Bir arama terimi sağlamalısınız'  
    })  
  }  
  console.log(req.query.search)  
  res.send({  
    products: []  
  })  
})
```

Return kullandıktan sonra sayfada hata mesajı alsak da konsolda sıkıntı yaşamadık yani birden fazla istek gönderebildik.

Kodumuzun son halini bu şekilde bıraktık. Ardından hava durumunun son noktasını güncelledik. Bir adres kabul edip adres yoksa hata mesajı gönderme döndüreceğiz.

```
app.get('/weather', (req, res) => {  
  if (!req.query.address) {  
    return res.send({  
      error: 'Bir adres sağlamalısınız'  
    })  
  }  
  res.send({  
    forecast: 'Kar yağıyor',  
    location: 'Bursa',  
    address: req.query.address  
  })  
})
```

Bu güncelleme sonunda adres olmama durumunda return olarak hata mesajı döndürecek. Fakat konsolda herhangi bir hataya sebep olmayacak.



Önceden hazırladığımız kodumuzu (geocode) buraya yapıştırdık. Tüm utils klasörünü web-server/src nin altına yapıştırdık. Fakat kullandığımız bazı npm modülleri dahil olmadığından nodemon u sonlandırıp onları da dahil ettik.

```
PS D:\dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src> npm i request
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
added 43 packages, and audited 150 packages in 2s
18 packages are looking for funding
  run `npm fund` for details
2 moderate severity vulnerabilities
Some issues need review, and may require choosing
a different dependency.
Run `npm audit` for details.
PS D:\dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src>
```

Request modülünü dahil ettik.

Ardından tekrardan nodemon kullanarak dosyamızı çalıştırdık ve forecast ve geocode dosyalarını app.js dosyamıza dahil ettik.

```
const geocode = require('./utils/geocode')
const forecast = require('./utils/forecast')
```

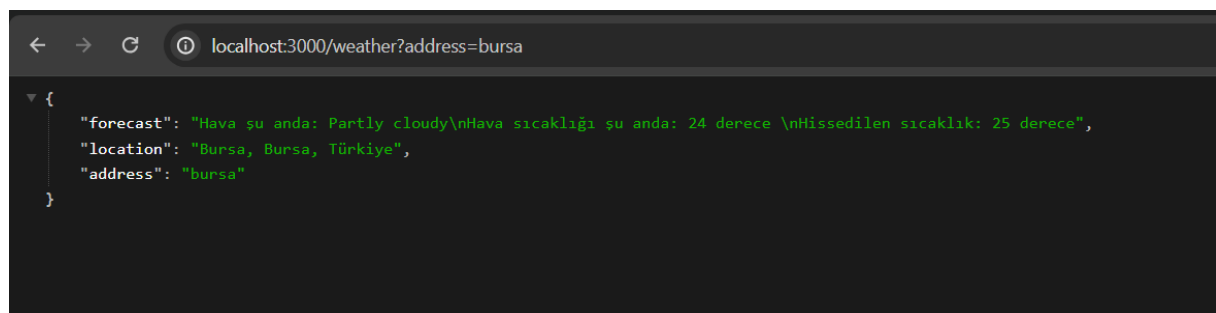
Hava durumu URL'sini güncelledik. Geocode iki argüman aldı. Eski yazdığımız res.send kısmını yorum satırına aldık. Artık oradaki bilgileri el ile yazmak yerine geocode kodu içerisinde API den çektiğimiz gerçek bilgiler ile doldurduk. O kısımda lokasyon, adres ve forecast bilgileri yer aldı.

```

app.get("/weather", (req, res) => {
  if (!req.query.address) {
    return res.send({
      error: "Bir adres sağlamalısınız",
    });
  }
  geocode(req.query.address, (error, { latitude, longitude, location }) => {
    if (error) {
      return res.send({ error });
    }
    forecast(latitude, longitude, (error, forecastData) => {
      if (error) {
        return res.send({ error });
      }
      res.send({
        forecast: forecastData,
        location,
        address: req.query.address,
      });
    });
  });
  //res.send({
  // forecast: "Karlı",
  // location: "Bursa",
  // address: req.query.address
  //})
});

```

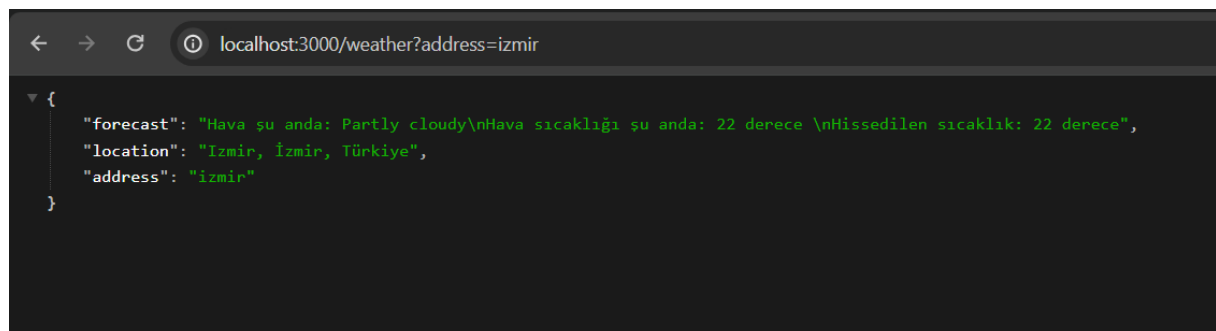
Kodumuzu bursa ve izmir ile denedik ve çıktı şu şekilde çıktı:



```

{
  "forecast": "Hava şu anda: Partly cloudy\nHava sıcaklığı şu anda: 24 derece \nHissedilen sıcaklık: 25 derece",
  "location": "Bursa, Bursa, Türkiye",
  "address": "bursa"
}

```



```

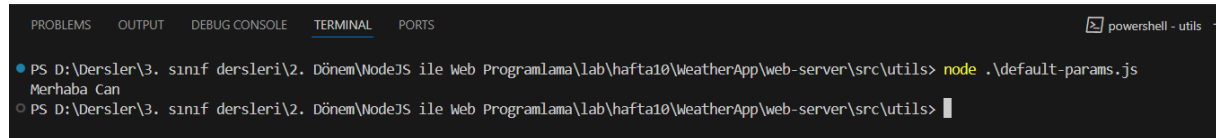
{
  "forecast": "Hava şu anda: Partly cloudy\nHava sıcaklığı şu anda: 22 derece \nHissedilen sıcaklık: 22 derece",
  "location": "Izmir, İzmir, Türkiye",
  "address": "izmir"
}

```

Ardından test için default-params.js adında dosya oluşturduk. Yeni dosya içerisinde:

```
const greeter = (name) => {  
  console.log("Merhaba " + name);  
};  
greeter("Can");
```

yazdık. Kodu kaydedip çalıştırdık ve:

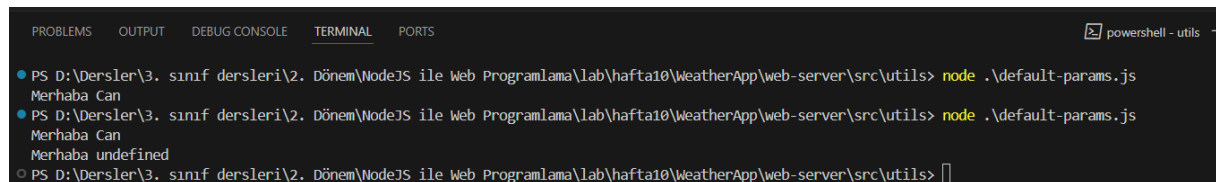


A terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing a PowerShell prompt. The command executed is `node .\default-params.js`, which outputs `Merhaba Can`. The prompt is now `PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src\utils>`.

Ardından argümansız bir şekilde çağırıp çalıştırdık.

```
const greeter = (name) => {  
  console.log("Merhaba " + name);  
};  
greeter("Can");  
  
greeter();
```

Bu kodun sonucu da şu çıktı:

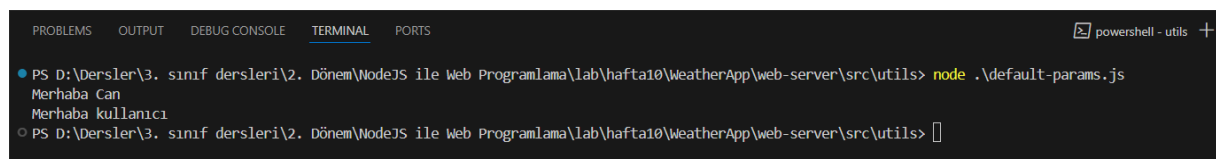


A terminal window showing the execution of `node .\default-params.js` twice. The first execution outputs `Merhaba Can`. The second execution outputs `Merhaba undefined`. The prompt is now `PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src\utils>`.

“undefined”. Eğer argüman vermezsek default olarak argüman değeri undefined çıkar. Varsayılan değeri kullanıcı olarak verirsek:

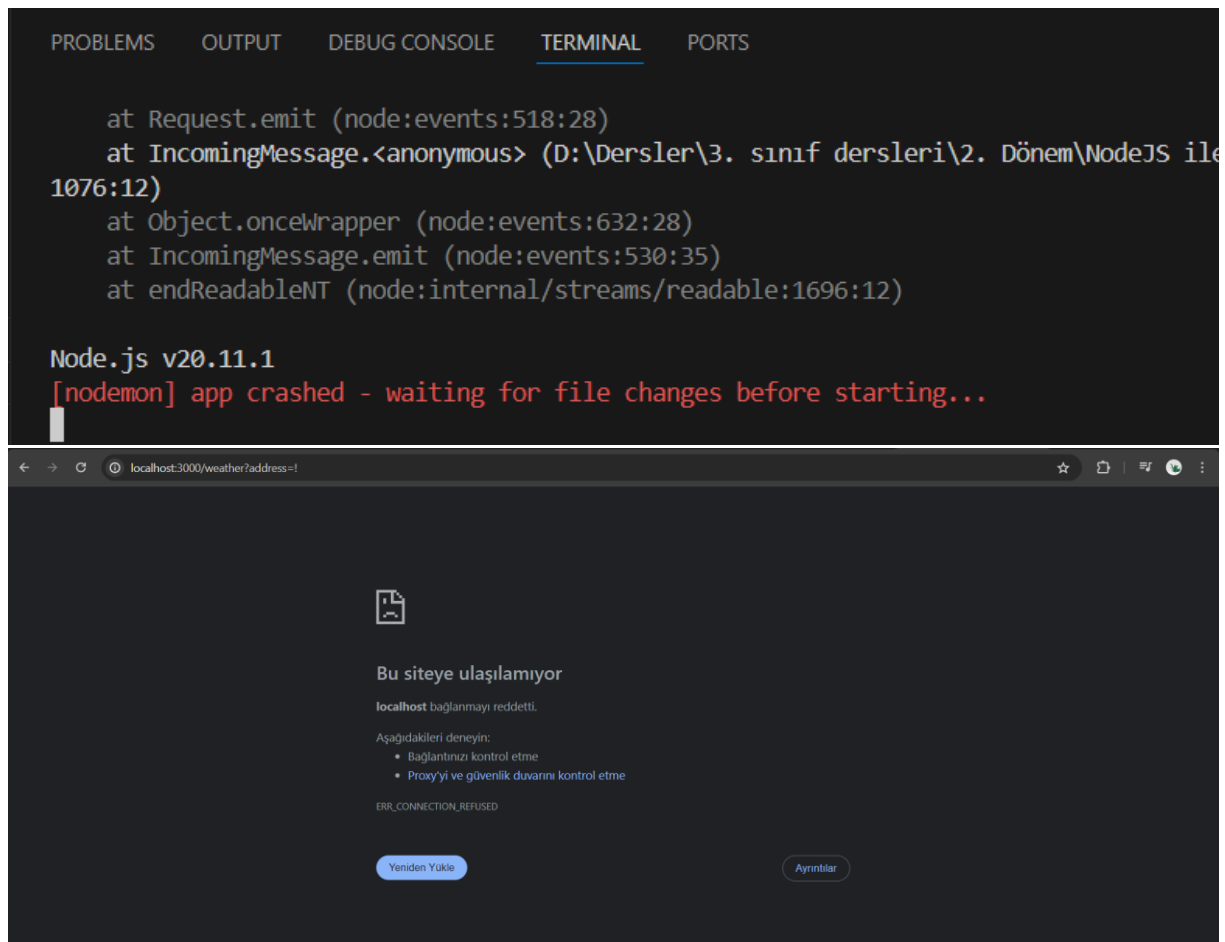
```
const greeter = (name = "kullanıcı", age) => {  
  console.log("Merhaba " + name);  
};  
greeter("Can");  
greeter();
```

Argümansız çağırma durumunda sonuç şu hali alır:



A terminal window showing the execution of `node .\default-params.js` twice. The first execution outputs `Merhaba Can`. The second execution outputs `Merhaba kullanıcı`. The prompt is now `PS D:\Dersler\3. sınıf dersleri\2. Dönem\NodeJS ile Web Programlama\lab\hafta10\WeatherApp\web-server\src\utils>`.

Ardından nodemonu tekrar çalıştırdık ve localhost:3000/weather?address=! sayfasına girdik.



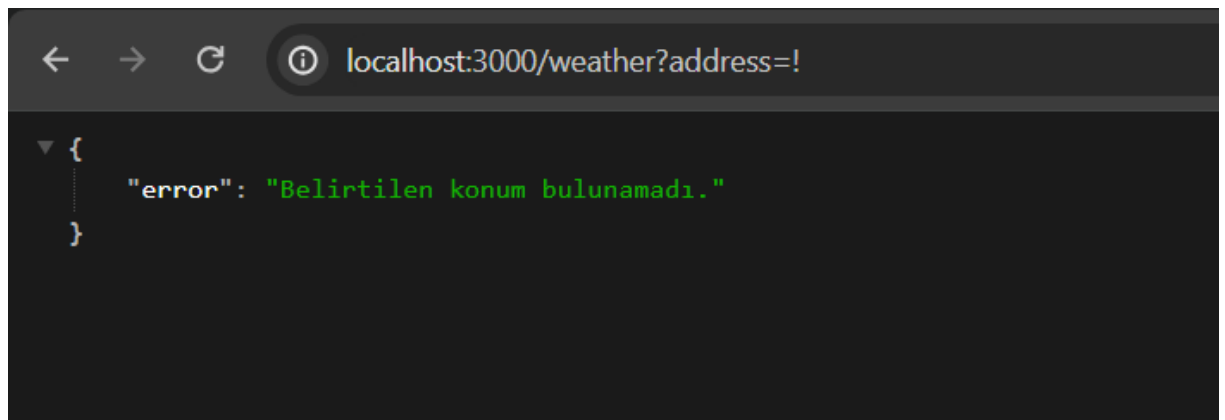
Ve sonuç bu oldu. Hata mesajı aldık ve sunucu çöktü.

Buradaki sorun geocode çağrısındaydı. Veriler için değer döndürmedik ve hata döndürdü. Nesneyi yapılandırmamız gerekir. Bunun içinde varsayılan bir değer sağlamamız gerekiyor.

Bunun içinde şunu yaptık:

```
geocode(req.query.address, (error, { latitude, longitude, location } = {})) => {
```

Bu sayede değer almama durumunda boş array'e eşitlenecek.





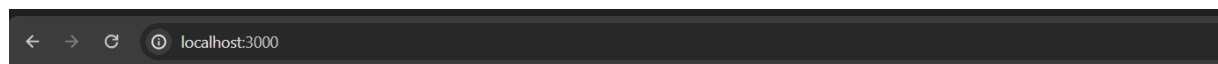
Ve sunucu çökmeyecek artı olarak da konsol üzerinden hata almayacağız. Yalnızca sayfamızda girilen değerin hatalı olduğuna dair bilgi mesajı alacağız.

Bu kısımdan sonra önyüzü güncelleme kısmına geçtik. Bir form sağladık.

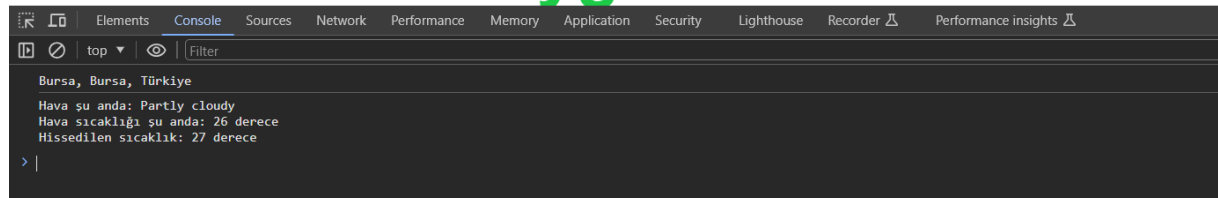
Js klasörü içerisindeki app.js dosyamıza şu kodu yazdık:

```
fetch("http://localhost:3000/weather?address=bursa").then((response) => {  
  response.json().then((data) => {  
    if (data.error) {  
      console.log(data.error);  
    } else {  
      console.log(data.location);  
      console.log(data.forecast);  
    }  
  });  
});
```

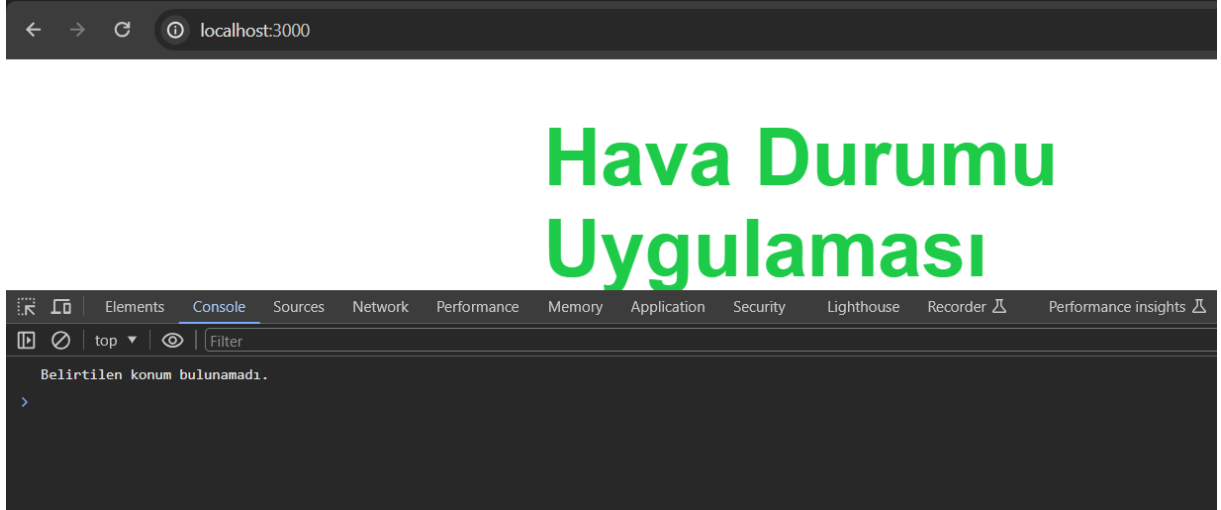
Ardından kodu çalıştırdık ve anasayfamızda konsola baktığımızda bursa nın hava durumu bilgisini almış olduk.



## Hava Durumu Uygulaması



Kod içerisinde adres bilgisini bursa yazmak yerine ! yazınca da çıktımız şu oldu



Ardından arama formunu oluşturmak için `inex.hbs` dosyamıza geçtik.

```
<html lang="en">
  <head>
    <link rel="stylesheet" href="./css/styles.css" />
    <script src="/js/app.js"></script>
    <link rel="icon" href="/img/indir.png">
    <title>Document</title>
  </head>
  <body>

    <div class="main-content">
      <h1>{{>header}}</h1>
      <form>
        <input placeholder="Lokasyon">
        <button>Ara</button>
      </form>
    </div>
    {{>footer}}
  </body>
</html>
```

Div bloğu arasına form yapısı oluşturduk. Ara butonu ve Lokasyon bilgisini soran bir arama butonu. Görüntüsü ise şu şekilde:



Ardından js içerisindeki app.js dosyasına gittik ve formdan alınan konumu aldık.

```
const weatherForm = document.querySelector("form");
```

Butona bir olay dinleyici ekledik.

```
weatherForm.addEventListener('submit', () => {  
  console.log('test')  
})
```

Bunun testini yaparken butona bastığımızda sayfa yenilenir. Bunun önüne geçmek için:

```
weatherForm.addEventListener("submit", (e) => {  
  e.preventDefault();  
  console.log("test");  
});
```

Kodunu yazdık. Ardından arama terimini aldık:

```
const weatherForm = document.querySelector("form");  
const search = document.querySelector("input");  
weatherForm.addEventListener("submit", (e) => {  
  e.preventDefault();  
  const location = search.value;  
  console.log(location);  
});
```

Fetch kodunu kestik ve olay dinleyicisine yapıştırdık.

```
const weatherForm = document.querySelector("form");
const search = document.querySelector("input");
weatherForm.addEventListener("submit", (e) => {
  e.preventDefault();
  const location = search.value;
  fetch("http://localhost:3000/weather?address=" + location).then(
    (response) => {
      response.json().then((data) => {
        if (data.error) {
          console.log(data.error);
        } else {
          console.log(data.location);
          console.log(data.forecast);
        }
      });
    }
  );
});
```

Ardından index.hbs de div içerisinde form altına şu kodu yazdık:

```
<p id="message-1"></p>
<p id="message-1"></p>
```

app.js dosyasına yeni bir querySelector ekledik.

Sınıflar . ile seçilir. Id'ler # ile seçilir.

Ardından js içerisindeki app.js dosyasına:

```
const messageOne = document.querySelector("#message-1");
const messageTwo = document.querySelector("#message-2");
```

yazdık.

Son olarak buton için stil ekledik:

```
input {
  border: 1px solid #cccccc;
  padding: 8px;
}
button {
  cursor: pointer;
  border: 1px solid #888888;
  background: #888888;
  color: white;
  padding: 8px;
}
```

Artık buton görüntüsü ve kodun son hali şu şekilde oldu:

# Hava Durumu Uygulaması

[Hava Durumu](#) [Hakkında](#) [Yardım](#)

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights

top Filter

Bursa, Bursa, Türkiye

Hava şu anda: Partly cloudy  
Hava sıcaklığı şu anda: 26 derece  
Hissedilen sıcaklık: 27 derece

# Hava Durumu Uygulaması

[Hava Durumu](#) [Hakkında](#) [Yardım](#)