

BURSA TEKNİK ÜNİVERSİTESİ

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



BLM0470 NodeJS İle Web Programlama

2023-2024 Bahar Dönemi

3.Hafta Raporu

Berkan SERBES – 22360859353

İçindekiler

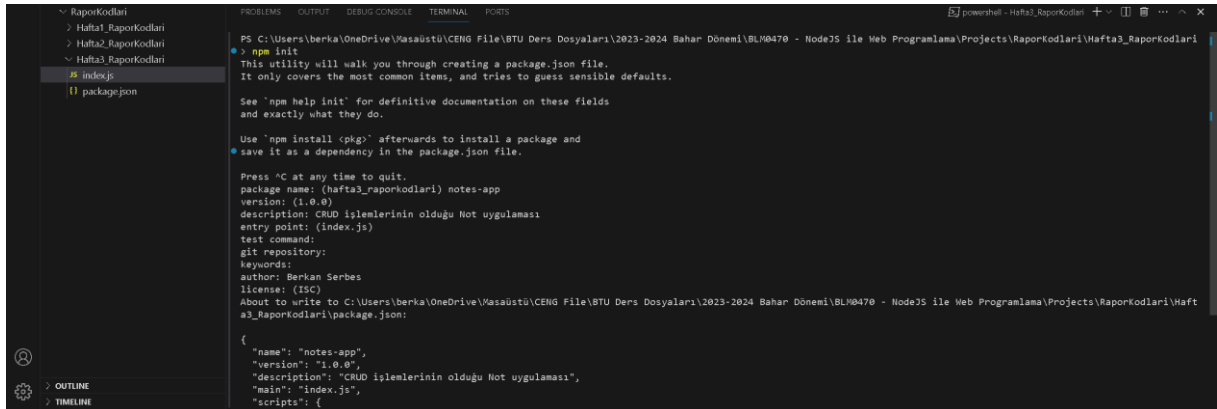
1. Not Uygulaması Geliştirme.....	3
1.1 Package.json Dosyasının Oluşturulması	3
1.2 NPM Paketlerinin Yüklenmesi.....	3
1.3 Fonksiyonların oluşturulması	4
1.3.1 Not Ekleme Fonksiyonunun Oluşturulması	4
1.3.2 Notları Getirme Fonksiyonunun Oluşturulması	5
1.3.3 Notları Kaydetme Fonksiyonunun Oluşturulması.....	9
1.3.4 Not Silme Fonksiyonunun Oluşturulması	9
1.3.5 Uygulamayı Çalıştıracığımız index.js Dosyasının Oluşturulması.....	10
1.4 Uygulamanın Test Edilmesi	12
2. Arrow Fonksiyonu.....	12
2.1 Arrow fonksiyonlarda this anahtar kelimesinin kullanımı	14
KAYNAKÇA	16

1. Not Uygulaması Geliştirme

1.1 Package.json Dosyasının Oluşturulması

Projeyi başlatmak için öncelikle bir boş dizin oluşturmamızdır. Bu dizine "index.js" adında bir dosya ekleyerek başlayabiliriz. Daha sonra terminal veya komut istemcisinde bu çalışma dizinine gidip "npm init" komutunu kullanarak "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 önemlidir.

npm init komutunu kullandıktan sonra, genellikle kullanıcıya 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 aşağıdaki gibi sorulara cevap verebilirsiniz.



```
PS C:\Users\berka\OneDrive\Kasaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLN0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta3_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.

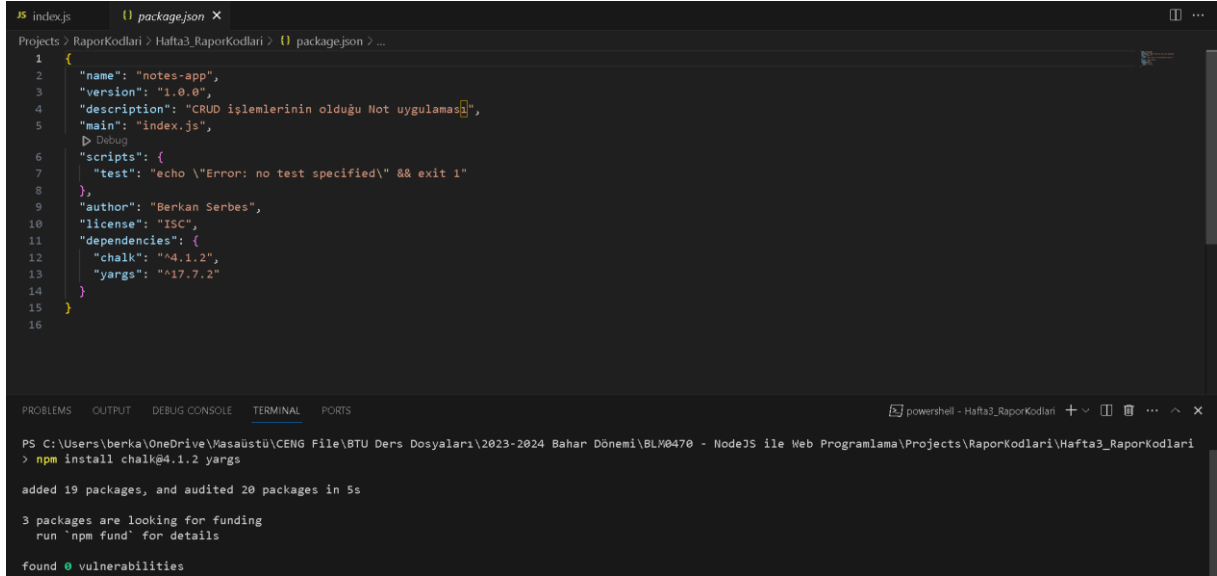
Press ^C at any time to quit.
package name: (hafta3_raporKodları) notes-app
version: (1.0.0)
description: CRUD işlemlerinin olduğu Not uygulaması
entry point: (index.js)
test command:
git repository:
keywords:
author: Berkan Serbes
license: (ISC)
About to write to C:\Users\berka\OneDrive\Kasaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLN0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta3_RaporKodları\package.json:
{
  "name": "notes-app",
  "version": "1.0.0",
  "description": "CRUD işlemlerinin olduğu Not uygulaması",
  "main": "index.js",
  "scripts": {

```

1.2 NPM Paketlerinin Yüklenmesi

Projemizi oluşturduktan sonra, kullanacağımız paketler yargs ve chalk olacaktır. Yargs, komut satırı argümanlarını işlemek için kullanılır ve karmaşık komut satırı arayüzlerinin oluşturulmasını kolaylaştırır. Chalk ise terminalde renkli ve biçimlendirilmiş çıktılar oluşturmak için kullanılır, bu da kullanıcıya 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. Dikkat ederseniz, chalk paketini yüklerken versiyon numarası belirtiyoruz (4.1.2). Bunun nedeni, chalk paketinin son sürümlerinin CommonJS modülleriyle uyumlu olmamasıdır. 4.1.2 sürümünü belirterek, bu uyumsuzluk probleminden kaçınmış oluyoruz.



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

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta3_RaporKodlari> npm install chalk@4.1.2 yargs

added 19 packages, and audited 20 packages in 5s

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

found 0 vulnerabilities
```

Bu iki paketi yükledikten sonra package.json dosyamızın içeriği yukarıdaki görseldeki gibi olacaktır.

1.3 Fonksiyonların oluşturulması

Projemizin daha modüler olması adına her bir fonksiyonu ayrı bir dosyada oluşturmayı tercih edeceğiz. Bu yaklaşım, birkaç önemli avantaj sağlar.

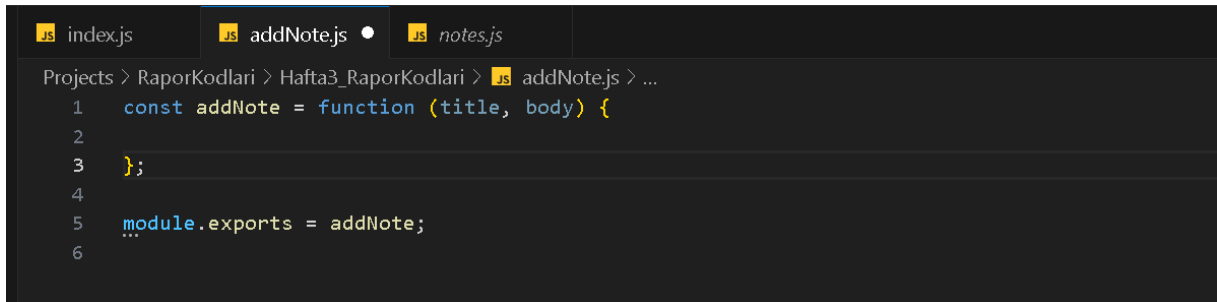
Öncelikle, her fonksiyon kendi dosyasında izole edilmiş bir şekilde bulunarak, kodun daha temiz ve okunabilir olmasını sağlar. Bu, kodun bölümlere ayrılmasını ve her bir bölümünün daha açık ve anlaşılır olmasını sağlar.

Ayrıca, her bir fonksiyonun ayrı bir dosyada bulunması, belirli bir işlevselliğin güncellenmesi veya değiştirilmesi gerektiğinde, sadece ilgili dosyanın üzerinde çalışmayı gerektirir. Bu durum, diğer fonksiyonların etkilenmesini önler ve kodun bakımını ve güncellenmesini kolaylaştırır.

Bu nedenlerle, projemizdeki her bir fonksiyonu ayrı bir dosyada oluşturarak, kodumuzu daha modüler ve yönetilebilir hale getirmeyi amaçlamaktayız.

1.3.1 Not Ekleme Fonksiyonunun Oluşturulması

addNote.js adında bir dosya oluşturalım. Bu dosya, not eklemek için kullanılacaktır. Not ekleme fonksiyonu, title(başlık) ve body(içerik) olmak üzere iki farklı parametre alacaktır. Bu parametreler, notun başlığını ve içeriğini belirtmek için kullanılacaktır. Sonrasında ise module.exports ile bu fonksiyonu dışarıya aktarıyoruz.



```
1 const addNote = function (title, body) {
2
3 };
4
5 module.exports = addNote;
6
```

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

```
.js index.js  .js addNote.js  .js notes.js
Projects > RaporKodlari > Hafta3_RaporKodlari > .js addNote.js > ...
1  const addNote = function (title, body) {
2    const notes = loadNotes();
3
4    if (notes.find((note) => note.title === title)) {
5      console.log(chalk.bgRed("Note title already exists"));
6      return;
7    }
8
9    notes.push({ title, body });
10
11    saveNotes(notes);
12  };
13
14  module.exports = addNote;
15
16
```

Fonksiyon içinde öncelikle, mevcut notları yüklemek veya boş bir diziyle (eğer herhangi bir not yoksa) başlatmak için loadNotes fonksiyonu kullanılır. Ardından, notes adında bir değişkende mevcut notlar tutulur.

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 fonksiyonu durdurulur ve konsola "Note title already exists" şeklinde bir uyarı mesajı yazdırılır.

Eğer not başlığı mevcut değilse, yeni not notes dizisine eklenir ve saveNotes fonksiyonu çağrılarak güncellenmiş notlar dosyaya kaydedilir.

Yukarıda görüldüğü üzere loadNotes ve saveNotes fonksiyonlarını da kullandık ancak henüz ilgili fonksiyonları oluşturmadık. Sıradaki yapacağımız işler bu fonksiyonları oluşturup bu dosyaya import etmek olacak.

1.3.2 Notları Getirme Fonksiyonunun Oluşturulması

Bu fonksiyon, notları yazdıracağımız dosyadan gelen verileri fs (file system) modülünde bulunan readFileSync fonksiyonu aracılığıyla alır. Bu fonksiyon, belirtilen dosyadan eşzamanlı olarak veri okur.

Okunan veriler, önce bir stringe dönüştürülür ve 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. Kodumuz aşağıdaki gibidir.

```
JS index.js JS addNote.js JS loadNotes.js JS notes.js
Projects > RaporKodlari > Hafta3_RaporKodlari > JS loadNotes.js > ...
1  const { readFileSync } = require("fs");
2
3  const loadNotes = function () {
4      const dataBuffer = readFileSync("./notes/notes.json");
5      const dataJSON = dataBuffer.toString();
6      return JSON.parse(dataJSON);
7  };
8
9
```

Bu fonksiyonda eksik bazı kısımlar mevcut. Örneğin, eğer kullanıcı var olmayan bir dosyadan o dosyanın içeriğini okumaya çalışırsa, dataBuffer değişkenimiz null değerini alacaktır. Sonraki satırlarda bu null değerini string bir değere dönüştürmeye çalıştığımızda hata alacağız. Aynı zamanda, 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.

4. satırda notes klasöründe yer alan notes.json dosyasından verileri okumaya çalışıyoruz. Ancak, şu anda notes.json dosyamız bulunmamaktadır. Eğer bu fonksiyonu çalıştırsak, dosya bulunamadığı için bir hata alırız. Kodumuzu çalıştıralım ve çıktıyı görelim.

```
index.js addNote.js loadNotes.js notes.js
Projects > RaporKodlari > Hafta3_RaporKodlari > JS loadNotes.js > ...
1  const { readFileSync } = require("fs");
2
3  const loadNotes = function () {
4      const dataBuffer = readFileSync("./notes/notes.json");
5      const dataJSON = dataBuffer.toString();
6      return JSON.parse(dataJSON);
7  };
8
9
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\Hafta3_RaporKodlari
> node loadNotes
node:fs:581
  return binding.open(
        ^
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\Hafta3_RaporKodlari\notes\notes.json'
    at Object.openSync (node:fs:581:18)
    at readFileSync (node:fs:457:35)
    at loadNotes (C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta3_RaporKodlari\loadNotes.js:4:22)
    at Object.<anonymous> (C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta3_RaporKodlari\loadNotes.js:9:1)
    at Module._compile (node:internal/modules/cjs/loader:1376:14)
    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\\Hafta3_RaporKodlari\\notes\\notes.json'
}
```

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.

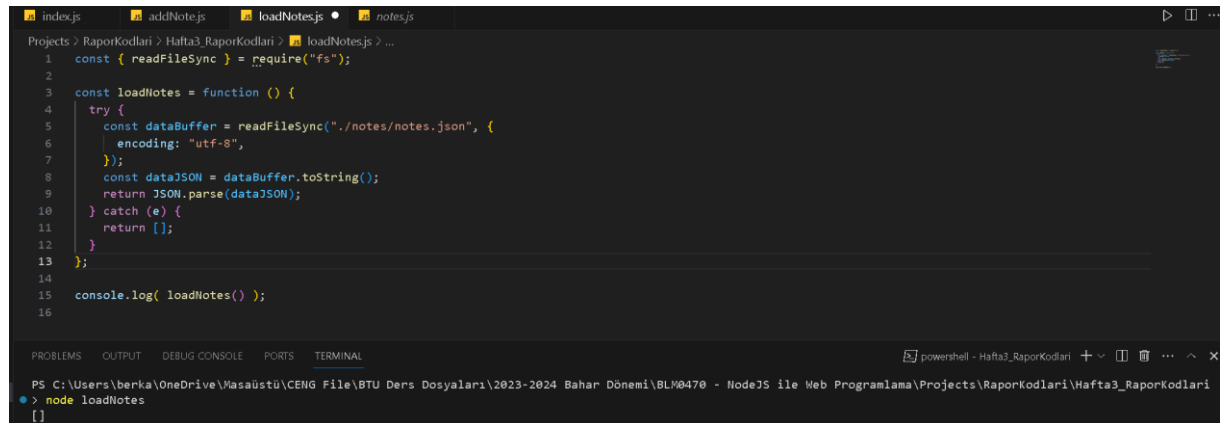
Not: Ek olarak `readFileSync` dosyasına ikinci bir argüman olarak bir nesne gönderdim içinde `encoding: "utf-8"` opsiyonu yer almaktadır. Bu bize okunan datanın buffer olarak değil de unicode karakterleri formatında okunmasını sağlayacaktır. Bu argümanı eklemesiniz de olur.

```
const { readFileSync } = require("fs");

const loadNotes = function () {
  try {
    const dataBuffer = readFileSync("./notes/notes.json", {
      encoding: "utf-8",
    });
    const dataJSON = dataBuffer.toString();
    return JSON.parse(dataJSON);
  } catch (e) {
    return [];
  }
};
```

Yaptığımız değişikle beraber eğer dosya okuma veya veri dönüştürme işlemlerinde herhangi bir hata oluşursa, `catch` bloğu çalışır ve fonksiyon boş bir dizi döndürür. Bu durumda, fonksiyon, dosyanın bulunamaması veya okunamaması gibi hatalarla karşılaşılması durumunda boş bir dizi döndürerek aldığımız hatayı bu şekilde düzeltip işlemine devam eder.

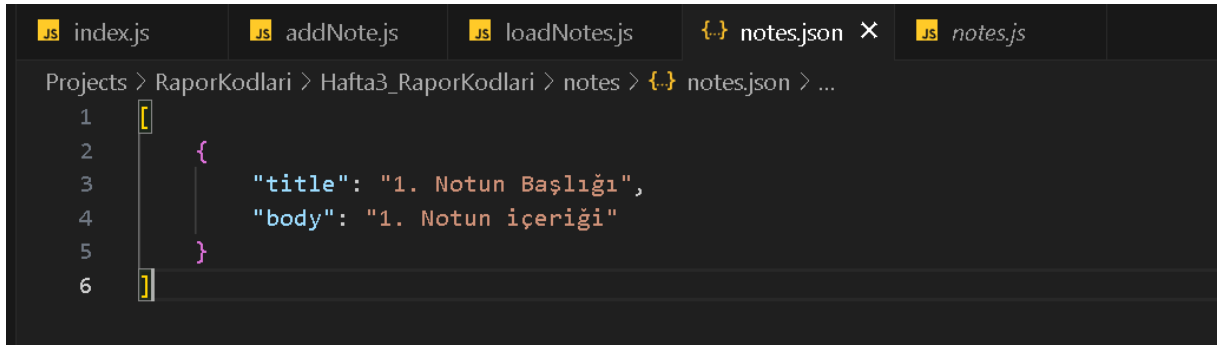
İlk olarak bu fonksiyonu olduğu haliyle yani `notes.json` dosyasını oluşturmadan çalıştıralım. Beklenen çıktı boş bir array dönmesi olacaktır.



The screenshot shows a VS Code editor with a project named 'RaporKodlari' and a file named 'loadNotes.js'. The code in the file is the same as shown in the previous block. The terminal at the bottom shows the command `node loadNotes` being executed, and the output is an empty array `[]`.

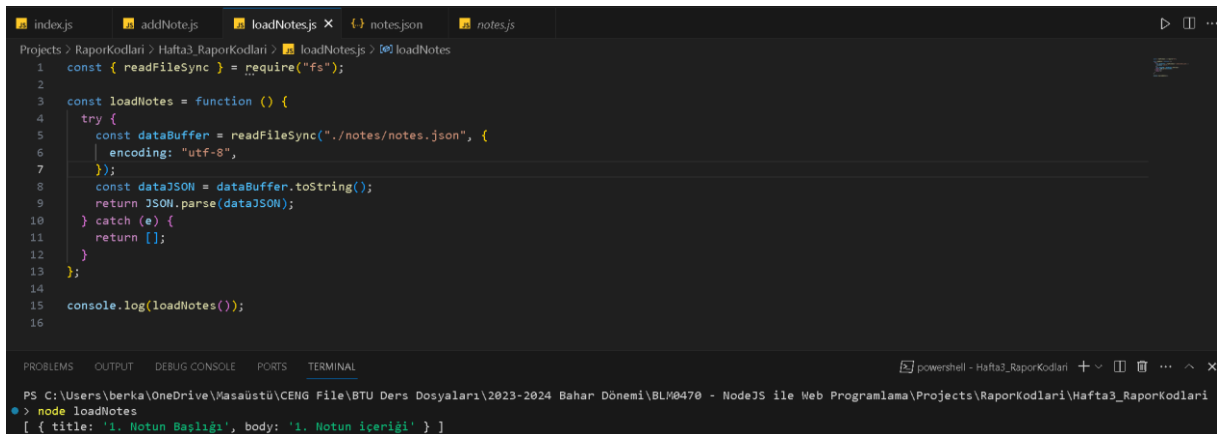
Yukarıdaki görselden görüldüğü üzere beklediğimiz gibi bir çıktı almaktayız.

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



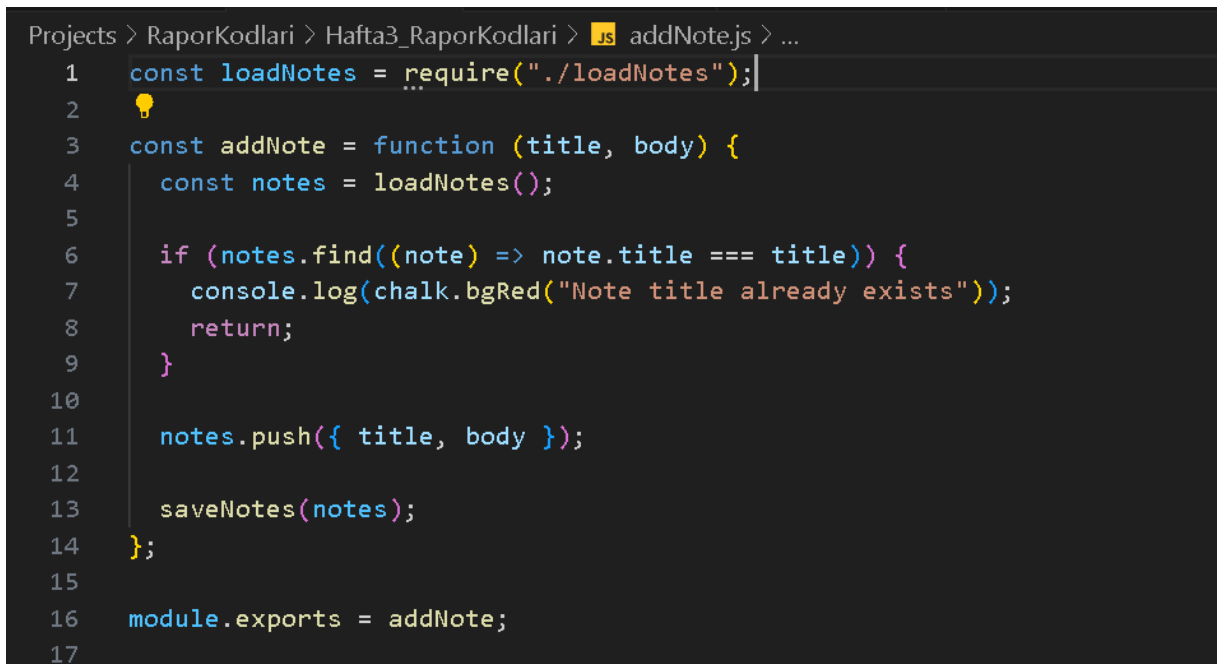
```
JS index.js JS addNote.js JS loadNotes.js notes.json X notes.js
Projects > Raporkodlari > Hafta3_Raporkodlari > notes > notes.json > ...
1 [
2   {
3     "title": "1. Notun Başlığı",
4     "body": "1. Notun içeriği"
5   }
6 ]
```

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



```
index.js addNote.js loadNotes.js X notes.json notes.js
Projects > Raporkodlari > Hafta3_Raporkodlari > loadNotes.js > loadNotes
1 const { readFileSync } = require("fs");
2
3 const loadNotes = function () {
4   try {
5     const dataBuffer = readFileSync("./notes/notes.json", {
6       encoding: "utf-8",
7     });
8     const dataJSON = dataBuffer.toString();
9     return JSON.parse(dataJSON);
10   } catch (e) {
11     return [];
12   }
13 };
14
15 console.log(loadNotes());
16
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\Hafta3_Raporkodlari
> node loadNotes
[ { title: '1. Notun Başlığı', body: '1. Notun içeriği' } ]
```

Yukarıdaki çıktıdan da görüldüğü üzere dosyamızın içeriğini başarılı bir şekilde okumaktayız. Şimdi de bu fonksiyonu notları ekleme fonksiyonumuza import edelim.



```
Projects > Raporkodlari > Hafta3_Raporkodlari > JS addNote.js > ...
1 const loadNotes = require("../loadNotes");
2
3 const addNote = function (title, body) {
4   const notes = loadNotes();
5
6   if (notes.find((note) => note.title === title)) {
7     console.log(chalk.bgRed("Note title already exists"));
8     return;
9   }
10
11   notes.push({ title, body });
12
13   saveNotes(notes);
14 };
15
16 module.exports = addNote;
17
```

Not ekleme fonksiyonumuza 1. Satırdaki kodu yazalım.

Şimdide notları kaydetme fonksiyonumuzu oluşturalım.

1.3.3 Notları Kaydetme Fonksiyonunun Oluşturulması

Aşağıda yer alan kod parçası, notları bir dosyaya kaydetmek için bir fonksiyon tanımlar. Fonksiyon, notes adında bir dizi parametresi alır ve bu diziyi bir JSON stringine dönüştürerek belirtilen dosyaya yazar.

```
Projects > RaporKodlari > Hafta3_RaporKodlari > JS saveNote.js > ...
1  const { writeFileSync } = require("fs");
2
3  const saveNote = function (notes) {
4    writeFileSync("./notes/notes.json", JSON.stringify(notes));
5  };
6
7  module.exports = saveNote;
```

Şimdi de bu fonksiyonu notları ekleme fonksiyonumuza aşağıdaki görseldeki gibi import edelim.

```
index.js  addNote.js  loadNotes.js  saveNote.js  notes.json
Projects > RaporKodlari > Hafta3_RaporKodlari > addNote.js > ...
1  const loadNotes = require("./loadNotes");
2  const saveNote = require("./saveNote");
3
4  const addNote = function (title, body) {
5    const notes = loadNotes();
6
7    if (notes.find((note) => note.title === title)) {
8      console.log(chalk.bgRed("Note title already exists"));
9      return;
10   }
11
12   notes.push({ title, body });
13   saveNote(notes);
14 };
15
16 module.exports = addNote;
```

1.3.4 Not Silme Fonksiyonunun Oluşturulması

Not silme fonksiyonumuz aşağıdaki görseldeki gibi olacaktır.

```
index.js  addNote.js  removeNote.js  loadNotes.js  saveNote.js  notes.json
Projects > RaporKodlari > Hafta3_RaporKodlari > removeNote.js > removeNote
1  const chalk = require("chalk");
2  const loadNotes = require("./loadNotes");
3  const saveNote = require("./saveNote");
4
5  const removeNote = function (title) {
6    const notes = loadNotes();
7
8    const filteredNotes = notes.filter((note) => note.title !== title);
9
10   if (notes.length > filteredNotes.length) {
11     console.log(chalk.green.inverse("Note removed!"));
12     saveNote(filteredNotes);
13   } else {
14     console.log(chalk.red.inverse("No note found!"));
15   }
16
17   saveNote(filteredNotes);
18 };
19
20 module.exports = removeNote;
```

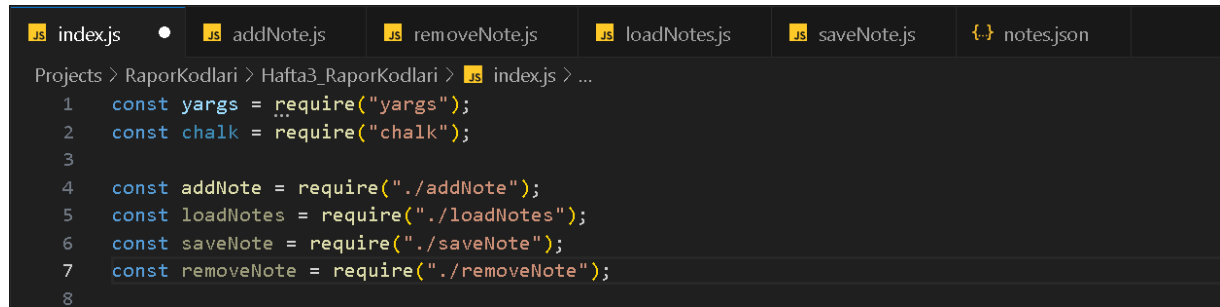
İlk olarak bu fonksiyonda kullanacağımız paketleri ve fonksiyonları import ettik. Daha sonra loadNotes fonksiyonunu kullanarak mevcut notları yükledik ve bu notları notes adlı değişkene atadık. Daha sonra bir array metodu olan filter fonksiyonunu kullanarak notes dizisinden kaldırılacak notları filtreleyerek yeni bir dizi olan filteredNotes dizisini oluşturduk. Burada, title değeri verilen title parametresiyle eşleşmeyen notlar korunur.

Sonra, orijinal notes dizisinin uzunluğu ile filteredNotes dizisinin uzunluğu karşılaştırılır. Eğer orijinal dizi ile filtrelenmiş dizi arasında fark varsa, yani bir notun kaldırıldıysa, kullanıcıya "Note removed!" mesajı gösterilir ve filteredNotes dizisi saveNote fonksiyonuna gönderilir. Bu, notları güncellemek için dosyaya kaydedilir. Eğer not bulunamazsa, yani orijinal dizi ile filtrelenmiş dizi aynı uzunlukta ise, kullanıcıya "No note found!" mesajı gösterilir. Son olarak, filteredNotes dizisi tekrar saveNote fonksiyonuna gönderilir ve notlar dosyaya kaydedilir. Bu işlem, her ihtimale karşı, notların güncel haliyle depolanmasını sağlar.

1.3.5 Uygulamayı Çalıştıracığımız index.js Dosyasının Oluşturulması

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

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 loadNotes = require('./loadNotes');
6 const saveNote = require('./saveNote');
7 const removeNote = require('./removeNote');
```

Bundan sonraki adımda yapmamız gereken şey, Yargs paketinin yardımıyla kullanıcıdan komut satırı yoluyla argüman olarak "add" komutunu almak olacaktır. Bu komut, yeni bir not eklemek için kullanılacaktır. Yargs paketi, kullanıcının belirttiği başlık ve içerik bilgilerini almak üzere gerekli olan argümanları işleyecektir. Daha sonra, bu argümanlar addNote fonksiyonuna iletilerek yeni notun eklenmesi sağlanacaktır.

Bu şekilde, kullanıcı uygulamayı kullanarak yeni notlar ekleyebilecektir. Aşağıdaki görseldeki gibi komutumuzu oluşturabiliriz.

```
index.js  x  addNote.js  removeNote.js  loadNotes.js  saveNote.js  notes.json

Projects > Raporkodlari > Hafta3_Raporkodlari > index.js > builder > title
1  const yargs = require("yargs");
2  const chalk = require("chalk");
3
4  const addNote = require("./addNote");
5  const loadNotes = require("./loadNotes");
6  const saveNote = require("./saveNote");
7  const removeNote = require("./removeNote");
8
9  yargs.command({
10    command: "add",
11    describe: "Add a new note",
12    builder: {
13      title: {
14        describe: "Note title",
15        demandOption: true,
16        type: "string",
17      },
18      body: {
19        describe: "Note body",
20        demandOption: true,
21        type: "string",
22      },
23    },
24    handler: function (argv) {
25      addNote(argv.title, argv.body);
26    },
27  });
```

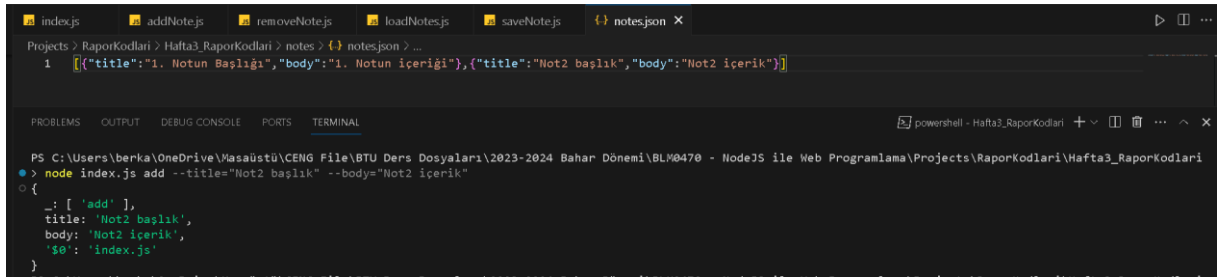
Notları silme ve listeleme fonksiyonları için de benzer şekilde komutlarımızı aşağıdaki gibi oluşturalım.

```
28
29  yargs.command({
30    command: "remove",
31    describe: "Remove a note",
32    builder: {
33      title: {
34        describe: "Note title",
35        demandOption: true,
36        type: "string",
37      },
38    },
39    handler: function (argv) {
40      removeNote(argv.title);
41    },
42  });
43
44  yargs.command({
45    command: "list",
46    describe: "List your notes",
47    handler: function () {
48      const notes = loadNotes();
49      console.log(chalk.inverse("Your notes"));
50      notes.forEach((note) => {
51        console.log(`${note.title}: ${note.body}`);
52      });
53    },
54  });
```

Not alma uygulamamızı bu şekilde tamamlamış oluyoruz. Şimdi de uygulamamızı test edelim.

1.4 Uygulamanın Test Edilmesi

İlk olarak add fonksiyonumuzu test edelim. Aşağıdaki komut ile notes.json dosyamıza girdiğimiz parametrelere göre notes.json dosyasına eklenen veri aşağıdaki görseldeki gibidir.

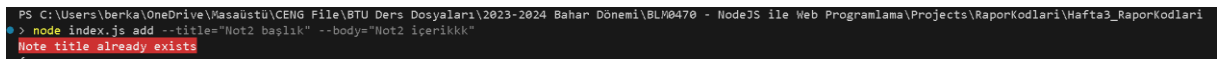


```
index.js  addNotes.js  removeNotes.js  loadNotes.js  saveNotes.js  notes.json x
Projects > Raporkodlari > Hafta3_Raporkodlari > notes > {} notes.json > ...
1 [{"title": "1. Notun Başlığı", "body": "1. Notun içeriği"}, {"title": "Not2 başlık", "body": "Not2 içerik"}]]

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
powershell - Hafta3_Raporkodlari + - - - - - x

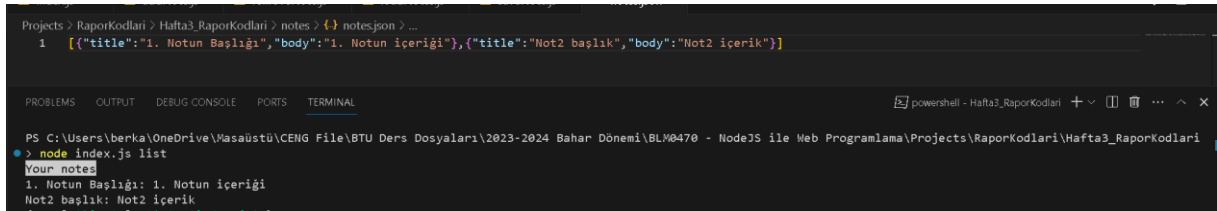
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\Raporkodlari\Hafta3_Raporkodlari
> node index.js add --title="Not2 başlık" --body="Not2 içerik"
{
  _: [ 'add' ],
  title: 'Not2 başlık',
  body: 'Not2 içerik',
  '$0': 'index.js'
}
```

Eğer var olan bir başlığı yine eklemeye çalıştığımızda ise aşağıdaki çıktıyı alıyoruz.



```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\Raporkodlari\Hafta3_Raporkodlari
> node index.js add --title="Not2 başlık" --body="Not2 içerik"
Note title already exists
```

Notları listeleme komutumuzu çalıştırdığımızda ise aşağıdaki çıktıyı almaktayız.

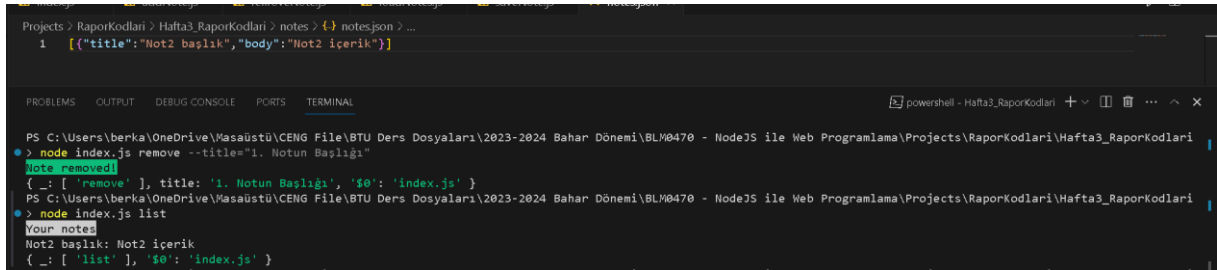


```
Projects > Raporkodlari > Hafta3_Raporkodlari > notes > {} notes.json > ...
1 [{"title": "1. Notun Başlığı", "body": "1. Notun içeriği"}, {"title": "Not2 başlık", "body": "Not2 içerik"}]]

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
powershell - Hafta3_Raporkodlari + - - - - - x

PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\Raporkodlari\Hafta3_Raporkodlari
> node index.js list
Your notes
1. Notun Başlığı: 1. Notun içeriği
Not2 başlık: Not2 içerik
```

Şimdi de ilk notumuzu dosyamızdan silelim ve notlarımızı yine listeleyelim.



```
index.js  addNotes.js  removeNotes.js  loadNotes.js  saveNotes.js  notes.json
Projects > Raporkodlari > Hafta3_Raporkodlari > notes > {} notes.json > ...
1 [{"title": "Not2 başlık", "body": "Not2 içerik"}]]

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
powershell - Hafta3_Raporkodlari + - - - - - x

PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\Raporkodlari\Hafta3_Raporkodlari
> node index.js remove --title="1. Notun Başlığı"
Note removed!
{
  _: [ 'remove' ],
  title: '1. Notun Başlığı',
  '$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\Hafta3_Raporkodlari
> node index.js list
Your notes
Not2 başlık: Not2 içerik
{
  _: [ 'list' ],
  '$0': 'index.js'
}
```

Görüldüğü gibi uygulamamız başarılı bir şekilde çalışmaktadır.

2. Arrow Fonksiyonu

Arrow fonksiyonları, ECMAScript 6 (ES6) ile birlikte JavaScript'e eklenen bir özelliktir. ES6, JavaScript'in 2015 yılında yayınlanan bir sürümüdür ve birçok yeni dil özelliği ve gelişmiş sözdizimler içerir. Arrow fonksiyonları da bu yeniliklerden biridir.

Arrow fonksiyonları, kısa ve temiz bir sözdizim sunarak, özellikle kısa ve tek satırlık işlevlerin tanımlanmasını kolaylaştırır. Klasik fonksiyon ifadelerine göre daha okunabilir ve anlaşılır bir kod yazımı sağlarlar.

Arrow fonksiyonları, => ok işareti ile tanımlanır. Genel sözdizimi şu şekildedir:

```
Projects > RaporKodlari > Hafta3_RaporKodlari > js arrow-func.js > ...  
1  const functionName = (parameters) => {  
2    // function body  
3  };
```

Arrow fonksiyonunun nasıl kullanıldığını bir örnekle daha iyi anlayalım.

```
let sum = (a, b, c) => a + b + c;  
  
/* This arrow function is a shorter form of:  
let sum = function(a, b, c) {  
    return a + b + c;  
}  
*/  
  
alert( sum(1, 3, 5) )    // Output: 9
```

Yukarıdaki kod parçasında görüldüğü üzere `(a, b, c) => a + b + c;` ifadesi *a*, *b* ve *c* adlı 3 adet parametre alır ve bunların toplamını geriye döndürür. Burada *return* anahtar kelimesini kullanmamamızın sebebi eğer geriye döndürülecek ifademiz tek bir ifadeden(statement) oluşuyorsa *return* anahtar kelimesini kullanmaya gerek yoktur.

Çok ifadeli arrow function kullanımı:

Eğer fonksiyonumuz birden fazla ifade içeren bir gövdeden oluşuyorsa bu gövdeyi süslü parantez `{ }` ile sarmalamamız gerekir. Örneğin,

```
let sum = (a, b, c) => {  
    let result = a + b + c;  
    return result;  
}
```

Parametresiz arrow function kullanımı:

Eğer fonksiyonumuz herhangi bir parametre içermiyorsa boş parantez kullanmalıyız. Örneğin,

```
let greet = () => alert("Hello everyone");
```

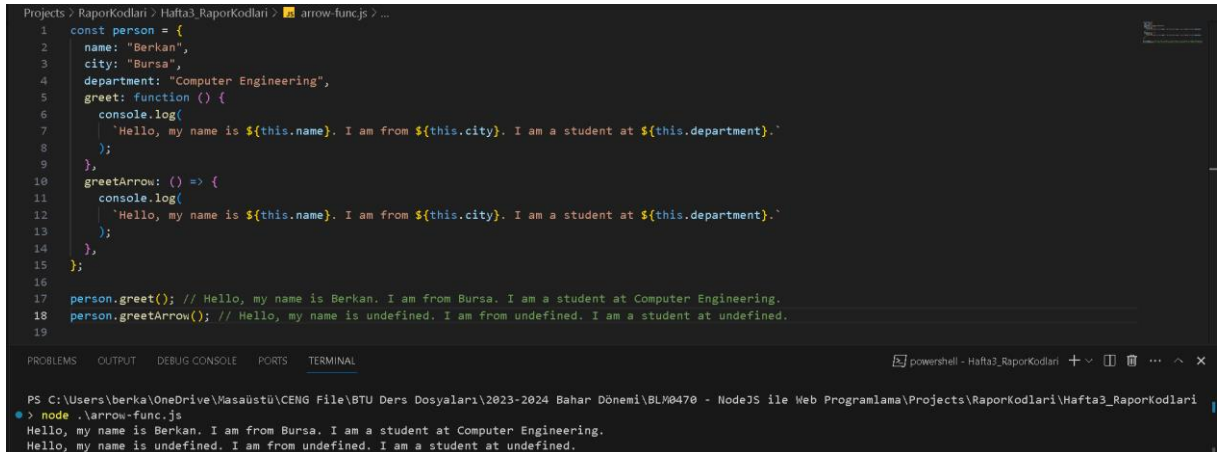
Tek parametrelili arrow function kullanımı:

Eğer fonksiyonumuz tek bir parametre alıyorsa bu parametreyi parantezle sarmalamadan da yazabiliriz. Örneğin,

```
let greet = name => alert(`Hello ${name}`);
```

2.1 Arrow fonksiyonlarda this anahtar kelimesinin kullanımı

Arrow fonksiyonlarıyla this keyword'ünün kullanımı, klasik fonksiyon ifadelerinden farklıdır ve önemli bir noktadır. Arrow fonksiyonları, bir işlemin this bağlamını belirlemek için kendi bağlamını oluşturmazlar. Bunun yerine, arrow fonksiyonları, tanımlandıkları yerdeki dış bağlamı korur ve o bağlamın this değerini miras alırlar.



```
1 const person = {
2   name: "Berkan",
3   city: "Bursa",
4   department: "Computer Engineering",
5   greet: function () {
6     console.log(
7       `Hello, my name is ${this.name}. I am from ${this.city}. I am a student at ${this.department}.`
8     );
9   },
10  greetArrow: () => {
11    console.log(
12      `Hello, my name is ${this.name}. I am from ${this.city}. I am a student at ${this.department}.`
13    );
14  },
15 };
16
17 person.greet(); // Hello, my name is Berkan. I am from Bursa. I am a student at Computer Engineering.
18 person.greetArrow(); // Hello, my name is undefined. I am from undefined. I am a student at undefined.
19
```

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\RaporKodlari\Hafta3_RaporKodlari

➤ node .\arrow-func.js

Hello, my name is Berkan. I am from Bursa. I am a student at Computer Engineering.

Hello, my name is undefined. I am from undefined. I am a student at undefined.

Örneğin yukarıdaki kod parçasında görüldüğü gibi person adında bir obje tanımlanmıştır. Bu objenin içinde iki adet fonksiyon bulunmaktadır: greet ve greetArrow. greet fonksiyonu klasik bir fonksiyon ifadesi olarak tanımlanmıştır, bu nedenle this bağlamı person objesinin kendisidir. Ancak greetArrow fonksiyonu arrow fonksiyonu olarak tanımlanmıştır, bu nedenle this bağlamı arrow fonksiyonunun tanımlandığı kapsamdır, yani genel kapsam olan global kapsamdır. Bu nedenle this.name ve this.age değerleri undefined olarak değerlendirilir.

KAYNAKÇA

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

<https://medium.com/@berkanserbes/javascriptte-fonksiyonlar-b9f81eb9804e>