


Nama : Salman Alfarisi

Kelas : S1SE-07-01

NIM : 231104036

1.



```
{  
  "nama": "Salman Alfarisi",  
  "nim": "2311104036",  
  "kelas": "S1SE-01-07",  
  "angkatan": 2023,  
  "hobi": ["Mendaki", "Gambar", "Badminton"]  
}
```

```

// Import module filesystem
const fs = require('fs');

class SalmanAlfarisi2311104036 {
  constructor(nama, nim, kelas, angkatan, hobi) {
    this.nama = nama;
    this.nim = nim;
    this.kelas = kelas;
    this.angkatan = angkatan;
    this.hobi = hobi;
  }

  static ReadJSON(filePath) {
    try {
      const data = fs.readFileSync(filePath, 'utf-8');
      const obj = JSON.parse(data);

      const mahasiswa = new SalmanAlfarisi2311104036(
        obj.nama,
        obj.nim,
        obj.kelas,
        obj.angkatan,
        obj.hobi
      );

      // Output hasil deserialisasi
      console.log("==== DATA MAHASISWA =====");
      console.log(`Nama      : ${mahasiswa.nama}`);
      console.log(`NIM       : ${mahasiswa.nim}`);
      console.log(`Kelas    : ${mahasiswa.kelas}`);
      console.log(`Angkatan  : ${mahasiswa.angkatan}`);
      console.log(`Hobi      : ${mahasiswa.hobi.join(', ')} `);
    } catch (error) {
      console.error("Gagal membaca atau parse file JSON:", error.message);
    }
  }
}

SalmanAlfarisi2311104036.ReadJSON('./jurnal7_1_2311104036.json');

```

Bagian pertama dari program bertanggung jawab untuk membaca data pribadi seorang mahasiswa dari file JSON bernama jurnal7_1_2311104036.json. Data yang dimuat meliputi nama, NIM, kelas, angkatan, dan hobi dalam bentuk array. Melalui class SalmanAlfarisi2311104036, program memanfaatkan konsep konstruktor untuk menginisialisasi properti data mahasiswa, lalu menampilkan isi data tersebut dengan rapi di terminal.

Metode ReadJSON() menggunakan modul fs bawaan Node.js untuk membaca file secara sinkron, kemudian melakukan parsing string JSON menjadi objek JavaScript. Setelah objek terbentuk, data ditampilkan satu per satu dengan format yang mudah dibaca. Program juga menangani kemungkinan kesalahan seperti file tidak ditemukan atau data tidak valid dengan menggunakan blok try...catch. Bagian ini menunjukkan pemahaman Anda dalam menangani struktur JSON sederhana dan menerapkannya menggunakan prinsip OOP (Object-Oriented Programming).

2.

```
{
  "members": [
    {
      "nim": "2311104036",
      "firstname": "Salman",
      "lastname": "Alfarisi",
      "age": 20,
      "gender": "Male"
    },
    {
      "nim": "2311104037",
      "firstname": "Dina",
      "lastname": "Kurnia",
      "age": 21,
      "gender": "Female"
    },
    {
      "nim": "2311104038",
      "firstname": "Raka",
      "lastname": "Pratama",
      "age": 22,
      "gender": "Male"
    }
  ]
}
```

```

const fs = require('fs');

class tim2311104036 {
  constructor(members) {
    this.members = members;
  }

  static ReadJSON(filePath) {
    try {
      const jsonData = fs.readFileSync(filePath, 'utf-8');
      const obj = JSON.parse(jsonData);

      const team = new tim2311104036(obj.members);

      console.log("Team member list:");
      team.members.forEach(member => {
        console.log(`${member.nim} ${member.firstname} ${member.lastname} (${member.age} ${member.gender})`);
      });
    } catch (error) {
      console.error("Gagal membaca atau parse JSON:", error.message);
    }
  }
}

tim2311104036.ReadJSON('./jurnal7_2_2311104036.json');

```

Bagian kedua dari program digunakan untuk membaca data tim yang terdiri dari beberapa anggota, tersimpan dalam file jurnal7_2_2311104036.json. File JSON ini berisi array members, di mana setiap elemen adalah objek yang memuat informasi seperti NIM, nama depan, nama belakang, usia, dan gender. Class tim2311104036 menerima array anggota tim tersebut dan menyimpannya ke dalam properti members.

Metode ReadJSON() akan membaca file, menguraikan JSON, dan kemudian mencetak daftar semua anggota tim dengan format satu baris per anggota. Iterasi dilakukan menggunakan forEach(), yang merupakan cara elegan dalam JavaScript untuk memproses array. Bagian ini menunjukkan kemampuan dalam menangani data berformat array dan menampilkannya secara efisien, sekaligus memperkuat praktik penggunaan class dan metode statis untuk mengorganisasi kode.

3.

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

```

const fs = require('fs');

class Glossary2311104036 {
  constructor(glossEntry) {
    this.glossEntry = glossEntry;
  }

  static ReadJSON(filePath) {
    try {
      const rawData = fs.readFileSync(filePath, 'utf-8');
      const parsed = JSON.parse(rawData);

      const glossEntry = parsed.glossary?.GlossDiv?.GlossList?.GlossEntry;

      if (!glossEntry) {
        throw new Error("GlossEntry tidak ditemukan di file JSON");
      }

      const item = new Glossary2311104036(glossEntry);

      console.log("==== GLOSS ENTRY =====");
      console.log(`ID : ${item.glossEntry.ID}`);
      console.log(`GlossTerm : ${item.glossEntry.GlossTerm}`);
      console.log(`Acronym : ${item.glossEntry.Acronym}`);
      console.log(`Abbrev : ${item.glossEntry.Abbrev}`);
      console.log(`Definisi : ${item.glossEntry.GlossDef.para}`);
      console.log(`See Also : ${item.glossEntry.GlossDef.GlossSeeAlso.join(', ')}`);
      console.log(`GlossSee : ${item.glossEntry.GlossSee}`);
    } catch (err) {
      console.error("Gagal membaca GlossEntry:", err.message);
    }
  }
}

Glossary2311104036.ReadJSON('./jurnal7_3_2311104036.json');

```

Bagian ketiga merupakan bagian paling kompleks karena menangani struktur JSON yang bersifat bersarang (nested). File jurnal7_3_2311104036.json memuat data glossary (daftar istilah) yang memiliki hierarki objek di dalamnya, seperti GlossDiv, GlossList, hingga GlossEntry. Class Glossary2311104036 dirancang untuk mengakses langsung objek GlossEntry, yang merupakan bagian inti dari glossary tersebut.

Metode ReadJSON() menggunakan optional chaining (?.) agar dapat dengan aman mengakses properti-properti bersarang tanpa menyebabkan error jika ada yang tidak ditemukan. Setelah berhasil mengakses GlossEntry, program menampilkan berbagai informasi seperti ID, istilah, akronim, definisi paragraf, daftar istilah lain yang berkaitan, serta referensi lainnya. Bagian ini menunjukkan kemampuan Anda dalam membaca dan mengekstrak data dari struktur JSON bertingkat secara efisien, serta memahami pentingnya validasi data dalam konteks yang lebih kompleks.