Furkan Kazım Çam -21360859036 -3.hafta rapor

app.js ve notes.js adı iki dosya açınız.

App.js'in içine aşağıdaki kodu yazınız.

```
const yargs=require("yargs")
yargs.version("1.1.0")


yargs.command({
    command:"add",
    describe:"Add a new note",
    builder:{
        title:{
            describe:"Note title",
            demandOption:true,
            type:"string"
        },
        body:{
            describe:"Note body",
            demandOption:true,
            type:"string"
        }
    },

    handler:function(argv){

    }

})
```

Bir dosyadan diğer dosyaya fonksiyon export etmeyi gördük. Ama bir sorun var. Tek bir fonksiyon gönderiyoruz. Peki ya birden fazla fonksiyon export etmek istersek ne yapmamız gerekmektedir?

Notes.js dosyasının içine yazınız.

```
module.exports={

notGetir:getNotes,
notEkle:addNote,
notSil:removeNote

}
```

Burada anlatılan ; notes.js dosyasında bu fonksiyonların olduğu ve onları export ettiğimizdir. Bu foknsiyonları kullanmak istediğimizde diğer dosyalardan notes.js dosyasını require etmemiz gerekmektedir. Bu fonskiyonları app.js dosyasında kullanacağımız için “const notes=require("./notes.js")” kodunu app.js’in içine yazıyoruz.

Add command’ın handler bölümüne “notes.notEkle(argv.title,argv.body)” kodunu giriyoruz. Bir not ekleyeceğiz ve notun title ve body özellikleri olacak.

Notes.js dosyasına girip getNotes fonksiyonun içini yazalım. Bunun için notları kaydedeceğimiz dosyadan okuma yapacağız. loadNotes fonksiyonunu yazalım.

```
 Const loadNotes = function () {

}
```

addNote fonksiyonunu yazalım .

```
const addNote = function (title, body) {

        const notes = loadNotes()

}
```

Bir dosyaya yazma ve okuma işlemi olduğundan dolayı const fs = require(‘fs’) kod satırını dosyamızın en tepesine yazmalısınız. Şimdi loadNotes fonksiyonunun içini dolduralım.

Burada dosyadan okuma yapıp json formuna dönüştüreceğiz. Eğer dosya boş ise bize bir hata döndürmeli bunu sağlamak için ise try-catch bloklarına ihtiyacımız var. Son durumda fonksiyonumuzun içi şu durumda olmalıdır.

```
const loadNotes=function(){
```

```
    try {
        const databuffer=fs.readFileSync("notes.json")//okur
        const dataJSOn=databuffer.toString()//JSONa çevirir
        return JSON.parse(dataJSOn)//JSONdan stringe çevirir
    }
    catch (e) {
        return []
    }
}
```

loadNotes fonksiyonumuz hazır sirada ise addNote fonksiyonu var.

```
const addNote = function (title, body) {
const notes = loadNotes()
notes.push({
title: title,
body: body
})
saveNotes(notes)
}
```

İlk önce loadNotes'i çalıştırıyoruz ve sonrasında ise alınan notu saveNotes ile notes dizisine ekliyoruz. SaveNotes fonksiyonun içi ise

```
const saveNote = function (notes) {
const dataJson = JSON.stringify(notes)
fs.writeFileSync('notes.json', dataJSON)
}
```

Böyle olmalıdır çünkü alınan not json dosyası olarak kaydedilecektir.

Notu eklemeden önce eğer not önceden eklenmiş ise o onu eklemeyen yani bir notu bir defadan fazla eklemememizi sağlayan kodu yazmamız gerekmektedir. Bu değişikliği ise

addNote fonksiyonunda yapmamız gerekmektedir.

```
const addNote = function (title, body) {
const notes = loadNotes()
const duplicateNotes = notes.filter(function (note) {
return note.title === title
})
if (duplicateNotes.lenth === 0) { //no duplicates
notes.push({
title: title,
body: body
})
saveNotes(notes)
console.log('New note added!')
} else {
console.log('Note title taken')
}
}
```

Son durumda addNote fonksiyonunu içi böyle olmadır. Notu kaydetmeden önce yani saveNotes fonksiyonu çalışmadan önce bu not daha önce kaydedilmiş mi diye kontrol ediecektir. Bir filtreleme kullanılır bunun için. Yeni not ile dosyadan çekilen not title'ları eşit mi değil mi diye kontrol eden bir filtredir. Bu filtre eşit olanları geriye döndüreceğinden dolayı lenght'ı 0 ise demek ki not daha önce kaydolmamıştır bu yüzden notu kaydedebiliriz else durumunda ise not başlığı alındığı için kaydolunmaz ve hata mesajı yazdırılır.

Sıradaki işlem ise not silme.

Bu adıma app.js'in içinde yeni bir komut tanımlayarak başlayalım.

```
yargs.command({
```

```
 command: 'remove',
 describe: 'Remove a new note',
builder: {
title: {
describe: ’Note title’,
demandOption: true,
type: ‘string’
}
}
 handler: function () {
 console.log('Removing the note!')
 }
}
)
```

Basitçe böyle yazılabilir. Bir title alıyoru tipi string ve girdi alınması zorunlu. Ekrana ise removing the note! Yazdırıyor. Notes.js’in içine ise bir removeNote adlı fonksiyon oluşturalım. Ve tabiiki onu export edelim.

```
const removeNote = function (title) {
}
```

Son durumda modüle.exports kısmı böyle olmalıdır.

```
module.exports={
notGetir:getNotes,
notEkle:addNote,
notSil:removeNote
}
```

Ve tabiki yazdığımız fonksiyonu kullanabilmemiz için tanımladığımız remove komutunun handler kısmına fonksiyonu eklememiz gerekmektedir

```
    handler:function(argv){
        notes.notSil(argv.title)
```

```
}
```

Evet bir removeNote fonksiyonu yazdık ama fonksiyonun içi boş. Hadi onu dolduralım.

İlk önce var olan dosyayı okumamız gerekmektedir. Onun için fonksiyonu yazmaya loadNotes fonksiyonu ile başlayalım.

```
const notes = loadNotes()
```

Bir filtre aracılığıyla sileceğimiz notun dosyada olup olmadığını anlamamız gerekmektedir. Filtre ise şöyle olmalıdır.

```
const removeNote = function (title) {
const notes = loadNotes()
const notesToKeep = notes.filter(function (note) {
return note.title !== title
})
saveNotes(notesToKeep)
}
```

Biraz eğlenceli hale getirelim. Eğer not silinirse yeşil bir şekilde not silindi, eğer ki not yok ise yani silinemez ise kırmızı bir şekilde not bulunamadı şeklinde bir yazı yazdıralım.

Bunun için önceki haftalarda öğrendiğimiz chalk modülünü kullanacağız.

Require ederek işleme başlayalım. Dosyanın başına yazınız.

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

removeNotes fonksiyonuna aşağıdaki kodları ekleyelim

```
if (notes.length > notesToKeep.length) {
console.log(chalk.green.inverse('Note removed!'))
saveNotes(notesToKeep)
} else {
```

```
console.log(chalk.red.inverse('No note found!'))
}
```

Eğer not silinirse yeşil biçimde 'Note removed! Eğer ki silinemezse 'No note found!' yazısı konsolda görünecektir.

Aynı işemleri addNote fonksiyonu içinde yapabilirsiniz.

Arrow Functions

Arrow.js adına bir dosya açınız.

```
const square = function (x) {
return x*x
}
console.log(square(3))
```

bu kodu çalıştırır iseniz 9 cevabını alırsınız. Peki ben bu fonksiyonu daha kısa bir şekilde yazabilir miyim?

Elbette

```
const square = (x) => x*x
```

yukardaki kod ile aynı çıktıyı üretir. Bu şekilde tanımlanan fonksiyonlara arrow function denir. Bir fonksiyonu böyle tanımlayabilmeniz için fonksiyonun argüman almadan tek satırda işini bitirebilmesi gerekmektedir.

Diğer bir yöntem ve en çok kullanılan yöntem ise şöyle kullanılır.

```
const event = {
name: 'Birthday Party',
printGuestList: function () {
console.log('Guest list for ' + this.name)
}
```

}

Yeni bir değişken tanımı yapılır ve özellikleri verilir. İçindeki bir fonksiyonda ise değişkenin kendi özelliklerine this komutu ile ulaşılabilir. Fonksiyonu arrow function ile şöyle düzenleyebiliriz.

printGuestList: () => {

console.log(‘Guest list for ’ + this.name)

}

Daha da kısaltıp

printGuestList() {

console.log(‘Guest list for ’ + this.name)

}

Şeklinde yazılabilir.

Event değişkenine yeni bir özellik katıp arrow function’u pekiştirelim.

guestList:["furkan","kazım","betul"],

özelliğini ekleyiniz.

```
printGuestList(){
    console.log(this.name+" katılımcı  listesi")
    this.guestList.forEach((guest)=>{
        console.log(guest+" is attending "+this.name)
    })
}
```

Foreach kullanarak bütün guestList’i yazdırabilirsiniz.