

Node Js Ödevi

7th Week

Farhan Ahmad

20360859096

Geçen hafta **API'da oluşan Hatalar, Geri Çağırma Fonksiyonu** gibi konuları incelemiştik. Bu hafta ise **Object Property Shorthand, Destructuring ve Expressjs** gibi konuları ele alacağız.

Object Property Shorthand ve Destructuring:

Bu rapor JavaScript'te Object Property Shorthand ve Destructuring konularını anlatmaktır. Bu konseptler, JavaScript kodunu daha kısa ve okunabilir hale getirmek için kullanılan önemli tekniklerdir.

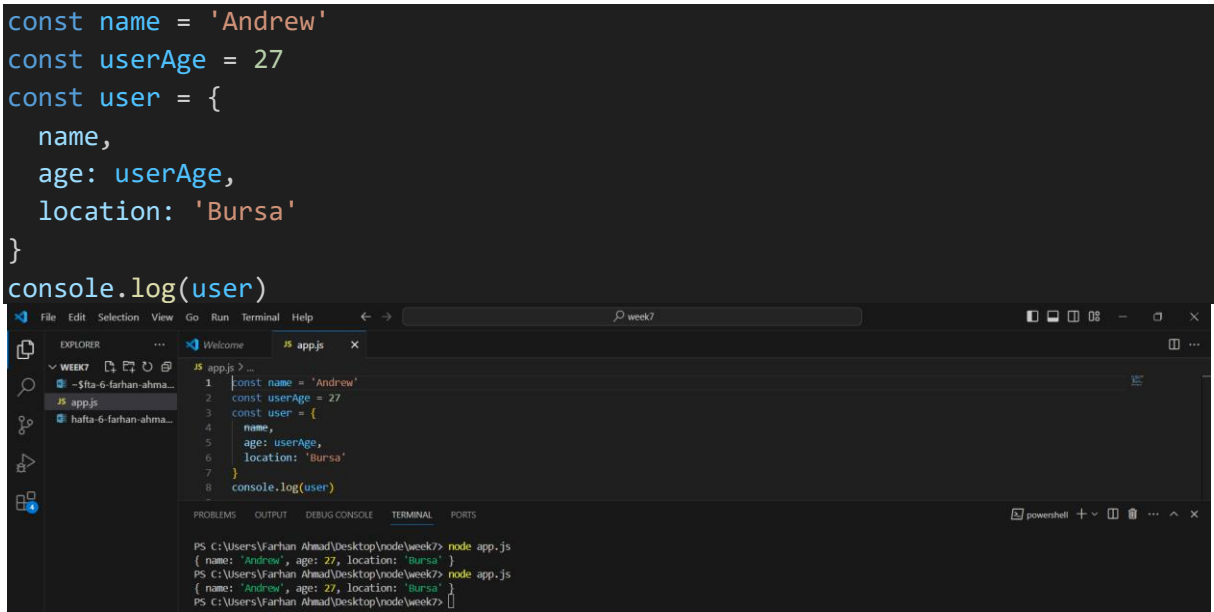
Object Property Shorthand:

Object Property Shorthand, bir nesne oluştururken değişkenlerin değerlerini kullanma kolaylığı sağlar. Örneğin:

```
const name = 'Andrew'  
const userAge = 27  
const user = {  
  name: name,  
  age: userAge,  
  location: 'Bursa'  
}  
console.log(user)
```



Yukarıdaki örnekte, name ve userAge değişkenlerini kullanarak bir kullanıcı nesnesi oluşturuyoruz. Ancak, aynı sonucu daha kısa bir şekilde elde edebiliriz:



Bu şekilde, değişkenin adıyla aynı isme sahip bir özellik oluşturabiliriz.

Object Destructuring:

Bir nesnenin içindeki özellikleri ayrı ayrı değişkenlere atamak için kullanılan bir JavaScript özelliğidir. Bu sayede, nesne içindeki belirli özelliklere kolayca erişebiliriz.

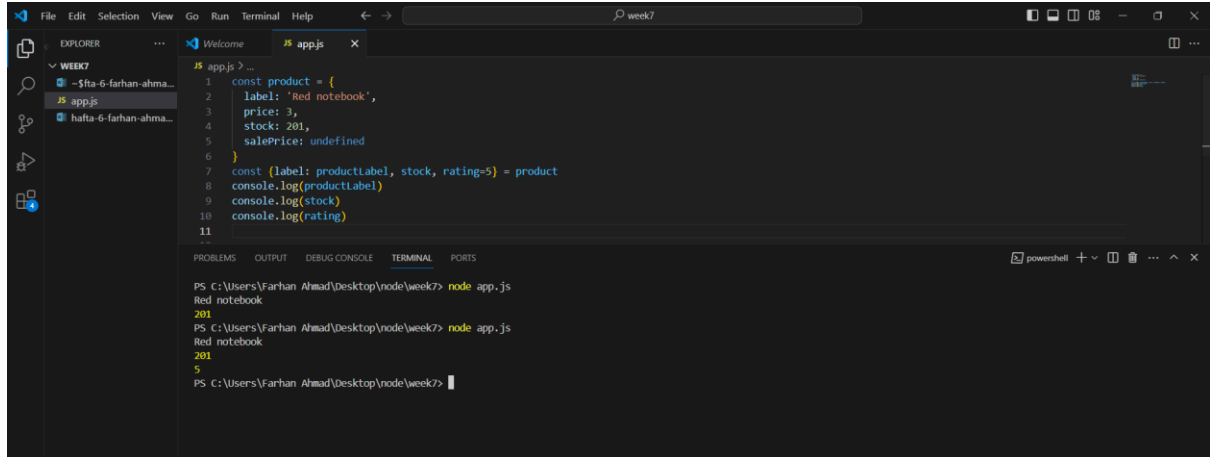
```
const product = {  
  label: 'Red notebook',  
  price: 3,  
  stock: 201,  
  salePrice: undefined  
}  
  
const {label, stock} = product  
console.log(label)  
console.log(stock)
```



Yukarıdaki örnekte, product nesnesinin label ve stock özelliklerini ayrı değişkenlere atıyoruz. Bu şekilde, nesnenin belirli özelliklerine kolayca erişebiliriz.

Destructuring sırasında, eğer nesnede olmayan bir özellik atamaya çalışırsak, o değişken undefined olur ancak kod çökmeyecektir.

Ayrıca, destructuring işlemi sırasında özellikleri yeniden adlandırabilir veya varsayılan bir değer atayabiliriz:



```
1 const product = {
2   label: 'Red notebook',
3   price: 3,
4   stock: 201,
5   salePrice: undefined
6 }
7 const {label: productLabel, stock, rating=5} = product
8 console.log(productLabel)
9 console.log(stock)
10 console.log(rating)
11
```

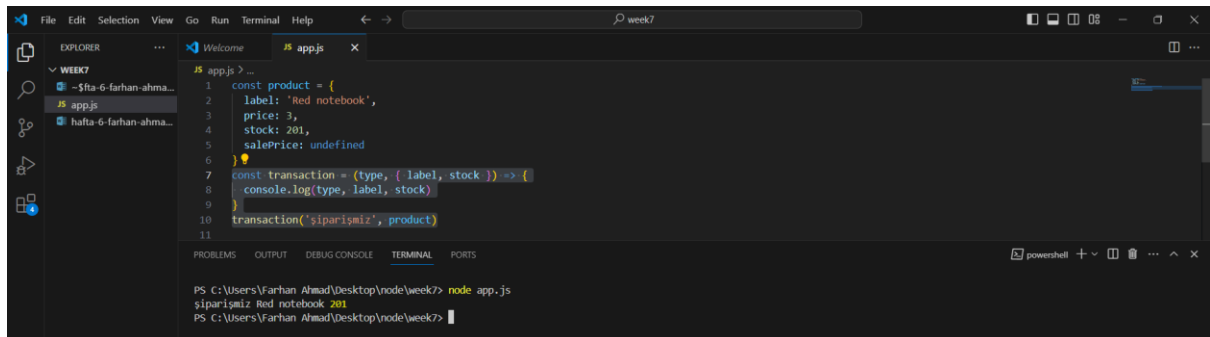
```
PS C:\Users\Farhan Ahmad\Desktop\node\week7> node app.js
Red notebook
201
PS C:\Users\Farhan Ahmad\Desktop\node\week7> node app.js
Red notebook
201
5
PS C:\Users\Farhan Ahmad\Desktop\node\week7>
```

Yukarıdaki örnekte, label özelliğini productLabel olarak yeniden adlandırdık ve rating özelliği için varsayılan bir değer belirledik.

Fonksiyonlarda Destructuring:

Bir fonksiyonun parametrelerini daha temiz ve okunabilir hale getirmek için kullanılan bir tekniktir. Bir nesnenin belirli özelliklerine fonksiyon parametreleri olarak doğrudan erişim sağlar. Bu, fonksiyonun içinde nesnenin özelliklerine tekrar tekrar erişme ihtiyacını ortadan kaldırarak kod tekrarını azaltır.

```
const transaction = (type, { label, stock }) => {
  console.log(type, label, stock)
}
transaction('siparişimiz', product)
```



```
1 const product = {
2   label: 'Red notebook',
3   price: 3,
4   stock: 201,
5   salePrice: undefined
6 }
7
8 const transaction = (type, { label, stock }) => {
9   console.log(type, label, stock)
10 }
11 transaction('siparişimiz', product)
```

```
PS C:\Users\Farhan Ahmad\Desktop\node\week7> node app.js
siparişimiz Red notebook 201
PS C:\Users\Farhan Ahmad\Desktop\node\week7>
```

Yukarıdaki örnekte, transaction fonksiyonu type ve bir ürün nesnesi alır ve bu nesnenin label ve stock özelliklerini kullanarak bir işlem gerçekleştirir.

Destructuring ve Property Shorthand Kullanımı:

app.js, forecast.js ve geocode.js dosyalarında Destructuring ve Property Shorthand kullanarak kodu daha temiz ve okunabilir hale getirmektir.

1. app.js dosyasında, sadece latitude, longitude ve location verilerini kullanıyoruz. Kodu şu şekilde değiştirin:

```
2. const request = require('request');
3.
4. const geocode = require('./geocode.js');
5. const forecast = require('./forecast.js');
6.
7. geocode('Bursa', (error, {latitude, longitude, location} = {}) ) => {
8.   if (error) {
9.     return console.log(error); // Eğer bir hata varsa, devam etmeyecek.
10.  }
11.  forecast(latitude, longitude, (error, forecastData) => {
12.    if (error) {
13.      return console.log(error);
14.    }
15.    console.log(location);
16.    console.log(forecastData);
17.  });
18. });
19.
```

Bu şekilde, geocode fonksiyonunun callback'inde doğrudan latitude, longitude ve location özelliklerine erişebiliriz.

- forecast.js dosyasında, URL ve yanıt gövdesini şu şekilde değiştirin:

```
• const axios = require('axios');
•
• const forecast = (latitude, longitude, callback) => {
•   const url =
•     'http://api.weatherstack.com/current?access_key=81c6957beda8a6484399ecabcfdd7eae&query=' + latitude + ',' + longitude + '&units=f';
•
•   axios.get(url)
•     .then(response => {
•       const data = response.data;
•       if (data.error) {
•         callback('Konum bulunamadı!', undefined);
•       } else {
•         callback(undefined, 'Hava şu anda: ' +
```

```

•         data.current.weather_descriptions[0] + ' Hava sıcaklığı şu
anda: ' +
•         data.current.temperature + ' derece ve hissedilen sıcaklık: '
+
•         data.current.feelslike + ' derece');
•     }
• })
• .catch(error => {
•     callback('Hava durumu servisine bağlanılamıyor!', undefined);
• });
• };
•
• module.exports = forecast;
•

```

Bu şekilde, request fonksiyonunun callback'inde doğrudan body özelliğine erişebiliriz.

- geocode.js dosyasında, URL ve yanıt gövdesini şu şekilde değiştirin:

```

• const axios = require('axios');
• address='Bursa';
• const geocode = (address, callback) => {
•     const url = 'https://api.mapbox.com/geocoding/v5/mapbox.places/' +
encodeURIComponent(address) +
•     '.json?access_token=pk.eyJ1IjoiYWhtYWQwOCIsImEiOiJjbHU0dDg3YWxaWw5Mmlu
eHhlenNkempzIn0.UdvbLgMJdWENZGpG1cSsog&limit=1';
•
•     axios.get(url)
•         .then(response => {
•             const data = response.data;
•             if (data.features.length === 0) {
•                 callback('Konum bulunamadı!', undefined);
•             } else {
•                 callback(undefined, {
•                     latitude: data.features[0].center[1],
•                     longitude: data.features[0].center[0],
•                     location: data.features[0].place_name
•                 });
•             }
•         })
•         .catch(error => {
•             callback('Hava durumu servisine bağlanılamıyor!', undefined);
•         });
•
• };
• console.log(geocode)
•

```

```
• module.exports = geocode;  
•
```

Bu şekilde, request fonksiyonunun callback'inde doğrudan body özelliğine erişebiliriz.

Destructuring ve Property Shorthand kullanarak JavaScript kodunu daha temiz, daha okunabilir ve daha etkili hale getirdik. Bu teknikler, özellikle bir nesnenin belirli özelliklerine erişmek ve bu özellikleri farklı değişkenlerde kullanmak istediğimizde oldukça kullanışlıdır. Bu şekilde, kod tekrarını azaltabilir ve kodumuzu daha modüler hale getirebiliriz.

ExpressJs

Express.js, Node.js tabanlı minimalist ve esnek bir web çerçevesidir. Temel HTTP sunucu işlevselliğini sağlayarak web uygulamaları oluşturmayı kolaylaştırır. Yönlendirme sistemi ile URL'lerin yönlendirilmesini ve middleware'ler ile HTTP istekleri ve yanıtlarının işlenmesini sağlar. Ayrıca, şablon motorları ve statik dosya sunumu gibi özelliklerle dinamik ve statik içerik sunabilir. Express.js, geniş topluluğu ve basit API'siyle web geliştirme sürecini hızlandırır ve esneklik sağlar.

Express Kurulumu ve Temel Sunucu Oluşturma:

Öncelikle, projenizin bulunduğu dizinde terminali açın ve şu komutları çalıştırarak Express ve nodemon paketlerini yükleyin:

npm init -y

npm i express nodemon

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Farhan Ahmad\Desktop\node\week7> npm init -y
Wrote to C:\Users\Farhan Ahmad\Desktop\node\week7\package.json:

{
  "name": "week7",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  },
  "devDependencies": {},
  "description": ""
}

PS C:\Users\Farhan Ahmad\Desktop\node\week7> npm i express

up to date, audited 65 packages in 2s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Farhan Ahmad\Desktop\node\week7> |
```

Daha sonra, src klasörü içinde app.js adında bir dosya oluşturun ve şu kodları yazın:

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello express!');
});

app.listen(3000, () => {
  console.log('Server is up on port 3000.');
```

```
File Edit Selection View Go Run Terminal Help
week7
EXPLORER
WEEK7
  node_modules
  - $ita-6-farhan-ahma...
  app.js
  forecast.js
  geocode.js
  hafta-6-farhan-ahma...
  map.js
  package-lock.json
  package.json
  weatherapp.js
  app.js > ...
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => {
5   res.send('Hello express!');
6 });
7
8 app.listen(3000, () => {
9   console.log('Server is up on port 3000.');
```

```
localhost:3000
localhost:3000
Hello express!
```


Bu kod, bir Express uygulaması oluşturur ve ana sayfada "Hello express!" metnini gösteren bir HTTP GET isteği yöneticisi ekler. `app.listen` metoduyla da sunucuyu belirtilen portta (burada 3000) dinlemeye başlar.

Yeni Sayfalar Eklemek:

Yeni sayfalar eklemek için, `app.get` metoduyla isteğe bağlı olarak ek sayfalar oluşturabilirsiniz:

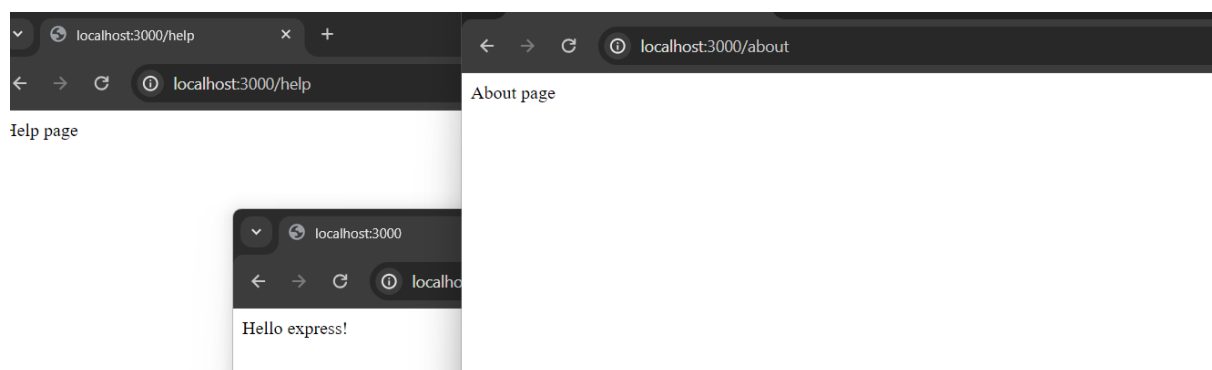
```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello express!');
});

app.get('/help', (req, res) => {
  res.send('Help page');
});

app.get('/about', (req, res) => {
  res.send('About page');
});

app.listen(3000, () => {
  console.log('Server is up on port 3000.');
```



Bu kod, `/help` ve `/about` URL'lerine yapılan isteklere karşılık gelen metinleri gönderir.

HTML ve JSON İçerikleri Sunmak:

Express, HTML veya JSON gibi farklı türlerde içerikleri sunabilir. Örneğin:

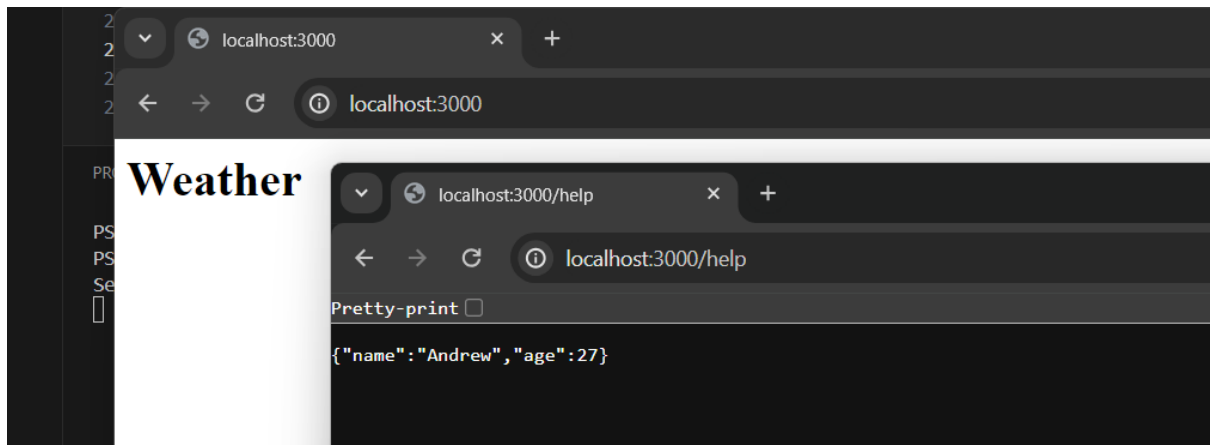
```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('<h1>Weather</h1>');
});

app.get('/help', (req, res) => {
  res.send({
    name: 'Andrew',
    age: 27
  });
});

app.listen(3000, () => {
  console.log('Server is up on port 3000.');
```

Bu kodlar, ana sayfada bir başlık gösterirken, /help sayfasında bir JSON nesnesi döndürür.



Statik Dosyaları Sunmak:

Express, `express.static` yöntemiyle statik dosyaları sunabilir.

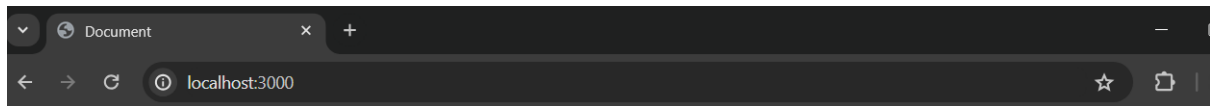
public Klasörü Oluşturma: İlk olarak, Express.js uygulamanızın ana dizininde "public" adında bir klasör oluşturun. Bu klasör, static dosyalarınızın (örneğin, HTML, CSS, JavaScript dosyaları) bulunduğu yer olacak.

index.html Dosyası Oluşturma: "public" klasörü içinde "index.html" adında bir dosya oluşturun. Bu dosya, tarayıcılarda varsayılan olarak açılacak ana sayfa dosyanız olacak.

Express Uygulamasında Static Dosyaları Sunmak: Express.js uygulamanızın ana dosyasını (genellikle "app.js" olarak adlandırılır) açın ve aşağıdaki gibi bir yapı ekleyin:



```
1 const express = require('express');
2 const app = express();
3 const path = require('path');
4 const publicDirectoryPath = path.join(__dirname, 'public');
5 app.use(express.static(publicDirectoryPath));
6
7 app.listen(3000, () => {
8   console.log('Sunucu 3000 portunda çalışıyor');
9 });
```



Express ile Static Dosya Sunumu

Welcome