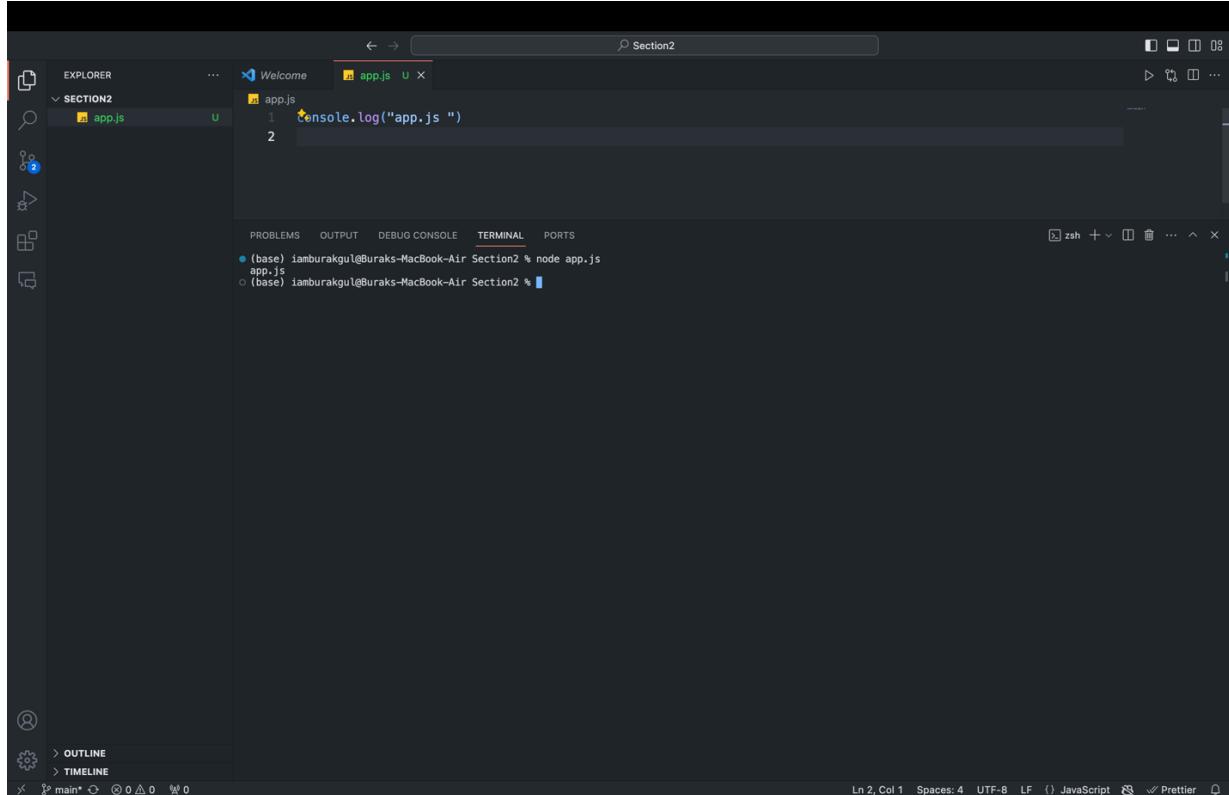


Node.js Section 2

Merhaba bugün terminal işlemleri üzerinden Node.js serimize devam ediyoruz ve 2.bölümdeyiz.

Section2 folderine gittim ve bir tane app.js adında file oluştururdum.İçinde olduğunu görmek için “console.log(“App.js”)” yazıyorum ve bu file i Section2 folderinin içinde node app.js ile çalıştırıralım ve çıktımızı alalım.



The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a 'SECTION2' folder containing an 'app.js' file. The main area shows the 'app.js' file content:

```
1 console.log("app.js")
2
```

The 'TERMINAL' tab is selected at the bottom, showing the command line output:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js
app.js
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

At the bottom status bar, it says 'Ln 2, Col 1 Spaces: 4 UTF-8 LF () JavaScript Prettier'.

Peki ben mesela burada app.js dosyasının çalışmasını isterken parametre verme şansım olsaydı onu nasıl yapardım bunu öğreneceğiz bu bölümde.

Birde node app.js BurakGül ile çalıştırıralım aynı sonucu alacağız.

```
console.log("app.js")
```

```
(base) iamburakgul@buraks-MacBook-Air Section2 % node app.js
app.js
(base) iamburakgul@buraks-MacBook-Air Section2 % node app.js BurakGÜL
app.js
(base) iamburakgul@buraks-MacBook-Air Section2 %
```

Peki bu eklediğim BurakGÜL argümanına nasıl erişirim ?

Bunun için app.js file'ına console.log(process.argv) yazıyoruz.

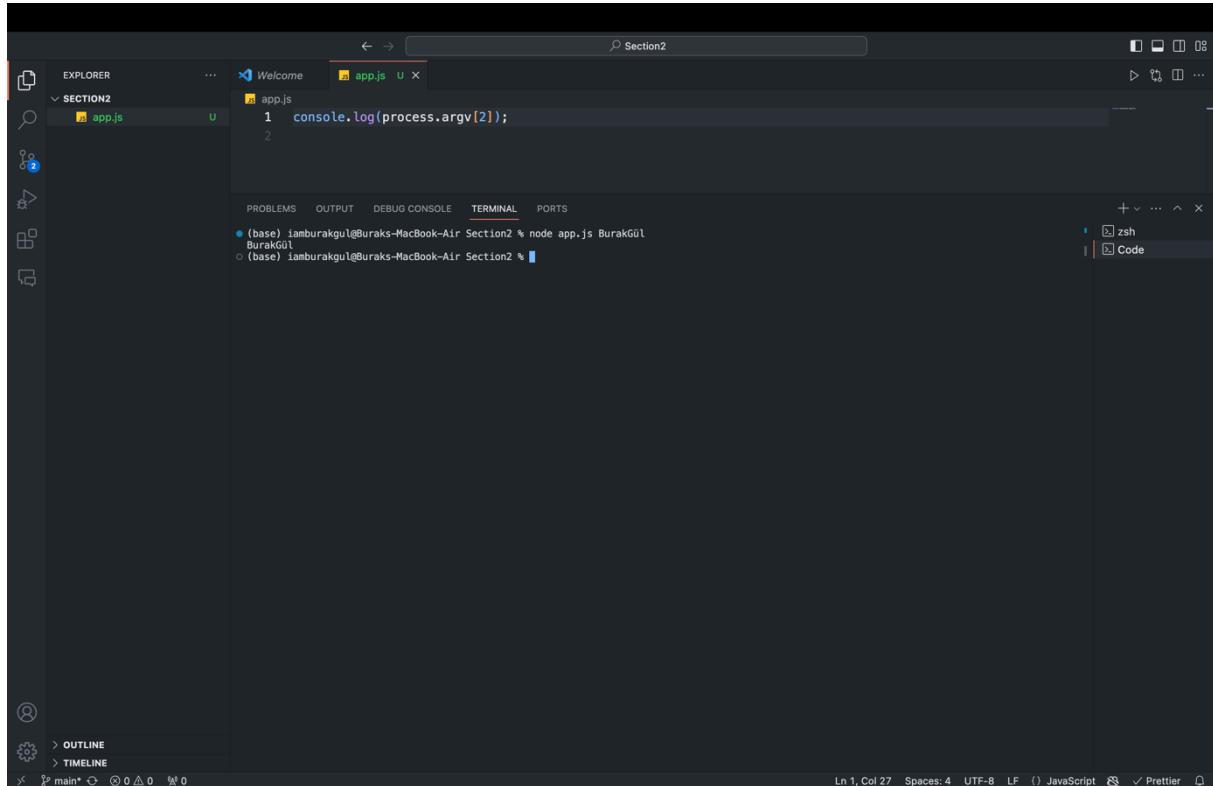
```
1 console.log(process.argv)
```

```
(base) iamburakgul@buraks-MacBook-Air Section2 % node app.js
app.js
(base) iamburakgul@buraks-MacBook-Air Section2 % node app.js BurakGÜL
app.js
(base) iamburakgul@buraks-MacBook-Air Section2 % node app.js BurakGÜL
[
  '/usr/local/bin/node',
  '/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section2/app.js',
  'BurakGÜL'
]
(base) iamburakgul@buraks-MacBook-Air Section2 %
```

Peki burada yazanlar nedir ?

Node.js'te terminalden bir komutla birlikte oluşumlar eklediğinizde, bu kolonilere process.argv dizisi üzerinden erişebilirsiniz. process.argv bir dizi olup, Node.js işlemi için komut satırlarının depolanmalarını içerir. Dizi, ilk eleman olarak node'un yoluyla başlar, ikinci eleman olarak çalıştırılan dosyanın yolunu içerir ve sonrasında komut satırlarından oluşan oluşumlar sırasıyla yer alır.

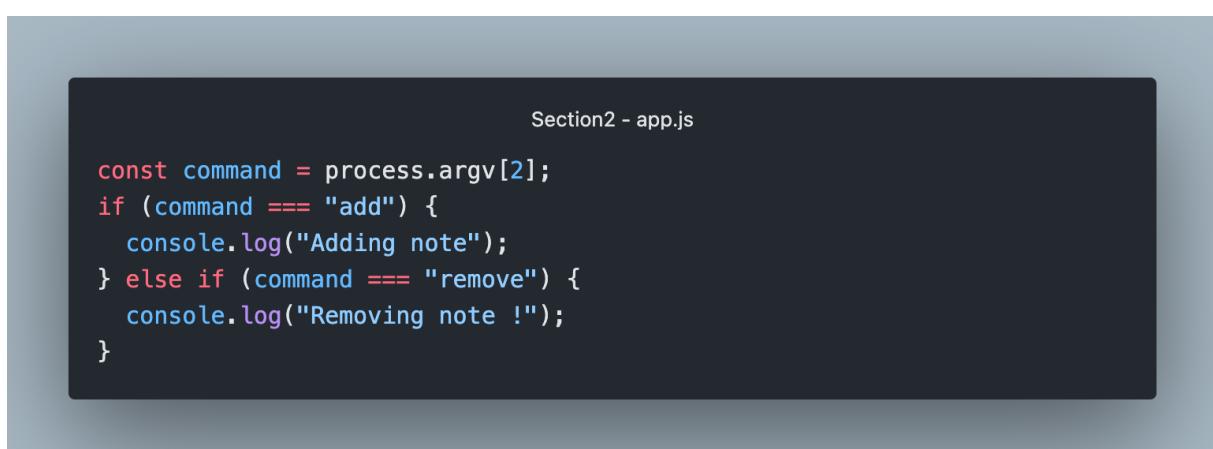
Sadece girilen argümanı yazdırmak için ise "console.log(process.argv[2])" ile yazdırabiliriz.



The screenshot shows the VS Code interface with a dark theme. In the center, there's a terminal window titled 'Section2'. It contains the command 'node app.js' and its output, 'BurakGÜL'. The terminal tab bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. On the left, the 'EXPLORER' sidebar shows a folder named 'SECTION2' containing an 'app.js' file. The bottom status bar displays 'Ln 1, Col 27' and other terminal settings.

Şimdi not alma uygulamasına çevirelim ve BurakGül yerine add,remove vs yazabilelim ve ona göre işlemlerimizi düzenleyelim.

Ve app.js fileımız şu şekilde olsun.



```
Section2 - app.js

const command = process.argv[2];
if (command === "add") {
  console.log("Adding note");
} else if (command === "remove") {
  console.log("Removing note !");
}
```

The screenshot shows the VS Code interface with a dark theme. In the Explorer sidebar, there is a folder named 'SECTION2' containing an 'app.js' file. The code in 'app.js' is as follows:

```
const command = process.argv[2];
if (command === "add") {
  console.log("Adding note");
} else if (command === "remove") {
  console.log("Removing note !");
}
```

The Terminal tab is active, displaying the output of running the script:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add
Adding note
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js remove
Removing note !
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

The status bar at the bottom indicates the code is in JavaScript and has been prettified.

Şimdi gördüğümüz üzere add ve remove çalışıyor.

Not kısmına gelince notumun bir title ve bir de content e ihtiyacı olacak onları da girebilmeliyim.

The screenshot shows the VS Code interface with a dark theme. In the Explorer sidebar, there is a folder named 'SECTION2' containing an 'app.js' file. The code in 'app.js' is as follows:

```
const command = process.argv[2];
if (command === "add") {
  console.log("Adding note");
} else if (command === "remove") {
  console.log("Removing note !");
}
console.log(process.argv)
```

The Terminal tab is active, displaying the output of running the script with an argument:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add --title="Title1"
Adding note
[
  '/usr/local/bin/node',
  '/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section2/app.js',
  'add',
  '--title=Title1'
]
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

The status bar at the bottom indicates the code is in JavaScript and has been prettified.

Fakat gördüğünüz gibi title a erişmem gerekiyor.

process.argv dizisinden verileri alırm title parametresi var mı yok mu tek tek kontrol ederim fonksiyonlar vs vs uğraşırıım. Bu çok yorucu olacaktır zaten bunu halleden paketler vardır.

Bunun için yarg packageını kullanacağız.

yargs'ı kullanmadan önce "npm init" ile npm başlatalım. Ve okeyleyelim hepsini daha sonra yarg'a geçelim.

<https://www.npmjs.com/package/yargs> bu siteden bakabilirsiniz.

"npm i yarg" ile terminalden indirelim.

The screenshot shows the Visual Studio Code interface. In the center, the code editor displays the file `app.js` with the following content:

```
1 const command = process.argv[2];
2 if (command === "add") {
3   console.log("Adding note");
4 } else if (command === "remove") {
5   console.log("Removing note !");
6 }
7
8 console.log(process.argv)
```

Below the code editor is the terminal pane, which shows the output of the command `npm i yarg`:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % npm i yarg
added 16 packages, and audited 17 packages in 1s
2 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

Haloldu gördüğünüz gibi.

Şimdi yarg'ı ekleyelim dosyamıza.

Şimdi ise yarg'ı kullanarak parametre'lere erişelim.

The screenshot shows a VS Code interface with the following details:

- Explorer:** Shows a folder named "SECTION2" containing "node_modules", "app.js", "package-lock.json", and "package.json".
- Terminal:** Displays the command `(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js --title="iOS Bakılacaklar "` followed by its output: `{ _: [], title: 'iOS Bakılacaklar ', '\$0': 'app.js' }`.
- Status Bar:** Shows "Ln 12, Col 1" and "Spaces: 2, UTF-8, LF". It also indicates the file is a JavaScript file and has been prettified.

Peki nedir bu yargs.argv ?

yargs.argv, yargs tarafından işlenen komut satırı argümanlarının bir objesini döner. Bu obje, argümanların isimlerini anahtar olarak ve argümanların kendilerini değer olarak içerir. Örneğin, komut satırına node app.js add --title="My Note" yazıldığında yargs.argv şu objeyi dönecektir:

```
Section2 - yargs.argv.json

{
  "_": ["add"],
  "title": "My Note",
  "$0": "app.js"
}
```

Bize bir JSON dönecektir.

Bu objede:

_ anahtarı, isimsiz argümanları bir dizi olarak içerir. Bu durumda, "add" komutu bir isimsiz argümandır.

"title" anahtarı, --title ile belirtilen argümanın değerini tutar.

"\$0" anahtarı, çalıştırılan scriptin adını içerir.

Tamam bunu da anladık.

Peki ben bu app.js nin nasıl çalıştığını nasıl bileceğim. Komut satırına "node app.js – help" yazalım bu bize gösterecektir.

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command `node app.js --help` and its output:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js --help
Options:
  --help      Show help
  --version   Show version number
  [boolean]
  [boolean]
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

Bana böyle bir şey geldi ama ben add remove bunları bu komut satırı arayüzünde help kısmına bir şeyler eklemek istiyorum bunun için yargs versiyonunu özelleştirelim.

Kodumuza `yargs.version("1.1.0")` I ekleyelim ve burada if else ile olan kodlara ihtiyacımız yok çünkü yargs bizim için bu ayrıştırma işlemini yapacaktır. Kodlarımızı silelim if else kısımdakileri.

Daha sonra yargs'a command ekleyelim.

yargs.command metodunun kullanımı, Node.js tabanlı komut satırı araçları (CLI) geliştirirken çok önemli bir rol oynar. Bu metod, özel komutların tanımlanması, bu komutlara ait seçeneklerin (options) belirlenmesi ve komutlar çalıştırıldığında çağrılacak işlevlerin (handler functions) atanması için kullanılır. Bu sayede, geliştirdiğiniz araçların kullanıcı dostu ve esnek bir yapıda olmasını sağlayabilirsiniz.

yargs.command Parametreleri

yargs.command metodunun temel parametreleri şunlardır:

- 1- command: Komutun adı ve gerekiğinde komutun alacağı parametreler. Komut parametreleri, <param> şeklinde zorunlu parametreler ve [param] şeklinde istege bağlı parametreler olarak belirtilebilir. Örneğin, add <file> komutu, add komutu için zorunlu bir file parametresi tanımlar.
- 2- describe: Komutun ne işe yaradığını açıklayan kısa bir açıklama. Bu açıklama, kullanıcının yardım (--help veya -h) talep ettiğinde gösterilir ve komutun amacını anamasına yardımcı olur.
- 3- builder: Komuta özgü seçenekleri (options) tanımlamak için kullanılır. Bu parametre, bir obje veya bir fonksiyon olabilir. Objede her bir seçenekin tanımı yapılırken, fonksiyon şeklinde kullanıldığında, fonksiyon yargs instance'ını parametre olarak alır ve seçenekleri zincirleme (chaining) yöntemi ile tanımlamanıza olanak tanır.
- 4- handler: Komut çağrıldığında yürütülecek işlevi tanımlar. Bu işlev, komutla birlikte girilen argümanları ve seçenekleri içeren bir obje (argv) alır. Genellikle, bu işlev komutun asıl işlemlerini gerçekleştirir.

Şimdi komut satırından help e gidelim bakalım neler çıkıyor daha sonra add komutunu çalıştıralım.

Ve app.js şu halde olunca çalışacaktır.

Section2 - app.js

```
const yargs = require("yargs");
yargs.version("1.1.0");
// Create add command
yargs.command({
  command: "add",
  describe: "Adds a new note",
  handler: function () {
    console.log("Adding a new note!");
  },
});
yargs.parse();
```

The screenshot shows a VS Code interface with the following details:

- Explorer View:** Shows a folder named "SECTION2" containing "node_modules", "app.js", "package-lock.json", and "package.json".
- Terminal View:** Displays the following command-line session:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js --help
app.js [command]
Commands:
  app.js add  Adds a new note
  Options:
    --help      Show help
    --version   Show version number
[boolean] [boolean]
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add
Adding a new note!
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```
- Status Bar:** Shows "Ln 1, Col 1 (230 selected)" and file statistics: "Spaces: 2", "UTF-8", "LF", "(JavaScript)", "Prettier".

Burada “node app.js –help” diyince ekran geldi. “node app.js add” diyince ise handler fonksiyonu çalıştı.

Şimdi aynı şekilde list, remove ve read command leri ekleyelim.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a folder named "SECTION2" containing "node_modules", "app.js", "package-lock.json", and "package.json".
- Terminal:** The title bar says "Section2". The terminal shows the following code in app.js:

```
    17  });
    18  yargs.command({
    19      command: "list",
    20      describe: "List your notes",
    21      handler: function () {
    22          console.log("Listing all notes !");
    23      },
    24  });
    25  yargs.command({
    26      command: "read",
    27      describe: "Read a note",
    28      handler: function () {
    29          console.log("Reading a note!");
    30      },
    31  });
    32
    33  yargs.parse();
    34
```

- Output:** The terminal also displays the help output for the app.js command:

```
● (base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js --help
app.js [command]

Commands:
  app.js add      Adds a new note
  app.js remove   Remove a new note
  app.js list     List your notes
  app.js read     Read a note

Options:
  --help           Show help
  --version        Show version number
  [boolean]
  [boolean]
```

Ekledik ve help diyince command ler geldi.

Şimdi tek tek deneyelim.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a folder named "SECTION2" containing "node_modules", "app.js", "package-lock.json", and "package.json".
- Terminal:** The title bar says "Section2". The terminal shows the following code in app.js:

```
    17  });
    18  yargs.command({
    19      command: "list",
    20      describe: "List your notes",
    21      handler: function () {
    22          console.log("Listing all notes !");
    23      },
    24  });
    25  yargs.command({
    26      command: "read",
    27      describe: "Read a note",
    28      handler: function () {
    29          console.log("Reading a note!");
    30      },
    31  });
    32
    33  yargs.parse();
    34
```

- Output:** The terminal shows the results of running the application with various commands:

```
● (base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add
Adding a new note!
● (base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js remove
Removing the note!
● (base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js list
Listing all notes !
● (base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js read
Reading a note!
○ (base) iamburakgul@Buraks-MacBook-Air Section2 %
```

Gördüğünüz gibi handler lar tek tek çalışıyor.

Şimdi her command e tek tek builder ekleyelim ben add için yapacağım diğerleri içinde siz yaparsınız.

Builder ne işe yarıyordu : komuta özgü optionları tanımlamak için kullanırız.

handler fonksiyonunda argv parametresi, kullanıcının komut satırı aracılığıyla girdiği argümanların ve seçeneklerin (options) işlenmiş bir formunu içeren bir objedir. args kütüphanesi, komut satırından gelen girdileri ayırtırır (parse eder) ve bu bilgileri argv objesinde toplar.

```
Section2 - app.js

yargs.command({
  command: "add",
  describe: "Adds a new note",
  builder:{
    title:{
      describe: "Note title"
    }
  },
  handler: function (argv) {
    console.log("Adding a new note!",argv);
  },
});
```

Bu command e builder ekledik ve çalıştıralım bakalım.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows a folder named "SECTION2" containing "node_modules", "app.js", "package-lock.json", and "package.json".
- CODE EDITOR**: The active file is "app.js". The code defines a command-line application with an "add" command:

```
3 // Create add command
4 args.command({
5   command: "add",
6   describe: "Adds a new note",
7   builder: {
8     title: {
9       describe: "Note title"
10    }
11 },
12 handler: function (argv) {
13   console.log("Adding a new note!", argv);
14 },
15});
```
- TERMINAL**: Shows the terminal output of running the application with and without parameters:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add --title="Burak Gül Note Başlığı 1"
Adding a new note! { _: [ 'add' ], title: 'Burak Gül Note Başlığı 1', '$0': 'app.js' }
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add
Adding a new note! { _: [ 'add' ], '$0': 'app.js' }
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```
- STATUS BAR**: Shows the current file is "app.js", line 10, column 6, spaces: 2, encoding: UTF-8, LF, and the file is a JavaScript file.

Parametreyi verip çalıştırıldım ve sonra parametresiz de deneyince çalışıyor yani title gerekli değildir bunu gerekli yapalım.

title in içine “demandOption: true” eklemeliyiz.

The screenshot shows the VS Code interface with the following details:

- CODE EDITOR**: The active file is "app.js". The code has been updated to include the "demandOption: true" option for the "title" parameter:

```
yargs.command({
  command: "add",
  describe: "Adds a new note",
  builder: {
    title: {
      describe: "Note title",
      demandOption: true
    }
  },
  handler: function (argv) {
    console.log("Adding a new note!", argv);
  },
});
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a file named "app.js" with a size of 168 bytes.
- Code Editor:** Displays the following code for a command-line application using yargs:

```
1 const yargs = require("yargs");
2 yargs.version("1.1.0");
3 // Create add command
4 yargs.command({
5   command: "add",
6   describe: "Adds a new note",
7   builder: {
8     title: {
9       describe: "Note title",
10      demandOption: true
11    }
12  },
13  handler: function (argv) {
14    console.log(`Adding a new note! { _: [ '$0' ], title: '${argv.title}' }`);
15  }
});
```

- Terminal:** Shows the following terminal output:
 - Running `node app.js add --title="Burak Gül Note Başlığı 1"` adds a note with title "Burak Gül Note Başlığı 1".
 - Running `node app.js add` without a title results in an error: "Missing required argument: title".
- Status Bar:** Shows the current file is "app.js", the line count is 168, and the file is saved.

Parametreli deneyince çalıştı ama parametresiz çalışmadı gördüğümüz gibi.

Ve tipide string olması gerektiği için title a "type: "string"" ekleyelim.

```
Section2 - app.js

yargs.command({
  command: "add",
  describe: "Adds a new note",
  builder:{
    title:{
      describe: "Note title",
      demandOption: true
    }
  },
  handler: function (argv) {
    console.log("Adding a new note!", argv);
  },
});
```

Son hali bu oldu command imizin.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a single file named "app.js".
- Editor:** Displays the code for the "add" command. Line 16 is highlighted in red, indicating an error or warning.
- Terminal:** Shows the command "node app.js add --title" being run, followed by the output "Adding a new note! { _: ['add'], title: '' }".
- Status Bar:** Shows file statistics: Ln 13, Col 5, Spaces: 2, UTF-8, LF, {}, JavaScript, Prettier.

Ve bu şekilde deneyince titleımız boş bir string oldu.

Şimdi notumuzun body si de olmalı ve onu da dolduralım builder içine ve handler fonksiyonunu da güncelleyelim.

Section2 - app.js

```
yargs.command({  
  command: "add",  
  describe: "Adds a new note",  
  builder: {  
    title: {  
      describe: "Note title",  
      demandOption: true,  
      type: "string",  
    },  
    body: {  
      describe: "Body title",  
      demandOption: true,  
      type: "string",  
    },  
  },  
  handler: function (argv) {  
    console.log("Title: ", argv.title)  
    console.log("Body: ", argv.body)  
  },  
});
```

Ve çalıştırıralım body miz de zorunlu / gerekli olduğundan ötürü vermezsek hata ile karşılaşırız.

```
1 // Create add command
2 args.command({
3   command: "add",
4   describe: "Adds a new note",
5   builder: {
6     title: {
7       describe: "Note title",
8       demandOption: true,
9       type: "string",
10    },
11    body: {
12      describe: "Body title",
13      demandOption: true,
14      type: "string",
15    },
16  },
17});
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add --title="Title Burak Gül"
app.js add
Adds a new note
Options:
  --help      Show help
  --version   Show version number
  --title    Note title
  --body     Body title
               [boolean]           [boolean]
               [string] [required] [string] [required]
```

Missing required argument: body
(base) iamburakgul@Buraks-MacBook-Air Section2 %

Şimdi body kısmını da ekleyelim.

```
1 // Create add command
2 args.command({
3   command: "add",
4   describe: "Adds a new note",
5   builder: {
6     title: {
7       describe: "Note title",
8       demandOption: true,
9       type: "string",
10    },
11    body: {
12      describe: "Body title",
13      demandOption: true,
14      type: "string",
15    },
16  },
17},
18 handler: function (argv) {
19   console.log("Title: ", argv.title)
20   console.log("Body: ", argv.body)
21 },
22});
23});
24});
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node app.js add --title="Title Burak Gül" --body="iOS Developer"
Title: Title Burak Gül
Body: iOS Developer
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

Şimdi geriye kalan remove list ve read kısımlarını da doldurmadan önce JSON ile data store etmeye bakalım. Remove list ve read kısımlarını sonradan yapacağız. Veriyi kaldırırmam okumam listlemem için önce kaydetmeliyim mantıken.

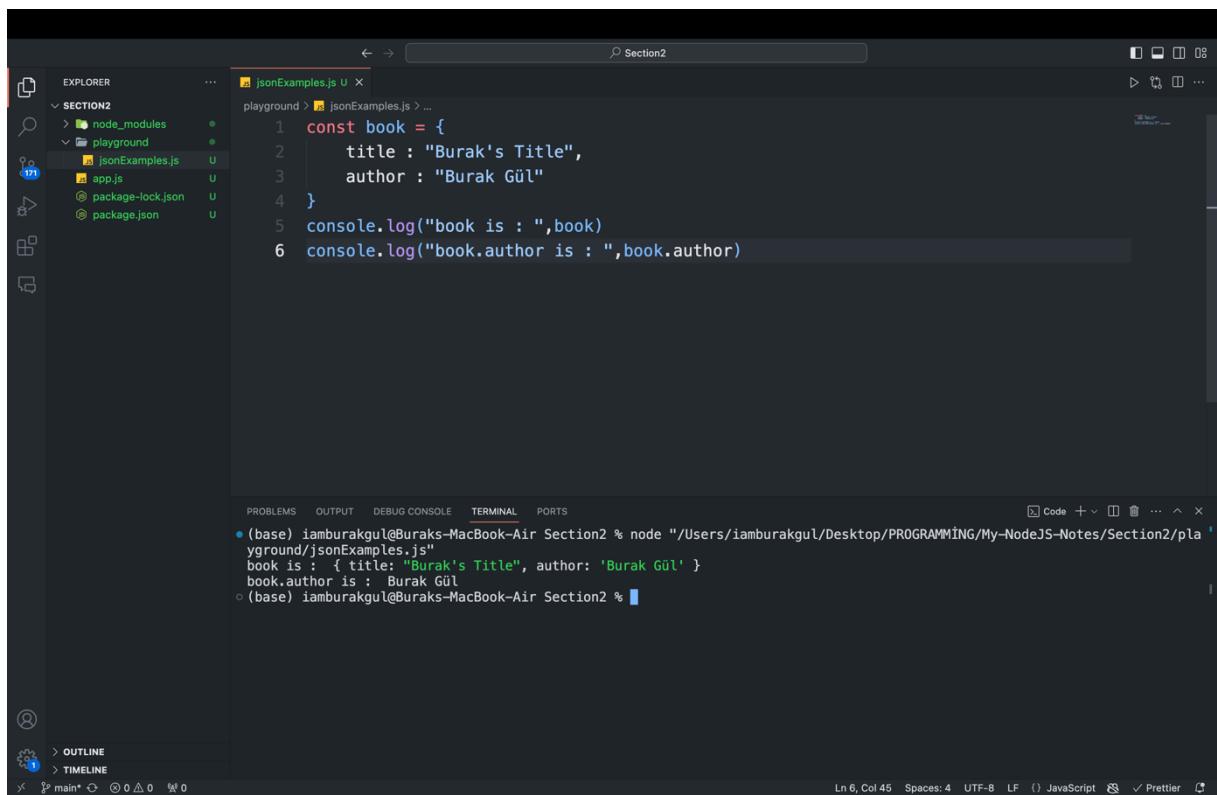
Store Data With JSON :

JSON ile işlem yapabilmek için bir şey indirmemize gerek yok JavaScript kendisi otomatik olarak desteklemektedir.

Yeni bir folder da işlemlerimizi halledelim.

Playground adında bir folder ve içinde de jsonExamples.js adında bir file oluşturalım ve içini tek tek dolduralım.

Önce bir object oluşturalım ve kendisini ve özelliklerini yazdırıralım.



The screenshot shows a dark-themed VS Code interface. In the Explorer sidebar, there's a folder named 'playground' containing files like 'app.js', 'package-lock.json', and 'package.json'. Inside 'playground', there's a sub-folder 'jsonExamples.js' which contains a single file named 'JsonExamples.js'. The code in 'JsonExamples.js' is as follows:

```
const book = {
  title : "Burak's Title",
  author : "Burak Gül"
}
console.log("book is : ",book)
console.log("book.author is : ",book.author)
```

In the Terminal tab, the output shows the execution of the script and its results:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section2/playground/jsonExamples.js"
book is : { title: "Burak's Title", author: 'Burak Gül' }
book.author is : Burak Gül
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

The status bar at the bottom indicates the code is in JavaScript mode, has 45 lines and 4 spaces, and is using LF line endings.

Şimdi ise

JSON sınıfının static bir methodu olan `stringify` i kullanarak objectimizi JSON stringine çevirelim.

The screenshot shows a VS Code interface with the following details:

- EXPLORER** sidebar: Shows a project structure with `SECTION2` containing `node_modules`, `playground` (which has `jsonExamples.js`), and files `app.js`, `package-lock.json`, and `package.json`.
- TERMINAL** tab: Displays the output of running `node jsonExamples.js`. The terminal shows the following code execution:

```
const book = {
  title : "Burak's Title",
  author : "Burak GÜL"
}
console.log("book is : ",book)
console.log("book.author is : ",book.author)
const bookJSON = JSON.stringify(book)
console.log(bookJSON)
```

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section2/playground/jsonExamples.js"
book is : { title: "Burak's Title", author: 'Burak GÜL' }
book.author is : Burak GÜL
{"title":"Burak's Title","author":"Burak GÜL"}
```
- STATUS BAR**: Shows file statistics: Ln 8, Col 22, Spaces: 4, UTF-8, LF, JavaScript, and Prettier icons.

Peki biz bu JSON verilerini ne yapacağız neden string haline getirdik.

XML ve JSON veri aktarımında kullanılan formatlardır.

Diyelim Java'da C#'ta Swift'te bir object oluşturduk ve bunu karşı tarafa göndermek istiyorsak(başka bir yere) verilerimizi string hale getiririz daha az yer kaplasınlar ve hızlı olsun diye. JavaScript nesne/object indeki boşluklar silinir,propertyler de string içine alınarak gönderilir.

JSON halinde olan book'dan yani bookJSON'dan book objectinin title property'sine erişemeyiz gelin deneyelim.

Gördüğünüz gibi bookJSON bir stringdir. Bu yüzden string methodları geldi.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project structure with 'SECTION2' expanded, containing 'node_modules', 'playground', and files like 'jsonExamples.js', 'app.js', 'package-lock.json', and 'package.json'. The 'jsonExamples.js' file is open in the editor. The code is as follows:

```
// ! Burada verimizi JSON String haline getirdik
const bookJSON = JSON.stringify(book)
console.log(bookJSON)

// ! Şimdi bu JSON String halinde olan veriyden ana veriye erişmeye çalışalım
console.log(bookJSON.)
```

The cursor is at the end of 'bookJSON.' on line 11. A code completion dropdown is open, listing methods starting with 'charAt':

- (method) String.charAt(p os: number): string
- Returns the character at the specified index.
- @param pos — The zero-based index of the desired character.
- charAt
- charCodeAt
- codePointAt
- concat
- endsWith
- includes
- indexOf
- lastIndexOf
- length
- localeCompare
- match
- matchAll
- normalize
- padEnd
- padStart
- repeat
- replace
- search
- slice

At the bottom of the interface, status bar text includes 'Ln 11, Col 22 Spaces: 4 UTF-8 LF () JavaScript ⚡ Prettier ⚡'.

Deneyelim title property'sine erişmeyi.

A screenshot of the Visual Studio Code interface, similar to the previous one but with different code in the editor and a terminal tab open at the bottom.

The 'jsonExamples.js' file now contains:

```
// ! Burada verimizi JSON String haline getirdik
const bookJSON = JSON.stringify(book)
console.log(bookJSON)

// ! Şimdi bu JSON String halinde olan veriyden ana veriye erişmeye çalışalım
console.log(bookJSON.title)
```

The terminal tab shows the output of running the script:

```
(base) iamburakgul@Buraks-MacBook-Air Section2 % node "/Users/iamburakgul/Desktop/PROGRAMMING/My-NodeJS-Notes/Section2/playground/jsonExamples.js"
{"title":"Burak's Title","author":"Burak Gül"}
undefined
(base) iamburakgul@Buraks-MacBook-Air Section2 %
```

At the bottom, the status bar shows 'Ln 11, Col 27 Spaces: 4 UTF-8 LF () JavaScript ⚡ Prettier ⚡'.

Undefined yazısından görüyoruz hata olduğunu.

The screenshot shows a VS Code interface with the following details:

- Explorer View:** Shows a project structure with a folder named "SECTION2" containing "node_modules", "playground", and "jsonExamples.js".
- Code Editor:** Displays a file named "jsonExamples.js" with the following content:

```
const book = {  
    title : "Burak's Title",  
    author : "Burak Gül"  
}  
// ! Burada verimizi JSON String haline getirdik  
const bookJSON = JSON.stringify(book)  
console.log("JSON String is : ",bookJSON);  
  
// ! Şimdi bu JSON String halinde olan veriyden ana veriye erişmeye çalışalım  
// console.log(bookJSON.title) // ! hata olacaktır  
  
// * Şimdi ise JSON String verisinden object elde edelim  
// * yani parse etmeliyiz , ayırtılmalıdır.  
const parsedObject = JSON.parse(bookJSON)  
console.log("Parsed JSON Object",parsedObject)
```
- Terminal:** Shows the output of running the script: "JSON String is : {"title":"Burak's Title","author":"Burak Gül"} Parsed JSON Object { title: 'Burak's Title', author: 'Burak Gül' }".
- Status Bar:** Shows the file is a JavaScript file ("JavaScript") and is prettier-formatted.

Buradan da göreceğiniz gibi JSON String verisi ile JavaScript object i farklı şeylerdir.

The screenshot shows a dark-themed code editor window with the following details:

- Title Bar:** "Section2 - jsonExamples.js"
- Code Editor:** Displays the same "jsonExamples.js" code as the previous screenshot.
- Output Area:** Shows the terminal output:

```
console.log(parsedObject.title)  
console.log(parsedObject.author)
```

Bunları da yazarak parse ettiğimiz JSON Stringinden gelen verilerin property'lerine erişebiliriz.