



Node.JS ile Web Programlama Dersi

Rapor - 1

Mahire Zühal Özdemir

İÇİNDEKİLER

ŞEKİLLER DİZİNİ	3
Node JS	5
Node Js Nedir?	5
Node Js Kurulumu	5
Node Js Çalışma Ortamları Nelerdir?	7
Node Js Uygulamaları	8
Node JS Versiyon Kontrol İşlemi	8
Js kodunu Browser ve Node Uygulaması İle Çalıştırmak	9
Ekranı Metin Yazdırmak	10
Oluşturulan Dosyayı Çağırma	10
Node JS Dosyaları İçerisinde Değişken Çağırma	11
Node JS Dosyaları İçerisinde Fonksiyon Çağırma	12
Node JS ile Dosya İşlemleri	12
NPM Nedir?	14
Npm Validator Modülü	14
Node Js Paket Kurma / CHALK	15
Npm Initialize İşlemi	17

ŞEKİLLER DİZİNİ

Şekil 1: Node.Js Download Ekranı	6
Şekil 2: Node.JS Download Sayfası	6
Şekil 3: Visual Studio Code Anasayfası	7
Şekil 4: Atom Ide Anasayfası	7
Şekil 5: Sublime Text Editör Anasayfası	8
Şekil 6: Webstorm Editör Anasayfası	8
Şekil 7: Node JS Versiyon Kontrol İşlemi	8
Şekil 8: Browser Üzerinde Basit Toplama İşlemi	9
Şekil 9: Komut Satırında Node Uygulamasını Açmak	9
Şekil 10: Komut Satırında Basit Toplama İşlemi	9
Şekil 11: index.js Dosyası	10
Şekil 12: Komut satırında index.js dosyasını çalıştırmak	10
Şekil 13: utils.js dosyası içeriği	10
Şekil 14: index.js dosyası	11
Şekil 15: index.js Çıktısı	11
Şekil 16: Exports Komutu	11
Şekil 17: Index.js Dosyası Require Komutu	11
Şekil 18: Index dosyası çıktısı	12
Şekil 19: Add fonksiyonu exports işlemi	12
Şekil 20: index.js dosyası içerisinde add fonksiyonu çağırma	12
Şekil 21:index.js çıktısı	12
Şekil 22: Dosya İşlemleri	13
Şekil 23: Txt içeriği	13
Şekil 24: notes.js dosyası	13
Şekil 25: app.js dosyası	14
Şekil 26: app.js çıktısı	14
Şekil 27: Validator Modülü Yükleme	15
Şekil 28: Validator modülü	15
Şekil 29: app.js Çıktısı	15
Şekil 30: Chalk Npm Anasayfası	16
Şekil 31: Chalk Kurulum	16

Şekil 32: Uninstall Chalk	16
Şekil 33: Chalk 4.0 Kurma	17
Şekil 34: Chalk Blue Kullanımı	17
Şekil 35:npm init	18
Şekil 36: package.json dosyası	18

Node JS

Node Js Nedir?

Node.js, JavaScript tabanlı bir çalıştırma ortamıdır. İstemci taraflı JavaScript'in sınırlamalarını aşmak için geliştirilmiştir.

JavaScript, öncelikle istemci tarafında çalışan bir dildi. İstemci tarafında çalışmanın dezavantajı, manipülasyona açık olması ve yalnızca tarayıcı tarafından sağlanan izinlerle sınırlı olmasıydı.

Daha sonraları Chrome V8 Engine kullanıma sunuldu ve böylece sunucu tarafında da JS işlemleri gerçekleştirilebildi.

NodeJs; V8 Engine üzerinden geliştirilmiş sunucu tarafında çalışan Javascript çalışma ortamı olarak tanımlanabilir. Hem backend hem frontend'de kullanılabilir olduğu için Node JS bilerek full-stack olarak uygulamalar geliştirmek mümkündür.

NodeJS'in diğer dillerden farkı; asenkron ve non-blocking olarak çalışmasıdır.

Asenkron çalışma; bir işlemin sonucunu beklemeden diğer işlemlerin çalışmasına denir. Bu sayede işlemler çok zaman almadan çalıştırılır.

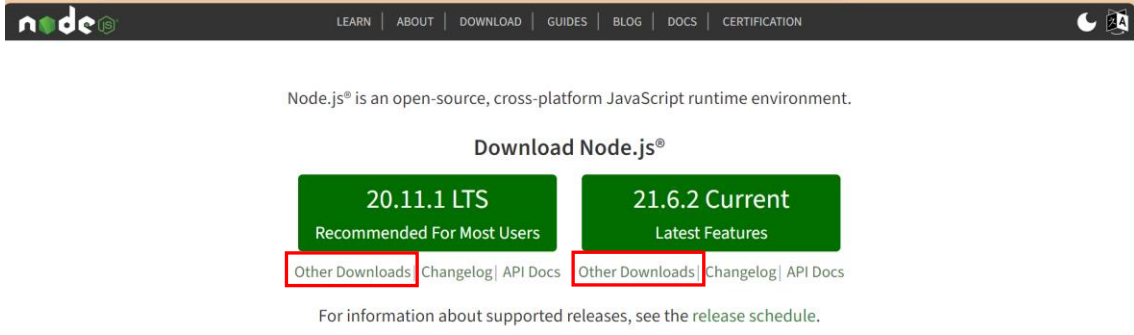
Non-blocking; Sürdürülen işlemlerin birbirinden bağımsız çalışması ve birbirini bloke etmemesi işlemine non-blocking denir.

Event-Driven; Node.js, olayları (events) dinler ve bu olaylar gerçekleştiğinde belirtilen işlevleri çağırır. Bu sayede asenkron ve etkili bir şekilde çalışabilir.

Callback; Node.js'te geri arama (callback), asenkron programlama modelinin temel bir parçasıdır. Geri aramalar, işlevleri diğer işlevlerin içine parametre olarak iletmek ve bu işlevlerin tamamlanmasından sonra çağrılmak üzere kullanılır. Bu özellik, Node.js'in etkili bir şekilde asenkron çalışmasını sağlar.

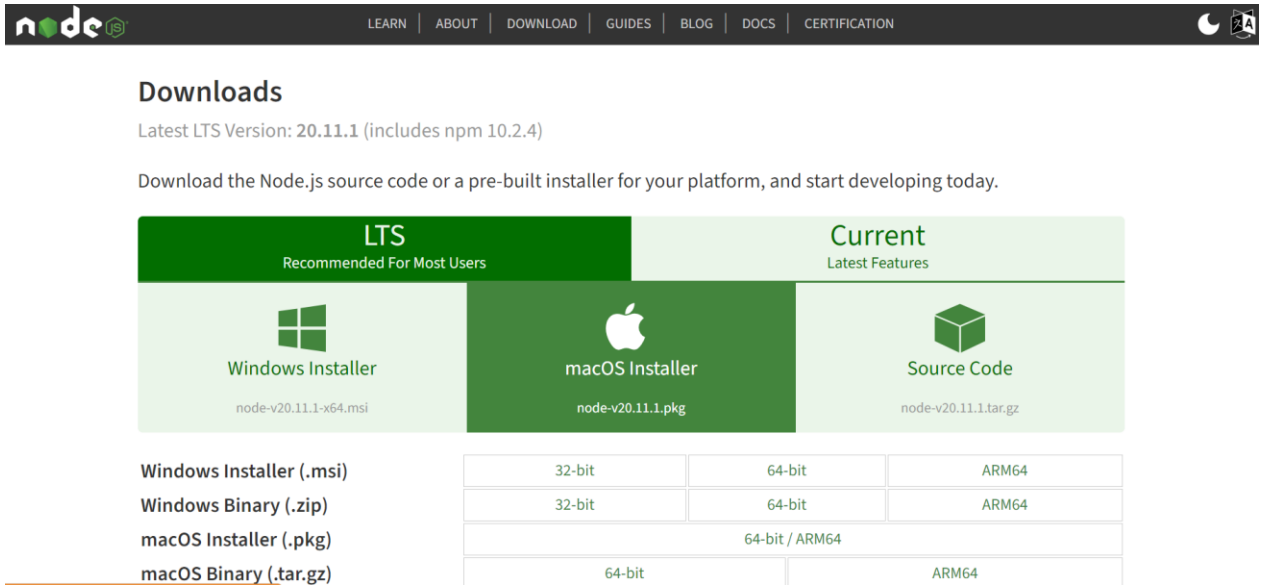
Node Js Kurulumu

- [Node Js web sayfasına](#) gidilir.
- Windows işletim sistemi için sayfada görülen indirme seçeneklerinden uygun olan seçilerek indirme işlemi tamamlanır.



Şekil 1: Node.Js Download Ekranı

- Diğer işletim sistemleri için '**Other Downloads**' seçeneği seçilerek işletim sistemi için uygun seçenek seçilerek indirme işlemi tamamlanır.

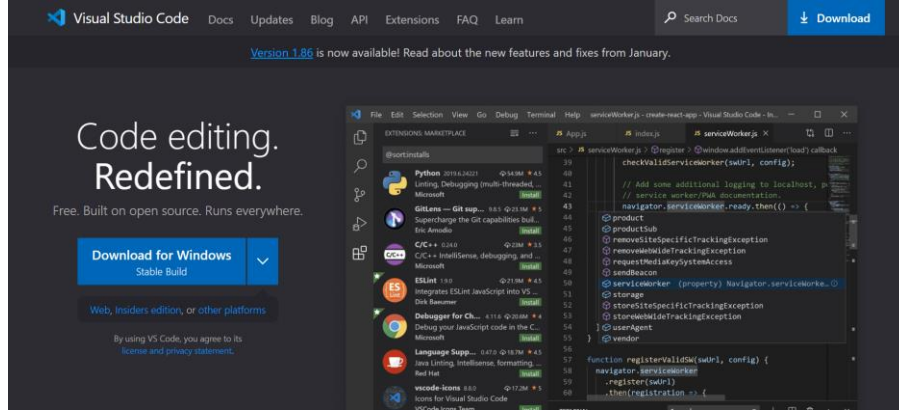


Şekil 2: Node.JS Download Sayfası

Node Js Çalışma Ortamları Nelerdir?

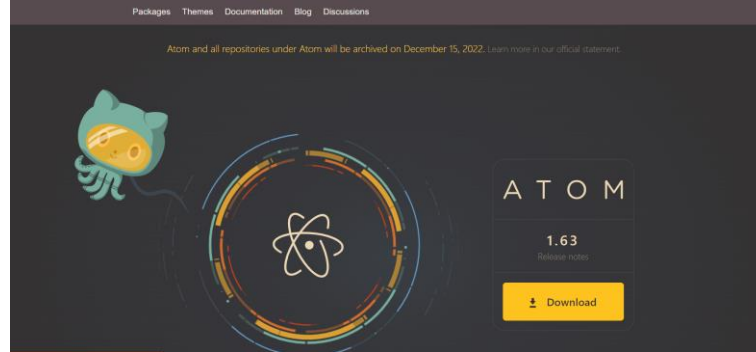
Node.js uygulamaları, geliştirilirken Visual Studio Code, Atom, Sublime Text ve WebStorm gibi uygulamalar kullanılabilir.

Visual Studio Code indirmek için [Tıklayınız](#).



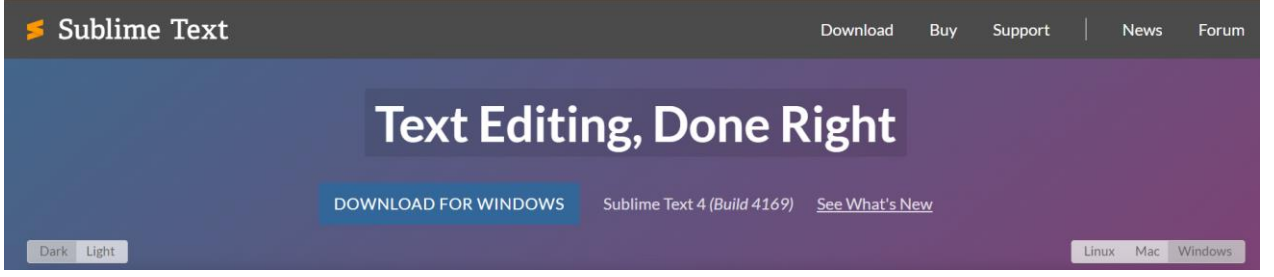
Şekil 3: Visual Studio Code Anasayfası

Atom Ide indirmek için [Tıklayınız](#).



Şekil 4: Atom Ide Anasayfası

Sublime Text indirmek için [Tıklayınız](#).



Şekil 5: Sublime Text Editör Anasayfası

Jet Brains tarafından geliştirilmiş webstorm indirmek için [tıklayınız](#).



Şekil 6: Webstorm Editör Anasayfası

Node Js Uygulamaları

Node JS Versiyon Kontrol İşlemi

İlk olarak komut satırını açarak versiyon kontrol işlemini gerçekleştirelim.

```
C:\Users\zuhal>node -v  
v16.15.0
```

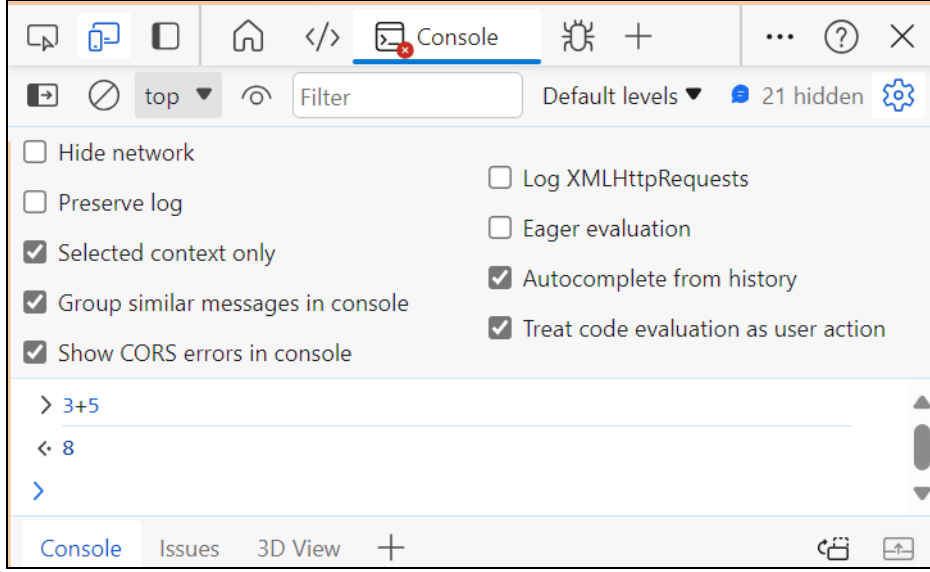
Şekil 7: Node JS Versiyon Kontrol İşlemi

Versiyonumuzu kontrol ettikten sonra uygulama geliştirme adımlarına geçebiliriz.

Js kodunu Browser ve Node Uygulaması İle Çalıştırmak

Node Js'ten önce Javascript kodlarının istemci tarafında çalıştığını biliyoruz. Bunu basit bir şekilde denersek;

Chrome üzerinde **F12** tuşuna tıklayarak console uygulamasını açalım. Konsol üzerinde basit olarak 3+5 toplama işlemini yapalım.



Şekil 8: Browser Üzerinde Basit Toplama İşlemi

Komut satırı uygulamasında **node** komutu ile node js açılır.

```
C:\Users\zuhal>node
Welcome to Node.js v16.15.0.
Type ".help" for more information.
```

Şekil 9: Komut Satırında Node Uygulamasını Açmak

Basit bir toplama işlemi yapılarak js komutları denenir.

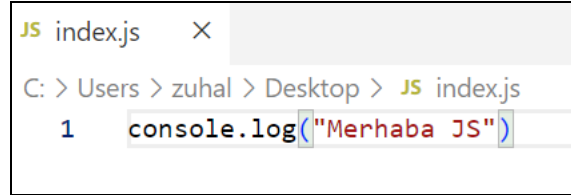
```
> 3+5
8
```

Şekil 10: Komut Satırında Basit Toplama İşlemi

Ekrana Metin Yazdırmak

Çalışma klasöründe **index.js** dosyası oluşturulur.

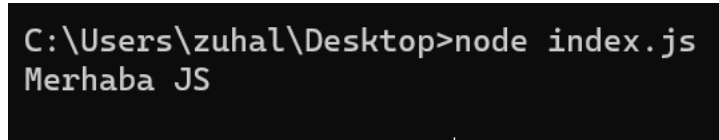
index.js dosyası içerisinde ekrana yazdırma komutu olan **console.log** yazdırılır.



```
JS index.js  X
C: > Users > zuhal > Desktop > JS index.js
1 console.log("Merhaba JS")
```

Şekil 11: index.js Dosyası

Komut satırı üzerinde **node index.js** komutu ile dosya çalıştırılır.



```
C:\Users\zuhal\Desktop>node index.js
Merhaba JS
```

Şekil 12: Komut satırında index.js dosyasını çalıştırmak

Oluşturulan Dosyayı Çağırarak

Çalışma klasörümüz içerisine utils.js dosyası oluşturalım. Oluşturulan dosyanın içeriğini aşağıdaki gibi dolduralım.



```
JS utils.js  X
C: > Users > zuhal > Desktop > JS utils.js
1 console.log("Utils Dosyası Çalışıyor...")
2
```

Şekil 13: utils.js dosyası içeriği

Daha sonra index.js içerisinde utils.js dosyamızı import ederek çağıralım.

```
JS index.js  X
C: > Users > zuhal > Desktop > JS index.js > ...
1 console.log("Merhaba JS")
2 const utils = require("./utils.js")
```

Şekil 14: index.js dosyası

index.js dosyamızı çalıştırdığımızda hem utils hem de index dosyaları çalıştırılır.

```
C:\Users\zuhal\Desktop>node index.js
Merhaba JS
Utils Dosyası Çalışıyor...
```

Şekil 15: index.js Çıktısı

Node JS Dosyaları İçerisinde Değişken Çağırma

Utils dosyasında oluşturduğumuz bir değişkeni index.js dosyası içerisinde çağırmak için **exports** terimini kullanırız. Utils.js dosyası içerisinde **const name** ile değişken tanımlarız. Bu dosyanın çağrıldığı dosyalarda **name** değişkeninin kullanılabilmesi için module.exports ile aşağıdaki gibi çağırılması gerekir.

Oluşturulan utils.js dosyası aşağıdaki gibidir.

```
JS utils.js  ●
C: > Users > zuhal > Desktop > JS utils.js > ...
10
11 const name = 'Ali'
12 module.exports = name
```

Şekil 16: Exports Komutu

index.js dosyasının içeriği aşağıdaki gibidir.

```
const name = require('./utils.js')
console.log(name)
```

Şekil 17: Index.js Dosyası Require Komutu

index.js dosyasının çıktısı aşağıdaki gibidir:

```
C:\Users\zuhal\Desktop>node index2.js  
Ali
```

Şekil 18: Index dosyası çıktısı

Node JS Dosyaları İçerisinde Fonksiyon Çağırma

Dosyalar arasında fonksiyonları çağırmak için yine **exports** terimi kullanılır.

Utils.js dosyası içerisinde oluşturulan add fonksiyonu aşağıdaki gibidir.

```
const add = function (a,b) {return a+b}  
module.exports = add
```

Şekil 19: Add fonksiyonu exports işlemi

```
const add = require('./utils.js')  
const sonuc = add(4,5)  
console.log(sonuc)
```

Şekil 20: index.js dosyası içerisinde add fonksiyonu çağırma

index.js dosyası çıktısı aşağıdaki gibidir.

```
C:\Users\zuhal\Desktop>node index2.js  
9
```

Şekil 21:index.js çıktısı

Node JS ile Dosya İşlemleri

Node.js'de, FS modülü kullanılarak dosya işlemleri gerçekleştirilebilir. **writeFileSync** yöntemiyle bir dosya oluşturulabilir ve içine belirtilen metinler yazılabilir.

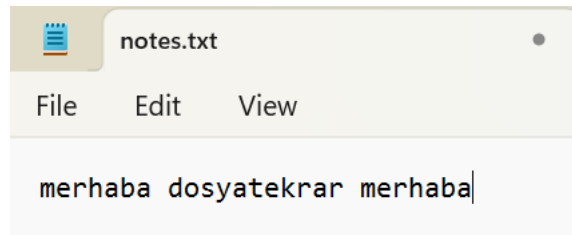
appendFileSync komutu ile açılan dosya içerisine girdiler eklenir.

index.js dosyası oluşturularak içeriği aşağıdaki gibi doldurulur.

```
const fs = require('fs')
fs.writeFileSync('notes.txt', 'merhaba dosya')
fs.appendFileSync('notes.txt', 'tekrar merhaba')
```

Şekil 22: Dosya İşlemleri

index.js dosyası çalıştırıldıktan sonra notes.txt dosyası oluşur.



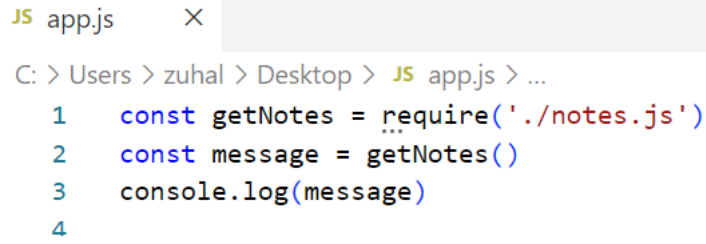
Şekil 23: Txt içeriği

Daha sonra notes.js dosyası oluşturarak içine bir fonksiyon tanımlayalım ve bu fonksiyon aracılığıyla ekrana bir mesaj yazdıralım.

```
JS notes.js  X
C: > Users > zuhal > Desktop > JS notes.js > ...
1  const getNotes = function () {return "this's a message"}
2
3  module.exports = getNotes
4
5
```

Şekil 24: notes.js dosyası

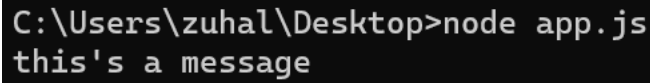
GetNotes fonksiyonunu index.js dosyası içerisinde aşağıdaki gibi çağırırız.



```
JS app.js X
C: > Users > zuhal > Desktop > JS app.js > ...
1  const getNotes = require('./notes.js')
2  const message = getNotes()
3  console.log(message)
4
```

Şekil 25: app.js dosyası

App.js dosyası çıktısı aşağıdaki gibidir:



```
C:\Users\zuhal\Desktop>node app.js
this's a message
```

Şekil 26: app.js çıktısı

NPM Nedir?

npm (Node Package Manager), JavaScript ve Node.js geliştiricilerinin kütüphane ve paketlerin yönetimini kolaylaştıran bir paket yöneticisi ve dağıtım platformudur.

npm, Node.js'nin resmi paket yöneticisidir ve geniş bir paket deposuna erişim sağlar.

npm, geliştiricilere projelerinde kullanacakları binlerce önceden oluşturulmuş paketi arama, indirme, yükleme, güncelleme ve paylaşma imkanı sunar.

Bu paketler, genellikle JavaScript kütüphaneleri, framework'leri, araçları ve diğer kod bileşenlerini içerir.

Node.JS kurulduktan sonra npm hazır olarak gelir.

Npm Validator Modülü

Validator modülü, kullanıcı girdilerinin doğruluğunu kontrol etmek için kullanılan kütüphanedir. Bu modül, kullanıcıların sağladığı verilerin belirli kriterlere uygun olup olmadığını kontrol etmek için çeşitli fonksiyonlar sağlar. Örneğin, bir kullanıcının bir formda girdiği e-posta adresinin geçerli olup olmadığını kontrol etmek, bir url adresinin belirli şartları karşılayıp karşılamadığını kontrol etmek için kullanılır.

İlk olarak Npm ile validator modülünü indirmek için ***npm i validator*** komutu yazılır.

```
C:\Users\zuhal\Desktop>npm i validator

added 1 package, and audited 16 packages in 2s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Şekil 27: Validator Modülü Yükleme

Yüklenen modül **require** komutu ile import edilir. **isEmail** ve **isURL** fonksiyonları ile çeşitli kontroller sağlanır.

```
const validator = require('validator')
console.log(validator.isEmail('0000000000@btu.edu.tr'))
console.log(validator.isURL('www.btu.edu.tr'))
```

Şekil 28: Validator modülü

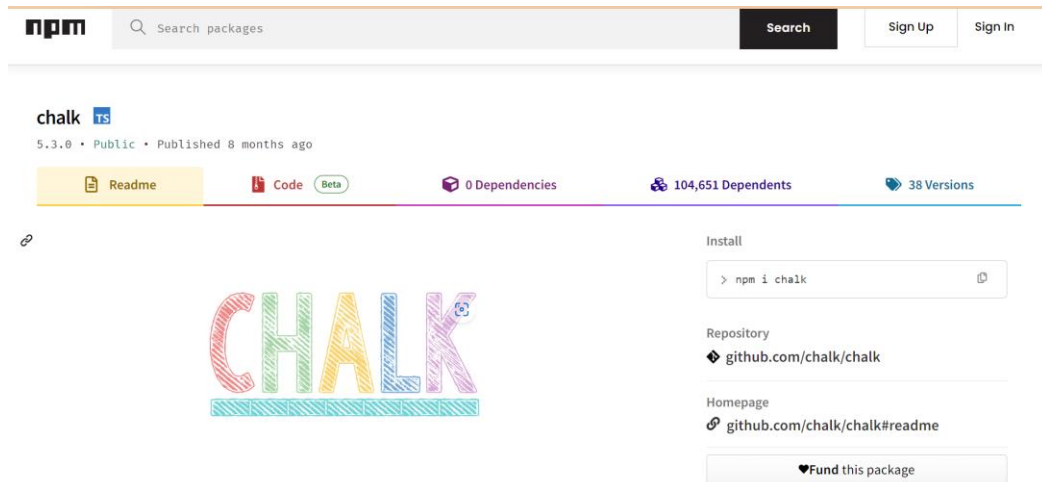
app.js dosyasını çalıştırdıktan sonra çıktı aşağıdaki gibi olur.

```
C:\Users\zuhal\Desktop>node app.js
true
true
```

Şekil 29: app.js Çıktısı

Node Js Paket Kurma / CHALK

CHALK; Node.js ortamında kullanılan bir konsol renklendirme kütüphanesidir. Bu modül, terminalde metin çıktılarını renklendirmek için kullanılır. Konsol çıktılarını renklendirmek, hata mesajlarını vurgulamak veya belirli bilgileri görsel olarak ayırt etmek gibi durumlarda faydalıdır.



Şekil 30: Chalk Npm Anasayfası

Npm ile kurmak için; ***npm i chalk*** komutu kullanılır.

```
C:\Users\zuhal>npm i chalk

added 1 package, and audited 10 packages in 1s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Şekil 31: Chalk Kurulum

```
C:\Users\zuhal>npm uninstall chalk

removed 1 package, and audited 9 packages in 752ms

found 0 vulnerabilities
```

Şekil 32: Uninstall Chalk


```
C:\Users\zuhal>npm i chalk@4.0

added 6 packages, and audited 15 packages in 1s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Şekil 33: Chalk 4.0 Kurma

```
C:\Users\zuhal\Desktop>node index.js
Merhaba JS
```

Şekil 34: Chalk Blue Kullanımı

```
const chalk = require('chalk')
console.log(chalk.green.bold('Success'))
```

```
C:\Users\zuhal\Desktop>node app.js
Success
```

Npm Initialize İşlemi

Npm initialize işlemi; bir Node.js projesi oluştururken veya mevcut bir proje için bir package.json dosyası oluştururken kullanılan bir komuttur. Bu komut, bir projenin temel bilgilerini (proje adı, sürüm, açıklama, geliştirici bilgileri vb.) girmenizi ve bu bilgilere dayalı olarak bir package.json dosyası oluşturmanızı sağlar.

```
C:\Users\zuhal\Desktop>npm -init
```

Şekil 35:npm init

Komutu çalıştırdıktan sonra, npm sizden projenizin hakkında çeşitli bilgiler girmenizi isteyecektir. Örneğin, proje adı, sürüm numarası, açıklama, ana dosya adı, geliştirici bilgileri vb. gibi bilgileri girmeniz istenecektir. Bu bilgileri girdikten sonra, npm package.json dosyasını oluşturacak ve girdiğiniz bilgileri içerecektir.

```
{
  "name": "notes-app",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  ▶ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "chalk": "4.0",
    "validator": "^13.11.0"
  }
}
```

Şekil 36: package.json dosyası

