

Node Js Ödevi

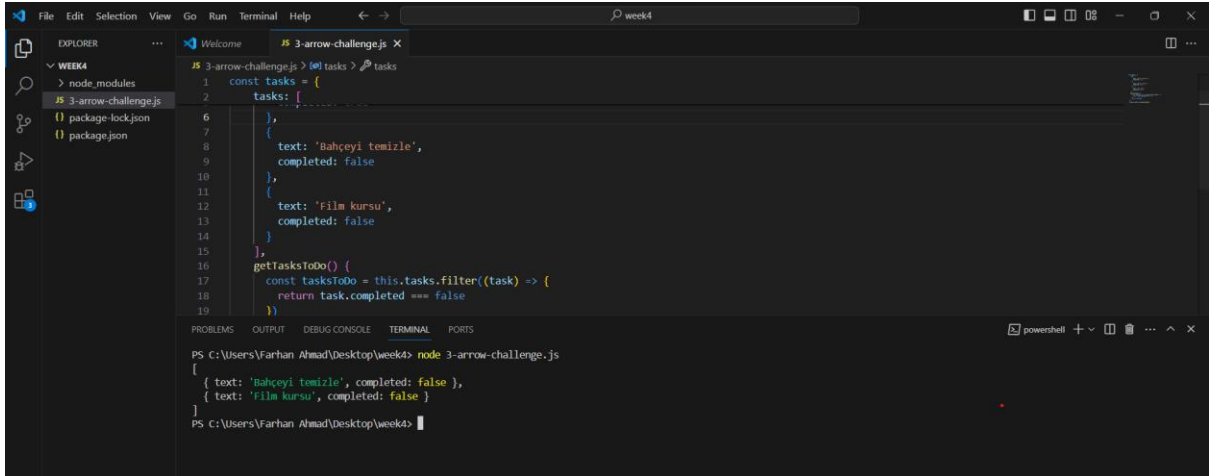
4th Week

Farhan Ahmad

20360859096

Geçen hafta **Es Modülünü ve Nodemon Modülünü** gibi konuları incelemiştik. Bu hafta **Nodemon ile Arrow(OK) Modülünün kullanışını ve geçmiş haftanın kodlarını Arrow modülü ile yazması ve Debbuger** gibi konuları ele alacağız.

Arrow Modülünün Kullanışı:



```
1 const tasks = {
2   tasks: [
3     {
4       text: 'Bahçeyi temizle',
5       completed: false
6     },
7     {
8       text: 'Film kursu',
9       completed: false
10    }
11  ],
12  getTasksToDo() {
13    const tasksToDo = this.tasks.filter(task => {
14      return task.completed === false
15    })
16  }
17 }
18
19 PS C:\Users\Farhan Ahmad\Desktop\week4> node 3-arrow-challenge.js
[
  { text: 'Bahçeyi temizle', completed: false },
  { text: 'Film kursu', completed: false }
]
```

Yukarıdaki kodda nodeJs'te bir nesne içinde yer alan metotları Arrow modülünü (=>) kullanarak tanımlamanın nasıl yapıldığı açıklanmaktadır. Verilen örnekte, bir görev listesi nesnesi içinde, tamamlanmamış görevleri filtreleyen bir getTasksToDo metodu tanımlanmaktadır.

İlk olarak, tasks adında bir nesne oluşturulur ve içine tasks adında bir dizi görev eklenir. Her görev, metin ve tamamlanma durumu özelliklerine sahiptir. Ardından, getTasksToDo metodu tasks dizisini filtreleyerek tamamlanmamış görevleri seçer ve bu görevleri yeni bir dizi olarak döndürür.

Metodun arrow function syntax'ıyla tanımlanması, kodun daha kısa ve okunabilir olmasını sağlar. Arrow function, function keyword'ünün yerine kullanılabilir ve bu durumda this bağlamı da otomatik olarak doğru şekilde ayarlanır.

Sonuç olarak, console.log(tasks.getTasksToDo()) ifadesi çağrıldığında, tamamlanmamış görevlerin listesi konsola yazdırılır ve beklendiği gibi çalışır.

```
1 const tasks = {
2   tasks: [
3     {
4       text: 'Market alışverişisi',
5       completed: true
6     },
7     {
8       text: 'Bahçeyi temizle',
9       completed: false
10    },
11    {
12      text: 'Film kursu',
13      completed: false
14    }
15  ],
16  getTasksToDo() {
17    return this.tasks.filter((task) => task.completed !== true);
18  }
19 }
20 console.log(tasks.getTasksToDo());
```

```
PS C:\Users\Farhan Ahmad\Desktop\week4> node 3-arrow-challenge.js
[
  { text: 'Bahçeyi temizle', completed: false },
  { text: 'Film kursu', completed: false }
]
```

Handler fonksiyonunun kullanımı:

Bir çağrı işleyici (call handler), her Node.js işlev çağrısını işlemek için kullanılan bir yöntemdir. Bir işlev sürümü oluştururken, dosya adını ve istek işleyici adını içeren giriş noktasını belirtmelisiniz (örneğin, `index.myFunction`).

The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the file structure for a project named 'NODE1', including files like 'node_modules', '3-arrow-challenge.js', 'app.js', 'arrow-fonskiyon.js', 'notes.js', 'notes.json', 'package-lock.json', and 'package.json'. The 'app.js' file is selected and open in the editor. The code in 'app.js' defines a REST client for a notes application, with endpoints for GET, POST, PUT, and DELETE. The terminal at the bottom shows the command 'node app.js' being executed, and the output indicates that the application is running successfully, displaying 'note title taken'.

```

JS app.js
1  const notes = require('./notes.js');
2  const argv=require('yargs').argv;
3  const handler = (argv) => {
4      notes.addNote(argv.title, argv.body);
5  };
6
7  handler(argv);
8
9
10
11

```

```

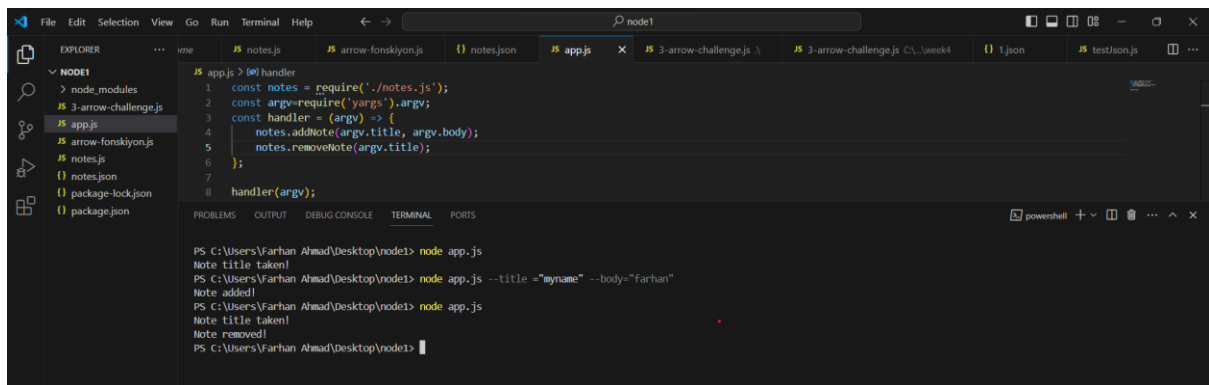
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js
note title taken
PS C:\Users\Farhan Ahmad\Desktop\node1>

```

Arrow modülü ile add fonksiyonun kullanışı:

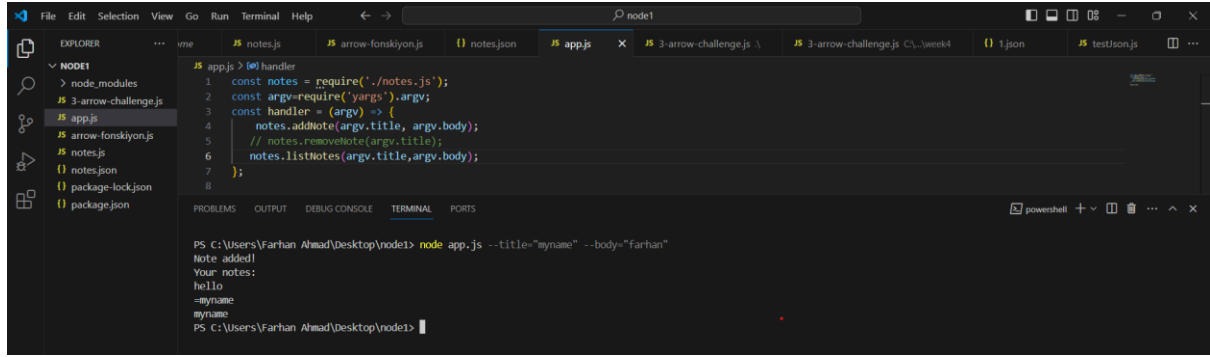
Arrow modülü ile remove fonksiyonun kullanışı:

```
const removeNote = (title) => {  
  let notes = loadNotes();  
  const notesToKeep = notes.filter((note) => note.title !== title);  
  
  if (notes.length > notesToKeep.length) {  
    saveNotes(notesToKeep);  
    console.log('Note removed!');  
  } else {  
    console.log('Note not found!');  
  }  
};
```



Arrow modülü ile listleme fonksiyonun kullanışı:

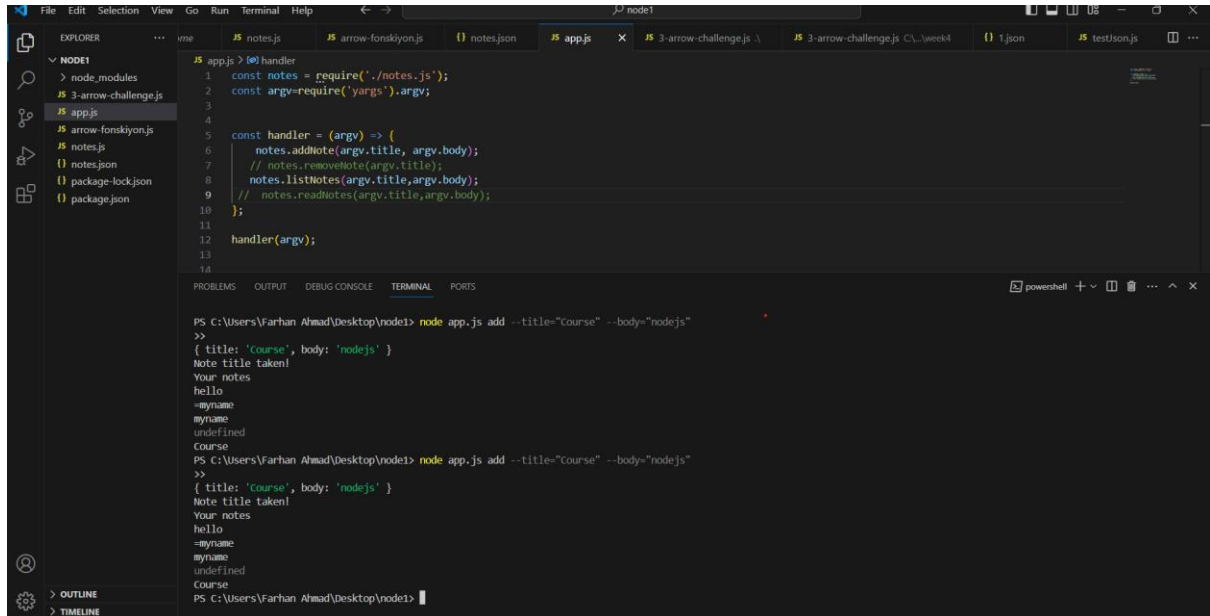
```
const listNotes = () => {  
  const notes = loadNotes();  
  console.log('Your notes:');  
  notes.forEach((note) => {  
    console.log(note.title);  
  });  
};
```



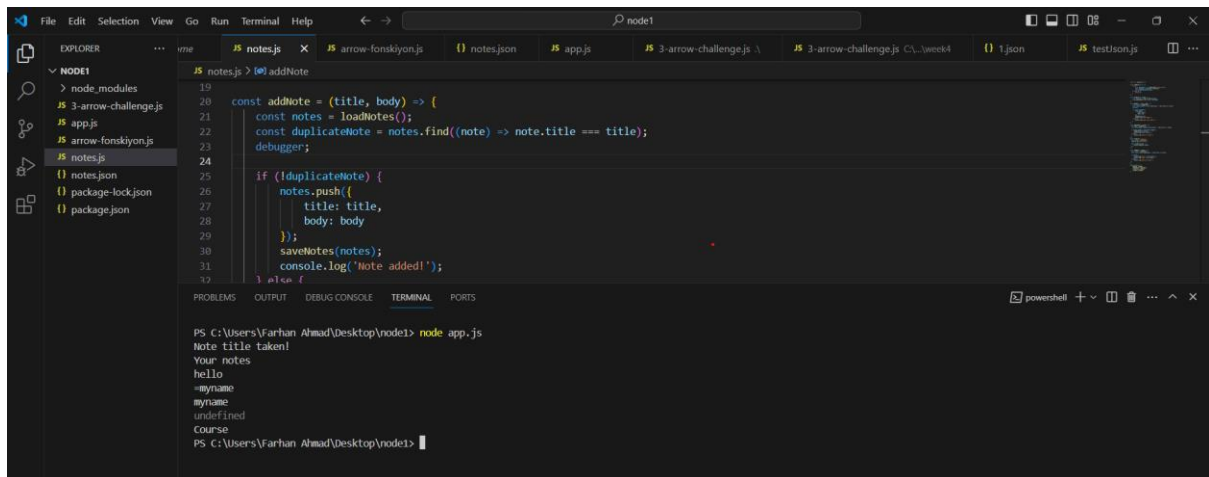
```
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js --title="myname" --body="farhan"
Note added!
Your notes:
hello
=myname
myname
PS C:\Users\Farhan Ahmad\Desktop\node1>
```

Node.js Hata Ayıklama (Debugging)

Node.js'te hata ayıklama yapmak için `console.log` kullanılmaktadır. Örneğin, `notes.js` dosyasında `addNote` fonksiyonunu düşünelim. Else bloğunun hiçbir zaman çalışmadığını varsayalım. Bunun nedenini anlamak için `duplicateNote` değişkenini yazdırabiliriz.



```
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js add --title="Course" --body="nodejs"
>>
{ title: 'Course', body: 'nodejs' }
Note title taken!
Your notes
hello
=myname
myname
undefined
Course
PS C:\Users\Farhan Ahmad\Desktop\node1> node app.js add --title="Course" --body="nodejs"
>>
{ title: 'Course', body: 'nodejs' }
Note title taken!
Your notes
hello
=myname
myname
undefined
Course
PS C:\Users\Farhan Ahmad\Desktop\node1>
```



node inspect app.js add --title="Course" --body="nodejs"

