

# Node Js Ödevi

## 3<sup>rd</sup> Week

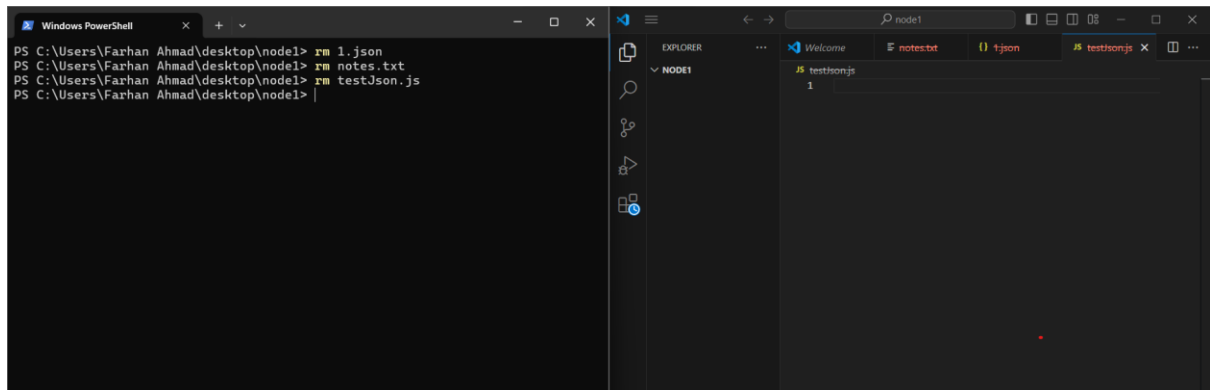
**Farhan Ahmad**

**20360859096**

Geçen hafta “**add**”, “**remove**” ve “**list**” argümanlarını incelemiştik. Bu hafta ise oluşturduğumuz notları nasıl silebileceğimizi(**Delete**) ve daha sonra farklı konuları ele alacağız. Yani, bu hafta not silme işlemlerine odaklanacağız ve ardından başka konuları ele alacağız.

İlk olarak, notes.txt, testJson.js ve 1.json dosyalarının silinmesi ve ardından notes.js dosyasında yeni bir not oluşturmak için bir fonksiyon tanımlanması gerekmektedir.

### 1. Dosya Silmesi



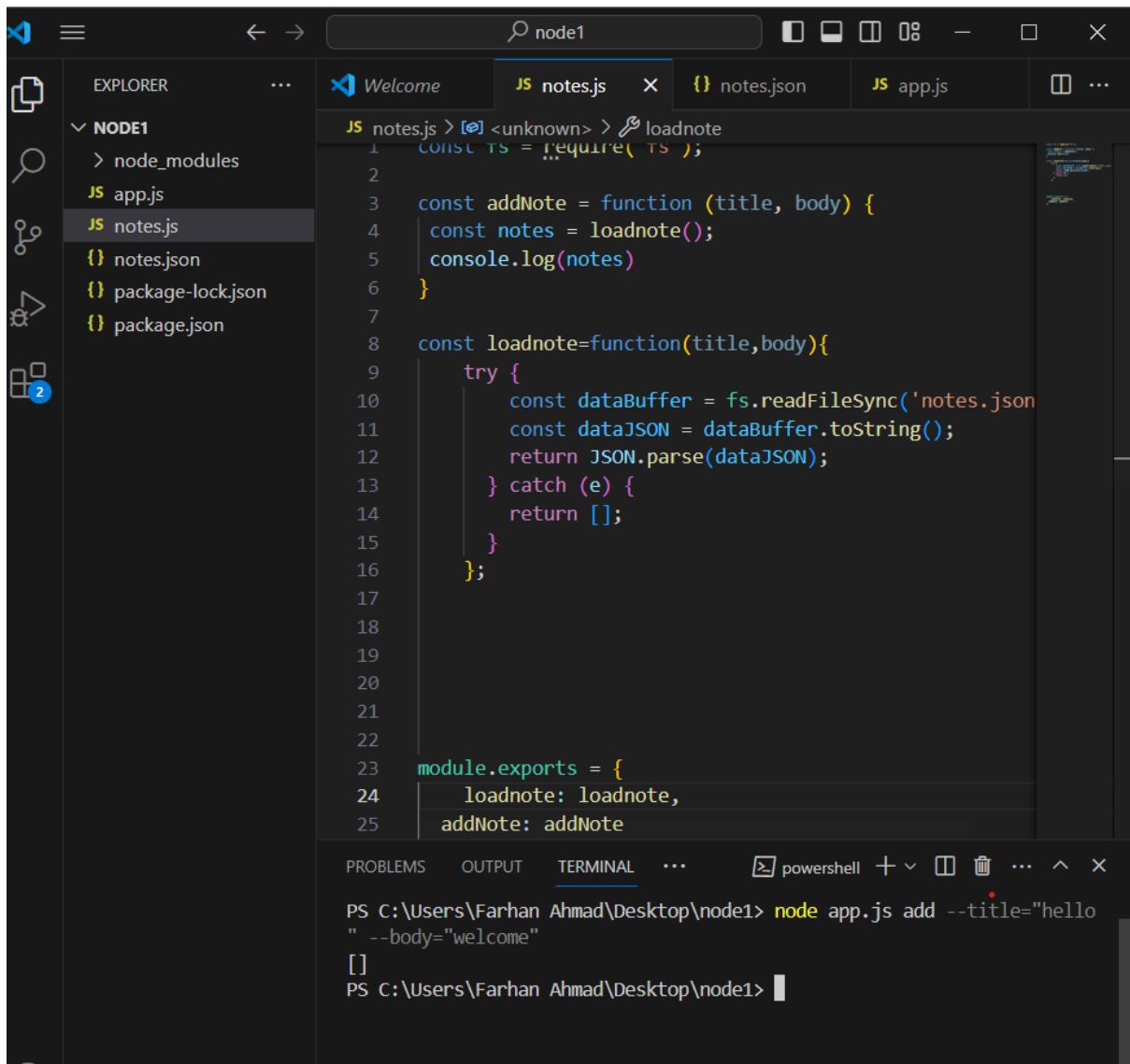
## 2. Yeni not olusturmak

İlk olarak, fs modülü require ile içe aktarılır. Ardından, loadnote adlı bir fonksiyon tanımlanır. Bu fonksiyon, 'notes.json' adlı bir dosyadan notları yüklemek için kullanılır. Eğer dosya mevcut değilse veya bir hata oluşursa, boş bir dizi döndürülür.

Daha sonra, addNote adlı bir fonksiyon tanımlanır. Bu fonksiyon, bir notun başlığını ve içeriğini parametre olarak alır. loadnote fonksiyonu çağrılarak mevcut notlar yüklenir ve konsola yazdırılır.

Son olarak, module.exports kullanılarak loadnote ve addNote fonksiyonları dışa aktarılır. Bu sayede, bu fonksiyonlar diğer dosyalardan kullanılabilir hale gelir.

Bu kod parçası, notları yüklemek ve yeni notlar eklemek için temel işlevleri sağlar. Ancak, notları dosyaya kaydetme veya var olan notları güncelleme gibi işlevler henüz yerine getirilmemi

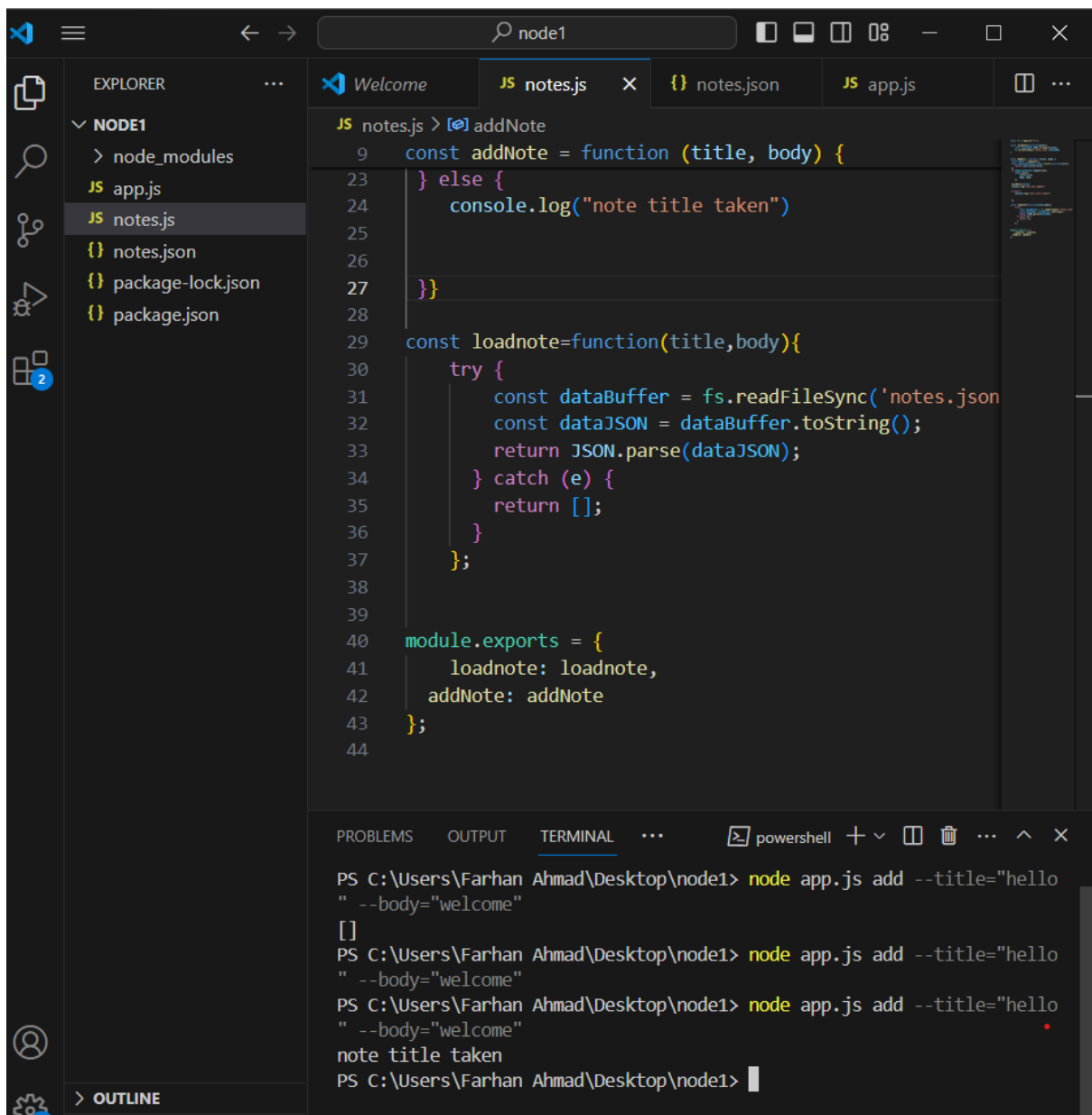
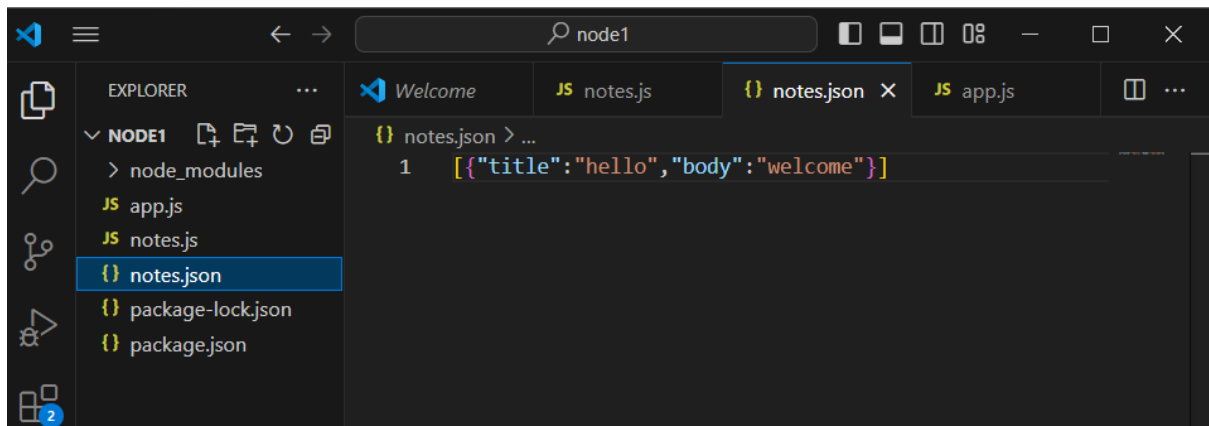


```
node1
Welcome JS notes.js X {} notes.json JS app.js
JS notes.js > [?] addNote
1  const fs = require('fs');
2
3  const saveNotes=function(notes){
4      const dataJSON= JSON.stringify(notes)
5      fs.writeFileSync('notes.json',dataJSON)
6  }
7
8
9  const addNote = function (title, body) {
10     const notes = loadnote();
11     notes.push({
12         title:title,
13         body: body
14     })
15     saveNotes(notes)
16 }
17
18 const loadnote=function(title,body){
19     try {
20         const dataBuffer = fs.readFileSync('notes.json')
21         const dataJSON = dataBuffer.toString();
22         return JSON.parse(dataJSON);
23     } catch (e) {
24         return [];
25     }
26 }

PROBLEMS OUTPUT TERMINAL ... powershell + v [] [] ... ^ x
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js add --title="hello" --body="welcome"
[]
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js add --title="hello" --body="welcome"
[]
PS C:\Users\Farhan Ahmad\Desktop\node1> 
```

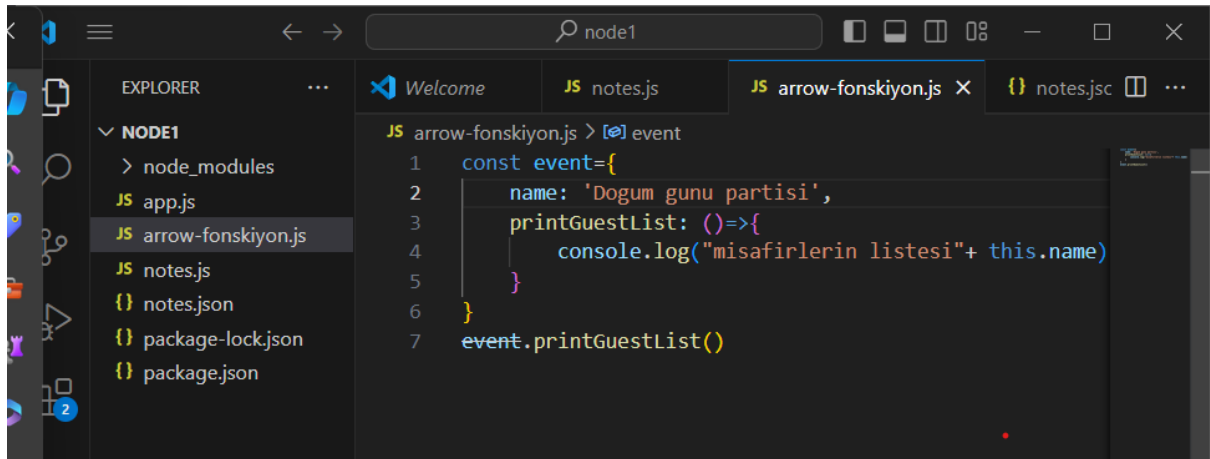
Ve running command olarak :

**Node app.js add—title="hello" –body="welcome" olarak calistirdik**



Not uygulamamıza bir fonksiyon daha ekleyelim. Bu sefer not silme fonksiyonunu ekleyeceğiz. Not silme fonksiyonunu eklemek için yargs modülünü kullanacağız. Builder nesnesini geçen sefer olduğu gibi kullanacağız. Son olarak, başarılı uygulamalar için yeşil renkte ve hatalar için kırmızı renkte yazılar yazmak için chalk modüllerini kullanacağız.

**Raporun ikinci bölümünde, ES modülü nodemon'dan bahsediyoruz.**



```
JS arrow-fonskiyon.js > [⌘] event
1  const event={
2      name: 'Dogum gunu partisi',
3      printGuestList: ()=>{
4          console.log("misafirlerin listesi"+ this.name)
5      }
6  }
7  event.printGuestList()
```

```
JS arrow-fonskiyon.js > [event] > printGuestList
1  const event={
2      name: 'Dogum gunu partisi',
3      printGuestList: ()=>{
4          console.log('misafirlerin listesi' + this.name
5      }
6  }
```

```
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
PS C:\Users\Farhan Ahmad\Desktop\node1>
```

İlk olarak, kare alma işlemi için kısa bir fonksiyon tanımı kullanılabilir. Daha sonra, bir nesne tanımlanarak bu nesne üzerinde fonksiyonların kullanımı gösterilir. Arrow function olarak tanımlandığında, `this` değerine erişilemediği için hata alınır. Daha sonra, daha kısa bir fonksiyon tanımı gösterilir. Son olarak, dizi üzerinde `forEach` döngüsü kullanılarak her eleman için belirli bir işlem yapılması gösterilir. Arrow function kullanarak `this` bağlamına erişim sağlanabilir ve bu sayede `this.name` gibi değerlere rahatlıkla erişilebilir.

The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'NODE1' with files: 'node\_modules', 'app.js', 'arrow-fonskiyon.js' (selected), 'notes.js', 'notes.json', 'package-lock.json', and 'package.json'. The main editor area has three tabs: 'Welcome', 'JS notes.js', and 'JS arrow-fonskiyon.js' (active). The code in 'arrow-fonskiyon.js' is as follows:

```
JS arrow-fonskiyon.js > [?] myEvent > printGuestList
1  const square = (x) => x*x;
2  const myEvent={
3      name: 'Dogum gunu partisi',
4      gustList:['farhan','ahmad','furkan','ali'],
5      printGuestList (){
6          console.log(this.name+ ' katilan liste');
7          this.gustList.forEach((guest)=>{
8              console.log('misafirlerin listesinden '+guest
9          })
10     }
11 }
12 myEvent.printGuestList()
```

The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt. The command executed is 'node arrow-fonskiyon.js'. The output is:

```
PS C:\Users\Farhan Ahmad\Desktop\node1> node arrow-fonskiyon.js
Dogum gunu partisi katilan liste
misafirlerin listesinden farhan Dogum gunu partisi katiliyor.
misafirlerin listesinden ahmad Dogum gunu partisi katiliyor.
misafirlerin listesinden furkan Dogum gunu partisi katiliyor.
misafirlerin listesinden ali Dogum gunu partisi katiliyor.
PS C:\Users\Farhan Ahmad\Desktop\node1>
```