Resume Materi NodeJS/ 20230140066/Halim Bagus Al Hakim

NodeJS adalah runtime environment berbasis JavaScript yang berjalan di sisi server. Berbeda dengan JavaScript yang biasanya dijalankan di browser, NodeJS memungkinkan developer untuk menggunakan JavaScript untuk membangun aplikasi backend.

Beberapa konsep dasar NodeJS untuk pemula:
- Non-blocking & Asynchronous: NodeJS menggunakan event-driven architecture sehingga efisien untuk menangani banyak request.
- Package Manager (NPM): Digunakan untuk mengelola library eksternal.
- Module System: Memungkinkan kode dipisahkan ke dalam file/module.
- HTTP Module: Modul bawaan NodeJS untuk membuat server HTTP sederhana.

Contoh membuat server HTTP dasar:

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.write('Hello World');
  res.end();
});
server.listen(3000, () => console.log('Server running...'));
```

- Sejarah Singkat: NodeJS dibuat oleh Ryan Dahl pada tahun 2009 dengan tujuan agar JavaScript bisa berjalan di sisi server.
- Event Loop: Mekanisme utama NodeJS yang memungkinkan menjalankan banyak operasi tanpa blocking.
    - Callback → cara lama untuk menangani asynchronous.
    - Promise & Async/Await → cara modern agar kode lebih mudah dibaca.
- NPM (Node Package Manager):
    - Perintah dasar:
        - npm init -y → inisialisasi project.
        - npm install express → menginstal library.
        - npm run start → menjalankan script.

 Database Layer

- Menggunakan **MySQL** sebagai sistem manajemen basis data.
- Konfigurasi koneksi dilakukan pada file config/db.js.
- Berfungsi sebagai penghubung antara aplikasi dan penyimpanan data.

 Model (User Model)

- Merepresentasikan tabel users di dalam database.
- Menyediakan fungsi-fungsi CRUD (Create, Read, Update, Delete).
- Contoh fungsi: getAll(), getById(), create(), update(), delete().

 Controller (UserController)

- Menangani logika bisnis utama aplikasi.
- Mengatur autentikasi dengan JWT.
- Contoh fungsi: register, login, getUsers, updateUser.

 Routes (userRoutes.js)

- Mendefinisikan endpoint API yang akan diakses client.
- Contoh endpoint:
    - POST /register → Pendaftaran user baru.
    - POST /login → Autentikasi user.
    - GET /users → Mengambil data semua user.
    - GET /users/:id → Mengambil data user berdasarkan ID.

 Middleware (auth.js)

- Berfungsi memvalidasi **JSON Web Token (JWT)**.
- Memastikan hanya request yang memiliki token valid yang bisa mengakses endpoint tertentu.
- Memberikan respon 401 Unauthorized jika token tidak valid atau tidak ada.

 App Entry Point (app.js)

- Menjadi titik masuk utama aplikasi.

- Menghubungkan semua route, middleware, dan konfigurasi server.
- Biasanya server dijalankan dengan perintah node app.js atau npm start.

 Manfaat Arsitektur MVC

- Memisahkan logika bisnis, query database, dan pengaturan routing.
- Membuat aplikasi lebih mudah dikembangkan dan dipelihara.
- Memastikan keamanan lebih baik melalui middleware autentikasi JWT

Unit testing adalah pengujian bagian terkecil kode untuk memastikan fungsionalitas berjalan dengan benar. Jest merupakan framework testing yang populer untuk NodeJS.

 Pengertian

- Unit testing adalah pengujian terhadap unit terkecil kode program (fungsi/metode).
- Tujuannya memastikan setiap unit kode bekerja sesuai fungsinya dan mencegah bug sejak dini.

 Mengapa Menggunakan Jest?

- Mudah dipelajari dengan sintaks sederhana.
- Cepat dieksekusi karena dioptimalkan untuk NodeJS.
- Mendukung integrasi dengan framework populer seperti React, Vue, dan NextJS.
- Dilengkapi dengan fitur bawaan seperti mocking, snapshot test, dan code coverage tanpa harus menambah library tambahan.

 Fitur Utama Jest

1. **Matcher**
    - toBe() → untuk membandingkan nilai primitif.
    - toEqual() → untuk membandingkan objek/array secara mendalam.
    - toContain() → untuk memeriksa elemen dalam array.
    - toThrow() → untuk menguji apakah sebuah fungsi menghasilkan error.
2. **Async Testing**

- Mendukung Promise dan async/await.
- Menyediakan matcher khusus seperti .resolves dan .rejects untuk memudahkan pengujian asynchronous.

3. **Mocking**
    - Membuat tiruan fungsi agar tidak bergantung pada database atau API eksternal.
    - Bisa berupa mock function (jest.fn()), mock class, atau mock module.
    - Sangat berguna untuk menguji logika bisnis tanpa mengganggu sistem lain.
4. **Setup Function**
    - beforeEach / afterEach → dijalankan sebelum/sesudah setiap pengujian.
    - beforeAll / afterAll → dijalankan sekali sebelum/sesudah semua pengujian dalam satu file.
    - Memudahkan persiapan kondisi awal dan pembersihan data setelah test.
5. **Code Coverage**
    - Menunjukkan seberapa banyak kode sudah diuji.
    - Memberikan laporan detail meliputi *statements*, *functions*, *branches*, dan *lines*.
    - Dapat diatur threshold minimal, misalnya 80%, untuk menjaga kualitas kode.

 Kelebihan Penggunaan Unit Test dengan Jest

- Membantu menemukan bug lebih cepat.
- Mempermudah refactoring kode tanpa khawatir merusak fungsi lain.
- Menjadi bagian penting dari praktik *Continuous Integration* (CI/CD).
- Meningkatkan kepercayaan diri developer saat aplikasi berkembang semakin kompleks.

4. NodeJS Todolist API

Class Diagram

- **TodolistService** → bertanggung jawab menyimpan dan mengelola daftar todo.
    - Method utama: getTodoList(), createTodo(), updateTodo(), deleteTodo().
- **app.mjs** → file utama server yang menghubungkan request dari client ke TodolistService.

Implementasi Teknis

- Menggunakan http.createServer() dari modul bawaan NodeJS.
- Server menangani request berdasarkan metode HTTP:
    - **GET** → Mengambil daftar todo.
    - **POST** → Menambah todo baru.
    - **PUT** → Memperbarui todo berdasarkan ID.
    - **DELETE** → Menghapus todo berdasarkan ID.
- Setiap response dikirim dengan Content-Type: application/json.

.