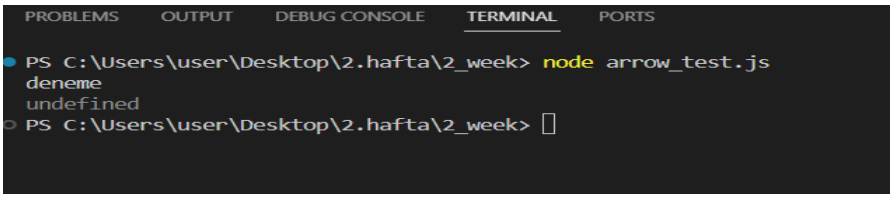
ÖZLEM ARSLAN 19360859071

BURSA TEKNİK ÜNİVERSİTESİ NODE.JS ile WEB PROGLAMA DERSİ 4.HAFTA TEORİ DERSİ RAPORU

Bu hafta ilk olarak yeni bir dosya oluşturalım ve adını arrow_test.js diyelim. Daha sonra bir obje oluşturalım ve bu obje içinde arry bulundursun. 3 elemanlı bir arry tanımlayalım. Elimizde ki objenin elemanlarına teker teker bakıp eğer completedları true ise yapılmış anlamına gelecektir ve tamamlanmamış görevleri getiren getTasksToDo() isimli bir fonksiyon yazalım. Bu sayede yapılmamış görevleri çekebilelim.

```
// obje tanımlaması yapalım.
// bu obje arry içersin
const tasks={
   // arryin 3 elemanı var
   tasks:[
      {
         text:"alışveriş", // arry eleman için görev ismi
         completed:true,  // arry elemanın tamamlanma durumu
      },
      {
         text:"temizlik",
         completed:false,
      },
      {
         text:"ödev",
         completed:false
      }
   ],
   getTasksToDo:function(){ // yapılacak işlemleri getiren fonk
      console.log("deneme")
   }
}
console.log(tasks.getTaskSToDo())
```

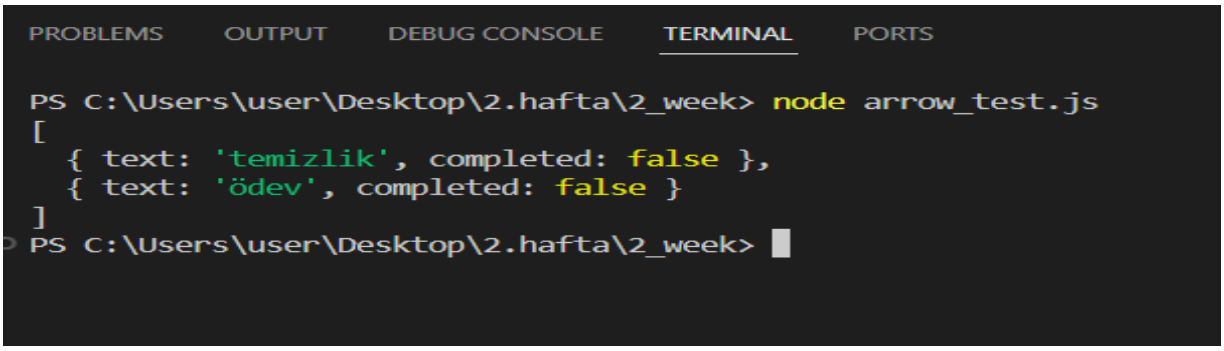
Çıktısı:



Görevler tamamlanmamış ise geriye bu görevleri döndüren bir arrow fonksiyonu tanımlayalım.

```
// obje tanımlaması yapalım.
// bu obje arry içersin
const tasks={
   // arryin 3 elemanı var
   tasks:[ // arrayın her bir elemanına task dedik
      {
         text:"alışveriş", // arry eleman için görev ismi
         completed:true,  // arry elemanın tamamlanma durumu
      },
      {
         text:"temizlik",
         completed:false,
      },
      {
         text:"ödev",
         completed:false
      }
   ],
   getTasksToDo:function(){ // tamamlanmamış görevleri arrow fonksiyonla getirme
      // filter fonksiyonu arry üzerindeki elemanları filterler
     const tasksToDo=this.tasks.filter((task)=>{
      return task.completed===false // filtreleme işlemini completed özelliği ile kontorl edelim.
     })
     return tasksToDo // geriye dönen değer tamamlanmamış görevelerdir.
   }
}
console.log(tasks.getTasksToDo())
```

Çıktısı:



Bu fonksiyonu daha da düzenleyelim ve tek satırlık bir kod haline getirelim. İlk başta filtreleme yapıp bunu bir değişkene atıp geriye döndürmek yerine daha tümleşik hale getirelim.

```
// obje tanımlaması yapalım.
// bu obje arry içersin
const tasks={
  // arryin 3 elemanı var
  tasks:[ // arrayın her bir elemanına task dedik
    {
      text:"alışveriş", // arry eleman için görev ismi
      completed:true,  // arry elemanın tamamlanma durumu
    },
    {
      text:"temizlik",
      completed:false,
    },
    {
      text:"ödev",
      completed:false
    }
  ],
  getTasksToDo : function (){
    return this.tasks.filter((task) =>task.completed === false) }
}
console.log(tasks.getTasksToDo())
```

fonksiyonu tek satırlık hale getirmiş olduk.

Çıktısı:

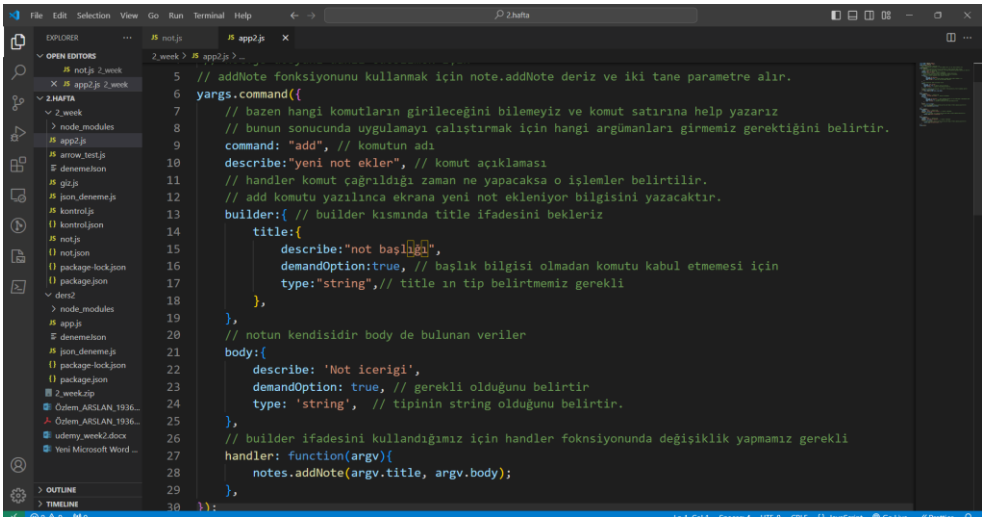
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\user\Desktop\2.hafta\2_week> node arrow_test.js
[
  { text: 'temizlik', completed: false },
  { text: 'ödev', completed: false }
]
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Yine aynı çıktıyı elde ederiz.

Bu kısımda geçen haftaki yazdığımız kodlardan devam edelim. App2.js dosyamızı açalım. Function key Wordü kaldıralım ve tüm fonksiyonları arrow fonksiyon haline getirelim.

İlk olarak app2.js içine gidelim burada handler kısmında fonksiyon tanımlamaları var bunlarda değişiklik yapmamız lazım.

Eskiden not ekleme fonksiyonu şu şekildeydi.



```
// addNote fonksiyonunu kullanmak için note.addNote deriz ve iki tane parametre alır.
yargs.command({
    // bazen hangi komutların girileceğini bilemeyiz ve komut satırına help yazarız
    // bunun sonucunda uygulamayı çalıştırmak için hangi argümanları girmemiz gerektiğini belirtir.
    command: "add", // komutun adı
    describe:"yeni not ekler", // komut açıklaması
    // handler komut çağrıldığı zaman ne yapacaksa o işlemler belirtilir.
    // add komutu yazılınca ekrana yeni not ekleniyor bilgisini yazacaktır.
    builder:{ // builder kısmında title ifadesini bekleriz
        title:{
            describe:"not başlığı",
            demandOption:true, // başlık bilgisi olmadan komutu kabul etmemesi için
            type:"string",// title ın tip belirtmemiz gerekli
        },
    },
    // notun kendisidir body de bulunan veriler
    body:{
        describe: 'Not icerigi',
        demandOption: true, // gerekli olduğunu belirtir
        type: 'string',  // tipinin string olduğunu belirtir.
    },
    // builder ifadesini kullandığımız için handler foknsiyonunda değişiklik yapmamız gerekli
    handler: function(argv){
        notes.addNote(argv.title, argv.body);
    },
});
```

Hadler kısmını şu hale getirelim. : Funciton ifadelerini silelim.

```
yargs.command({
  // bazen hangi komutların girileceğini bilemeyiz ve komut satırına help yazarız
  // bunun sonucunda uygulamayı çalıştırmak için hangi argümanları girmemiz gerektiğini belirtir.
  command: "add", // komutun adı
  describe:"yeni not ekler", // komut açıklaması
  // handler komut çağrıldığı zaman ne yapacaksa o işlemler belirtilir.
  // add komutu yazılınca ekrana yeni not ekleniyor bilgisini yazacaktır.
  builder:{ // builder kısmında title ifadesini bekleriz
    title:{
      describe:"not başlığı",
```

```
        demandOption:true, // başlık bilgisi olmadan komutu kabul etmemesi için
        type:"string",// title ın tip belirtmemiz gerekli
      },
  },
  // notun kendisidir body de bulunan veriler
  body:{
    describe: 'Not icerigi',
    demandOption: true, // gerekli olduğunu belirtir
    type: 'string',  // tipinin string olduğunu belirtir.
  },
  // builder ifadesini kullandığımız için handler foknsiyonunda değişiklik yapmamız gerekli
  handler(argv){
    notes.addNote(argv.title, argv.body);
  },
});
```

Aynı işlemleri remove fonksiyonu içinde yapalım.

İlk hali:

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

Değiştirilmiş hali:

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

Aynı işlemleri list ve read fonksiyonları içinde yapalım son halleri şu şekildedir.

```
yargs.command({
   command: 'list', // komutun adı
   describe: 'mevcut notları listeler', // konutun açıklaması
   handler(){
      // komut satırında list argümanı girildiğinde hangi işlemin yapılacağını belirtir.
      console.log("notlar listeleniyor...")
   }
})
yargs.command({
   command: 'read', // komutun adı
   describe: 'seçilen notu gösterir', // konutun açıklaması
   handler(){
      // komut satırında remove argümanı girildiğinde hangi işlemin yapılacağını belirtir.
      console.log("not gösteriliyor...")
   }
})
```

Yapılan tüm işlemleri kaydedelim ve tüm fonksiyonlar çalışıyor mu diye kontrol edelim.

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js remove --title="dört"
not silindi
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js remove --title="dört"
Silmek istediginiz not bulunamamıstır.
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="yeni" --body="notlar"
yeni not eklendi
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="yeni" --body="notlar"
Bu başlık daha once kullanıldı.Not eklenemiyor!!!
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js read --title="yeni"
not gösteriliyor...
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js list
notlar listeleniyor...
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Tüm fonksiyonlarımız çalışmaktadır.

Not.js ye gelelim ve fonksiyonlarımızı düzenleyelim function ifadesini {} ve return ifadesini silelim arrow fonksiyon haline getirelim. Sildiğimiz function ifadelerinin yerine arrow fonksiyonunu temsil eden => sembolünü koyalım. addNote() fonksiyonun ilk hali bu şekildedir.

```
const addNote = function (title, body) {
  //notları dosyaya kaydedıcez
  const notes = loadNotes(); //array dondurcek
  const duplicateNotes = notes.filter(function (note) {
    //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
    //note.title // notesin içindeki elemanlardan bır tanesi
    return note.title === title;
  });
  if (duplicateNotes.length === 0) {
    //hiçbir eşleşme yok,yenı not eklenebılır
    notes.push({
      title: title,
      body: body,
    }); //arraye eleman eklemek
    console.log(chalk.green.inverse("yeni not eklendi")); // arrayin içindekileri consola yazdırma
    saveNotes(notes);
  } else {
    //o baslık daha once alınmıs
    console.log(
      chalk.red.inverse("Bu başlık daha once kullanıldı.Not eklenemiyor!!!")
    );
  }
};
```

Arrow fonksiyonu hali ise:

```
const addNote =  (title, body)=>{
  //notları dosyaya kaydedıcez
  const notes = loadNotes(); //array dondurcek
  const duplicateNotes = notes.filter((note)=>note.title === title)
```

```
    //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
    //note.title // notesin içindeki elemanlardan bır tanesi
   if (duplicateNotes.length === 0) {
    //hiçbir eşleşme yok,yenı not eklenebılır
    notes.push({
     title: title,
     body: body,
    }); //arraye eleman eklemek
    console.log(chalk.green.inverse("yeni not eklendi")); // arrayin içindekileri consola yazdırma
    saveNotes(notes);
   } else {
    //o baslık daha once alınmıs
    console.log(
     chalk.red.inverse("Bu başlık daha once kullanıldı.Not eklenemiyor!!!")
    );
   }
  };
```

Ekleme denemesi yapalım.

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="yen2i" --body="notlar"
yeni not eklendi
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Bu hale getirdik filter fonksiyonuna arrow fonksiyonu haline getirmiş olduk. Tüm bu işlemleri diğer fonksiyonlar üzerinde de uygulayalım.

removeNote fonksiyonun ilk hali bu şekildedir.

```
const removeNote = function (title) {
  // silme işleminden önce notları listelememiz lazım
  const notes = loadNotes(); //notları aldık
  const notesToKeep = notes.filter(function (note) { // kaydetmek istediğimiz notları tutalım
      return note.title !== title; // notların title esit degilse saklarız.
  });
  if (notes.length > notesToKeep.length) {
    //baslangıçtaki not sayısı sonrakinden büyükse silinmiştir aksi halde silme yoktur
  //console.log("Note remove");
    console.log(chalk.green.inverse("not silindi"))// inverse terse çekiyor rekleri arkadakilar ile
değişir
  saveNotes(notesToKeep);
  } else {
  console.log(chalk.red.inverse("Silmek istediginiz not bulunamamıstır."));
  }
  };
```

Arrow hali:

```
const removeNote = (title)=> {
```

```
    // silme işleminden önce notları listelememiz lazım
    const notes = loadNotes(); //notları aldık
    const notesToKeep = notes.filter((note)=>note.title !== title ) //
kaydetmek istediğimiz notları tutalım
            // notların title esit degilse saklarız.
    if (notes.length > notesToKeep.length) {
        //baslangıçtaki not sayısı sonrakinden büyükse silinmiştir aksi
halde silme yoktur
    //console.log("Note remove");
        console.log(chalk.green.inverse("not silindi"))// inverse terse
çekiyor rekleri arkadakilar ile değişir
    saveNotes(notesToKeep);
    } else {
    console.log(chalk.red.inverse("Silmek istediginiz not
bulunamamıstır."));
    }
    };
```

Diğer fonksiyonların ilk halleri şu şekildedir:

```
const loadNotes = function () {
    try{ // dosya mevcut değilse hata almamak için try catch kullanalım
    const dataBuffer = fs.readFileSync("not.json") // dosyadan okuma işlemi
yapacak
     // . ile propertylere erişemeyiz.
    // dosyadan okuma yaptığında veriler json formatında gelir ama bunu
stringe çevir.
    const dataJSON = dataBuffer.toString()
    //json formatına parse etmemiz gerekli
    return JSON.parse(dataJSON) // . ile propertylere erişebiliriz
    }
    catch (e) {
        return [] // [] boş bir arry demek
        }
};
const saveNotes = function (notes) {
    const dataJson = JSON.stringify(notes) ;// json formatı bozmadan string
haline çevirir
    fs.writeFileSync("not.json", dataJson) ;// aynı dosya ismini vermen
gerekli
    }
```

Her iki fonksiyonu arrow haline getirelim.

```
const loadNotes = () =>{
    try{ // dosya mevcut değilse hata almamak için try catch kullanalım
```

```
    const dataBuffer = fs.readFileSync("not.json") // dosyadan okuma işlemi
yapacak
     // . ile propertylere erişemeyiz.
    // dosyadan okuma yaptığında veriler json formatında gelir ama bunu
stringe çevir.
    const dataJSON = dataBuffer.toString()
    //json formatına parse etmemiz gerekli
    return JSON.parse(dataJSON) // . ile propertylere erişebiliriz
    }
    catch (e) {
        return [] // [] boş bir arry demek
        }
};
const saveNotes =  (notes)=> {
    const dataJson = JSON.stringify(notes) ;// json formatı bozmadan string
haline çevirir
    fs.writeFileSync("not.json", dataJson) ;// aynı dosya ismini vermen
gerekli
    }
```

Tekrardan fonksiyonlarımız çalışıyor mu diye kontrol edelim.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js remove --title="deneme"
not silindi
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="ders" --body="zamanı"
yeni not eklendi
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="ders" --body="zamanı"
Bu başlık daha once kullanıldı.Not eklenemiyor!!!
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js list
notlar listeleniyor...
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Şimdiki kısımda listeleme ve okuma fonksiyonlarının içlerini dolduralım.

Listelemede fonksiyonu notların titlellarını listelesin okuma fonksiyonu ise title verip notun içeriğini okuma işlemlerini yapalım. Ve tüm işlemleri arrow fonksiyon şeklinde tanımlayalım. Not.js içine listNotes() fonksiyonunu tanımlayalım ve export edelim.

```
const listNotes=()=>{

}
module.exports = {
    //  isimlerde karışıklık olmaması için ayni isimler verildi
    //  iki noktanın soldaki isimi farklı bir dosyada (app.js) fonksiyonları kullanırken
    //  tanımlanan isimdir. app.js eğer fonksiyonu notEkleme() isminde kullanmak istiyorsak
    //  o zaman burada sol kısımda notEkleme yapmalısın.
    //  sağdaki isim bu dosyadaki tanımla yapılırken tanımlanan isimle aynı olmalıdır.
    getNotes: getNotes,
    addNote:addNote,
    removeNote:removeNote,
    listNotes:listNotes

    }
```

Şimdi app2.js içine girelim

```
yargs.command({
    command: 'list', // komutun adı
    describe: 'mevcut notları listeler', // konutun açıklaması
    handler(){
        // komut satırında list argümanı girildiğinde hangi işlemin
yapılacağını belirtir.
        console.log("notlar listeleniyor...")
    }
})
```

List bu şekildeydi şimdi düzenleme yapalım fonksiyon çağrısı yapalım ve kaydedelim.

```
yargs.command({
    command: 'list', // komutun adı
    describe: 'mevcut notları listeler', // konutun açıklaması
    handler(){
        // komut satırında list argümanı girildiğinde hangi işlemin
yapılacağını belirtir.
        //console.log("notlar listeleniyor...")
        notes.listNotes()
    }
})
```

Not.js ye geçelim ve boş olan listeleme fonksiyonunu dolduralım.

```
const listNotes =() => {
  // ilk olarak dosyadan okuma işlemi yapmalı bunuda zaten load fonksiyonu
yapıyordu
  const notes= loadNotes() // okunan notları arry olarak alır
  // notes yani arrydir bu arry üzerinde for each ile dolaşıp ekrana yazdır
```

```
  console.log(chalk.inverse("kayıtlı notlar"))
  notes.forEach((note)=>{
    console.log(note.title) // arryin elemanın başlığını ekrana yazar
  })

}
```

Çıktısı:

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js list
kayıtlı notlar
test
deneme2
okul
yeni
yen2i
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Read fonksiyonunu düzenlemeden önce performans artışı sağlamak amacıyla not.js içine gidelim ve addNote() fonksiyonunda bazı değişiklikler yapalım biz yeni bir not eklediğimizde mevcut notların başlıklarına bakıyoruz ve bu başlık daha önceden kullanılmışsa yeni not eklememeli bu kısımda kullandığımız filter fonksiyonu arryin her bir elemanına teker teker bakar problem şu kısımda oluşmaktadır eğer bizim 100 tane notumuz olmuş olsaydı toplamda 100 defa bakacaktık belki eklenecek not 2. Sırada kullanılmış olsaydı filter fonksiyonu hala bakmaya devam edecekti. Bu problemi düzeltelim. Alternatif bir fonksiyon kullanalım.

Eskiden filter fonksiyonu ile şu şekildeydi:

```
const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      const duplicateNotes = notes.filter((note)=>note.title === title)
        //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
        //note.title // notesin içindeki elemanlardan bır tanesi
```

Find fonksiyonu ile yapalım

```
const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      //const duplicateNotes = notes.filter((note)=>note.title === title)
      const duplicateNote=notes.find((note)=>note.title===title)
```

find fonksiyonu ilk kez ifadeyi bulduğu zaman bulma işlemini tamamladığı için arry den çıkıp kontrole devam etmez. Filter fonksiyonu geriye bir arry döndürebilirdi bu sebeple uzunluğuna bakıyorduk şu

an da kullandığımız fonksiyon tek bir eleman döndürecek bu sebeple kontrol işlemini uzunluğa göre değil böyle bir not tanımlanmış mı yoksa tanımlanmamış mı diye kontrol edelim.

Eski hali:

```
  const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      //const duplicateNotes = notes.filter((note)=>note.title === title)
      const duplicateNote=notes.find((note)=>note.title===title)
        //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
      if (duplicateNotes.length === 0) {
        //hiçbir eşleşme yok,yenı not eklenebılır
        notes.push({
          title: title,
          body: body,
        }); //arraye eleman eklemek
        console.log(chalk.green.inverse("yeni not eklendi")); // arrayin
içindekileri consola yazdırma
        saveNotes(notes);
      } else {
        //o baslık daha once alınmıs
        console.log(
          chalk.red.inverse("Bu başlık daha once kullanıldı.Not
eklenemiyor!!!")
        );
      }
    };
```

Yeni hali:

```
const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      //const duplicateNotes = notes.filter((note)=>note.title === title)
      const duplicateNote=notes.find((note)=>note.title===title)
        //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
      //if (duplicateNotes.length === 0) {
      if (!duplicateNotes) {
        //hiçbir eşleşme yok,yenı not eklenebılır
        notes.push({
          title: title,
          body: body,
        }); //arraye eleman eklemek
        console.log(chalk.green.inverse("yeni not eklendi")); // arrayin
içindekileri consola yazdırma
        saveNotes(notes);
```

```
      } else {
        //o baslık daha once alınmıs
        console.log(
          chalk.red.inverse("Bu başlık daha once kullanıldı.Not
eklenemiyor!!!")
        );
      }
    };
```

Bu kısımda app2.js içine gidelim ve read fonksiyonumuz için handler kısmını düzenleyelim.

Eski hali:

```
yargs.command({
    command: 'read', // komutun adı
    describe: 'seçilen notu gösterir', // konutun açıklaması
    handler(){
        // komut satırında remove argümanı girildiğinde hangi işlemin
yapılacağını belirtir.
        console.log("not gösteriliyor...")
    }
})
```

Bu şekilde iken builder eklenmemiş. Bizim builder i eklememizdeki amaç: çalıştırmak istediğimiz commandi kaç tane argüman ile çalıştırmak istediğimizi belirtmemizdir. Read fonksiyonu ile okuma işlemi yaparken title bilgisini almak istediğimiz için builder ekleyelim.

```
yargs.command({
    command: 'read', // komutun adı
    describe: 'seçilen notu gösterir', // konutun açıklaması
    builder:{
        title:{
            describe:"not başlığı", // help kısmında komutun ne işe
yaradığının açıklamasını yapar.
            demandOption:true, // require haline getirir.
            type: "string", // parametre olarak boş title girilmesini
önler.
        },
    },
    handler(){
        // komut satırında remove argümanı girildiğinde hangi işlemin
yapılacağını belirtir.
        console.log("not gösteriliyor...")
    }
})
```

Bu ekleme işlemini hallettikten sonra not.js ye gidip arrow fonksiyon tipinde tanımlayalım.

Argüman olarak title almalı ve export edilmelidir.

```
const readNotes = (title) => {
}
```

Şu anlık içini boş bırakalım.

```
module.exports = {
    //  isimlerde karışıklık olmaması için ayni isimler verildi
    //  iki noktanın soldaki isimi farklı bir dosyada (app.js) fonksiyonları kullanırken
    //  tanımlanan isimdir. app.js eğer fonksiyonu notEkleme() isminde kullanmak istiyorsak
    //  o zaman burada sol kısımda notEkleme yapmalısın.
    //  sağdaki isim bu dosyadaki tanımla yapılırken tanımlanan isimle aynı olmalıdır.
    getNotes: getNotes,
    addNote:addNote,
    removeNote:removeNote,
    listNotes:listNotes, // app2.js içinde listNotes() e erişim sağlanır
    readNotes:readNotes // app2.js içinde readNotes() e erişim

    }
```

App2.js ye geri gelip read commandin handler kısmında bazı değişiklikler yapalım.

Eski hali:

```
handler(){
        // komut satırında remove argümanı girildiğinde hangi işlemin yapılacağını belirtir.
        console.log("not gösteriliyor...")
    }
```

Yeni hali:

```
 handler(argv){
        // komut satırında remove argümanı girildiğinde hangi işlemin yapılacağını belirtir.
        notes.readNotes(argv.title)
       //console.log("not gösteriliyor...")
    }
```

Title i argüman olarak gönderebilmek için

Not.js içine gidip fonksiyonumuzun içini dolduralım.

İlk olarak dosyadan notları çekmemiz lazım bunun için loadNote() fonksiyonunu çağıralım.

Arry içinde gelen notların herhangi birinde bizim okumak istediğimiz notun title a eşit mi diye find fonksiyonu ile arama yapmamız lazım.

```
const readNotes = (title) => {
  const notes = loadNotes(); // dosyadan notaları getirelim
  const note = notes.find((note) =>note.title ===title)

}
```

Eğer varsa not tanımlı olacaktır bulunamazsa undefined olarak kalacaktır şimdi bu kontrol işlemini yapalım.

```
const readNotes = (title) => {
  const notes = loadNotes(); // dosyadan notaları getirelim
  const note = notes.find((note) =>note.title ===title)
  if(note){ // not bulunmuşsa
    console.log(chalk.inverse(note.title))
    console.log(note.body)
    }
    else{ // istenen not bulunamamıştır.
    console.log(chalk.red.inverse("Bu başlığasahip bit not bulunamadı"))
    }

}
```

Çıktısı:

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js read --title="test"
test
icerik
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js read --title="bir"
Bu başlığasahip bit not bulunamadı
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Bu kısımda node.js de debugging işlemini nasıl yapabiliriz onu inceleyelim. Tabi ki de herkesin ilk aklına gelen console.log() u kullanmaktır. Basit bir örnek yapalım.

Not.js içindeki addNote() fonksiyonunda deneme işlemini yapalım. Diyelim ki else bloğuna hiç düşemiyoruz bunun sebebini öğrenmek için duplicateNote u yazdırmak isteyelim.

```
const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      //const duplicateNotes = notes.filter((note)=>note.title ===
title)
      const duplicateNote=notes.find((note)=>note.title===title)
        //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
      //if (duplicateNotes.length === 0) {
      console.log(duplicateNote)
```

```
      if (!duplicateNote) {  // yeni not eklenebilir
        //hiçbir eşleşme yok,yenı not eklenebılır
```

Çıktısı:

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="pazar" --body="günü"
undefined
yeni not eklendi
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Görüldüğü üzere undefined olarak gelmiştir. Aynı komutu tekrar çalıştıralım.

```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="pazar" --body="günü"
{ title: 'pazar', body: 'günü' }
Bu başlık daha once kullanıldı.Not eklenemiyor!!!
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Undefined olarak değilde title ve body bilgisi ile gelmiştir.

İkinci bir yöntem ise debugger komutunu kullanmaktır.

Debug yapmak istediğimiz noktaya breakpoint oluşturalım. Kod başlangıçtan breakpoint konulan noktaya kadar çalışır breakpointten itibaren de adım adım çalışmaya devam eder. Şu şekilde debugger komutunu ekleyelim.

```
    const addNote =  (title, body)=>{
      //notları dosyaya kaydedıcez
      const notes = loadNotes(); //array dondurcek
      //const duplicateNotes = notes.filter((note)=>note.title ===
title)
      const duplicateNote=notes.find((note)=>note.title===title)
        //aynı baslık olma durumunda kosula sokmak ıcın bu fonk yazdık.
      //if (duplicateNotes.length === 0) {
      console.log(duplicateNote)
      debugger
      if (!duplicateNote) {  // yeni not eklenebilir
        //hiçbir eşleşme yok,yenı not eklenebılır
        notes.push({
          title: title,
          body: body,
```

Daha sonra çalıştıralım herhangi bir değişiklik gözlemlenmez.

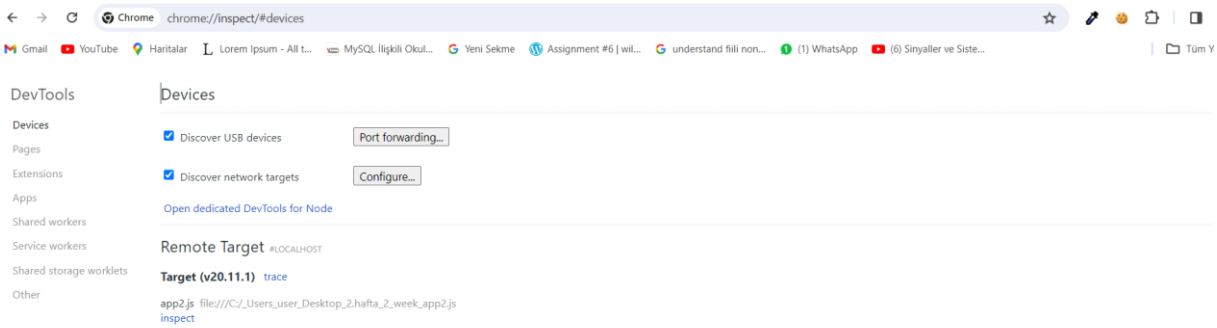
```
PS C:\Users\user\Desktop\2.hafta\2_week> node app2.js add --title="pazar" --body="günü"
{ title: 'pazar', body: 'günü' }
Bu başlık daha once kullanıldı.Not eklenemiyor!!!
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Özel bir komut var onu yazmamız lazım node inspect app2.js diye başlamamız lazım.

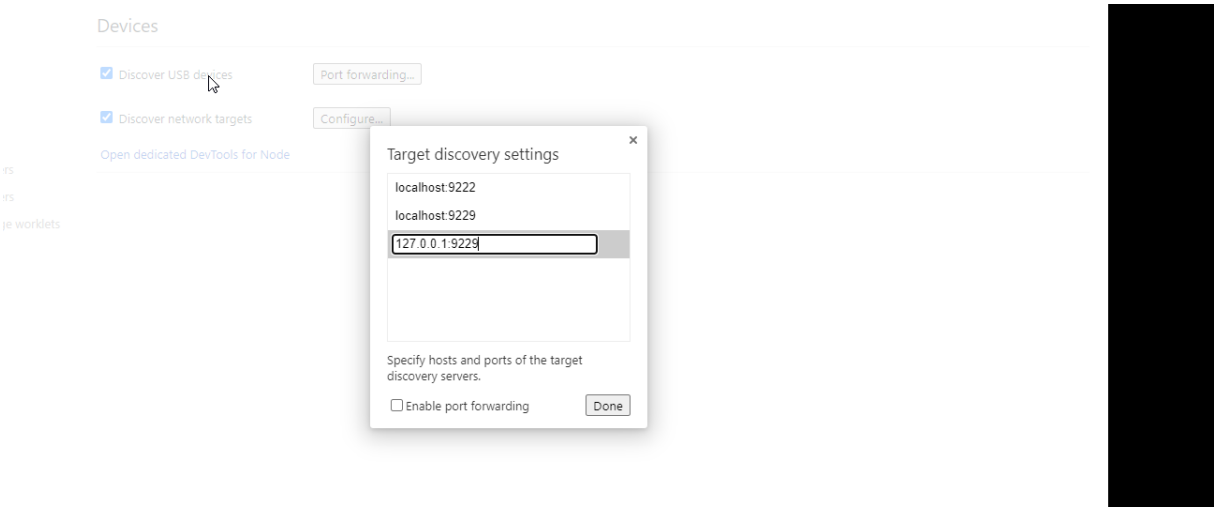
```
PS C:\Users\user\Desktop\2.hafta\2_week> node inspect app2.js add --title="pazartesi" --body="günü"
< Debugger listening on ws://127.0.0.1:9229/7e006f3b-e659-4182-9d66-590fd31b4c7f
< For help, see: https://nodejs.org/en/docs/inspector
<
connecting to 127.0.0.1:9229 ... ok
< Debugger attached.
<
Break on start in app2.js:1
> 1 const yargs= require('yargs')
  2 const notes = require('./not')
  3 yargs.version('1.1.0')
debug>
```

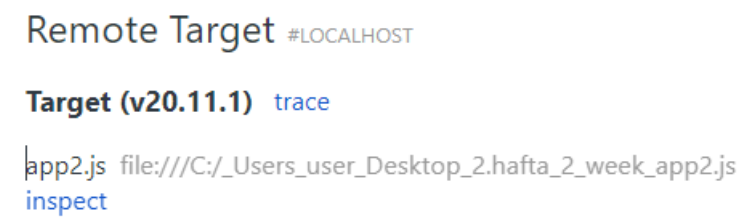
Bu debuggera erişmek için en kolay yöntem Chrome kullanmaktır. Chrome u açın yeni bir boş sayfa açın ve chrome://inspect/#devices bu adres bilgisini girin.

Sizi şu şekilde bir sayfa karşılamalıdır.

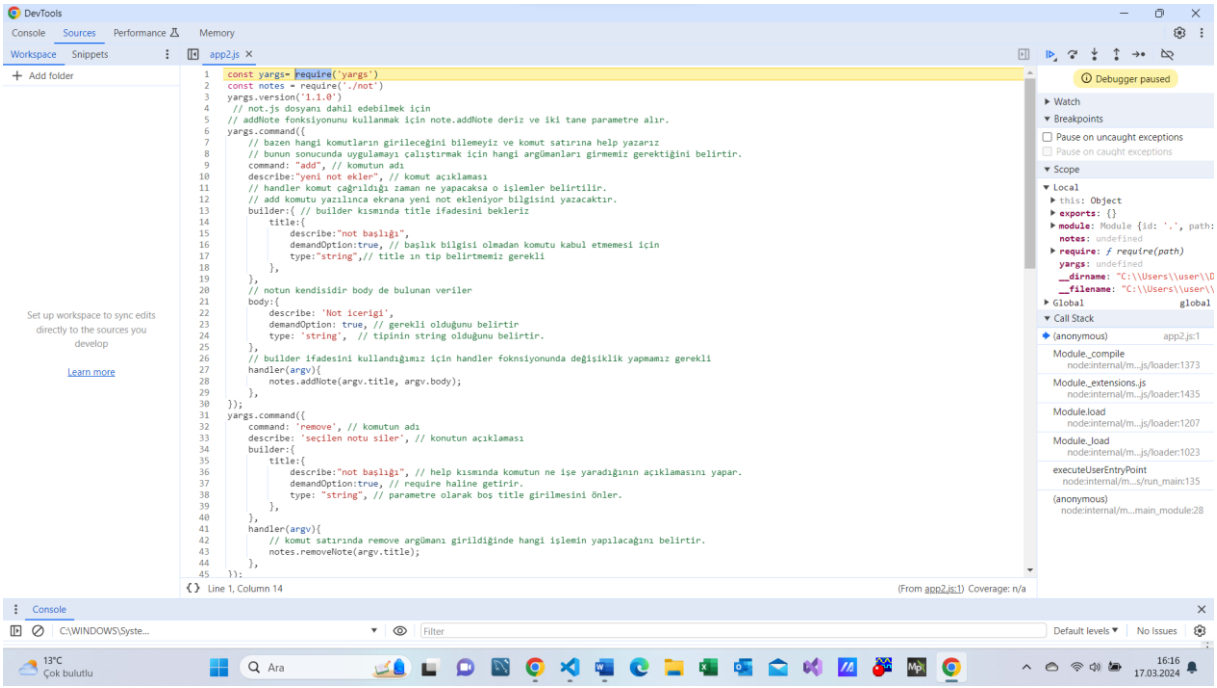


Eğer Remote Target kısmı gelmediyse configure kısmında bu port numarasını yazabilirisniz.

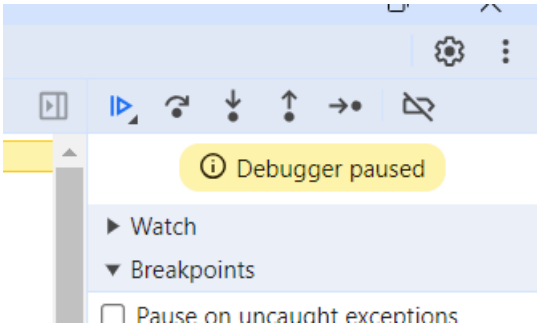




1127.0.0.1 demek ile localhost demek aynı şeydir her ikisi de sizin kendi bilgisayarınız anlamına gelmektedir. İnspect e tıklayınız.

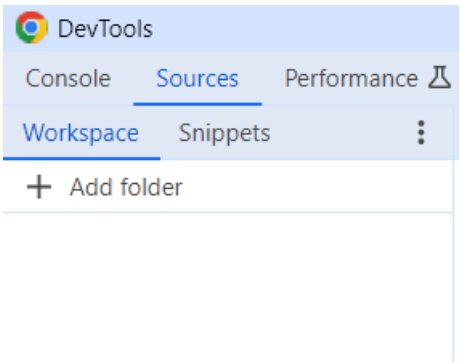


Bu şekilde bir sayfa karşınıza çıkmaktadır.

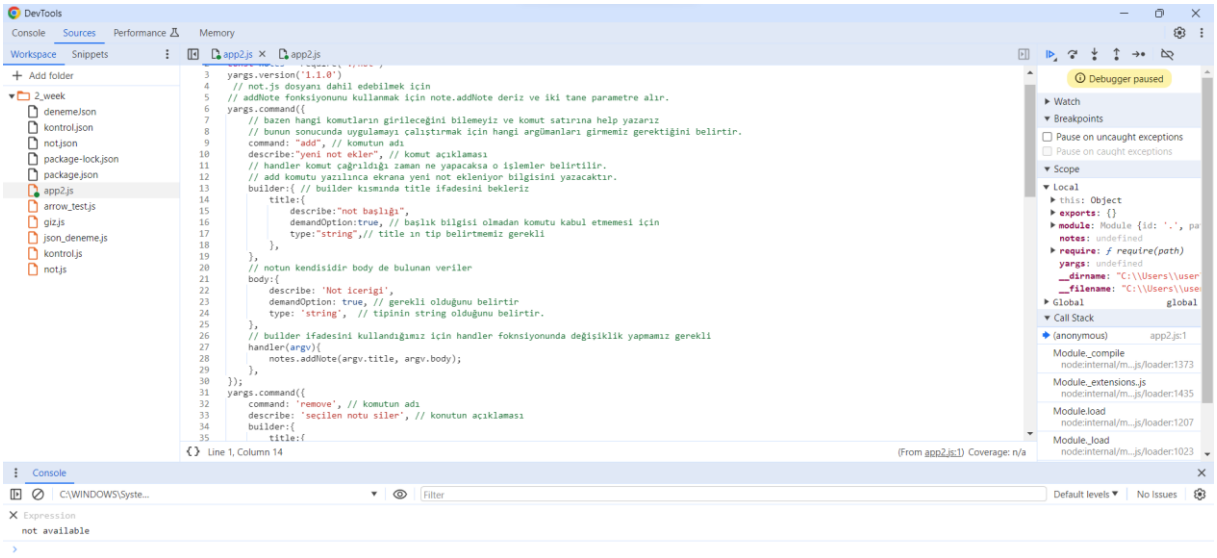


Sağ üst köşede bulunan oka bastığımızda debugger yazdığımız yere kadar debugging işlemi başlayacaktır.

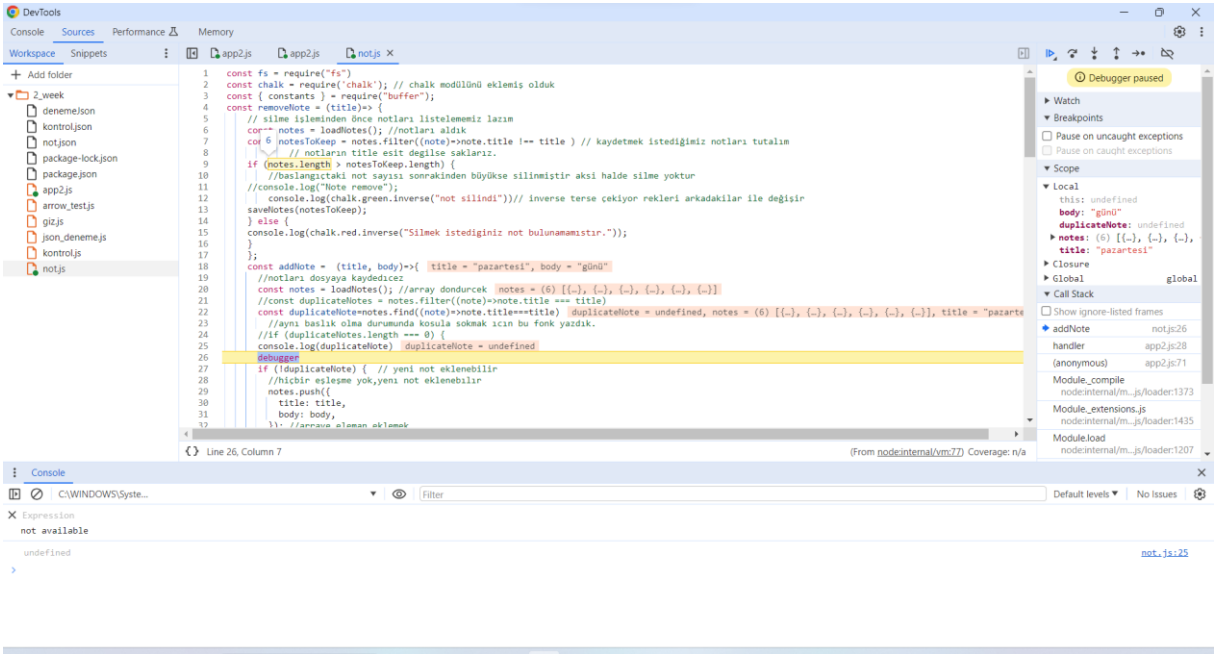
Sol üst köşede bulunan add folder kısmına tıklayın kendi kaynak kodunuzun bulunduğu konumu seçin ve açın.



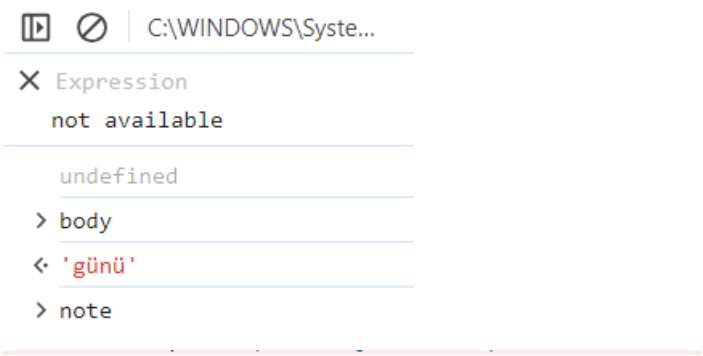
Daha sonra açılan sayfada dosyalara erişim sağlayabilmesi için izin vere tıklayınız.

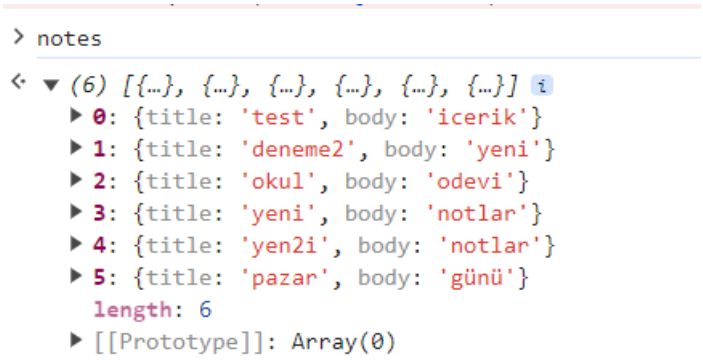


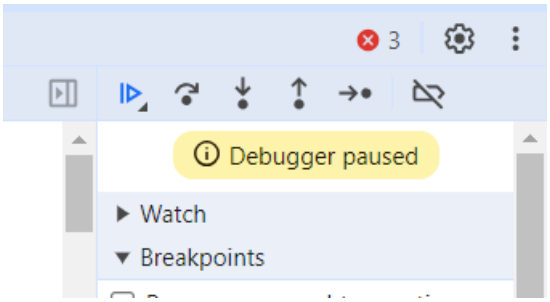
açılan sayfa bu şekildedir. Sağ da bulunan ok tuşuna basın ve debugging işlemini başlatın. kendisi otomatik olarak not.js dosyasını açar ve debugger komutunu görene kadar çalışıp durur.



Console body yazalım

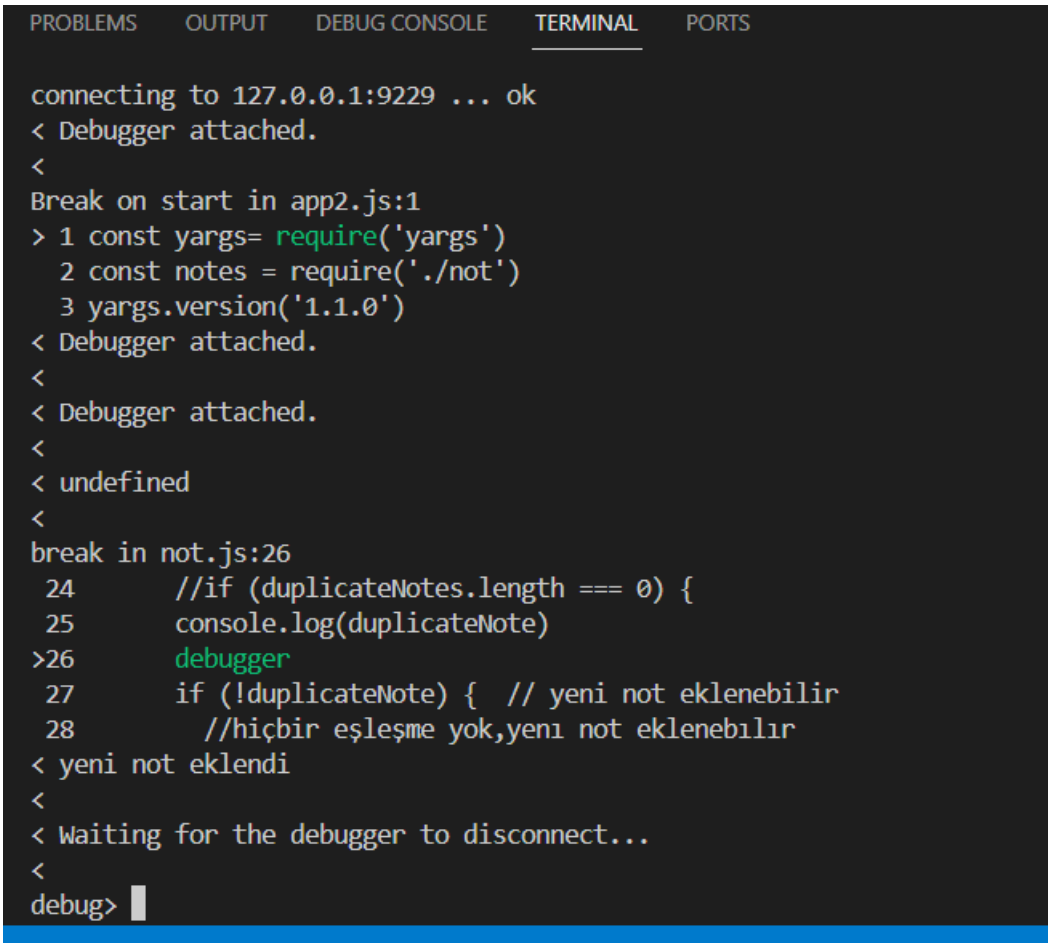




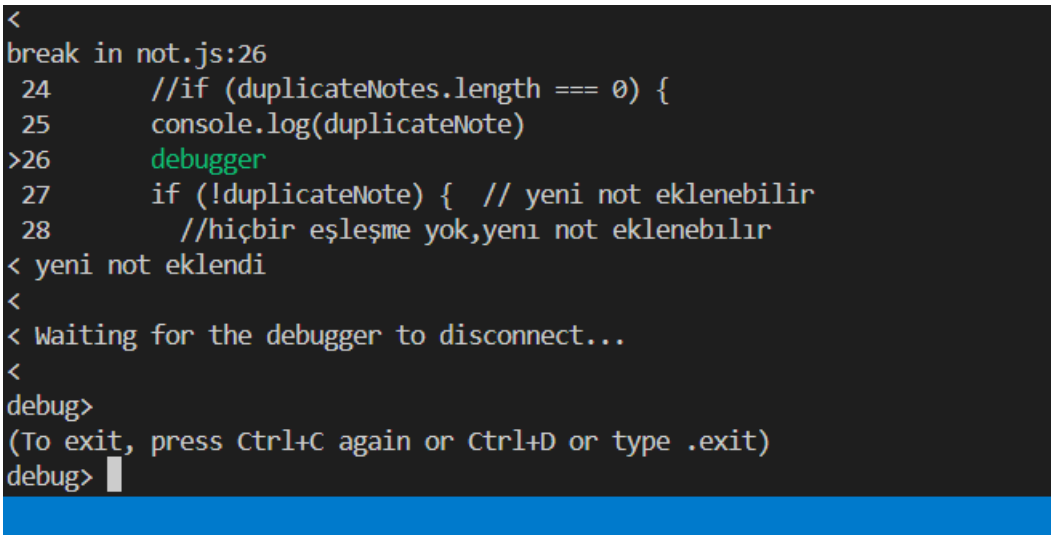


Mavi oka basarsanız kaldığı noktadan devam eder.

Visual studio code geri geldiğimizde debugging işleminin hala devam ettiğini görürüz.



Ctrl+c ye basalım.



Tekrar ctrl+c ye basalım ve çıkalım.

```
< Waiting for the debugger to disconnect...
<
debug>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
debug>
PS C:\Users\user\Desktop\2.hafta\2_week>
```

Debugging işelmini sonlandırdığımız için Remote Target kısmı boş görünecektir.