

BURSA TEKNİK ÜNİVERSİTESİ

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



BLM0470 NodeJS İle Web Programlama

2023-2024 Bahar Dönemi

2.Hafta Raporu

Berkan SERBES – 22360859353

İçindekiler

1. Kullanıcıdan Veri Alma	3
1.1 Process Nesnesi	3
1.1.1 Process.argv Özelliği Yardımıyla Komut Satırından Argüman Alma	3
1.2 Yargs Paketi	7
1.2.1 Yargs Paketinin Yüklenmesi	7
1.2.2 Yargs Paketiyle Birlikte Konsoldan Argüman Alma	8
2. JSON	11
2.1 JSON Nedir?	11
2.2 JSON Nesnesi Oluşturma	11
2.2.1 JSON.stringify() Fonksiyonu	12
2.2.2 JSON.parse() Fonksiyonu	13
Kaynakça	14

1. Kullanıcıdan Veri Alma

Kullanıcıdan veri alma işlemi, bir programın etkileşimli olmasını sağlayan temel bir mekanizmadır. Bu mekanizma, kullanıcıların programla doğrudan etkileşime geçmesine ve programın çalışmasını yönlendirmesine olanak tanır. Örneğin, bir kullanıcı bir metin editöründe dosya adını girdiğinde, program bu girdiyi alır ve kullanıcının isteğine göre ilgili dosyayı açar. Bu, programın kullanıcıların taleplerini hızlı ve etkili bir şekilde yanıtlayabilmesini sağlar.

Ayrıca, kullanıcıdan veri alma işlemi programların kullanıcı dostu olmasını sağlar. Kullanıcılar, programlarla etkileşime geçerken beklentilerini ve gereksinimlerini iletebilirler. Programlar, bu girdileri doğru bir şekilde işlemeli ve uygun geri bildirimlerle kullanıcıya yanıt vermeli. Bu, kullanıcı deneyiminin geliştirilmesine ve programların daha popüler hale gelmesine yardımcı olur.

1.1 Process Nesnesi

NodeJS'te **process** nesnesi, bir NodeJS uygulamasının merkezi kontrol noktasıdır ve işletim sistemi ve uygulama arasında köprü görevi görür. Bu nesne, çalışan uygulamayla ilgili bir dizi bilgi ve işlev sağlayarak uygulamanın davranışını yönetir ve çeşitli sistem düzeyi işlevler sağlar.

Öncelikle, process nesnesi işletim sistemi ve çalışan uygulama hakkında bilgi sağlar. Örneğin, '**process.env**' özelliği, ortam değişkenlerine erişim sağlar. Bu, uygulamanın çalıştığı ortamla ilgili bilgileri içerir. Aynı şekilde, process.arch ve process.platform gibi özellikler, uygulamanın çalıştığı işletim sistemi ve işlemci mimarisi hakkında bilgi sağlar.

Ayrıca, process nesnesi, uygulamanın çalışma zamanındaki davranışını kontrol etmek için bir dizi işlevi de sağlar. Örneğin, **process.exit()** fonksiyonu, NodeJS uygulamasının çalışmasını sonlandırır. Bir çıkış kodu belirtilirse, bu kod NodeJS programının çıkış kodu olarak kullanılır. Bu, uygulamanın başarı durumunu veya hata durumunu döndürmek için kullanılabilir.

Bunun yanı sıra, process nesnesi, işlevsel olarak çeşitli sistem düzeyi işlevleri sağlar. Örneğin, **process.cwd()** işlevi, uygulamanın çalışma dizinini döndürür. **process.memoryUsage()** işlevi, uygulamanın bellek kullanımı hakkında bilgi sağlar. Bu tür işlevler, uygulamanın çalışma zamanında sistem kaynaklarını izlemek ve yönetmek için kullanılabilir. Bu nedenle, process nesnesi, NodeJS uygulamalarının temel yapı taşlarından biridir ve uygulama geliştirme sürecinde önemli bir rol oynar.

1.1.1 Process.argv Özelliği Yardımıyla Komut Satırından Argüman Alma

process.argv, NodeJS'te bir dizi (array) objesidir ve programın çalıştırıldığı komut satırı komutunu ve komut satırına eklenen argümanları içerir. İlk iki öğe, NodeJS çalıştırılabilir dosyasının yolu ve programın adını içerir.

```
index.js 21.02.2024  index.js ...example1  app.js x
RaporKodlari > Hafta2_RaporKodlari > app.js
1 console.log(process.argv);
2 console.log(Array.isArray(process.argv));
3
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\Hafta2_RaporKodlari
> node app.js
[
  'C:\\Program Files\\nodejs\\node.exe',
  'C:\\Users\\berka\\OneDrive\\Masaüstü\\CENG File\\BTU Ders Dosyaları\\2023-2024 Bahar Dönemi\\BLM0470 - NodeJS ile Web Programlama\\Projects\\RaporKodlari\\Hafta2_RaporKodlari\\app.js'
]
true
```

Yukarıdaki çıktıdan görüldüğü üzere, ikinci satırdaki kodun çıktısı process.argv verisinin dizi olup olmadığını kontrol etmektedir ve çıktı olarak **true** değerini döndürmüştür. Bu, bize process.argv verisinin bir dizi(array) olduğunu kanıtlar niteliktedir. İlk satırdaki kodun çıktısı ise bir dizidir ve ilk elemanı NodeJS'in çalıştırılabilir dosyasının yolunu, ikinci eleman ise şu anda çalışan programın dosya yolunu içermektedir.

Şimdide programımızı argüman ekleyerek çalıştıralım ve çıktıyı görelim.

```
RaporKodlari > Hafta2_RaporKodlari > app.js
1 //console.log(Array.isArray(process.argv));
2 console.log(process.argv);
3
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\Hafta2_RaporKodlari
> node app.js 'third argument'
[
  'C:\\Program Files\\nodejs\\node.exe',
  'C:\\Users\\berka\\OneDrive\\Masaüstü\\CENG File\\BTU Ders Dosyaları\\2023-2024 Bahar Dönemi\\BLM0470 - NodeJS ile Web Programlama\\Projects\\RaporKodlari\\Hafta2_RaporKodlari\\app.js',
  'third argument'
]
```

Programı çalıştırırken sadece **node app.js** yazmak yerine node app.js 'e ek olarak bir string değer girerek argüman eklemiş oluyoruz. Çıktıdan görüldüğü üzere dizinin üçüncü elemanı bizim girdiğimiz string verisi olarak eklendi.

Basit bir örnek üzerinde kullanıcıdan veri alalım ve o verinin doğruluğunu kontrol ederek konsol ekranına mesaj yazdıralım.

```
RaporKodlari > Hafta2_RaporKodlari > app.js > ...
4 const command = process.argv[2];
5
6 if (command === "add") {
7   console.log("Ekleme işlemi yapıldı");
8 } else {
9   console.log("Ekleme işlemi yapılamadı");
10 }
Terminal (Ctrl+)
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\Hafta2_RaporKodlari
> node app.js
Ekleme işlemi yapılamadı
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js add
Ekleme işlemi yapıldı
```

Yukarıdaki örnek, kullanıcının girdiği **ilk argüman değerini** command adındaki değişkende tutar ve o değişkenin değeri eğer **'add'** kelimesi ise konsola **'Ekleme işlemi yapıldı'** eğer farklı bir kelime veya herhangi bir argüman değeri girmediyse konsola **'Ekleme işlemi yapılamadı'** mesajını yazdırmaktadır.

Şimdi de örneğimizi daha da genişletelim ve not uygulaması geliştirelim. Bu not uygulamasında kullanıcı, not başlığı ve not içeriğini girerek not oluşturabilir, notları okuyabilir ve notları silebilir olsun.

```
1 const { writeFileSync, readFileSync, unlinkSync } = require("fs");
2
3 const command = process.argv[2];
4 const title = process.argv[3];
5 const content = process.argv[4];
6
7 if (command === "add") {
8   console.log("Yeni not ekleniyor");
9   writeFileSync("./notes/note1.txt", `${title}\n ${content}\n`, {
10     encoding: "utf-8",
11     flag: "a",
12   });
13   console.log("Notunuz başarıyla eklendi");
14 } else if (command === "read") {
15   console.log("Not okunuyor...");
16   const data = readFileSync("./notes/note1.txt", { encoding: "utf-8" });
17   console.log(data);
18 } else if (command === "remove") {
19   console.log("Not siliniyor...");
20   unlinkSync("./notes/note1.txt");
21   console.log("Notunuz başarıyla silindi");
22 }
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js add "Not başlığı" "Not içeriği"
Yeni not ekleniyor
Notunuz başarıyla eklendi
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js read
Not okunuyor...
Not başlığı
Not içeriği
```

Yukarıdaki kod parçası kullanıcıdan sırasıyla komutun adı, not başlığı ve not içeriği olmak üzere üç tane argüman alır. Girilen komut 'add' ise aynı dizinde bulunan notes klasörünün içerisinde notes1.txt adlı dosyaya kullanıcının girdiği not başlığı ve not içeriği argümanlarını o dosyaya yazdırır. Girilen komut 'read' ise notes1.txt adlı dosyanın içeriğini okur. Eğer girilen komut 'remove' ise notes1.txt dosyasını siler.

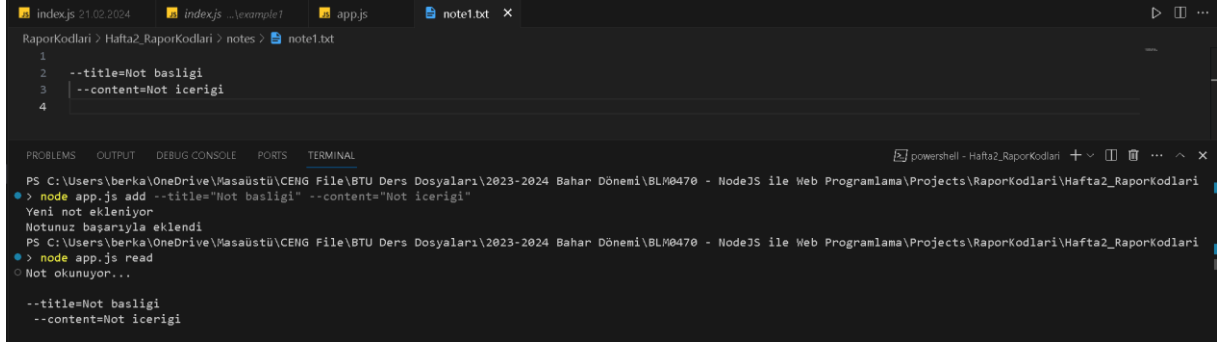
Girdiğimiz argümanlara göre notları yazdırdığımız dosyanın içeriği ise aşağıdaki gibi oluşmuştur.

```
1 Not başlığı
2 Not içeriği
3
```

Bu şekilde argüman girmenin çeşitli dezavantajları vardır. Birinci dezavantajı, kullanıcı argümanlarını hangi sırayla girdiğini bilmeli eğer ilk argümana komutun adı yerine notun başlığını girerse yazılan program doğru bir şekilde çalışmaz. İkinci dezavantajı, kullanıcı girdiği argümanlara ön ek(prefix) eklemek isteyebilir. Örneğin, notun başlığını salt string

olarak göndermek yerine --title='Not başlığı' şeklinde göndermek isteyebilir. Bu durumda kod çalışırken gönderilen argümanın kendisini direkt olarak ilgili dosyaya içerik olarak ekler ve bu da bizim istediğimiz bir durum değil.

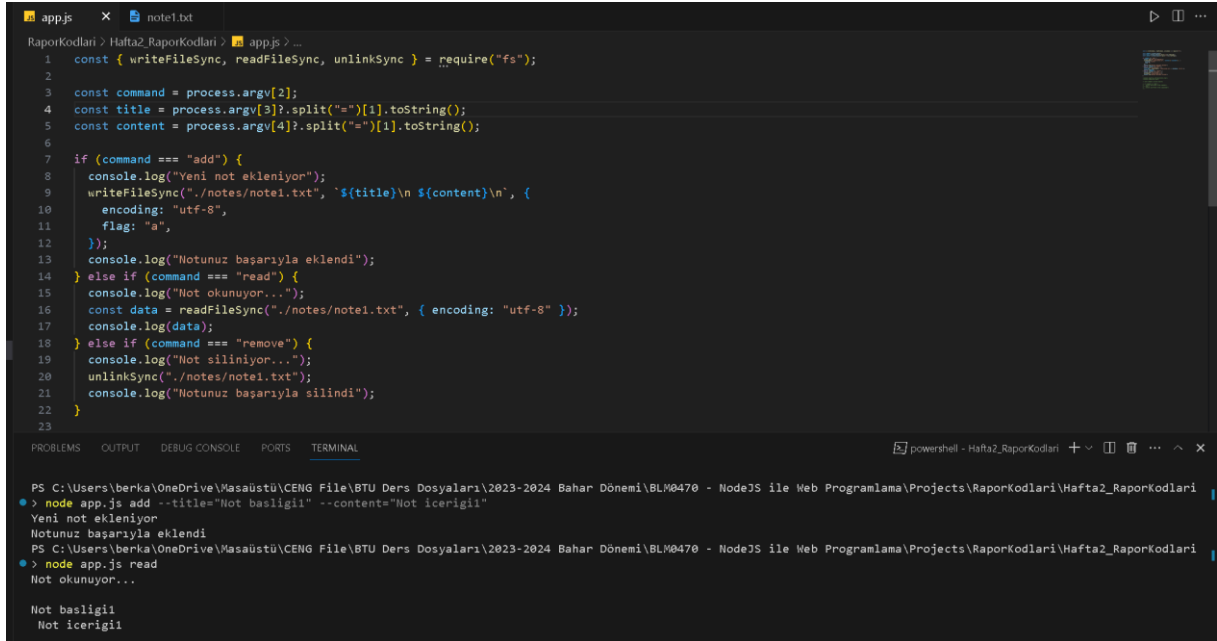
Aşağıdaki kodda görüldüğü üzere argümanları salt olarak göndermek yerine ön ekli biçimde göndermek istediğimizde bu ön ekleri de belgeye yazmaktadır. Bu ön eklerden kurtulmak için girilen argümanı = harfinden sonrasını formatlamamız gerekmektedir.



```
1
2 --title=Not basligi
3 --content=Not icerigi
4
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\M0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js add --title="Not basligi" --content="Not icerigi"
Yeni not ekleniyor
Notunuz başarıyla eklendi
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\M0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js read
Not okunuyor...
```

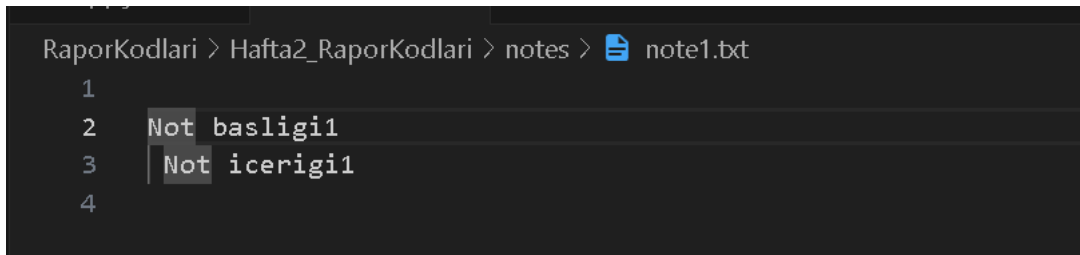
Aşağıdaki kodda 4. ve 5. satırda kullanıcının girdiği title ve content argümanları = harfinden sonrası parse edilmiştir yani = değerinden sonraki yazılar title ve content değişkenlerine atanmıştır.



```
1 const { writeFileSync, readFileSync, unlinkSync } = require("fs");
2
3 const command = process.argv[2];
4 const title = process.argv[3].split("=")[1].toString();
5 const content = process.argv[4].split("=")[1].toString();
6
7 if (command === "add") {
8   console.log("Yeni not ekleniyor");
9   writeFileSync("./notes/note1.txt", `${title}\n ${content}\n`, {
10     encoding: "utf-8",
11     flag: "a",
12   });
13   console.log("Notunuz başarıyla eklendi");
14 } else if (command === "read") {
15   console.log("Not okunuyor...");
16   const data = readFileSync("./notes/note1.txt", { encoding: "utf-8" });
17   console.log(data);
18 } else if (command === "remove") {
19   console.log("Not siliniyor...");
20   unlinkSync("./notes/note1.txt");
21   console.log("Notunuz başarıyla silindi");
22 }
23
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\M0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js add --title="Not basligi" --content="Not icerigi"
Yeni not ekleniyor
Notunuz başarıyla eklendi
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\M0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node app.js read
Not okunuyor...
```

Düzenlenen kodlara göre not içeriğinin çıktısı aşağıdaki gibidir.



```
RaporKodlari > Hafta2_RaporKodlari > notes > note1.txt
1
2 Not basligi
3 Not icerigi
4
```

1.2 Yargs Paketi

Yargs paketi, NodeJS tabanlı konsol uygulamaları için komut satırı arayüzlerini oluşturmayı sağlayan bir pakettir. Konsol uygulamaları, çeşitli komutlar, seçenekler ve argümanlar alabilir. Bu girdilerin analiz edilmesi, doğrulanması ve işlenmesi, özellikle karmaşık veya geniş kapsamlı uygulamalarda zor olabilir. yargs, bu tür zorlukları ortadan kaldırmak ve konsol uygulamalarının geliştirilmesini kolaylaştırmak için tasarlanmıştır.

Yargs kullanarak, konsol uygulamasının komutlarını, seçeneklerini ve argümanlarını tanımlayabilirsiniz. Bu tanımlamalar, uygulamanın hangi komutları desteklediğini, hangi seçenekleri kabul ettiğini ve hangi argümanları beklediğini belirler. Örneğin, bir dosya adını veya bir işlemi belirtmek için argümanlar ekleyebilir ve uygulamanın çeşitli işlevlerini temsil eden komutlar tanımlayabilirsiniz.

Yargs ayrıca kullanıcıya sunulan seçenekleri ve bunların karşılık gelen anahtarlarını belirlemenize olanak tanır. Bu seçenekler, uygulamanın davranışını etkileyen parametreler olabilir. Örneğin, bir **-v** veya **--version** seçeneği ekleyerek uygulamanın sürümünü görüntüleyebilirsiniz. Ayrıca, seçeneklerin varsayılan değerlerini, veri türlerini ve doğrulama kurallarını belirleyebilirsiniz.

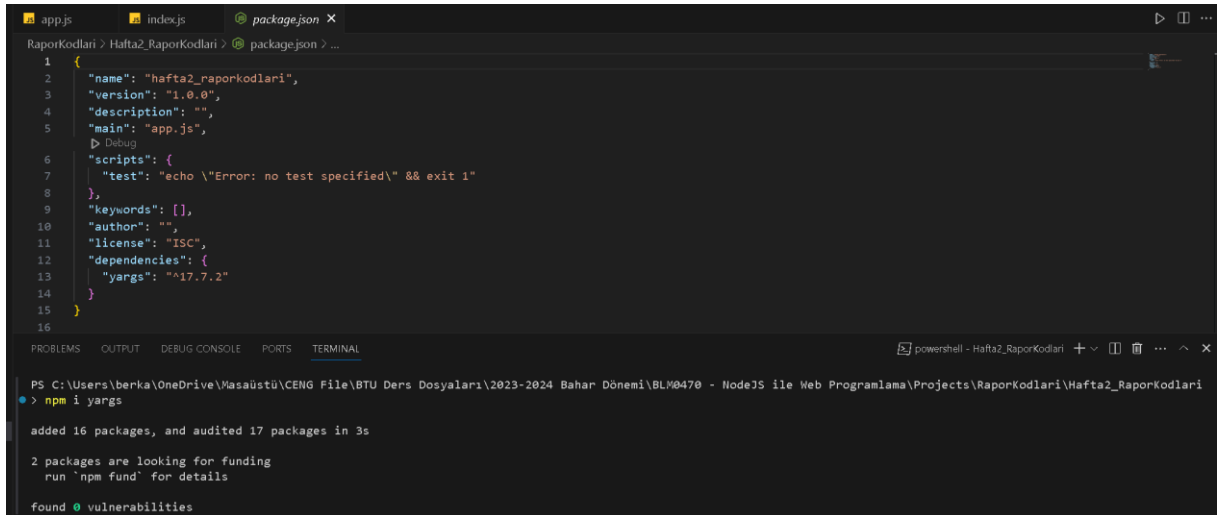
Kullanıcı girdilerinin doğrulanması ve analiz edilmesi de yargs tarafından kolaylaştırılır. yargs, girdilerin doğru formatta ve gereksinimleri karşılayan olduğundan emin olmak için bir dizi doğrulama ve dönüşüm işlevi sağlar. Bu, uygulamanın güvenilir ve tutarlı bir şekilde çalışmasını sağlar.

Son olarak, yargs otomatik olarak belgelendirme oluşturur ve kullanıcılara yardım metinleri sunar. Bu, uygulamanızın nasıl kullanılacağı hakkında kapsamlı bir kaynak sağlar ve kullanıcıların hızlıca başlamasını sağlar.

Bu özellikler sayesinde, yargs kullanarak karmaşık konsol uygulamalarını kolayca oluşturabilir, yönetebilir ve belgelendirebilirsiniz. Bu, geliştirme sürecini hızlandırır, kod tekrarını azaltır ve kullanıcı dostu bir deneyim sağlar.

1.2.1 Yargs Paketinin Yüklenmesi

Yargs paketini npm i yargs veya npm install yargs komutlarını kullanarak uygulamanıza dahil edebilirsiniz. Paketin yüklenip yüklenmediğini package.json dosyasında yer alan dependencies alanında görebilirsiniz aşağıdaki kodda olduğu gibi.



```
1 {
2   "name": "hafta2_raporkodlari",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "yargs": "^17.7.2"
14  }
15 }
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari> npm i yargs

added 16 packages, and audited 17 packages in 3s

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

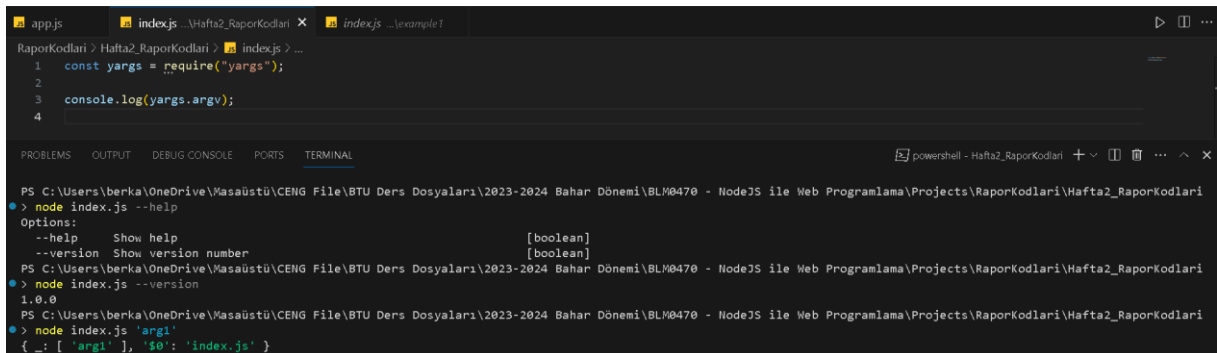
found 0 vulnerabilities
```

1.2.2 Yargs Paketiyle Birlikte Konsoldan Argüman Alma

Yargs paketi ile birlikte gelen bazı komutlar vardır. Bu komutlardan ikisi **--version** ve **--help** komutlarıdır.

Help komutu, kullanıcılara uygulamanın mevcut komutlarını, seçeneklerini ve bunların ne işe yaradığını açıklayan bir yardım mesajı sunar. Kullanıcılar bu komutu kullanarak, uygulamanın özelliklerini keşfedebilir ve nasıl kullanacaklarını öğrenebilirler.

Version komutu, kullanıcıların uygulamanın hangi sürümünü kullandıklarını öğrenmelerini sağlar. Kullanıcılar bu komutu kullanarak, uygulamanın sürüm numarasını hızlıca kontrol edebilirler. Bu özellik, kullanıcıların uygulamanın hangi sürümünde hangi özelliklerin bulunduğunu belirlemelerine ve güncellemeleri takip etmelerine yardımcı olur. Ayrıca, uygulamanın sürüm numarası hakkında bilgi almak, kullanıcıların uygulamayla ilgili herhangi bir sorun yaşadıklarında destek almak için de önemlidir. Aşağıdaki görselde bu iki komutun kullanımına yer verilmiştir.



```
1 const yargs = require("yargs");
2
3 console.log(yargs.argv);
4
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari> node index.js --help
Options:
  --help    Show help                                     [boolean]
  --version Show version number                             [boolean]
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari> node index.js --version
1.0.0
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari> node index.js 'arg1'
{ _: [ 'arg1' ], '$0': 'index.js' }
```

İsterseniz kodunuza **yargs.version('1.1.0')** yazıp çalıştırırsanız versiyonunuz güncellenecektir.


```
1 const yargs = require("yargs");
2
3 yargs.command({
4   command: "add",
5   describe: "Yeni not ekleme komutu",
6   handler: () => {
7     console.log("Yeni not eklendi");
8   },
9 });
10
11 yargs.command({
12   command: "read",
13   describe: "Not okuma komutu",
14   handler: () => {
15     console.log("Not okundu");
16   },
17 });
18
19 yargs.command({
20   command: "remove",
21   describe: "Not silme komutu",
22   handler: () => {
23     console.log("Not silindi");
24   },
25 });
26
27 console.log(yargs.argv);
28
```

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\Hafta2_RaporKodlari

> node index.js --help

index.js [command]

Commands:

index.js add	Yeni not ekleme komutu
index.js read	Not okuma komutu
index.js remove	Not silme komutu

Options:

--help	Show help	[boolean]
--version	Show version number	[boolean]

Yukarıda görselde yazılan kod, yargs paketinin basitçe kullanımını göstermektedir. İlk olarak, **yargs.command()** fonksiyonuyla farklı komutlar tanımlanır. **Bu fonksiyon parametre olarak bir nesne alır** ve bu nesnede komutumuzla ilgili bazı özellikler yer alır. Her bir komut, bir komut adı (command), komutun ne yaptığını açıklayan bir açıklama (describe) ve komutun çalıştırılması durumunda gerçekleştirilecek işlemleri belirten bir işleyici fonksiyon (handler) içerir.

İlk komut "add" komutunu tanımlar. Bu komut, yeni bir not eklemek için kullanılır. İkinci komut "read" komutunu tanımlar ve bu komut notları okumak için kullanılır. Üçüncü komut "remove" komutunu tanımlar ve bu komut notları silmek için kullanılır.

Her komut, ilgili işlemin gerçekleştirilmesinden sorumlu olan bir işleyici fonksiyon içerir. Örneğin, "add" komutu çalıştırıldığında, "Yeni not eklendi" mesajını görüntüleyen bir işleyici fonksiyon çalışır. Aşağıdaki görselde tanımlanan komut adlarının argüman olarak gönderilmesi sonucu programın çıktısı yer almaktadır.

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node index.js add
Yeni not eklendi
{ _: [ 'add' ], '$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\Hafta2_RaporKodlari
> node index.js read
Not okundu
{ _: [ 'read' ], '$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\Hafta2_RaporKodlari
> node index.js remove
Not silindi
{ _: [ 'remove' ], '$0': 'index.js' }
```

Şimdi de asıl önemli olan noktaya gelelim. Yargs paketini kullanmaya başlamadan önce kullanıcı komut girerken ön ek eklerse bu ön ekin metin dosyasında görünmemesi için kodumuzu çeşitli javascript fonksiyonlarıyla formatlamıştık. Yargs paketinde ise bu işlemi otomatik olarak yapan **builder** özelliği vardır. Builder özelliği, yargs paketinde kullanılan bir özelliktir ve komutlara bağlı olarak argümanları ve seçenekleri tanımlamak için kullanılır. Bir komutun builder özelliği, o komuta özgü argümanları ve seçenekleri tanımlamak için kullanılır. Bu argümanlar ve seçenekler, kullanıcının komutu çağırırken belirtmesi gereken

değerleri temsil eder. Builder özelliği, her bir argüman veya seçeneğin adını (key), açıklamasını (describe), veri türünü (type), zorunluluk durumunu (demandOption) ve diğer özel ayarları belirtmek için kullanılabilir. Bu özellik, komutların kullanımını daha belirgin hale getirir ve kullanıcıların hangi argümanları veya seçenekleri belirtmeleri gerektiği konusunda rehberlik sağlar. Ayrıca, builder özelliği sayesinde kullanıcılar, komutlar arasında tutarlılık sağlanmasına ve doğru girdilerin sağlanmasına yardımcı olacak şekilde yapılandırılabilir.

Şimdi de bu anlatılanları kodumuzda pratiğe dökelim.

```
1 const yargs = require("yargs");
2
3 yargs.command({
4   command: "add",
5   describe: "Yeni not ekleme komutu",
6   builder: {
7     title: {
8       describe: "Not başlığı",
9       demandOption: true,
10      type: "string",
11    },
12    content: {
13      describe: "Not içeriği",
14      demandOption: true,
15      type: "string",
16    },
17  },
18  handler: () => {
19    console.log("Yeni not eklendi");
20  },
21 });
22
```

PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari

> node index.js add
index.js add

Yeni not ekleme komutu

Options:

--help	Show help	[boolean]
--version	Show version number	[boolean]
--title	Not başlığı	[string] [required]
--content	Not içeriği	[string] [required]

Missing required arguments: title, content

PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari

> node index.js add --title="Baslik" --content="icerik"

Yeni not eklendi

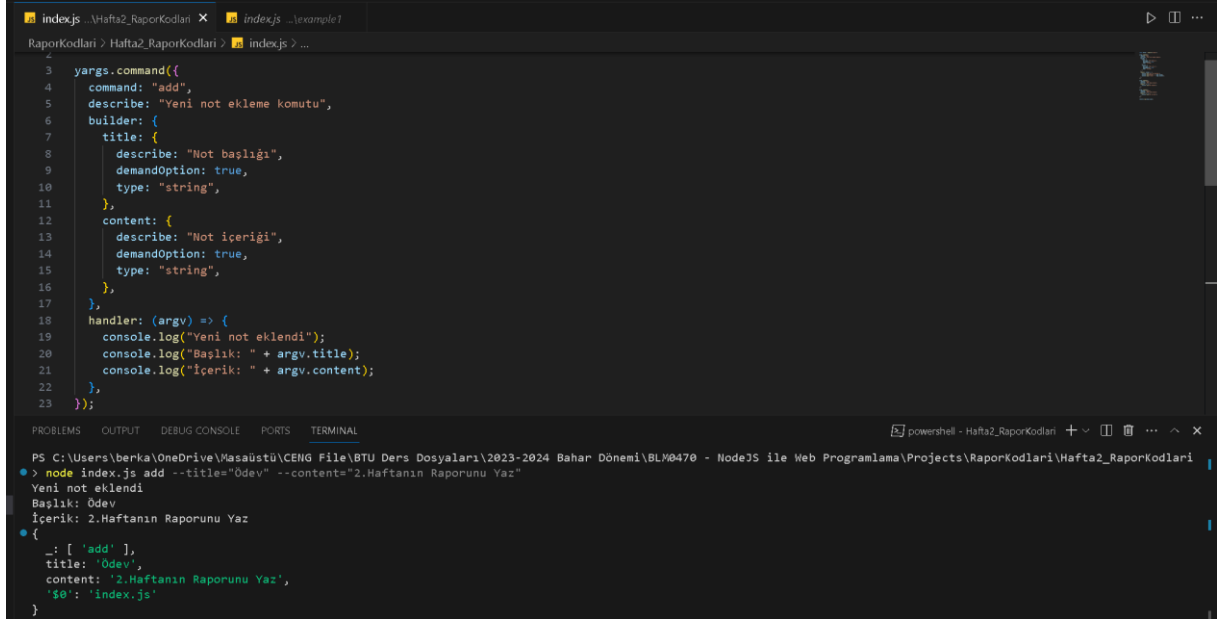
```
{ _: [ 'add' ], title: 'Baslik', content: 'icerik', '$0': 'index.js' }
```

Yukarıdaki örnekte, builder özelliği "add" komutu için kullanılmıştır. Bu komut, "title" ve "content" adında iki argüman içerir. Her bir argüman için aşağıdaki bilgiler belirtilmiştir:

- 'title': Argümanın adı
 - 'describe': Argümanın açıklamasını belirtir.
 - 'type': Girilecek argümanın hangi tipte olması gerektiğini belirtir.
 - 'demandOption': Argümanın zorunlu olup olmadığını belirtir. true olarak ayarlandığı için bu argümanın zorunlu olduğu belirtilir.
- 'content': Argümanın adı
 - Diğer veriler 'title' argümanı ile aynıdır.

Bu şekilde, builder özelliği kullanılarak "add" komutuna ait argümanlar ve seçenekler tanımlanmıştır. Kullanıcılar bu komutu çağırırken **--title** ve **--content** seçeneklerini belirtmek zorundadır. Bu sayede, komutun doğru ve tutarlı bir şekilde kullanılmasına yardımcı olunur. Eğer kullanıcı demandOption değeri true olan argümanları argüman olarak göndermezse yukarıdaki görselde olduğu gibi **Missing required argument** hatası alır.

Kullanıcının komut satırından girdiği argümanları yakalamak için handler fonksiyonunda parametre olarak argv değeri girerek bu fonksiyonları nesne şeklinde yakalayıp daha sonra **argv.argümanın adı** şeklinde girdileri alabiliriz. Aşağıda bunu gösteren örnek kod yer almaktadır.



```
1 // index.js
2
3 yargs.command({
4   command: "add",
5   describe: "Yeni not ekleme komutu",
6   builder: {
7     title: {
8       describe: "Not başlığı",
9       demandOption: true,
10      type: "string",
11    },
12    content: {
13      describe: "Not içeriği",
14      demandOption: true,
15      type: "string",
16    },
17  },
18  handler: (argv) => {
19    console.log("Yeni not eklendi");
20    console.log("Başlık: " + argv.title);
21    console.log("İçerik: " + argv.content);
22  },
23 });
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node index.js add --title="Ödev" --content="2.Haftanın Raporunu Yaz"
Yeni not eklendi
Başlık: Ödev
İçerik: 2.Haftanın Raporunu Yaz
{
  _: [ 'add' ],
  title: 'Ödev',
  content: '2.Haftanın Raporunu Yaz',
  '$0': 'index.js'
}
```

2. JSON

2.1 JSON Nedir?

JSON (JavaScript Object Notation), veri alışverişi için kullanılan hafif, metin tabanlı bir veri formatıdır özellikle web uygulamaları arasında yaygın olarak kullanılmaktadır. JSON'un temel yapısı, anahtar-değer (key-value) çiftlerinden oluşan nesneler ve sıralı öğelerden oluşan dizilerdir. Anahtarlar birer metin olarak yazılırken değerler sayılar, metinler, boolean değerler, diziler, başka JSON nesneleri veya null olabilirler. JSON dizileri ise virgülle ayrılmış bir dizi değer içerirler. Veriler basit bir sözdizimle temsil edilir, bu da okunabilirliği artırır. Ayrıca JSON, dil bağımsızdır, yani birçok programlama dilinde desteklenir. Bu nedenle, farklı platformlar arasında veri alışverişi yaparken uyumluluk sağlar. JSON'un avantajları arasında hafif olması ve doğruluk sağlaması da vardır. Metin tabanlı bir formatta olduğu için dosya boyutunu minimize eder ve veri transferi ve depolama maliyetlerini azaltır. Ayrıca, JSON verileri doğrulama için kolaydır, çünkü basit bir yapıya sahiptir ve hataları tespit etmek ve düzeltmek kolaydır. Genel olarak, JSON web tabanlı uygulamalarda, API'lerde, yapılandırma dosyalarında, veri depolama ve iletişimde ve birçok başka senaryoda yaygın olarak kullanılan bir veri formatıdır. Basit, hafif ve dil bağımsız olması nedeniyle, modern yazılım geliştirme süreçlerinde temel bir bileşen olarak kabul edilir.

2.2 JSON Nesnesi Oluşturma

JSON'da bir nesne oluşturmanın temel yapısı, anahtar-değer çiftlerinden oluşur. Anahtarlar bir metin olmalıdır ve bir değerle eşleştirilirler. Değerler, herhangi bir JSON veri türü olabilir: bir sayı, bir metin dizisi, bir başka JSON nesnesi, bir dizi veya boşluk.

Örneğin, bir kitabın bilgilerini içeren JSON nesnesi aşağıdaki gibi oluşturulabilir.

```
index.js  deneme.json  book.json  main.js
RaporKodlari > Hafta2_RaporKodlari > book.json > ...
1  {
2    "title": "What Men Live By",
3    "author": "Lev Nikolayevich Tolstoy",
4    "year": 1885,
5    "genre": "Fiction",
6    "quotes": [
7      "There is only one time that is important--Now! It is the most important time because it is the only time when we have any power.",
8      "Fear is the sharpest spur to a man's actions.",
9      "The truly wise man seeks knowledge not for his own benefit, but to share it with others."
10   ]
11 }
```

Aşağıdaki kod parçası, bir JSON dosyasından kitap bilgilerini içeren bir JSON nesnesini (book) içe aktarıyor ve bu nesneyi konsola yazdırıyor. Ardından, bu nesnenin veri türünü kontrol ediyor.

`const book = require("./book.json")` : Bu satır, `./book.json` dosyasından kitap bilgilerini içeren bir JSON nesnesini içe aktarır. `require()` fonksiyonu, JSON dosyasını bir JavaScript nesnesine dönüştürür ve bu nesneyi `book` değişkenine atar.

`console.log(book)` : Bu satır, `book` değişkenini konsola yazdırır. Bu, içeri aktarılan JSON nesnesinin tüm içeriğini konsola görüntüler.

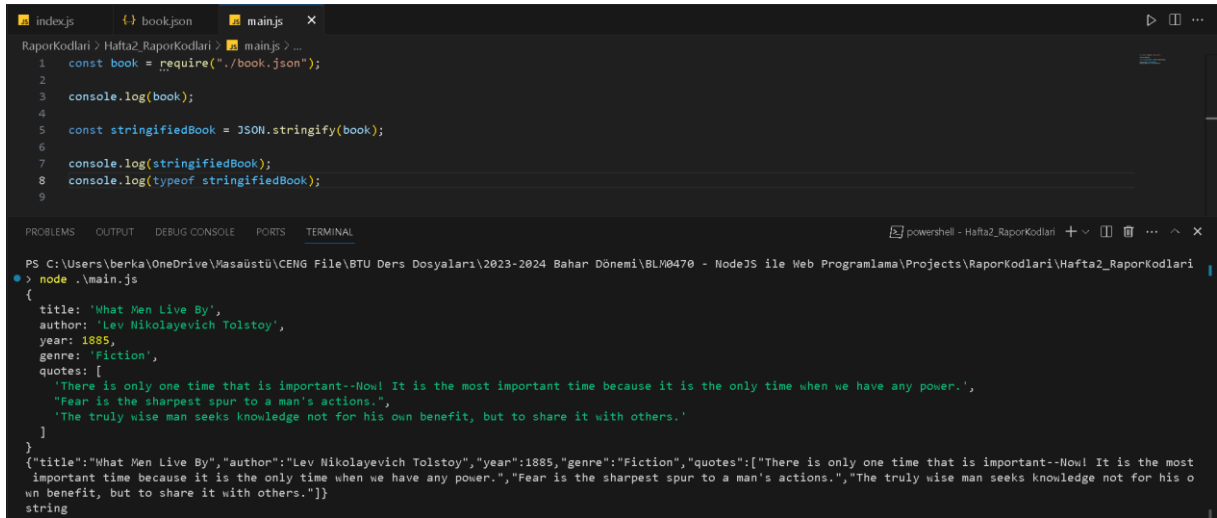
`console.log(typeof book)` : Bu satır, `book` değişkeninin veri türünü konsola yazdırır. Bu, içeri aktarılan JSON nesnesinin JavaScript'te bir nesne (object) türünde olduğunu gösterir.

```
index.js  book.json  main.js  x
RaporKodlari > Hafta2_RaporKodlari > main.js > ...
1  const book = require("./book.json");
2
3  console.log(book);
4  console.log(typeof book);
5
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
powershell - Hafta2_RaporKodlari
PS C:\Users\benka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari > node main.js
• {
  title: 'What Men Live By',
  author: 'Lev Nikolayevich Tolstoy',
  year: 1885,
  genre: 'Fiction',
  quotes: [
    'There is only one time that is important--Now! It is the most important time because it is the only time when we have any power.',
    'Fear is the sharpest spur to a man's actions.',
    'The truly wise man seeks knowledge not for his own benefit, but to share it with others.'
  ]
}
object
```

2.2.1 JSON.stringify() Fonksiyonu

JSON.stringify() fonksiyonu, JavaScript'te kullanılan bir fonksiyondur ve bir JavaScript nesnesini JSON biçimine dönüştürür. Bu fonksiyon, bir JavaScript nesnesini parametre olarak alır ve bu nesneyi JSON biçimine dönüştürerek bir JSON dizesi olarak döndürür. JSON dizesi, JSON biçimindeki verileri metin tabanlı olarak temsil eder.

Örneğin, aşağıdaki JavaScript nesnesini JSON dizisine dönüştüren bir kullanım örneği:



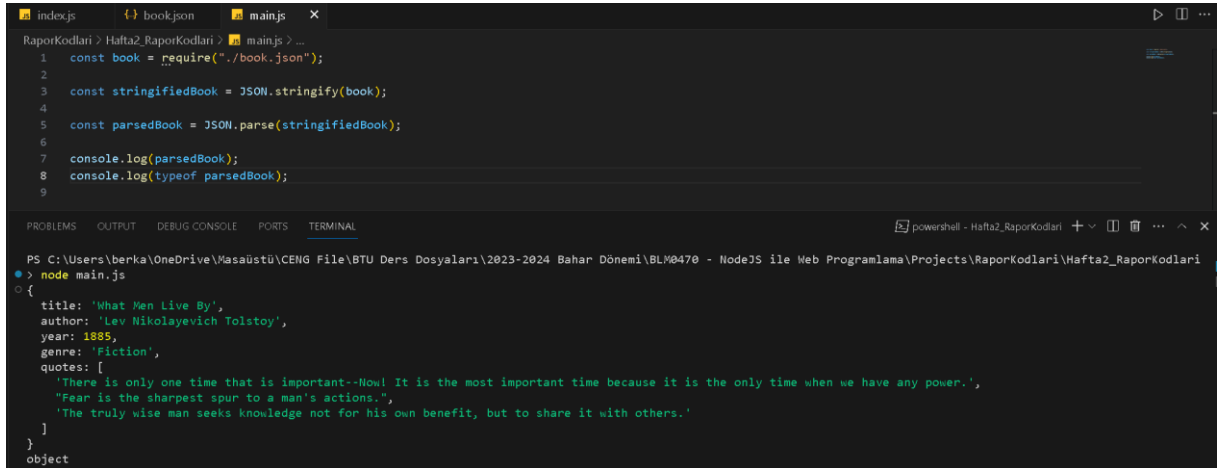
```
index.js book.json main.js X
RaporKodlari > Hafta2_RaporKodlari > main.js > ...
1 const book = require("../book.json");
2
3 console.log(book);
4
5 const stringifiedBook = JSON.stringify(book);
6
7 console.log(stringifiedBook);
8 console.log(typeof stringifiedBook);
9

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
powershell - Hafta2_RaporKodlari + - - - X
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node .\main.js
{
  title: 'What Men Live By',
  author: 'Lev Nikolayevich Tolstoy',
  year: 1885,
  genre: 'Fiction',
  quotes: [
    'There is only one time that is important--Now! It is the most important time because it is the only time when we have any power.',
    'Fear is the sharpest spur to a man's actions.',
    'The truly wise man seeks knowledge not for his own benefit, but to share it with others.'
  ]
}
{"title":"What Men Live By","author":"Lev Nikolayevich Tolstoy","year":1885,"genre":"Fiction","quotes":["There is only one time that is important--Now! It is the most important time because it is the only time when we have any power.", "Fear is the sharpest spur to a man's actions.", "The truly wise man seeks knowledge not for his own benefit, but to share it with others."]}
```

2.2.2 JSON.parse() Fonksiyonu

JSON.parse() fonksiyonu, JSON formatındaki bir metin dizesini alır ve bu metin dizesini JavaScript nesnesine dönüştürür. JSON formatındaki bir metin dizesi, **JSON.parse()** fonksiyonuna parametre olarak verilir ve bu fonksiyon tarafından işlenerek bir JavaScript nesnesi olarak döndürülür. Bu dönüşüm işlemi, JSON biçimindeki verileri JavaScript nesnelerine geri dönüştürerek, bu verileri JavaScript ortamında kullanılabilir hale getirir.

Örneğin, aşağıdaki JSON biçimini JavaScript nesnesine dönüştüren bir kullanım örneği:



```
index.js book.json main.js X
RaporKodlari > Hafta2_RaporKodlari > main.js > ...
1 const book = require("../book.json");
2
3 const stringifiedBook = JSON.stringify(book);
4
5 const parsedBook = JSON.parse(stringifiedBook);
6
7 console.log(parsedBook);
8 console.log(typeof parsedBook);
9

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
powershell - Hafta2_RaporKodlari + - - - X
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta2_RaporKodlari
> node main.js
{
  title: 'What Men Live By',
  author: 'Lev Nikolayevich Tolstoy',
  year: 1885,
  genre: 'Fiction',
  quotes: [
    'There is only one time that is important--Now! It is the most important time because it is the only time when we have any power.',
    'Fear is the sharpest spur to a man's actions.',
    'The truly wise man seeks knowledge not for his own benefit, but to share it with others.'
  ]
}
object
```

Kaynakça

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