Modul 2

Node.js Module System dan Command Line Arguments

TUJUAN PEMBELAJARAN

1. Mampu memahami dan menjelaskan tentang sistem modul Node.js diantaranya mengimport modul, file dan menginstal package melalui npm (node package manager)
2. Mampu memahami dan menjelaskan tentang command line arguments pada Node.js

Hardware & Software

1. PC (*Personal Computer)* dengan akses Internet
2. Visual Studio Code
3. Node.js
4. NPM (Node Package Manager)

URAIAN MATERI

A. Node.js Module System

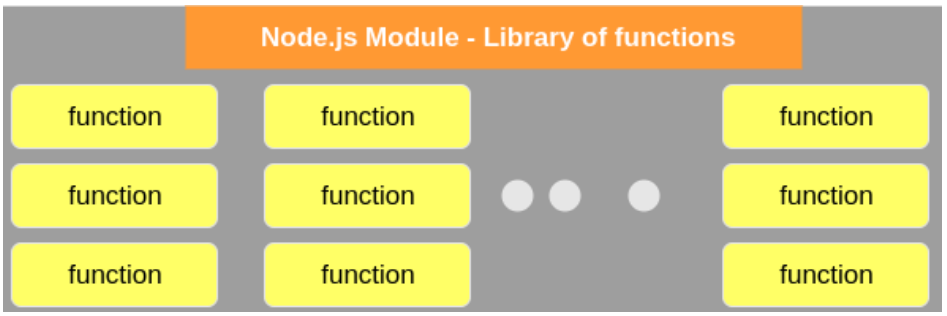
Sistem modul dalam Node.js adalah kerangka kerja yang memungkinkan pengembang perangkat lunak untuk mengorganisasi kode mereka menjadi unit yang lebih kecil dan dapat digunakan kembali yang disebut "modul." Ini memiliki beberapa konsep kunci:

1. **CommonJS**: Node.js mengikuti spesifikasi CommonJS, yang mendefinisikan cara modul didefinisikan, diimpor, dan digunakan dalam lingkungan Node.js. Spesifikasi ini bertujuan untuk memfasilitasi modularitas dalam pengembangan JavaScript di sisi server.
2. **Modul**: Modul adalah unit kode yang independen yang memungkinkan pengembang untuk memisahkan fungsionalitas dalam aplikasi mereka. Dalam

Node.js, setiap file JavaScript dianggap sebagai modul yang dapat digunakan dan diimpor dalam modul lain.

3. **Fungsi `require()`**: `require()` adalah fungsi yang digunakan untuk mengimpor modul lain ke dalam kode Node.js. Ini memungkinkan pengembang untuk mengakses dan menggunakan fungsionalitas yang ada dalam modul yang diimpor.
4. **`module.exports`**: `module.exports` adalah objek yang digunakan untuk mengekspor kode atau data dari suatu modul agar dapat diakses oleh modul lain. Ini memfasilitasi penggunaan kembali kode dan berbagi fungsionalitas antara modul.
5. **Modul Bawaan (Built-in Modules)**: Node.js juga menyertakan sejumlah modul bawaan yang disertakan secara default, seperti `**fs**` untuk operasi berkas, `**http**` untuk membuat server web, dan lainnya, yang dapat diimpor dan digunakan dalam aplikasi Node.js.

Secara ringkas, module pada Node.js dapat diartikan sebagai sebuah perpustakaan dari berbagai fungsi. Berikut adalah gambaran umum dari module pada Node.js



Gambar 1. Illustrasi Node.js Module System

B. NPM (Node Package Manager)

Node Package Manager (NPM) adalah alat pengelola paket yang kuat dan penting dalam ekosistem Node.js. NPM digunakan untuk mengelola, menginstal, dan mendistribusikan paket perangkat lunak dalam proyek-proyek Node.js. Ini memungkinkan pengembang untuk mengakses ribuan paket perangkat lunak open-source yang tersedia melalui repositori NPM untuk mempercepat pengembangan perangkat lunak. Beberapa konsep penting terkait NPM adalah

1. **Paket (Package)** : Paket adalah unit perangkat lunak yang terdiri dari kode JavaScript, berkas konfigurasi, dan informasi lainnya yang diperlukan untuk menggabungkan fungsionalitas tertentu. Setiap proyek Node.js biasanya memiliki file `package.json` yang mencatat semua paket yang diperlukan.
2. **`package.json`**: File `package.json` adalah berkas konfigurasi yang digunakan untuk mengelola proyek Node.js. Ini mencatat semua dependensi proyek, skrip yang dapat dijalankan, dan informasi lainnya tentang proyek. Ini juga

memungkinkan orang lain untuk dengan mudah menginstal semua dependensi proyek.

3. **Instalasi Paket** : Dengan NPM, pengembang dapat menginstal paket-paket yang diperlukan dalam proyek mereka dengan perintah `npm install`. NPM akan mengunduh paket-paket tersebut dari repositori NPM dan memasangnya dalam direktori proyek.
4. **Publikasi Paket (Package Publishing)** : Pengembang juga dapat menerbitkan paket mereka sendiri ke repositori NPM agar dapat digunakan oleh pengembang lain. Ini memungkinkan kolaborasi dan berbagi kode dalam komunitas Node.js.

C. COMMAND LINE ARGUMENTS

Dalam konteks Node.js, "command line arguments" adalah argumen atau parameter yang dapat disediakan oleh pengguna saat menjalankan sebuah skrip Node.js melalui baris perintah atau terminal. Argumen-argumen ini memungkinkan pengguna untuk memberikan input tambahan atau konfigurasi khusus kepada program Node.js yang sedang dijalankan. Node.js menyediakan cara untuk mengakses command line arguments melalui objek **process.argv.**

process.argv adalah array yang berisi argumen-argumen yang diberikan saat menjalankan skrip Node.js. Array ini mencakup informasi tentang cara skrip Node.js itu sendiri dijalankan, seperti path ke executable Node.js dan path ke skrip yang sedang dijalankan, diikuti oleh argumen-argumen yang diberikan oleh pengguna.

Argumen-argumen ini dapat digunakan untuk mengkonfigurasi atau mengoperasikan program Node.js sesuai dengan kebutuhan aplikasi. Pengguna dapat memasukkan argumen untuk mengatur parameter aplikasi, seperti port server, mode debug, nama file input/output, dan banyak lagi.

Dalam bahasa pemrograman dikenal juga istilah penguraian argument *(argument parsing)*. Peran argument parsing pada Node.js sering digunakan dalam proses *server-side scripting* dan ketika proses mengambil dan menguraikan argumen atau parameter yang diberikan kepada sebuah program JavaScript melalui baris perintah atau metode lainnya.

LATIHAN

A. NODE.JS MODULE SYSTEM

a. Import Core Module Node.js

1. Buatlah sebuah folder baru dengan nama **buku-catatan** dan dalam folder tersebut buatlah file javascript baru dengan nama **app.js** dan ketikan lah kode berikut ini

```
const fs = require('fs')
```

```
fs.writeFileSync('catatan.txt', 'Nama Saya Randi Proska')
//fs.appendFileSync('catatan.txt', ' Saya tinggal di
Padang')
```

2. Jalankan source code tersebut melalui terminal pada visual studio code dan perhatikan sebuah file baru bernama **catatan.txt** telah dibuat.
3. Jadikan baris **fs.writeFileSync** sebagai komentar dan kemudian uncomment baris **fs.appendFileSync.**
4. Jalankan Kembali source code dan lihat apa yang terjadi dengan **catatan.txt**

b. Import File Pada Node.js

1. Buatlah file baru pada folder buku-catatan dengan nama **catatan.js** dan ketikan kode berikut ini

```
const ambilCatatan = function () {
    return 'Ini Catatan Randi Proska...'
}
module.exports = ambilCatatan
```

2. Lalu ketikan kode berikut pada file app.js

```
const catatan = require('./catatan.js')
const pesan = catatan()
console.log(pesan)
```

3. Jalankan program melalui terminal dan perhatikan apa yang ditampilkan

c. Import npm (Node Package Manager) Pada Node.js

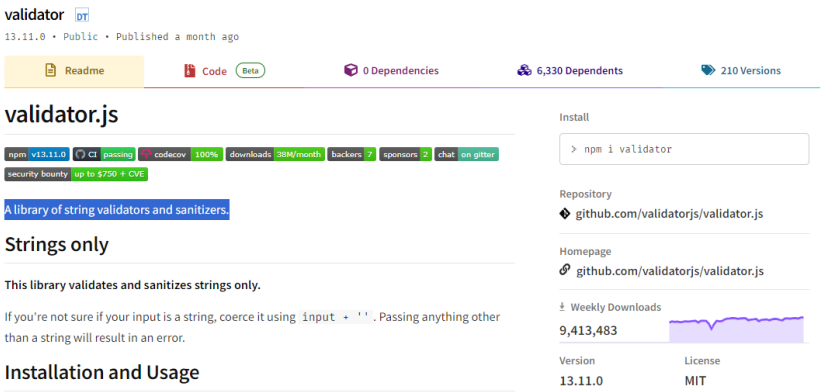
1. Silakan buka website https://www.npmjs.com/ untuk menemukan package yang akan digunakan
2. Buka terminal pada visual studio code dan pastikan anda berada pada direktori **buku-catatan.** Lalu ketikanlah **npm init** dan tekan **enter** hingga muncul seperti pada gambar dibawah ini

```
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

3. Setelah selesai akan muncul file baru pada direktori **buku-catatan** anda dengan nama **package.json**

4. Carilah package **validator** pada website npm. Validator adalah library untuk validasi string dan anda dapat menemukan informasi lengkapnya pada website npm, termasuk bagaimana cara menginstall dan menggunakannya



5. Lakukan instalasi package tersebut melalui terminal dengan perintah **`npm i validator@13.11.0.`** Penambahan **@13.11.0** pada akhir perintah instalasi bertujuan untuk menginstall package sesuai dengan versi yang diinginkan
6. Perhatikan bahwa sebuah folder baru **node-modules** telah muncul pada direktori anda. Selain itu, juga ada file baru dengan nama **package-lock.json** yang berisi informasi terkait package yang diinstal
7. Ketikanlah kode berikut pada file app.js dan jalankan program melalui terminal

```
const validator = require('validator')
const ambilCatatan = require('./catatan.js')
const pesan = ambilCatatan()
console.log(pesan)
console.log(validator.isURL('https://proska.com'))
```

d. Latihan 1

1. Lakukan instalasi package **chalk** versi 4.1.2 pada direktori aplikasi **buku-catatan**
2. Buatlah perintah untuk mencetak teks berikut menggunakan chalk **'print warna biru sukses'**. Perhatikan dokumentasi chalk pada website npm https://www.npmjs.com/package/chalk/v/4.1.2 untuk mengetahui bagaimana penggunaan nya.
3. **Penting**! Pastikan anda menambahkan variable **`const chalk`** pada awal program
4. Cobalah untuk mengganti warna teks biru dengan warna dan variasi lainnya

e. Latihan 2

1. Lakukan instalasi package **nodemon** versi terbaru dengan perintah **`npm install nodemon@[versi package terbaru] -g`**
2. Cek apakah nodemon berhasil terinstall dengan perintah **`nodemon -v`** pada terminal

3. Jika tidak muncul informasi versi nodemon, maka jalankan Windows PowerShell dan ketikan perintah **`Set-ExecutionPolicy Unrestricted,`** lalu coba jalankan lagi perintah **`nodemon -v`**
4. Apabila sudah terinstall dengan rapi, jalankan aplikasi anda menggunakan perintah nodemon, bukan lagi node. Contoh: **`nodemon app.js`**
5. Ubahlah salah satu coding program anda dan perhatikan bahwa program anda yang berjalan akan otomatis berubah tampilannya secara realtime

B. COMMAND LINE ARGUMENTS

a. Mendapatkan Input Dari Pengguna

1. Ketikanlah kode berikut pada file app.js anda

```
const ambilCatatan = require('./catatan.js')

const command = process.argv[2]
console.log(process.argv)

// if (command === 'tambah') {
//     console.log('Tambah Catatan')
// } else if (command === 'hapus') {
//     console.log('Hapus Catatan')
// }
```

2. Jalankan pada terminal dengan perintah node app.js [tambahkan *argument* anda dibelakang]. Misal: **`node app.js Randi Proska Sandra`**
3. Lalu, cobalah mengganti nilai **array[]** pada baris pertama, misal dengan angka 5. Anda juga bisa memasukan nilai array[] pada baris kedua dengan memberikan perintah seperti berikut

```
console.log(process.argv[2])
```

4. **Perhatikan dan pahami apa yang ditampilkan**
5. **Uncomment** source code yang ada pada komentar, lalau jalankan melalui terminal dengan perintah **`node app.js tambah`** atau **`node app.js hapus`**
6. **Perhatikan dan pahami apa yang ditampilkan**

b. Argument Parsing (Penguraian Argumen)

1. Lakukan instalasi package **yargs** pada direktori aplikasi **buku-catatan.** Yargs adalah satu package pada javascript yang berguna untuk *argument parsing.*
2. Masukanlah kode berikut pada file **app.js**

```
const yargs = require('yargs')
const catatan = require('./catatan.js')
// Kustomisasi versi yargs
yargs.version('10.1.0')

// Membuat perintah (command) 'tambah'
yargs.command({
    command: 'tambah',
    describe: 'tambah sebuah catatan baru',
    handler: function () {
        console.log('Sebuah catatan baru ditambahkan!')
    }
})

// Perintah hapus
yargs.command({
    command: 'hapus',
    describe: 'hapus catatan',
    handler: function () {
        console.log('Catatan berhasil dihapus')
    }
})
// Instruksi no.4 letakan disini

// letakan bagian ini pada baris terakhir
console.log(yargs.argv)
```

3. Jalankan program diatas melalui terminal pada visual studio code dengan perintah **`node app.js tambah`** atau **`node app.js hapus.`** Perhatikan dan pahami apa yang ditampilkan
4. Jika ini sebuah program yang lengkap dan memiliki banyak jumlah catatan didalam aplikasinya, maka diperlukan perintah untuk membaca catatan dan perintah untuk menampilkan semua catatan. **Lanjutkan** program diatas dengan menambahkan perintah (command) untuk menampilkan **list** (daftar) semua catatan serta perintah untuk membaca (**read**) sebuah catatan
5. Setelah anda menambahkan command pada nomor 4, ubahlah coding pada command tambah dengan source code dibawah ini

```
yargs.command({
    command: 'tambah',
    describe: 'tambah sebuah catatan baru',
    builder: {
        judul: {
            describe: 'Judul catatan',
            demandOption: true,
```

```
            type: 'string'
        },
        isi: {
            describe: 'Isi catatan',
            demandOption: true,
            type: 'string'
        }
    },
    handler: function (argv) {
        console.log('Judul: ' + argv.judul)
        console.log('Isi: ' + argv.isi)
    }
})
```

6. Lalu ubahlah baris terakhir pada program dengan code

```
yargs.parse()
```

7. Jalankan file app.js melalui terminal dengan perintah **node app tambah --judul="Catatan Randi", --isi="Ini adalah catatan pertama Randi"**
8. Perhatikan dan pahami apa yang ditampilkan

c. Menyimpan Data dengan JSON

1. Buatlah folder baru dengan nama **testsite**, lalu buatlah file baru dalam folder tersebut dengan nama **1-jsontest.js**
2. Ketikanlah source code berikut pada file tersebut

```
const fs = require('fs')

const buku = {
    judul: 'Pemrograman Jaringan',
    penulis: 'Randi Proska Sandra'
}

const bookJSON = JSON.stringify(book)
fs.writeFileSync('1-jsontest.json', bookJSON)
```

3. Perhatikan dan pahami apa yang terjadi.

d. Membaca Data JSON

1. Jadikan coding diatas menjadi komentar, **kecuali baris pertama**, lalu masukanlah code berikut ini

```
const dataBuffer = fs.readFileSync('1-jsontest.json')
const dataJSON = dataBuffer.toString()
const data = JSON.parse(dataJSON)
console.log(data)
```

2. Jalankan program app.js, lalu perhatikan dan pahami apa yang ditampilkan!.

3. Ubahlah **`console.log(data)`** menjadi **`console.log(data.judul),`** jalankan dan perhatikan perbedaan tampilannya dengan Langkah sebelumnya.

e. Latihan menyempurnakan aplikasi buku-catatan

1. Menambahkan catatan baru

a. Perhatikan Langkah pada latihan **bagian b nomor 5.** Ubahlah kode pada bagian **handler** menjadi seperti berikut

```
handler: function (argv) {
        catatan.tambahCatatan(argv.judul, argv.isi)
    }
```

b. Ubahlah kode pada file **catatan.js** menjadi berikut ini

```
const fs = require('fs')

const ambilCatatan = function () {
    return 'Ini Catatan Randi Proska...'
}

const tambahCatatan = function (judul, isi) {
    const catatan = muatCatatan()
    const catatanGanda = catatan.filter(function (note) {
        return note.title === judul
    })

    if (catatanGanda.length === 0) {
        catatan.push({
            judul: judul,
            isi: isi
        })
        simpanCatatan(catatan)
        console.log('Catatan baru ditambahkan!')
    } else {
        console.log('Judul catatan telah dipakai')
    }
}

const simpanCatatan = function (catatan) {
    const dataJSON = JSON.stringify(catatan)
    fs.writeFileSync('catatan.json', dataJSON)
}

const muatCatatan = function () {
    try {
        const dataBuffer = fs.readFileSync('catatan.json')
        const dataJSON = dataBuffer.toString()
        return JSON.parse(dataJSON)
    } catch (e) {
        return []
```

```
    }
}

module.exports = {
    ambilCatatan: ambilCatatan,
    tambahCatatan: tambahCatatan
}
```

c. Jalankan program diatas melalui terminal dengan perintah **`node app.js tambah --judul="Catatan 1", --isi="Isi catatan 1"`** dan perhatikan bahwa file baru **catatan.json** telah dibuat. Bukalah file tersebut dan lihat isinya

2. Menghapus catatan

a. Bukalah coding pada file app.js anda dan perhatikan bahwa perintah (command) **hapus** anda tidak memiliki **builder** sebagaiman command **tambah**

b. Buatlah builder untuk command **hapus** anda seperti pada command **tambah** namun **cukup judul saja** dan **tanpa** bagian **isi.**

c. Untuk bagian **handler**, gunakanlah code berikut

```
catatan.hapusCatatan(argv.judul)
```

d. Tambahkanlah fungsi baru pada file **catatan.js** dan letakan dibawah fungsi **tambahCatatan**. Masukan kode berikut!

```
const hapusCatatan = function (judul) {
    const catatan = muatCatatan()
    const catatanUntukDisimpan = catatan.filter(function (note) {
        return note.judul !== judul
    })

    if (catatan.length > catatanUntukDisimpan.length) {
        console.log(chalk.green.inverse('Catatan dihapus!'))
        simpanCatatan(catatanUntukDisimpan)
    } else {
        console.log(chalk.red.inverse('Catatan tidak
ditemukan!'))
    }
}
```

e. Kode diatas menggunakan perintah dari chalk pada bagian console.log, sehingga anda perlu menambahkan variabel chalk pada awal program di file catatan.js

f. Selain itu, anda juga perlu memperbarui module.exports dengan memanggil fungsi untuk menghapus catatan

g. Jalankan program diatas melalui terminal dengan perintah **`node app.js hapus --judul="Catatan 1”`**

3. Lanjutkanlah program diatas untuk fungsi menampilkan semua catatan (list) dan membaca catatan (read)

REFERENCES

1. Casciaro. (2014, December). Node.js Design Patterns (3rd ed.). Packt Publishing.
2. Pasquali, S., & Faaborg, K. (2017, December 29). Mastering Node.js: Build Robust and Scalable Real-Time Server-Side Web Applications Efficiently.
3. Guides | Node.js. (n.d.). Node.js. https://nodejs.org/en/docs/guides
4. *npm Docs*. (n.d.). Npm Docs. https://docs.npmjs.com/
5. Leka, M. (2022, June 8). *Exploring the JavaScript Ecosystem: Popular Tools, Frameworks, and Libraries*. Medium. https://mirzaleka.medium.com/exploring-javascript-ecosystem-popular-tools-frameworks-libraries-7901703ec88f