

BURSA TEKNİK ÜNİVERSİTESİ

**Mühendislik ve Doğa Bilimleri Fakültesi – Bilgisayar
Mühendisliği Bölümü**



BLM0470 NodeJS İle Web Programlama

2023-2024 Bahar Dönemi

4.Hafta Raporu

Berkan SERBES – 22360859353

İçindekiler

1.Arrow Fonksiyonlar	3
2.Argumantatif Komut Satırı Uygulaması Geliştirme	5
2.1 Projenin Oluşturulması ve Package.json Dosyasının Oluşturulması	5
2.2 Gerekli NPM Paketlerinin Yüklenmesi	6
2.3 Fonksiyonların Oluşturulması	6
2.3.1 Not Ekleme Fonksiyonu	7
2.3.2 Notları Getirme Fonksiyonu	8
2.3.3 Notların Bulunduğu JSON dosyasının oluşturulması	10
2.3.4 Notları Kaydetme Fonksiyonu	11
2.3.5 Notları Silme Fonksiyonu	12
2.3.6 Notları Listeleme Fonksiyonu	12
2.3.7 Ana Dosyanın (index.js) Oluşturulması	13
2.3.8 Programın Test Edilmesi	14
3. Hata Ayıklama (Debugging)	15
3.1 Uygulamamızdaki Hataları Debugger Yardımıyla Giderme	16
4. KAYNAKÇA	19

1.Arrow Fonksiyonlar

JavaScript'te Arrow Fonksiyonları, ECMAScript 6 (ES6) ile tanıtılan ve modern JavaScript kodlamasında sıklıkla kullanılan bir özelliktir. Geleneksel function ifadelerine bir alternatif olarak sunulan bu yapının temel özelliği, daha kısa ve daha okunabilir kod yazımını sağlamasıdır.

Bir arrow fonksiyonu tanımlamak için, geleneksel fonksiyon ifadesinde olduğu gibi function anahtar kelimesi yerine parametre listesinden sonra => işareti kullanılır.

Örneğin, aşağıdaki arrow fonksiyon kullanılmadan oluşturulan fonksiyon tanımını ele alalım:

```
JS index.js  X Search Hafta4_RaporKodlari — index.js - Hafta4_RaporKodlari
arrow-functions > JS index.js > [E] add
1  const add = function(num1, num2) {
2    return num1 + num2;
3  }
```

Bu fonksiyon, iki parametre alır ve onların toplamını döndürür. Aynı işlevi arrow fonksiyonu olarak tanımlarsak aşağıdaki görseldeki gibi tanımlarız.

```
JS index.js  X
arrow-functions > JS index.js > ...
1  const add = (num1, num2) => {
2    return num1 + num2;
3  }
4
5
```

Görüldüğü gibi, arrow fonksiyonu tanımlamak geleneksel fonksiyon tanımına göre daha az yazı gerektirir. Ayrıca, **tek bir ifade** döndüren arrow fonksiyonlarında süslü parantezler ({}) ve return ifadesi kullanmak gerekmez:

```
JS index.js  X
arrow-functions > JS index.js > ...
1  const multiplyByTwo = (num) => num * 2;
2
3
```

Ancak, birden fazla ifade döndürmek istenirse veya birden fazla satırdan oluşan bir gövdeye ihtiyaç duyulursa, süslü parantezler ve return ifadesi kullanılmalıdır:

```
JS index.js x
arrow-functions > JS index.js > ...
1 const getFullName = (name, lastName) => {
2   let modifyingName = name.slice(0,1).toUpperCase() + name.slice(1).toLowerCase();
3   let modifyingLastName = lastName.slice(0,1).toUpperCase() + lastName.slice(1).toLowerCase();
4
5   let fullName = `${modifyingName} ${modifyingLastName}`;
6
7   return fullName;
8 }
9
10 console.log(getFullName("bErkan", "sErbes"));
11
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta4_RaporKodları
arrow-functions> node index.js
Berkay Serbes
```

Şimdi genel olarak bir örnek yapalım arrow function ile ilgili.

```
arrow-challenge.js x
arrow-functions > arrow-challenge.js > ...
1 const tasks = {
2   tasks: [
3     {
4       text: "Grocery Shopping",
5       completed: true,
6     },
7     {
8       text: "Clean Yard",
9       completed: false,
10    },
11    {
12      text: "Film Course",
13      completed: false,
14    },
15  ],
16  getTasksToDo() {
17    return this.tasks.filter((task) => !task.completed);
18  },
19 };
20
21 console.log(tasks.getTasksToDo());
22
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta4_RaporKodları
arrow-functions> node .\arrow-challenge.js
[
  { text: 'Clean Yard', completed: false },
  { text: 'Film Course', completed: false }
]
```

Bu JavaScript kod bloğu, bir tasks nesnesi oluşturur. Bu nesne, tasks adında bir dizi özellik içerir. Her bir öğe, bir görevi temsil eden bir nesnedir ve text (metin) ve completed (tamamlanmış) özelliklerini içerir. Ardından, bu nesne içerisinde getTasksToDo adında bir metod tanımlanır.

getTasksToDo metodu, tasks dizisini dolaşır ve her bir görevin completed özelliğini kontrol ederek tamamlanmamış görevleri bir filtreleme işlemiyle döndürür. Yani, tamamlanmamış görevlerden oluşan bir alt dizi döndürür.

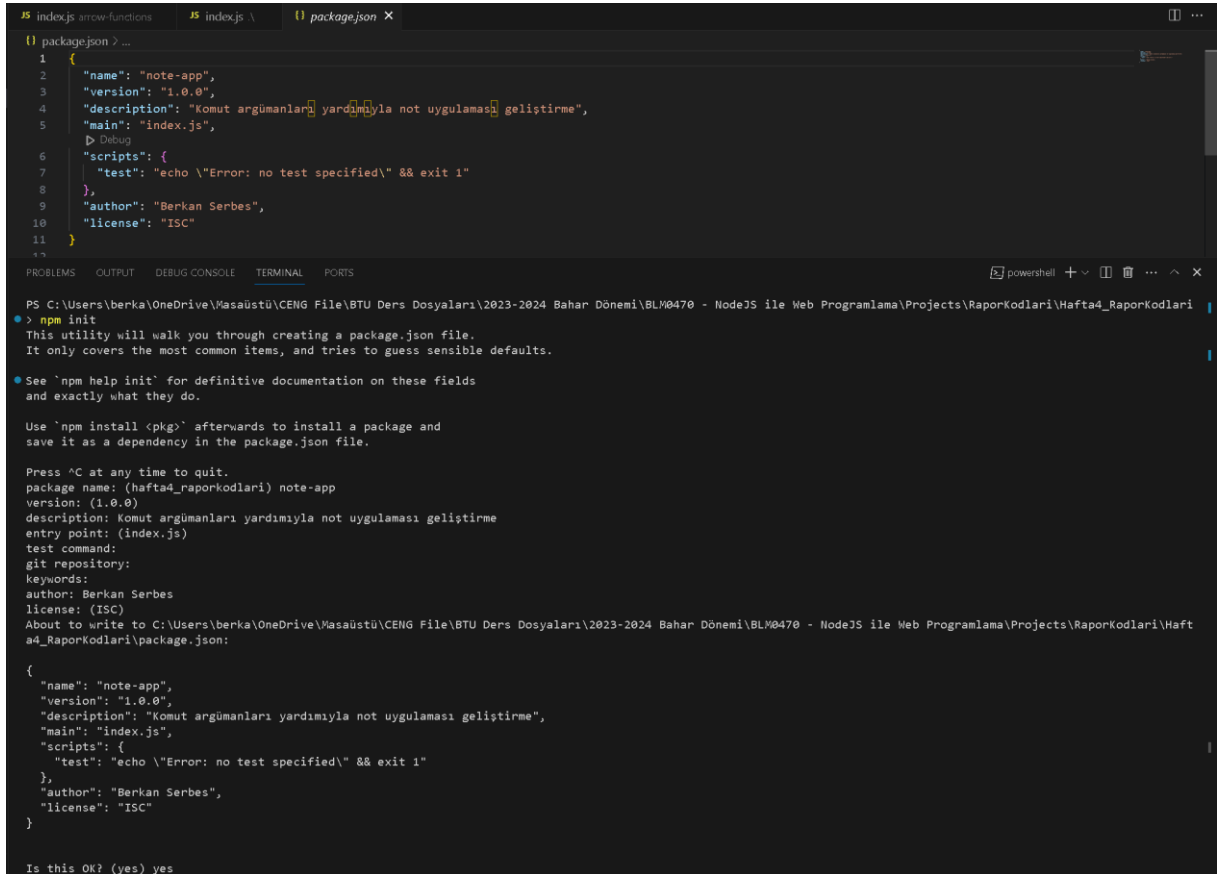
Son olarak, console.log ifadesi, getTasksToDo metodunu çağırır ve sonucunu konsola yazdırır. Bu sayede, tamamlanmamış görevlerin listesi konsola yazdırılır.

2.Argumantatif Komut Satırı Uygulaması Geliştirme

2.1 Projenin Oluşturulması ve Package.json Dosyasının Oluşturulması

Projeyi başlatmak için ilk adım, bir boş dizin oluşturmaktır. Bu dizine **index.js** adında bir dosya ekleyerek başlayabiliriz. Daha sonra editör terminalini veya komut istemcisini kullanarak bu çalışma dizinine gidip **npm init** komutunu çalıştırarak **package.json** dosyasını oluşturabiliriz. "package.json" dosyası, bir Node.js projesinin temelini oluşturur ve projenin bağımlılıklarını, betiklerini ve diğer önemli bilgilerini içerir. Bu dosya, projenin yapılandırılması, yönetilmesi ve dağıtılması için kritik bir rol oynar.

npm init komutunu kullandıktan sonra, genellikle projenin adı, sürümü, açıklaması, giriş noktası, test komutları, projenin sahibinin adı ve lisansı gibi bir dizi soru sorulur. Ancak, bu aşamaları hızlıca atlamak veya varsayılan değerleri kabul etmek istiyorsanız, npm init -y komutunu kullanarak bu işlemi otomatikleştirebilirsiniz. Bu komut, sizin adınıza tüm bu soruları varsayılan değerlerle doldurarak, size ek bir giriş yapma zorunluluğu olmadan hızlıca bir başlangıç yapmanıza olanak tanır. Eğer bu aşamayı manuel olarak yapmak istiyorsanız, npm init komutunu çalıştırdıktan sonra karşınıza çıkan soruları aşağıdaki gibi cevaplayabilirsiniz.



```
1 {
2   "name": "note-app",
3   "version": "1.0.0",
4   "description": "komut argümanları yardımıyla not uygulaması geliştirme",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "Berkant Serbes",
10  "license": "ISC"
11 }
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta4_RaporKodları> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

• See 'npm help init' for definitive documentation on these fields
  and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hafta4_raporKodları) note-app
version: (1.0.0)
description: komut argümanları yardımıyla not uygulaması geliştirme
entry point: (index.js)
test command:
git repository:
keywords:
author: Berkant Serbes
license: (ISC)
About to write to C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta4_RaporKodları\package.json:
{
  "name": "note-app",
  "version": "1.0.0",
  "description": "komut argümanları yardımıyla not uygulaması geliştirme",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Berkant Serbes",
  "license": "ISC"
}

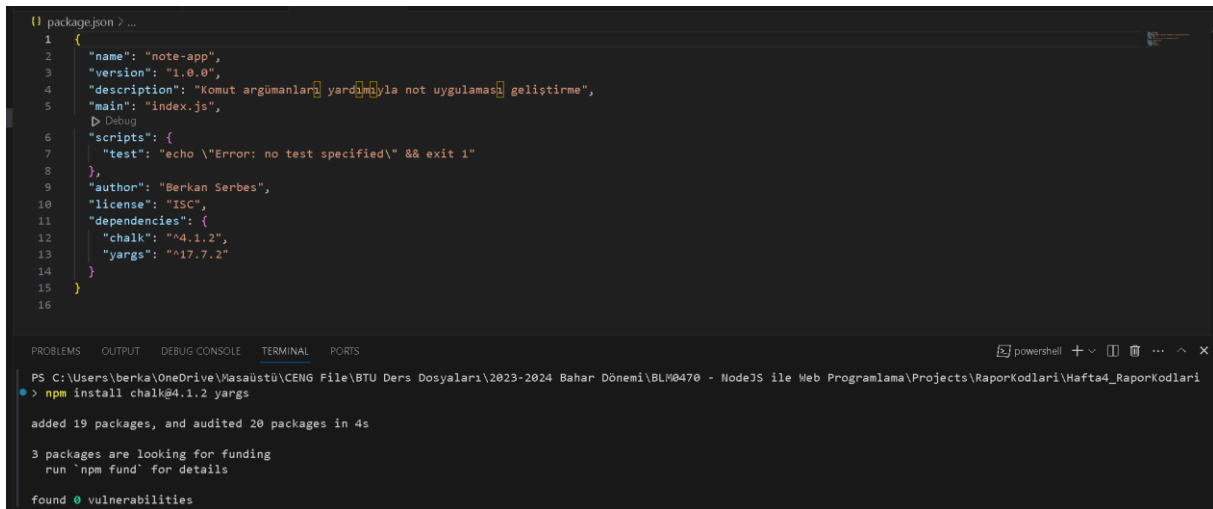
Is this OK? (yes) yes
```

Package.json dosyasını manuel bir şekilde oluşturulduktan sonra dosyanın içeriği yukarıdaki görseldeki gibi olacaktır.

2.2 Gerekli NPM Paketlerinin Yüklenmesi

Projemiz için gereksinim duyduğumuz paketler **yargs** ve **chalk** olacak. Yargs, komut satırı argümanlarını işlemek için kullanılır ve karmaşık komut satırı arayüzlerinin oluşturulmasını kolaylaştırır. Öte yandan, Chalk terminalde renkli ve biçimlendirilmiş çıktılar oluşturmak için kullanılır, bu da kullanıcılara daha okunaklı ve çekici bir deneyim sunar. Bu paketleri projemize eklemek için "npm install" komutunu kullanacağız. İki paketi aynı anda yüklemek için ise "npm install yargs chalk@4.1.2" komutunu kullanacağız. Chalk paketini yüklerken belirlediğimiz versiyon numarası (4.1.2), chalk paketinin son sürümlerinin CommonJS modülleriyle uyumlu olmaması nedeniyle tercih edilmektedir. Bu şekilde, uyumsuzluk problemlerinden kaçınarak projemizin stabilitesini ve uyumluluğunu sağlamış olacağız.

Paketleri yükledikten sonra, "package.json" dosyamızın içeriği aşağıdaki gibi olacaktır:



```
1 {
2   "name": "note-app",
3   "version": "1.0.0",
4   "description": "Komut argümanları yardımıyla not uygulaması geliştirme",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "Berkan Serbes",
10  "license": "ISC",
11  "dependencies": {
12    "chalk": "^4.1.2",
13    "yargs": "^17.7.2"
14  }
15 }
16
```

```
PS C:\Users\berka\OneDrive\Masüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\W470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta4_RaporKodları> npm install chalk@4.1.2 yargs
added 19 packages, and audited 20 packages in 4s
3 packages are looking for funding
run 'npm fund' for details
found 0 vulnerabilities
```

Yukarıdaki "package.json" dosyası, projenin temel bilgilerini ve bağımlılıklarını içerir. "chalk" ve "yargs" paketleri bağımlılıklar listesinde belirtilmiştir. Ayrıca, projenin adı, sürümü, ana dosyası, test komutları, anahtar kelimeler, yazar ve lisans bilgileri gibi diğer önemli bilgiler de dosyada yer alır. Bu şekilde, projenin yapılandırması ve yönetimi için gerekli olan temel bilgiler sağlanmış olur.

2.3 Fonksiyonların Oluşturulması

Projemizin parçalarını daha iyi organize etmek ve kodumuzu daha modüler hale getirmek için her bir fonksiyonu ayrı bir dosyada oluşturmayı tercih ediyoruz. Bu yaklaşımın birkaç önemli avantajı bulunmaktadır.

İlk olarak, her fonksiyonun kendi dosyasında izole bir şekilde bulunması, kodun daha temiz ve okunabilir olmasını sağlar. Bu, her bir fonksiyonun işlevselliğini daha kolay anlamamıza ve kodun daha iyi bir şekilde organize edilmesine yardımcı olur. Ayrıca, her bir fonksiyonun ayrı bir dosyada bulunması, belirli bir fonksiyonun güncellenmesi veya değiştirilmesi gerektiğinde sadece ilgili dosyanın üzerinde çalışmayı gerektirir. Bu, kodun bakımını ve güncellenmesini daha kolay ve hızlı hale getirir. Bu yaklaşım ayrıca kodun yeniden kullanılabilirliğini artırır. Her bir fonksiyonun ayrı bir dosyada bulunması, bu fonksiyonların başka projelerde veya farklı bileşenlerde yeniden kullanılmasını kolaylaştırır.

Bu nedenlerle, projemizdeki her bir fonksiyonu ayrı bir dosyada oluşturarak kodumuzu daha modüler hale getiriyoruz. Bu, projenin geliştirilmesini ve bakımını kolaylaştırırken, kodun daha sürdürülebilir ve genel olarak daha iyi organize edilmiş olmasını sağlar.

Ayrıca uygulama genelinde fonksiyonlarımızı oluştururken geleneksel function anahtar kelimesi yerine arrow fonksiyonlarını kullanacağız. Arrow fonksiyonların birçok avantajı bulunmaktadır.

Öncelikle, arrow fonksiyonları daha kısa ve daha temiz bir sözdizimine sahiptir. Fonksiyonu tanımlarken "function" anahtar kelimesini kullanmak yerine => operatörü kullanılır. Bu, kodun daha okunabilir ve anlaşılır olmasını sağlar.

Ayrıca, arrow fonksiyonları genellikle daha iyi performansa sahiptir. Geleneksel fonksiyonlar her çağrıldığında yeni bir kapsam oluştururken, arrow fonksiyonları kendi kapsamlarına sahip oldukları için daha hafif bir yapıya sahiptir. Tüm bu avantajlar göz önüne alındığında, projemizde arrow fonksiyonlarını kullanarak kodumuzu daha kısa, daha okunabilir ve daha tutarlı hale getireceğiz.

2.3.1 Not Ekleme Fonksiyonu

Aşağıdaki görseldeki gibi "addNote.js" adında bir dosya oluşturalım. Bu dosya, not eklemek için kullanılacaktır. İçeriğinde, "addNote" adında bir fonksiyon tanımlayacağız. Bu fonksiyon iki parametre alacak: "title" (başlık) ve "body" (içerik). Bu parametreler, notun başlığını ve içeriğini belirtmek için kullanılacaktır. Ardından, "module.exports" kullanarak bu fonksiyonu dışarıya aktaracağız.

```
JS index.js  {} package.json  JS addNote.js X
JS addNote.js > ...
1  const addNote = (title, body) => {
2
3  }
4
5  module.exports = addNote;
```

Not ekleme fonksiyonumuzun genel hatlarını oluşturduktan sonra fonksiyonun tamamlanmış hali aşağıdaki görseldeki gibi olacaktır.

```
JS index.js  {} package.json  JS addNote.js X
JS addNote.js > ...
1  const addNote = (title, body) => {
2      const notes = loadNotes();
3
4      if(notes.find(note => note.title === title)) {
5          console.log(chalk.bgRed(`${title} başlığında bir not zaten bulunmaktadır!`));
6          return;
7      }
8      notes.push({title, body});
9
10     saveNotes(notes);
11 }
12
13 module.exports = addNote;
```

Fonksiyon içinde ilk adım olarak, mevcut notları yüklemek veya herhangi bir not olmadığında boş bir diziyle başlatmak için "loadNotes" fonksiyonu kullanılır. Bu şekilde, mevcut notlar

"notes" adında bir değişkende saklanır. Daha sonra, verilen "title" parametresi ile mevcut notların başlıkları karşılaştırılarak, aynı başlığa sahip bir notun var olup olmadığı kontrol edilir. Eğer böyle bir not varsa, ekleme işlemi durdurulur ve konsola "ilgili notun başlığında bir not zaten bulunmaktadır!" şeklinde bir uyarı mesajı yazdırılır. Eğer not başlığı mevcut değilse, yeni not "notes" dizisine eklenir ve "saveNotes" fonksiyonu çağrılarak güncellenmiş notlar dosyaya kaydedilir.

Yukarıda belirtildiği gibi, "loadNotes" ve "saveNotes" fonksiyonlarını da kullanıyoruz ancak henüz ilgili fonksiyonları oluşturmadık. Sıradaki adım, bu fonksiyonları oluşturmak ve bu dosyaya import etmektir.

2.3.2 Notları Getirme Fonksiyonu

Bu fonksiyon, notları yazdıracağımız dosyadan gelen verileri fs (file system) modülünde bulunan readFileSync fonksiyonu aracılığıyla alır. readFileSync fonksiyonu, belirtilen dosyadan eşzamanlı olarak veri okur. Aşağıdaki görselden görüldüğü üzere readFileSync fonksiyonu ayrıca ikinci bir parametre olarak nesne türünde bir veri almış. {encoding: 'utf-8'} seçeneği, dosyadan okunan verinin Unicode karakter kodlamasıyla okunmasını sağlar, böylece metin verileri doğru bir şekilde yorumlanabilir ve işlenebilir. Eğer bu seçeneği eklemeseydik varsayılan olarak veriler buffer şeklinde okunacaktı ve daha sonra bizim bu buffer şeklinde okunan veriyi toString() fonksiyonu yardımıyla string'e dönüştürmemiz gerekecekti. Bu seçenekle beraber fazla kod yazımından kurtulmuş olduk.

Okunan veriler daha sonra JSON.parse() fonksiyonu kullanılarak bu string nesneye dönüştürülür. Bu işlem sayesinde, dosyadan alınan veriler bir JavaScript nesnesine çevrilir ve bu nesne notları temsil eder.

Bu süreç, dosyadan veri okumanın yanı sıra, okunan verilerin uygun bir formata dönüştürülmesini sağlar. Böylece, projemizdeki not verileri daha kolay işlenebilir ve yönetilebilir hale gelir.

```
1  const { readFileSync } = require("fs");
2
3  const loadNotes = () => {
4    const data = readFileSync("./notes/notes.json", { encoding: "utf-8" });
5
6    return JSON.parse(data);
7  };
8
9  module.exports = loadNotes;
10
11
```

Bu fonksiyonda bazı eksik kısımlar bulunmaktadır. Örneğin, eğer kullanıcı var olmayan bir dosyadan o dosyanın içeriğini okumaya çalışırsa hata alacağız. Ayrıca, okunan dosyanın içeriği boşsa, yani dosya hiçbir veri içermiyorsa, bu durumda boş bir dizi döndürmeliyiz. Bu şekilde, fonksiyonun her durumu kapsamasını ve kullanıcıya doğru bir geri bildirim sağlamasını sağlarız.

Aşağıdaki görselden de görüldüğü üzere var olmayan bir dosyadan veri okumaya çalıştığımızda "no such file or directory" hatası karşımıza çıkacaktır.


```
index.js  addNotes.js  loadNotes.js X
loadNotes.js > ...
1  const { readFileSync } = require("fs");
2
3  const loadNotes = () => {
4    const data = readFileSync("./notes/notes.json", { encoding: "utf-8" });
5
6    return JSON.parse(data);
7  };
8
9  console.log(loadNotes());
10
11 module.exports = loadNotes;
12

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node .\loadNotes.js
node:fs:453
  return binding.readFileUtf8(path, stringToFlags(options.flag));
                ^
Error: ENOENT: no such file or directory, open 'C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari\loadNotes.js:4:16'
    at readFileSync (node:fs:453:20)
    at loadNotes (C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari\loadNotes.js:4:16)
    at Object.<anonymous> (C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari\loadNotes.js:9:13)
    at Module._compile (node:internal/modules/cjs/loader:1375:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1435:10)
    at Module.load (node:internal/modules/cjs/loader:1207:32)
    at Module._load (node:internal/modules/cjs/loader:1023:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:135:12)
    at node:internal/main/run_main_module:28:49 {
  errno: -4058,
  code: 'ENOENT',
  syscall: 'open',
  path: 'C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari\loadNotes.js'
}
Node.js v20.11.1
```

Evet tahmin ettiğimiz gibi kodumuzu çalıştırdığımızda “no such file or directory” hatasını alıyoruz. Bu durumu düzeltmek için öncelikle var olmayan dosyaları kontrol etmeli ve gerekli dosyaların varlığını sağlamalıyız. Bu sayede, fonksiyonumuzun her durumu ele almasını ve beklenmeyen hatalarla karşılaşmamızı önlemesini sağlarız. Güncellenmiş kodumuz aşağıdaki görseldeki gibi olacaktır.

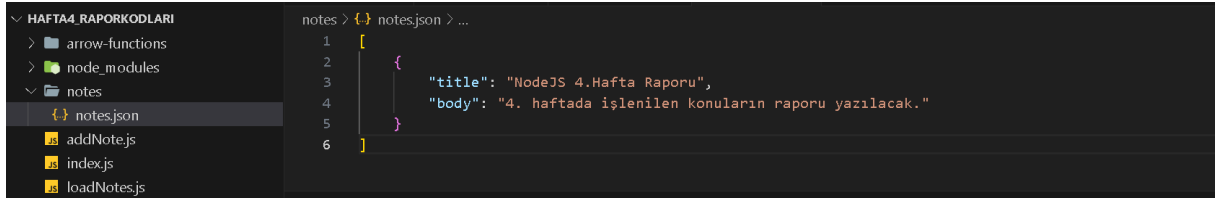
```
loadNotes.js X loadNotes
1  const { readFileSync } = require("fs");
2
3  const loadNotes = () => {
4    try {
5      const data = readFileSync("./notes/notes.json", { encoding: "utf-8" });
6      return JSON.parse(data);
7    } catch (error) {
8      return [];
9    }
10  };
11
12 console.log(loadNotes());
13
14 module.exports = loadNotes;
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node .\loadNotes.js
[]
```

Konsol ekranından görüldüğü üzere kodun güncellenmiş halini çalıştırdığımızda henüz böyle bir dosya olmamasına rağmen hata değil boş bir dizi dönmektedir.

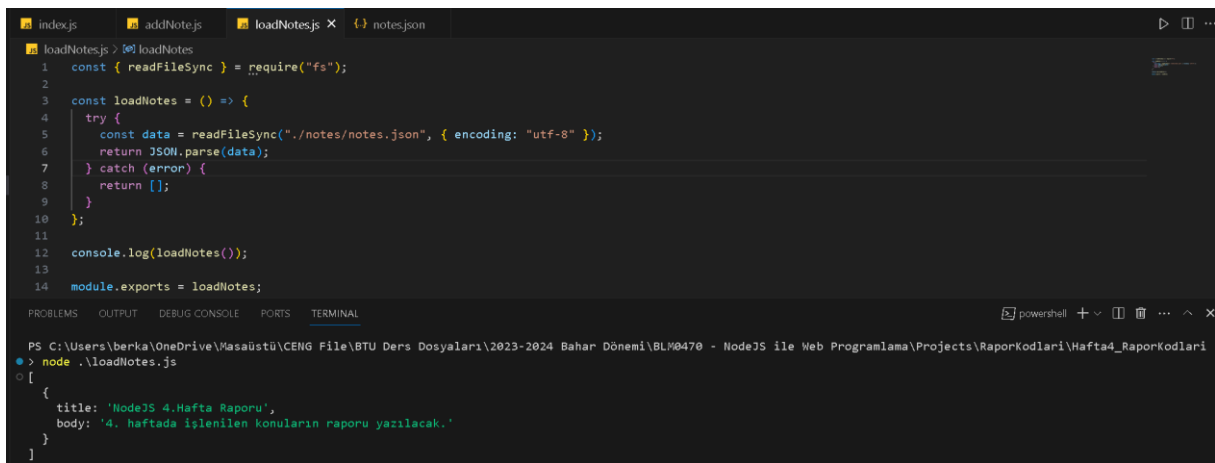
2.3.3 Notların Bulunduğu JSON dosyasının oluşturulması

Şimdi de bulunduğumuz dizine notes klasörünü ekleyip ilgili klasöre de notes.json dosyamızı oluşturalım ve içeriğini dolduralım.



```
notes > {} notes.json > ...
1  [
2    {
3      "title": "NodeJS 4.Hafta Raporu",
4      "body": "4. haftada işlenilen konuların raporu yazılacak."
5    }
6  ]
```

notes.json dosyamızı bu şekilde oluşturduktan sonra loadNotes fonksiyonumuzu yeniden çalıştıralım ve çıktısını görelim.



```
loadNotes.js > loadNotes
1  const { readFileSync } = require("fs");
2
3  const loadNotes = () => {
4    try {
5      const data = readFileSync("./notes/notes.json", { encoding: "utf-8" });
6      return JSON.parse(data);
7    } catch (error) {
8      return [];
9    }
10 };
11
12 console.log(loadNotes());
13
14 module.exports = loadNotes;
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node .\loadNotes.js
[
  {
    title: 'NodeJS 4.Hafta Raporu',
    body: '4. haftada işlenilen konuların raporu yazılacak.'
  }
]
```

Yukarıdaki görselden görüldüğü üzere notes.json dosyamızdaki içerik başarılı bir şekilde okunmaktadır.

Sıradaki yapacağımız düzenleme loadNotes fonksiyonunu addNotes fonksiyonuna import etmek olacak.



```
addNote.js > ...
1  const loadNotes = require("./loadNotes");
2
3  const addNote = (title, body) => {
4    const notes = loadNotes();
5
6    if (notes.find((note) => note.title === title)) {
7      console.log(
8        chalk.bgRed(`"${title}" başlığında bir not zaten bulunmaktadır!`)
9      );
10     return;
11   }
12   notes.push({ title, body });
13
14   saveNotes(notes);
15 };
16
```

2.3.4 Notları Kaydetme Fonksiyonu

saveNotes.js adlı bir dosya oluşturalım ve içeriğini aşağıdaki görseldeki gibi dolduralım.

```
JS saveNotes.js > ...
1  const { writeFileSync } = require("fs");
2
3  const saveNotes = (notes) => {
4    try {
5      writeFileSync("./notes/notes.json", JSON.stringify(notes));
6    } catch (err) {
7      console.log(err.message);
8    }
9  };
10
11 module.exports = saveNotes;
12
```

Bu fonksiyon, "notes" adında bir dizi(array) parametresi alır. Bu dizi, notların tutulduğu verileri içerir. Ardından, JSON.stringify() fonksiyonu kullanılarak bu dizi bir JSON formatında stringe dönüştürülür. Elde edilen JSON formatındaki veri, "./notes/notes.json" dosyasına writeFileSync() fonksiyonu aracılığıyla yazılır. Eğer herhangi bir hata meydana gelirse, try-catch bloğu içindeki catch bloğu devreye girer ve hata konsola yazdırılır. Son olarak, bu fonksiyon module.exports ile dışarıya aktarılır, böylece başka bir dosyada kullanılabilir hale gelir. Bu şekilde, fonksiyon notların dosyaya kaydedilmesini sağlar ve gerekli hata kontrolü ile birlikte çalışır.

Sıradaki yapacağımız düzenleme bu fonksiyonu not ekleme fonksiyonunun içerisine import etmek olacak aşağıdaki görseldeki gibi.

```
JS index.js  JS addNote.js X  JS saveNotes.js  JS loadNotes.js  notes.json
JS addNote.js > ...
1  const loadNotes = require("../loadNotes");
2  const saveNotes = require("../saveNotes");
3
4  const addNote = (title, body) => {
5    const notes = loadNotes();
6
7    if (notes.find((note) => note.title === title)) {
8      console.log(
9        chalk.bgRed(`${title} başlığında bir not zaten bulunmaktadır!`)
10      );
11      return;
12    }
13    notes.push({ title, body });
14
15    saveNotes(notes);
16  };
17
18 module.exports = addNote;
19
```

2.3.5 Notları Silme Fonksiyonu

removeNote.js adında bir dosya oluşturalım ve bu dosyanın içeriğini aşağıdaki görseldeki gibi yazalım.

```
index.js  addNote.js  removeNote.js  saveNotes.js  loadNotes.js  notes.json
removeNote.js > removeNote
1  const loadNotes = require("../loadNotes");
2  const saveNotes = require("../saveNotes");
3
4  const removeNote = (title) => {
5    try {
6      const notes = loadNotes();
7
8      const newNotes = notes.filter((note) => note.title !== title);
9      if (!(notes.length > newNotes.length)) {
10       console.log(chalk.red.inverse("Notunuz silinemedi"));
11       return;
12     }
13
14     saveNotes(newNotes);
15     console.log(chalk.green.inverse("Notunuz silindi"));
16   } catch (err) {
17     console.log(err.message);
18   }
19 };
20
21 module.exports = removeNote;
22
```

Bu fonksiyon, verilen bir başlık (title) parametresiyle çağrıldığında çalışır. İlk olarak, "loadNotes" fonksiyonunu kullanarak mevcut notları yükler. Daha sonra, filtreleme işlemi yaparak verilen başlığa sahip notu listeden kaldırır. Eğer not listesinde belirtilen başlıkla eşleşen bir not bulunamazsa veya not silinemezse (filtreleme sonrası listenin boyutu değişmezse), bir hata mesajı verilir ve işlem sonlandırılır. Aksi halde, yeni not listesi "saveNotes" fonksiyonu ile kaydedilir ve bir başarılı işlem mesajı kullanıcıya gösterilir. Herhangi bir hata durumunda, hata mesajı konsola yazdırılır. Bu şekilde, fonksiyon, belirtilen başlığa sahip bir notu not listesinden kaldırır ve güncellenmiş not listesini dosyaya kaydeder.

2.3.6 Notları Listeleme Fonksiyonu

listNotes.js adında bir dosya oluşturalım ve dosyanın içeriğini aşağıdaki görseldeki gibi yazalım.

```
index.js  addNote.js  removeNote.js  listNotes.js  saveNotes.js  loadNotes.js  notes.json
listNotes.js > ...
1  const loadNotes = require("../loadNotes");
2
3  const listNotes = () => {
4    try {
5      const allNotes = loadNotes();
6
7      if (allNotes.length <= 0) {
8        console.log(chalk.red.inverse("Kayıtlı not bulunamadı!"));
9        return;
10      }
11      console.log(chalk.inverse("Kayıtlı Notlar"));
12      allNotes.forEach((note) =>
13        console.log(
14          `Başlık: ${note.title} \n İçerik: ${note.body} \n *****\n`
15        )
16      );
17    } catch (e) {
18      console.log(e.message);
19    }
20  };
21
22 module.exports = listNotes;
```

Bu fonksiyon, çağrıldığında mevcut notları listeleyen bir işlevdir. İlk olarak, "loadNotes" fonksiyonu kullanılarak tüm notlar yüklenir. Ardından, eğer hiç not bulunamazsa (notlar dizisinin uzunluğu 0 veya daha az ise), kullanıcıya "Kayıtlı not bulunamadı!" şeklinde bir uyarı mesajı gösterilir ve fonksiyon sonlandırılır. Eğer notlar mevcutsa, "Kayıtlı Notlar" başlığı altında her bir not için başlık ve içerik bilgileri sıralanır. Her bir notun başlığı ve içeriği, ayrı ayrı satırlar halinde ve ***... ayraçları ile ayrılarak kullanıcıya gösterilir. Herhangi bir hata durumunda, hata mesajı konsola yazdırılır. Bu şekilde, fonksiyon, mevcut notları kullanıcıya düzenli bir şekilde sunar ve notlar hakkında bilgi sağlar.

2.3.7 Ana Dosyanın (index.js) Oluşturulması

Uygulamanın başlatılması için ilk adım olarak, index.js dosyasının oluşturulması gerekmektedir. Bu dosya, uygulamanın ana giriş noktası olarak işlev görecek.

index.js dosyamızı oluşturduktan sonra bu dosyada kullanılacak npm paketlerini ve oluşturduğumuz fonksiyonları import etmemiz gerekmektedir. Aşağıdaki görseldeki gibi ihtiyacımız olan modülleri import edelim.



```
1  const yargs = require("yargs");
2  const chalk = require("chalk");
3
4  const addNote = require("./addNote");
5  const removeNote = require("./removeNote");
6  const listNotes = require("./listNotes");
7  const loadNotes = require("./loadNotes");
8  const saveNotes = require("./saveNotes");
9
10
```

Şimdi, sıradaki adımımız Yargs paketinin kullanımıyla argüman odaklı fonksiyonlarımızı oluşturmaktır. Yargs paketi sayesinde, kullanıcılar komut satırında belirli işlevleri gerçekleştirmek için çeşitli seçenekler ve argümanlar kullanabilirler. Bu, programımızı daha esnek ve kullanıcı dostu hale getirerek, kullanıcı deneyimini geliştirecektir.

```
index.js x addNote.js removeNote.js listNotes.js saveNotes.js loadNotes.js notes.json
index.js > handler
3
4 const addNote = require("./addNote");
5 const removeNote = require("./removeNote");
6 const listNotes = require("./listNotes");
7 const loadNotes = require("./loadNotes");
8 const saveNotes = require("./saveNotes");
9
10 yargs.command({
11   command: "add",
12   describe: "Yeni bir not ekler",
13   builder: {
14     title: {
15       describe: "Not başlığı",
16       demandOption: true,
17       type: "string",
18     },
19     body: {
20       describe: "Not içeriği",
21       demandOption: true,
22       type: "string",
23     },
24   },
25   handler: (argv) => {
26     addNote(argv.title, argv.body);
27   },
28 });
29
30 yargs.command({
31   command: "remove",
32   describe: "Bir notu siler",
33   builder: {
34     title: {
35       describe: "Not başlığı",
36       demandOption: true,
37       type: "string",
38     },
39   },
40   handler: (argv) => {
41     removeNote(argv.title);
42   },
43 });
44
45 yargs.command({
46   command: "list",
47   describe: "Kayıtlı notları listeler",
48   handler: () => {
49     listNotes();
50   },
51 });
```

İlk olarak, "add" komutu, yeni bir not eklemek için kullanılır. Bu komut, "title" ve "body" parametrelerini gerektirir ve kullanıcıdan notun başlığını ve içeriğini alır. Ardından, "remove" komutu, belirtilen başlıkla bir notu silmek için kullanılır. Bu komut, sadece "title" parametresini gerektirir ve kullanıcıdan silinecek notun başlığını alır. Son olarak, "list" komutu, kayıtlı notları listelemek için kullanılır. Bu komut, herhangi bir parametre gerektirmez ve tüm kayıtlı notları konsola listeler. Her bir komut için "describe" özelliği kullanılarak, komutun ne işe yaradığı açıklanır ve kullanıcıya bilgi verilir. Her komutun "handler" kısmı, kullanıcının verdiği argümanlara göre ilgili fonksiyonları çağırır. Bu şekilde, Yargs paketi aracılığıyla argüman odaklı fonksiyonlarımızı tanımlayarak, kullanıcıların programı daha etkili bir şekilde kullanmalarını sağlarız.

2.3.8 Programın Test Edilmesi

İlk olarak add fonksiyonumuzu yani not ekleme fonksiyonumuzu test edelim.

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node index.js add --title="1.Görev" --body="Rapor düzenlenecek"
{
  _: [ 'add' ],
  title: '1.Görev',
  body: 'Rapor düzenlenecek',
  '$0': 'index.js'
}
```

Görüldüğü üzere add fonksiyonumuz düzgün bir biçimde çalışmaktadır. Gönderdiğimiz notun eklenip eklenmediğini list fonksiyonunu kullanıp test edelim.

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node index.js list
Kayıtlı Notlar
Başlık: NodeJS 4.Hafta Raporu
İçerik: 4. haftada işlenilen konuların raporu yazılacak.
*****
Başlık: 1.Görev
İçerik: Rapor düzenlenecek
*****
{ _: [ 'list' ], '$0': 'index.js' }
```

Görüldüğü üzere notumuz eklenmiş görülüyor.

Şimdide remove komutunu çalıştırıp eklediğimiz notu silelim.

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node index.js remove --title="1.Görev"
Notunuz silindi
{ _: [ 'remove' ], title: '1.Görev', '$0': 'index.js' }
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node index.js list
Kayıtlı Notlar
Başlık: NodeJS 4.Hafta Raporu
İçerik: 4. haftada işlenilen konuların raporu yazılacak.
*****
{ _: [ 'list' ], '$0': 'index.js' }
```

Görüldüğü üzere tüm fonksiyonlarımız beklenen çıktıyı vermektedir.

3. Hata Ayıklama (Debugging)

Hata ayıklama (debugging), yazdığımız programlarda oluşan hataları tanımlama, anlama ve düzeltme sürecidir. Node.js'te debugging, genellikle hataların kaynağını bulmak ve gidermek için çeşitli teknikler ve araçlar kullanılarak gerçekleştirilir.

Bir Node.js uygulamasında hata ayıklama yaparken, en yaygın kullanılan yöntemlerden biri "console.log()" ifadeleri eklemektir. Bu yöntem, programın belirli noktalarında değişkenlerin değerlerini, fonksiyonların çağırılma durumunu ve akışını konsola yazdırarak programın çalışma sürecini izlememizi sağlar. Bu şekilde, programın hangi aşamalarda ve nasıl davrandığını daha iyi anlayabiliriz.

Bununla birlikte, daha karmaşık hataları tespit etmek ve çözmek için Node.js'in dahili hata ayıklama modülü olan "debugger" kullanılabilir. Debugger, programın belirli noktalarında duraklatılmasını ve adım adım ilerletilmesini sağlayarak hataların kaynağını daha detaylı bir şekilde incelememizi sağlar. Bu sayede, programın içindeki değişken değerlerini ve kod akışını anlık olarak gözlemleyebiliriz.

Node.js'in geliştirme araçları da debugging sürecinde önemli bir rol oynar. Örneğin, Visual Studio Code gibi entegre geliştirme ortamları (IDE'ler), hata ayıklama için kullanışlı araçlar ve işlevler sağlar. Bu araçlar sayesinde, programı adım adım çalıştırabilir, değişken değerlerini izleyebilir, koddaki hataları kolayca tanımlayabilir ve düzeltebiliriz.

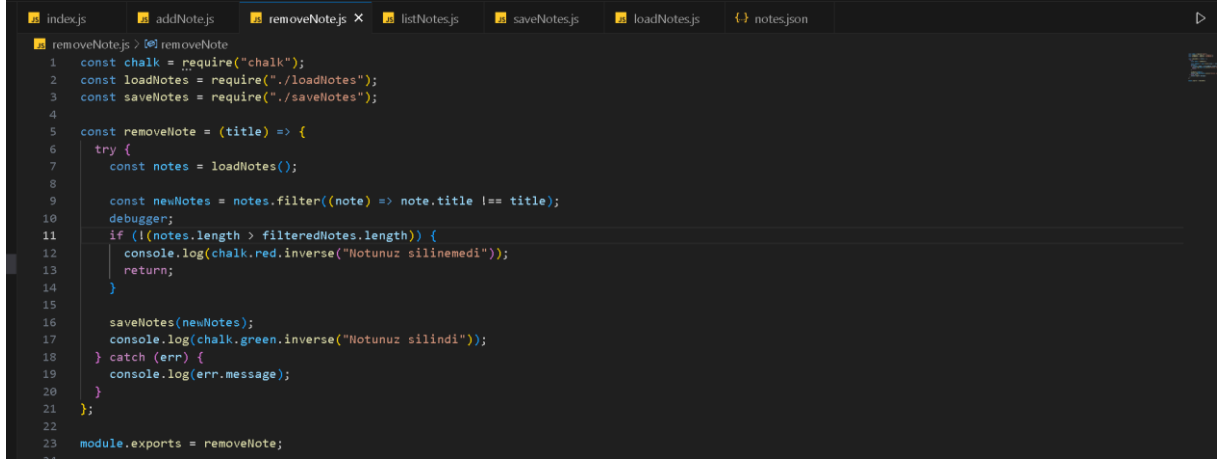
Ayrıca, Node.js'in "util" modülündeki "util.inspect()" fonksiyonu gibi araçlar da debugging sürecinde yardımcı olabilir. Bu fonksiyon, bir nesnenin içeriğini düzgün bir şekilde görselleştirmemizi sağlar, böylece nesnenin yapısı hakkında daha detaylı bilgi elde edebiliriz.

Tüm bu araçlar ve teknikler, Node.js uygulamalarında hata ayıklama sürecini daha etkili ve verimli hale getirir, böylece yazılım projelerimizi daha güvenilir hale getirme ve hataları daha hızlı bir şekilde çözme imkanı sağlar.

3.1 Uygulamamızdaki Hataları Debugger Yardımıyla Giderme

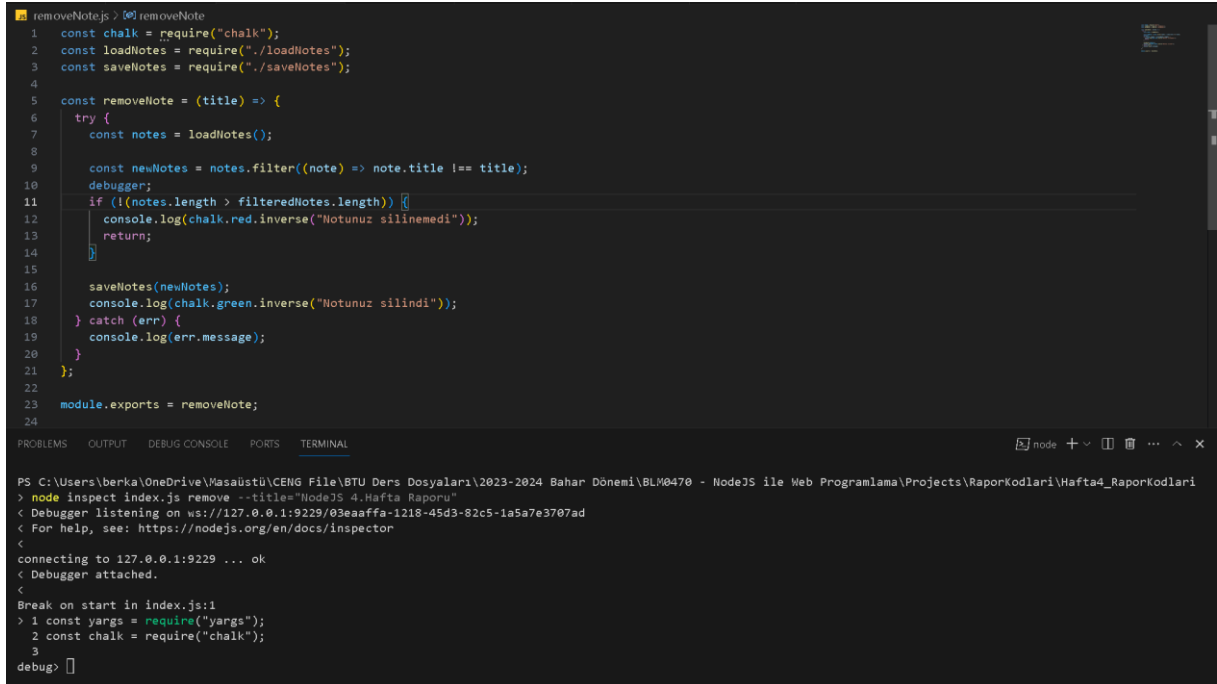
Uygulamamızda yer alan not silme fonksiyonunu yazarken bir satırda bilerek hatalı bir ifade yazdım. Bu hatayı debugger yardımıyla çözelim.

Öncelikle hata ayıklama işleminin nerden başlaması gerektiğini belirtmek için istediğimiz satıra debugger; ifadesini ekliyoruz. Aşağıdaki görselde 10.satıra eklenmiştir.



```
1 const chalk = require("chalk");
2 const loadNotes = require("../loadNotes");
3 const saveNotes = require("../saveNotes");
4
5 const removeNote = (title) => {
6   try {
7     const notes = loadNotes();
8
9     const newNotes = notes.filter((note) => note.title !== title);
10    debugger;
11    if (notes.length > filteredNotes.length) {
12      console.log(chalk.red.inverse("Notunuz silinemedi"));
13      return;
14    }
15
16    saveNotes(newNotes);
17    console.log(chalk.green.inverse("Notunuz silindi"));
18  } catch (err) {
19    console.log(err.message);
20  }
21 };
22
23 module.exports = removeNote;
```

İlgili ifadeyi ekledikten sonra izlemek istediğimiz dosyayı komut satırı yardımıyla node ifadesinden sonra inspect anahtar kelimesini ekliyoruz ve geri kalan komutu normal bir şekilde yazıyoruz. Aşağıdaki görseldeki gibi komutumuzu yazalım.



```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta4_RaporKodlari
> node inspect index.js remove --title="NodeJS 4. Hafta Raporu"
< Debugger listening on ws://127.0.0.1:9229/03eaaffa-1218-45d3-82c5-1a5a7e3707ad
< For help, see: https://nodejs.org/en/docs/inspector
<
< connecting to 127.0.0.1:9229 ... ok
< Debugger attached.
<
< Break on start in index.js:1
> 1 const yargs = require("yargs");
  2 const chalk = require("chalk");
  3
  debug>
```

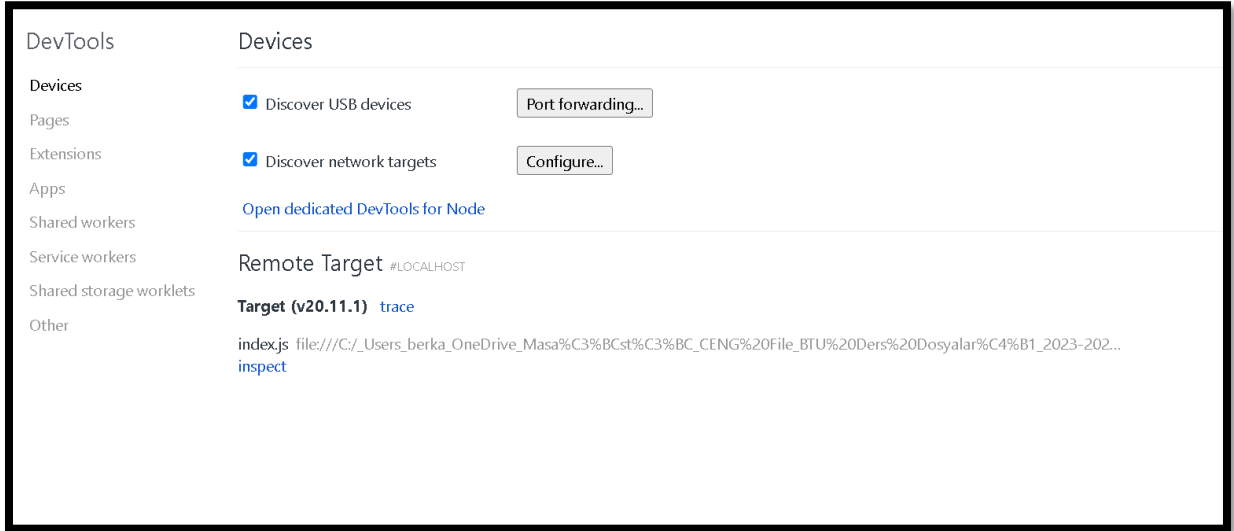
Komutumuz çalıştıktan sonra komut satırı arayüzünde çeşitli yazılar görmekteyiz. Bu yazılar, NodeJS'in hata ayıklama modunun etkinleştirildiğini ve hata ayıklama aracının dinleme modunda olduğunu belirtmektedir.

İlk satır, hata ayıklama aracının WebSocket (ws) üzerindeki dinleme adresini ve bağlantı noktasını (9229) gösterir. Bu adres ve bağlantı noktası, hata ayıklama aracının Node.js uygulamasına bağlanmak için kullanılacak olan adres ve bağlantı noktasıdır. UUID ("03eaaffa-1218-45d3-82c5-1a5a7e3707ad") ise eşsiz bir kimlik numarasıdır.

İkinci satır, hata ayıklama aracının kullanımı hakkında yardım almak için bir bağlantı verir. Bu bağlantı, Node.js'in resmi belgelerinde hata ayıklama aracının kullanımı hakkında daha fazla bilgi edinmek için bir kılavuz sunar.

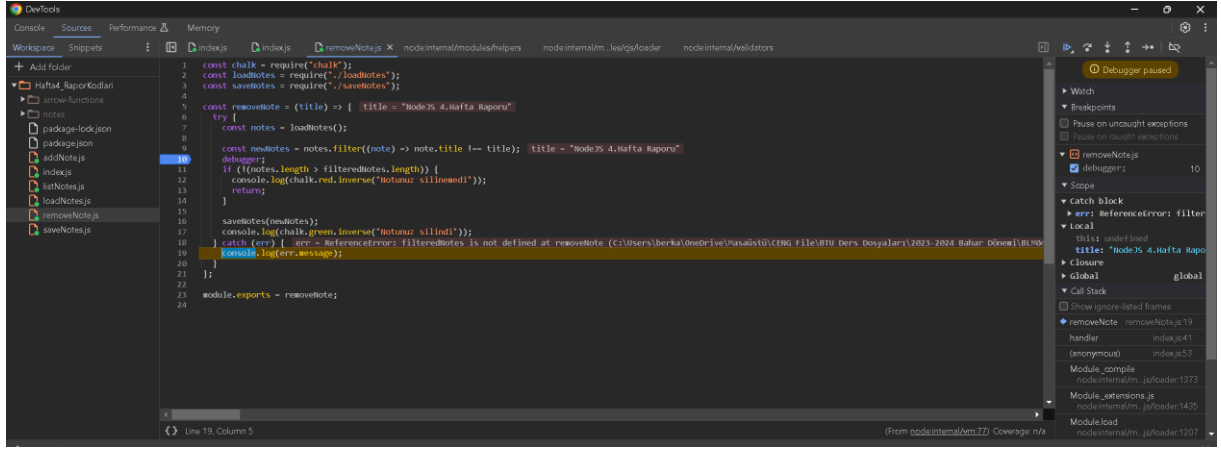
Sonraki satırlar, hata ayıklama aracının başlatıldığını ve belirtilen dosyada (index.js) bir kesme noktası oluşturulduğunu belirtir. Burada, hata ayıklama aracı, programın başlangıcında (index.js dosyasının ilk satırı) bir kesme noktası oluşturarak programın çalışmasını duraklatır. Bu, hata ayıklama aracının programın belirli bir noktasında duraklamasına ve kullanıcının kodu adım adım çalıştırmasına izin verir.

Hata ayıklama kısmına Google Chrome tarayıcımızdan ulaşmak için chrome://inspect adresine gidelim. İlgili sayfada local'de (yerel) çalıştığımız dosyanın hedef yolu çıkacak burada sağ tarafın en alt kısmında yer alan inspect yazan kısma tıklamalıyız.



İlgili kısma dokunduktan sonra boş bir pencere bizi karşılayacak. Bu pencerede Sources (kaynaklar) sekmesine gidip Workspace sekmesine tıklayıp hemen altında yer alan **add folder** kısmını tıklayıp projemizin olduğu dizini seçmeliyiz. Dizini seçtikten sonra klasörde yer alan dosyalar sol köşede görünecektir.

Daha sonra hata olduğunu düşündüğümüz yere breakpoint koymalıyız. Breakpointler hata ayıklama sürecinde, programın belirli bir yerinde kodun işlemlerini duraklatarak kullanıcıya kodu adım adım izleme ve değişken değerlerini kontrol etme imkanı verir. Ben aşağıdaki görseldeki gibi 10. Satıra breakpoint koydum sonrasında F9 kısayoluna basıp kodumu satır satır ilerlettim.



Kodumuzu 2 satır ilerlettikten sonra yukarıdaki şekilde görüldüğü üzere remove fonksiyonumuzda bir hata olduğunu görüyoruz. Bu hata bize filteredNotes adında bir değişkenin olmadığını söylüyor. Dikkat edilirse 9. Satırda filtrelenen notları newNotes adında bir değişkene atamışım ama 11. Satırda bu değişkeni kullanmak istemişim fakat değişkenin adını newNotes yazmak yerine filteredNotes yazmışım. Bu şekilde kodumuzdaki hataları bulup kolayca düzeltebiliriz.

4. KAYNAKÇA

<https://chat.openai.com/>