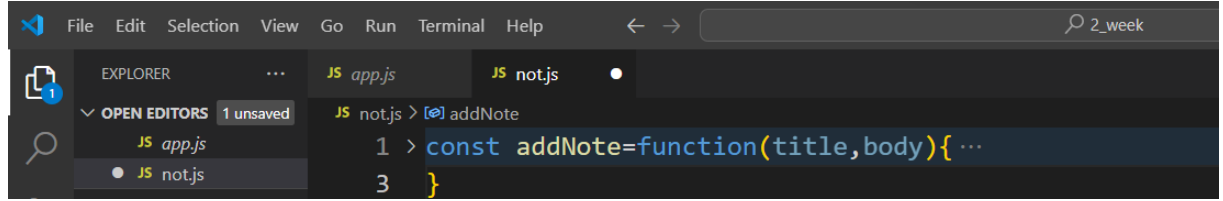


## 6 Mart 2024 3.Hafta Node js Teori Dersi Haftalık Rapor

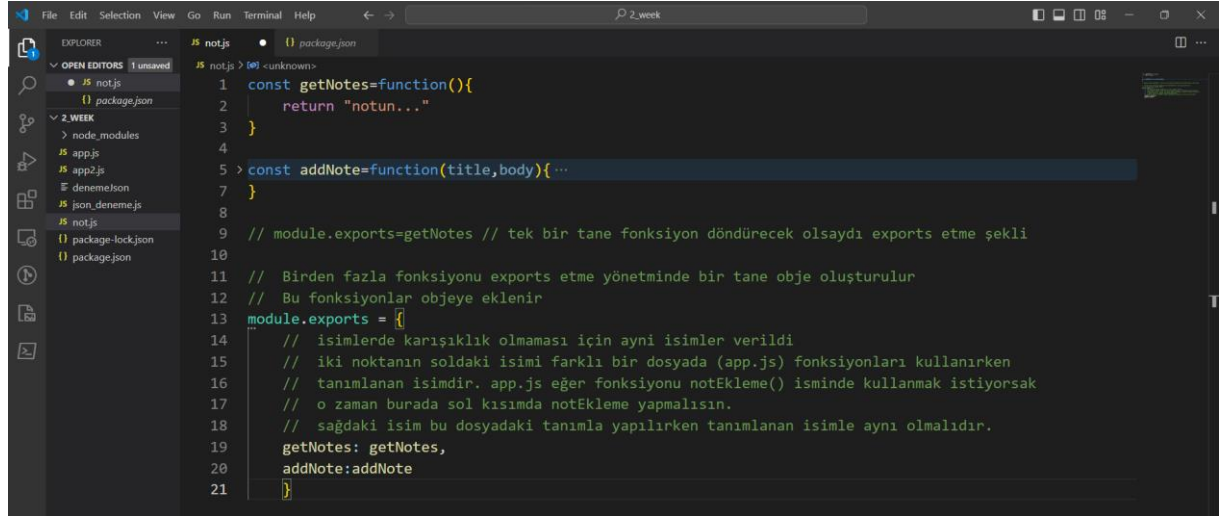
Not alma uygulaması geliştiriyoruz ve aldığımız notları json formatında dosyaya kaydedelim.

Notun başlığı ve içeriği bulunmaktadır. Projede geçen hafta oluşturduğumuz app2.js yi kullanacağız. Not.js dosyasını oluşturalım. Burada addNote isimli fonksiyon tanımlaması yapalım 2 tane parametre alacaktır title ve body olacaktır. AddNote fonksiyonu notları dosyaya kaydedecektir.



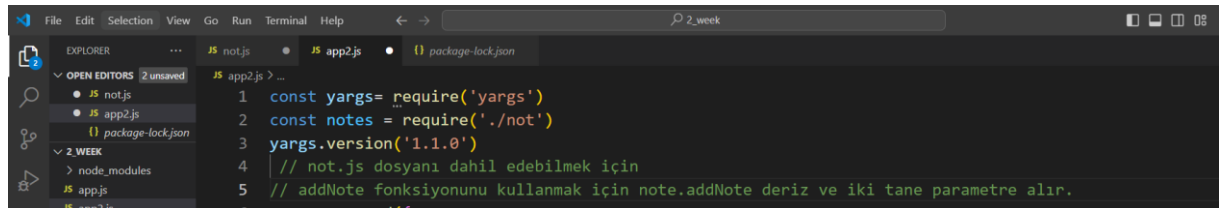
```
File Edit Selection View Go Run Terminal Help
EXPLORER JS app.js JS not.js
OPEN EDITORS 1 unsaved JS not.js > addNote
1 > const addNote=function(title,body){ ...
3 }
```

Fonksiyonu tanımladıktan sonra export etmemiz gerekli. Export etme işlemini modül.export diyerek gerçekleştirdik ama bu yöntem tek bir fonksiyon için geçerli bizim dosyamızda getNotes isimli fonksiyonumuz bulunduğu için bu şekilde export edemeyiz.



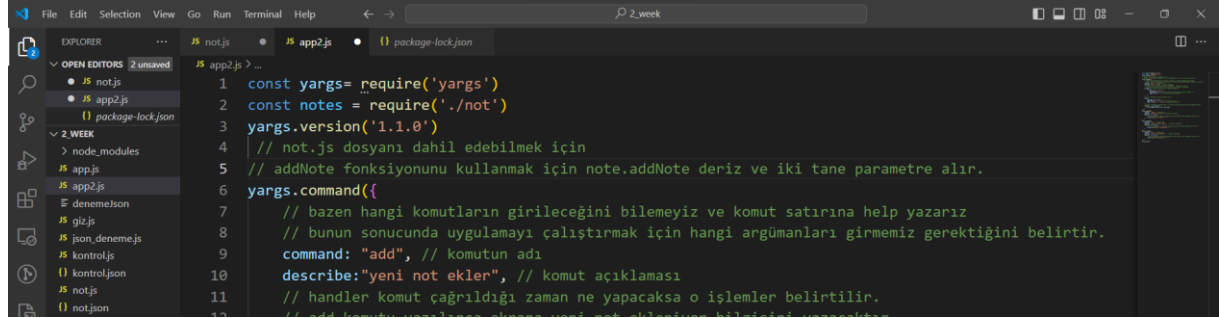
```
File Edit Selection View Go Run Terminal Help
EXPLORER JS not.js package.json
OPEN EDITORS 1 unsaved JS not.js > [unknown]
1 const getNotes=function(){
2   return "notun..."
3 }
4
5 > const addNote=function(title,body){ ...
7 }
8
9 // module.exports=getNotes // tek bir tane fonksiyon döndürecek olsaydı exports etme şekli
10
11 // Birden fazla fonksiyonu exports etme yönetimde bir tane obje oluşturulur
12 // Bu fonksiyonlar objeye eklenir
13 module.exports = {
14   // isimlerde karışıklık olmaması için aynı isimler verildi
15   // iki noktanın soldaki isimi farklı bir dosyada (app.js) fonksiyonları kullanırken
16   // tanımlanan isimdir. app.js eğer fonksiyonu notEkleme() isminde kullanmak istiyorsak
17   // o zaman burada sol kısımda notEkleme yapmalısın.
18   // sağdaki isim bu dosyadaki tanımla yapılırken tanımlanan isimle aynı olmalıdır.
19   getNotes: getNotes,
20   addNote:addNote
21 }
```

Şimdi ise not.js dosyasını geçen hafta oluşturduğumuz app.js dosyasına dahil edelim. Bunun require anahtar kelimesi ile yapalım içine parametre olarak dosya yolunu vermemiz lazım aynı dizinde bulunduğu belirtmek için ./ kullanılır ve devamında eklemek istediğimiz dosya adını yazarız.



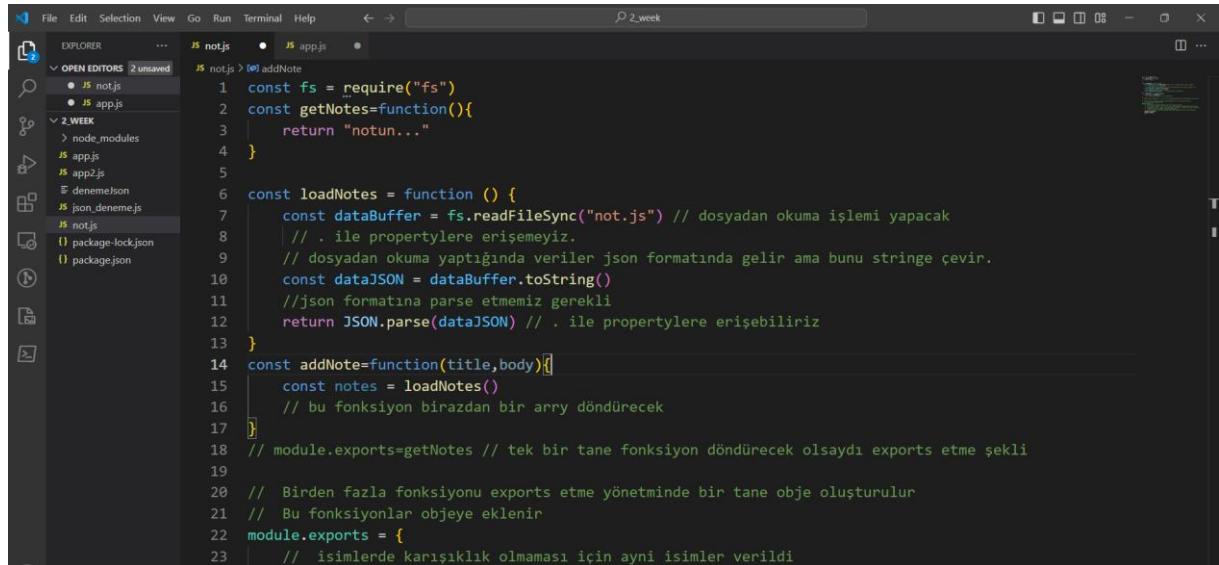
```
File Edit Selection View Go Run Terminal Help
EXPLORER JS not.js JS app2.js package-lock.json
OPEN EDITORS 2 unsaved JS not.js JS app2.js > ...
1 const yargs= require('yargs')
2 const notes = require('./not')
3 yargs.version('1.1.0')
4 // not.js dosyasını dahil edebilmek için
5 // addNote fonksiyonunu kullanmak için note.addNote deriz ve iki tane parametre alır.
```

Not.app dosyasında eskiden sadece getNotes fonksiyonumuz vardı ve bunu app.js de require ederken const ile isimlendirme verildiğinde direkt olarak const getNotes= require("./not") dememiz yeterliydi fakat burada require kelimesi ile artık bir fonksiyon gelmiyor bir obje geliyor dolayısıyla bu şekilde isimlendirme yanıltıcı olur daha genel geçer bir isimlendirme vermemiz gerekir o yüzden const notes =require("./not) kullanıldı. Fonksiyonları kullanabilmemiz için note.fonksiyon\_ismi ile erişebiliriz.

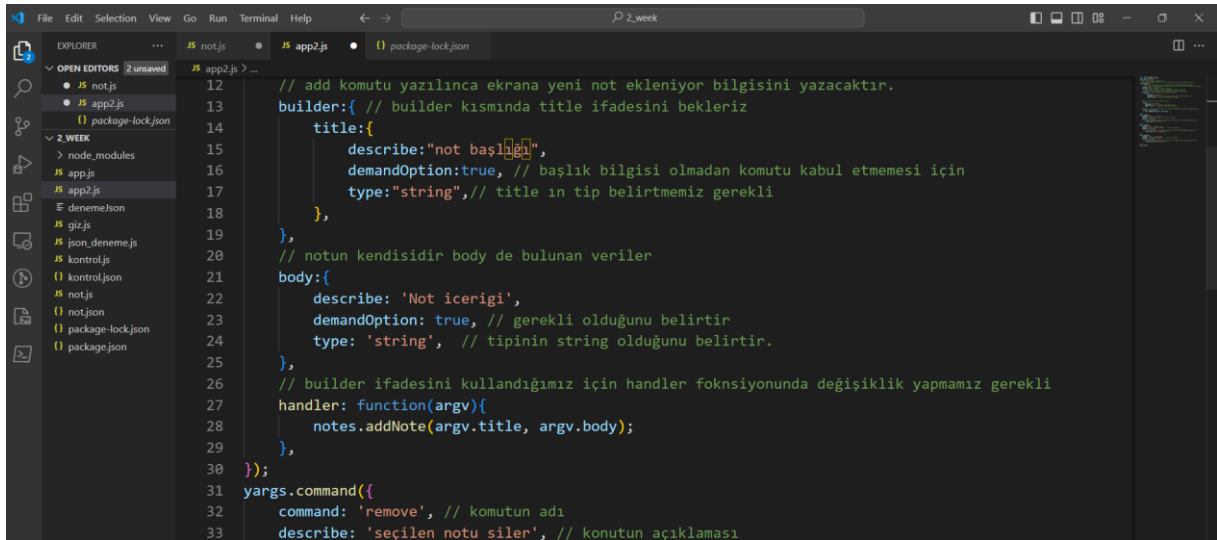


```
1 const yargs= require('yargs')
2 const notes = require('./not')
3 yargs.version('1.1.0')
4 // not.js dosyasını dahil edebilmek için
5 // addNote fonksiyonunu kullanmak için note.addNote deriz ve iki tane parametre alır.
6 yargs.command({
7   // bazen hangi komutların girileceğini bilemeyiz ve komut satırına help yazarız
8   // bunun sonucunda uygulamayı çalıştırmak için hangi argümanları girmemiz gerektiğini belirtir.
9   command: "add", // komutun adı
10  describe:"yeni not ekler", // komut açıklaması
11  // handler komut çağrıldığı zaman ne yapacaksa o işlemler belirtilir.
```

Madem biz not ekleme işlemi yapıyoruz o zaman eklediğimiz notu görmek isteriz bunun için not.js dosyasına geri dönelim. Dosyaya json formatında kaydetmeye çalışalım. Yeni bir fonksiyona (loadNotes) ihtiyacımız var çünkü ilk olarak mevcut notları çekip daha sonra yeni notu mevcut notların üzerine eklememiz ve dosyaya kaydetmemiz lazım. Karşılaştırma işlemi yapacağımız için dosyanın sonuna append ile ekleme işlemi yapmıyoruz. Dosya sistemlerinde okuma ve yazma işlemi yapmamız gerekecek bunun için fs modülü ekleyim ve dosyadan okuma işlemini loadNotes fonksiyonunda gerçekleştirelim.

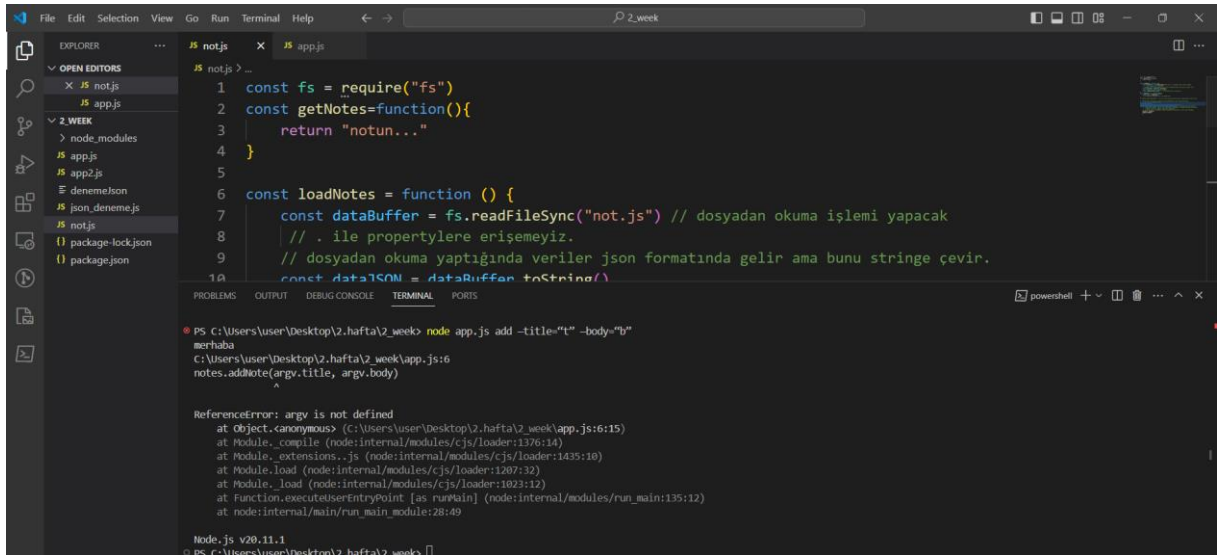


```
1 const fs = require("fs")
2 const getNotes=function(){
3   return "notun..."
4 }
5
6 const loadNotes = function () {
7   const dataBuffer = fs.readFileSync("not.js") // dosyadan okuma işlemi yapacak
8   // . ile propertylere erişemeyiz.
9   // dosyadan okuma yaptığında veriler json formatında gelir ama bunu stringe çevir.
10  const dataJSON = dataBuffer.toString()
11  //json formatına parse etmemiz gerekli
12  return JSON.parse(dataJSON) // . ile propertylere erişebiliriz
13 }
14 const addNote=function(title,body){
15   const notes = loadNotes()
16   // bu fonksiyon birazdan bir arry döndürecek
17 }
18 // module.exports=getNotes // tek bir tane fonksiyon döndürecek olsaydı exports etme şekli
19
20 // Birden fazla fonksiyonu exports etme yönetimde bir tane obje oluşturulur
21 // Bu fonksiyonlar objeye eklenir
22 module.exports = {
23   // isimlerde karışıklık olmaması için aynı isimler verildi
```



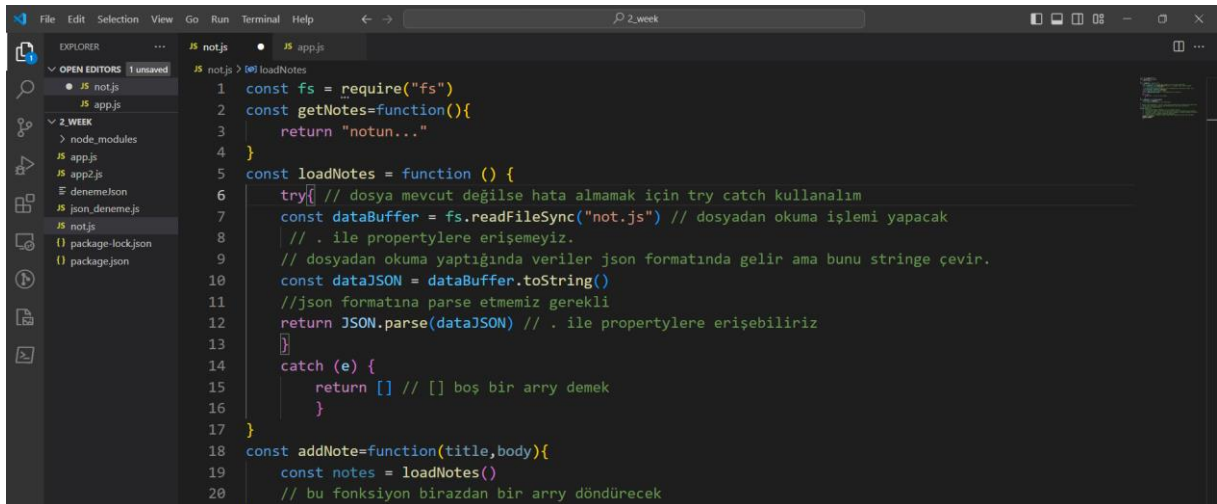
```
12 // add komutu yazılınca ekrana yeni not ekleniyor bilgisini yazacaktır.
13 builder: { // builder kısmında title ifadesini bekleriz
14   title: {
15     describe: "not başlığı",
16     demandOption: true, // başlık bilgisi olmadan komutu kabul etmemesi için
17     type: "string", // title in tip belirtmemiz gerekli
18   },
19 },
20 // notun kendisidir body de bulunan veriler
21 body: {
22   describe: 'Not içeriği',
23   demandOption: true, // gerekli olduğunu belirtir
24   type: 'string', // tipinin string olduğunu belirtir.
25 },
26 // builder ifadesini kullandığımız için handler fonksiyonunda değişiklik yapmamız gerekli
27 handler: function(argv) {
28   notes.addNote(argv.title, argv.body);
29 },
30 });
31 yargs.command({
32   command: 'remove', // komutun adı
33   describe: 'seçilen notu siler', // konunun açıklaması
```

Kodları kaydedelim ve çalıştıralım. Terminale `node app2.js add --title="t" --body="b"` yazdığımız zaman bu not kayıtlı olmadığı için hata alırız.



```
1 const fs = require("fs")
2 const getNotes=function(){
3   return "notun..."
4 }
5
6 const loadNotes = function () {
7   const dataBuffer = fs.readFileSync("not.js") // dosyadan okuma işlemi yapacak
8   // . ile propertylere erişemeyiz.
9   // dosyadan okuma yaptığında veriler json formatında gelir ama bunu stringe çevir.
10  const dataJSON = dataBuffer.toString()
11
12  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
13
14  PS C:\Users\User\Desktop\2.hafta\2_week> node app.js add --title="t" --body="b"
15  merhaba
16  C:\Users\User\Desktop\2.hafta\2_week> node app.js:6
17  notes.addNote(argv.title, argv.body)
18
19  ReferenceError: argv is not defined
20  at Object.<anonymous> (C:\Users\User\Desktop\2.hafta\2_week\app.js:6:15)
21  at Module._compile (node:internal/modules/cjs/loader:1376:14)
22  at Module._extensions..js (node:internal/modules/cjs/loader:1435:10)
23  at Module.load (node:internal/modules/cjs/loader:1207:32)
24  at Module._load (node:internal/modules/cjs/loader:1023:12)
25  at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:135:12)
26  at node:internal/main/run_main_module:28:49
27
28  Node.js v20.11.1
29  PS C:\Users\User\Desktop\2.hafta\2_week>
```

Dosya mevcut olmayabilir o zaman hataları önlemek için try catch blokları kullanalım. Dosya mevcut değilse demek ki henüz bir not eklememişsiniz.



```
1 const fs = require("fs")
2 const getNotes=function(){
3   return "notun..."
4 }
5
6 const loadNotes = function () {
7   try { // dosya mevcut değilse hata almamak için try catch kullanalım
8     const dataBuffer = fs.readFileSync("not.js") // dosyadan okuma işlemi yapacak
9     // . ile propertylere erişemeyiz.
10    // dosyadan okuma yaptığında veriler json formatında gelir ama bunu stringe çevir.
11    const dataJSON = dataBuffer.toString()
12    //json formatına parse etmemiz gerekli
13    return JSON.parse(dataJSON) // . ile propertylere erişebiliriz
14  } catch (e) {
15    return [] // [] boş bir array demek
16  }
17 }
18 const addNote=function(title,body){
19   const notes = loadNotes()
20   // bu fonksiyon birazdan bir array döndürecek
```

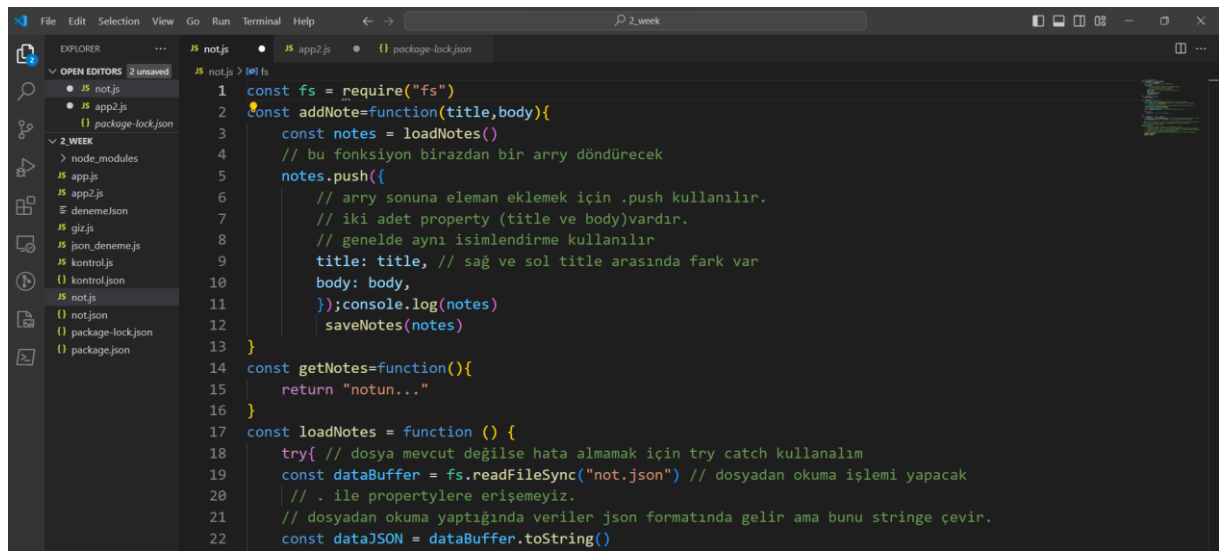
Not: Try catch bloklarını özelden genele doğru yazarız.

Notes artık bir arraydir. Array üzerine eleman eklemek için push kullanalım. Array obje alıyor ve bu objenin içinde iki tane property var bunlar title ve body propertyleridir.

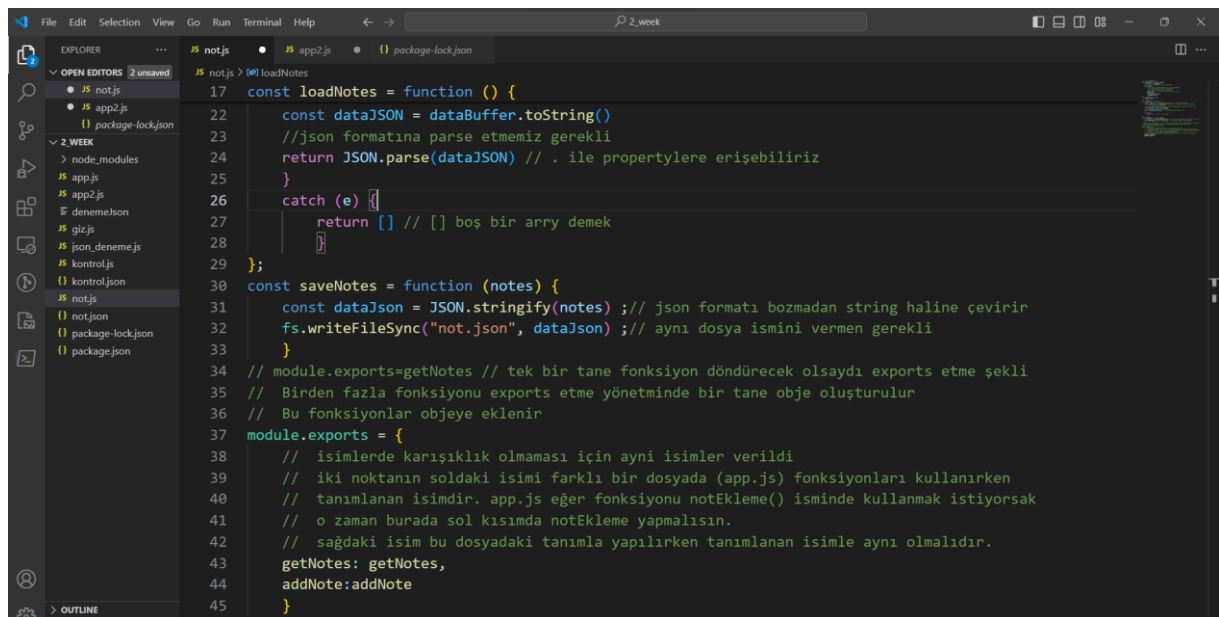
```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="başlık"-- body="içerik"
{ _: [ 'add', 'body=içerik' ], title: 'başlık--', '$0': 'app2.js' }
başlık: başlık--
içerik : undefined
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Bu kısımda dosyaya kopyalama işlemini gerçekleştirelim ve bunun için saveNotes() isimli fonksiyon tanımlayalım. Fonksiyona parametre olarak notes isimli bir array gönderdik.

SaveNotes() fonksiyonunu addNote() fonksiyonunda çağırma işlemi yapılır.



```
1 const fs = require("fs")
2 const addNote=function(title,body){
3   const notes = loadNotes()
4   // bu fonksiyon birazdan bir array döndürecek
5   notes.push({
6     // array sonuna eleman eklemek için .push kullanılır.
7     // iki adet property (title ve body)vardır.
8     // genelde aynı isimlendirme kullanılır
9     title: title, // sağ ve sol title arasında fark var
10    body: body,
11  });console.log(notes)
12  saveNotes(notes)
13 }
14 const getNotes=function(){
15   return "notun..."
16 }
17 const loadNotes = function () {
18   try{ // dosya mevcut değilse hata almamak için try catch kullanalım
19     const dataBuffer = fs.readFileSync("not.json") // dosyadan okuma işlemi yapacak
20     // . ile propertylere erişemeyiz.
21     // dosyadan okuma yaptığında veriler json formatında gelir ama bunu stringe çevir.
22     const dataJSON = dataBuffer.toString()
```



```
17 const loadNotes = function () {
22   const dataJSON = dataBuffer.toString()
23   //json formatına parse etmemiz gerekli
24   return JSON.parse(dataJSON) // . ile propertylere erişebiliriz
25 }
26 catch (e) {
27   return [] // [] boş bir array demek
28 }
29 };
30 const saveNotes = function (notes) {
31   const dataJSON = JSON.stringify(notes) ;// json formatı bozmadan string haline çevirir
32   fs.writeFileSync("not.json", dataJSON) ;// aynı dosya ismini vermen gerekli
33 }
34 // module.exports=getNotes // tek bir tane fonksiyon döndürecek olsaydı exports etme şekli
35 // Birden fazla fonksiyonu exports etme yönetiminde bir tane obje oluşturulur
36 // Bu fonksiyonlar objeye eklenir
37 module.exports = {
38   // isimlerde karışıklık olmaması için aynı isimler verildi
39   // iki noktanın soldaki ismi farklı bir dosyada (app.js) fonksiyonları kullanırken
40   // tanımlanan isimdir. app.js eğer fonksiyonu notEkleme() isminde kullanmak istiyorsa
41   // o zaman burada sol kısımda notEkleme yapmalısın.
42   // sağdaki isim bu dosyadaki tanımla yapılırken tanımlanan isimle aynı olmalıdır.
43   getNotes: getNotes,
44   addNote:addNote
45 }
```

Terminalde çalıştırdığımızda çıktısı şu şekilde olur.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="iki" --body="iki"
[
  { title: 'bir', body: 'bir' },
  { title: 'bir', body: 'iki' },
  { title: 'iki', body: 'bir' },
  { title: 'iki', body: 'iki' }
]
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Her bir notu ekler ve listeler. Her yeni eklenen not bir önceki nottan sonra eklenir. Yani append işlemini gerçekleştirmiş oldu.

Şimdi uygulamamıza yeni bir özellik ekleyelim bu özellik ise aynı başlıkta not eklenemiyor olsun. Aynı başlıklı not eklemeye izin verilmesin bunu yapmanın yolu mevcut notun title ile yeni notun title karşılaştırılmalı eğer eskilerden birisi ile aynı ise kabul etmeyip uyarı mesajını ekrana basmaktır. Bu işlemi gerçekleştirirken filter() fonksiyonunu kullanacağız filter() fonksiyonu arrayin içinde bir filtreleme işlemi gerçekleştirir. Filter fonksiyonu içinde bir fonksiyon tanımlayalım bu fonksiyona parametre olarak note verildi. Oluşturduğumuz fonksiyonun içeriğini kendimiz belirleyelim. Fonksiyonda eşitlik kontrolü yapalım. Note bir array bu arrayin elemanlarına teker teker erişim sağlayacaktır. AddNote fonksiyonunda yeni kod eklemeleri yapılır.

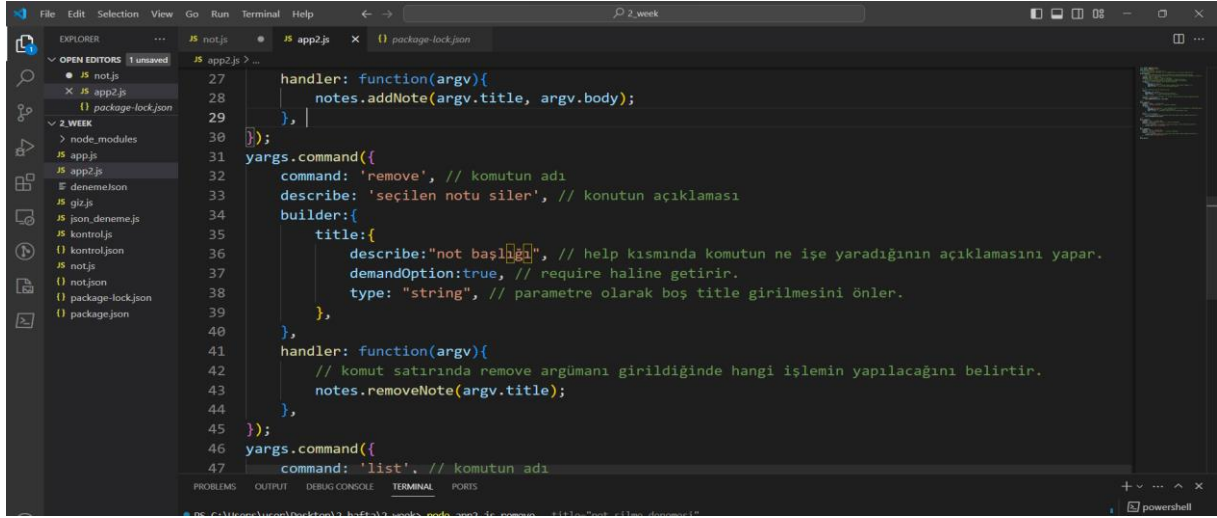
```
File Edit Selection View Go Run Terminal Help 2_week
EXPLORER JS not.js JS app2.js package-lock.json
OPEN EDITORS JS not.js
not.js
2 WEEK
node_modules
app.js
app2.js
deneme1son
giz.js
json_deneme.js
kontrol.js
kontrol.json
not.js
not.json
package-lock.json
package.json
OUTLINE
1 const fs = require("fs")
2 const addNote=function(title,body){
3   const notes = loadNotes()
4   // bu fonksiyon birazdan bir array döndürecek
5   const duplicateNotes = notes.filter(function (note) { // eğer aynı başlık olursa uyarı vermek için
6     // filter fonksiyonu kullanıldı
7     return note.title === title
8   });
9   if(duplicateNotes.length==0){
10    // yeni eklenen not daha önceki notlardan farklı o zaman ekleme işlemi yapılır.
11    notes.push({
12      // array sonuna eleman eklemek için .push kullanılır.
13      // iki adet property (title ve body)vardır.
14      // genelde aynı isimlendirme kullanılır
15      title: title, // sağ ve sol title arasında fark var
16      body: body,
17    });console.log(notes)
18    saveNotes(notes)
19  }else {
20    // bu kısım çalışırsa demekki daha önce girilen başlıktan not ekleme yapılıyor
21    // uyarı mesajı verilir.
22    console.log("bu başlık daha önce eklendi.Not eklenemez.")
23  }
24 }
25 const getNotes=function(){
```

Çıktısında aynı başlık bilgisine sahip not eklediğimizde hata mesajı verecektir.

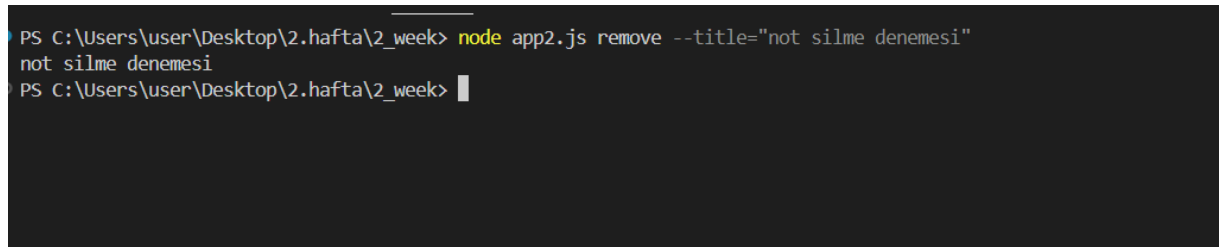
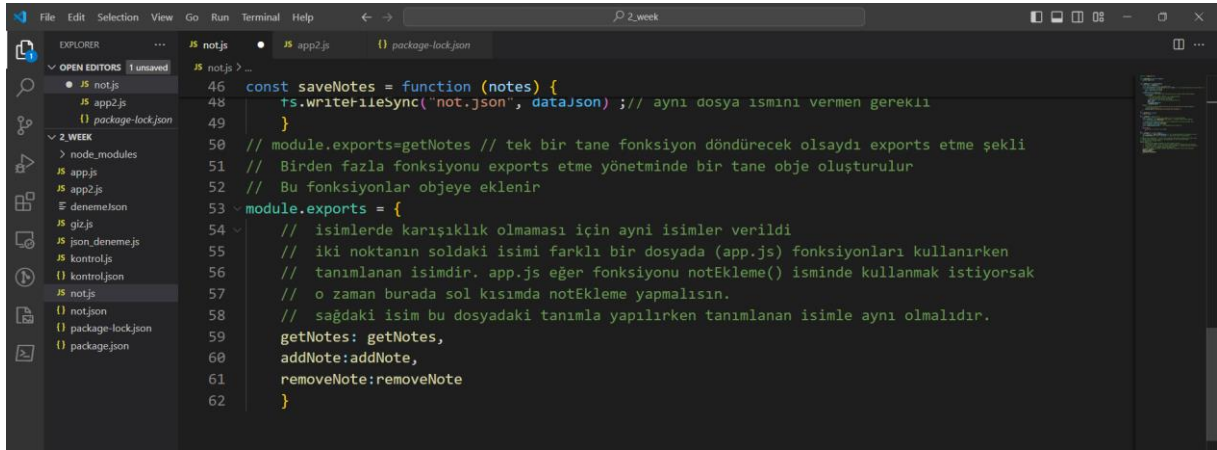
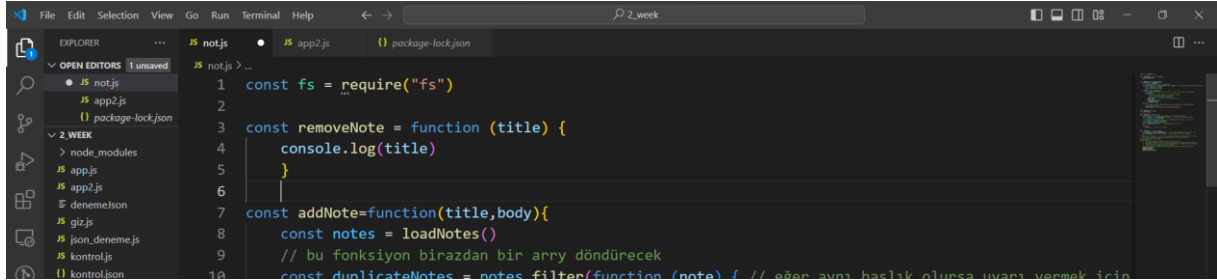
```
[
  { title: 'bir', body: 'bir' },
  { title: 'bir', body: 'iki' },
  { title: 'iki', body: 'bir' },
  { title: 'iki', body: 'iki' }
]
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="iki" --body="iki"
bu başlık daha önce eklendi.Not eklenemez.
```



Bu kısımda not silme işlemlerine yönelelim. App2.js dosyasına gidelim ve remove komutu düzenleyelim.



Not.js içinde silme fonksiyonunu tanımlayalım ve sonunda export edelim.

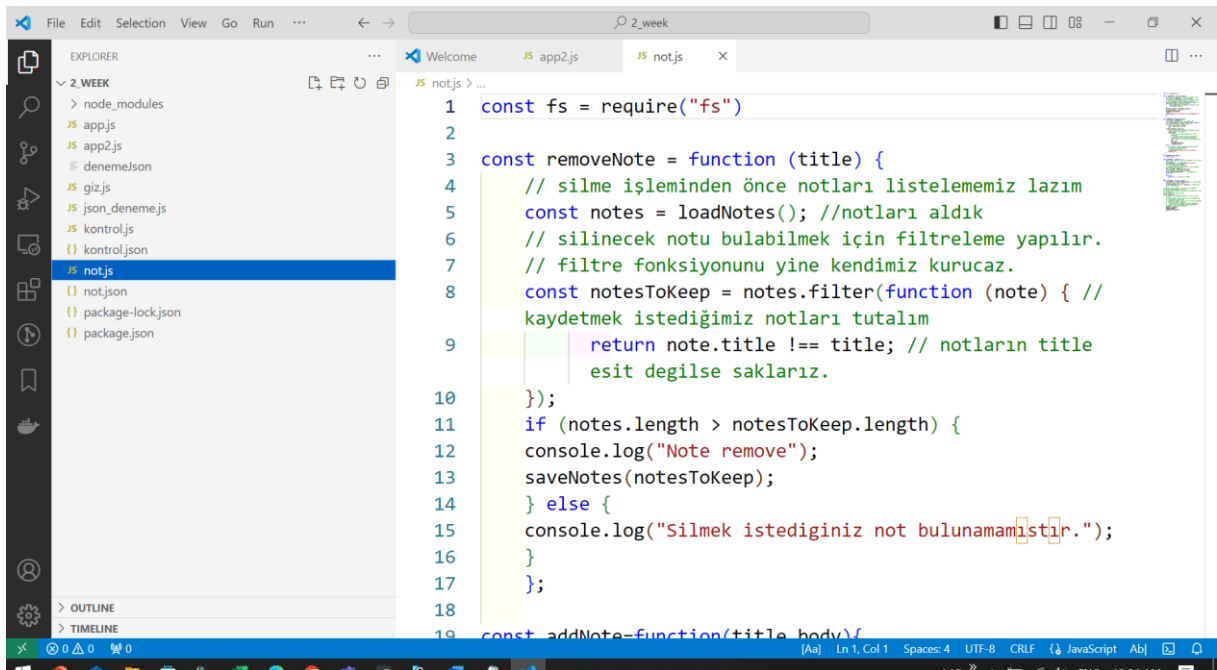


Not silme işlemini yapmadan önce tüm notları listelememiz lazım bu yüzden loadNotes fonksiyonunu çağıralım. Silme işleminden önce silmemiz gereken notu tespit etmemiz lazım.

App.js dosyamızda:

```
3 });  
1 yargs.command({  
2   command: 'remove', // komutun adı  
3   describe: 'seçilen notu siler', // konutun açıklaması  
4   builder: {  
5     title: {  
5       describe: "not başlığı", // help kısmında komutun  
        ne işe yaradığının açıklamasını yapar.  
7       demandOption: true, // require haline getirir.  
3       type: "string", // parametre olarak boş title  
        girilmesini önler.  
3     },  
3   },  
1   handler: function(argv) {  
2     // komut satırında remove argümanı girildiğinde  
        hangi işlemin yapılacağını belirtir.  
3     notes.removeNote(argv.title);  
4   },  
5 });
```

şeklinde ekleme yapalım.



```
1 const fs = require("fs")  
2  
3 const removeNote = function (title) {  
4   // silme işleminden önce notları listelememiz lazım  
5   const notes = loadNotes(); //notları aldık  
6   // silinecek notu bulabilmek için filtreleme yapılır.  
7   // filtre fonksiyonunu yine kendimiz kurucuz.  
8   const notesToKeep = notes.filter(function (note) { //  
9     // kaydetmek istediğimiz notları tutalım  
10    return note.title !== title; // notların title  
    esit değilse saklarız.  
11  });  
12  if (notes.length > notesToKeep.length) {  
13    console.log("Note remove");  
14    saveNotes(notesToKeep);  
15  } else {  
16    console.log("Silmek istediğiniz not bulunamamıştır.");  
17  }  
18  };  
19  const addNote=function(title, body){
```

Bu kod parçası ise not.js de eklenen kodlardır. Silmemiz gereken başlığı belirlemek için filtreleme işleme yaptık bunun sonucunda elde edilen verileri bir değişken üzerine atayıp koşul yapıları ile kıyaslama yaptık.

Bu kısımda projemize chalk modülünü ekleyerek devam edelim. Not.app dosyasına ilk olarak require anahtar kelimesi ile chalk modülünü ekleyelim.

```
Const chalk = require('chalk')
```

Daha sonra ekrana yazdığımız ifadelerin renklerini değiştirebilmek için

```
console.log(chalk.green.inverse('New note added!')) \\ arka plan rengi yeşil
```

```
console.log(chalk.red.inverse('Note title taken!')) \\ arka plan rengi kırmızı
```

kodları ekleyelim .

Son olarak arrow fonksiyonuna bakalım normalde temel olarak tanımlanan bir fonksiyonu arrow ile daha kullanışlı ve estetik bir hale getirebiliriz örenk olarak aşağıdaki kodu inceleyebilirsiniz.

Normal kare alma fonksiyonu şu şekilde iken:

```
const square = function (x) {  
  return x*x  
}
```

Bu fonksiyonu arrow haline getirelim:

```
const square = (x) => {  
  return x*x  
}
```

Not: Arrow fonksiyonun da bir noktaya dikkat etmemiz lazım o nokta ise this parametresi arrow çok da kullanıma elverişli olan bir parametre değildir.