Node.js

Apa Itu Node.js?

- Node.js adalah runtime environment JavaScript yang bersifat open-source dan berbasis JavaScript V8 engine dari Google.
- Dirancang untuk menjalankan kode JavaScript di sisi server, bukan hanya di browser.
- Menggabungkan kemampuan bahasa JavaScript dengan operasi I/O non-blocking yang efisien.

> Runtime environment: is a program that provides a software platform for applications to run.

Runtime vs Runtime Environment

Feature	Runtime	Runtime environment
What it is	The phase when a program is executing	The platform or system that allows a program to run
When it happens	After compilation	During runtime
Analogy	Cooking a dish	The kitchen and its tools
Example	JavaScript code running in the Chrome browser	Node.js script connecting to a database

Karakteristik Utama Node.js

- **Non-Bloking dan Event-Driven**: Operasi I/O non-blocking memungkinkan Node.js untuk menangani banyak koneksi secara efisien.
- **Server-Side**: Digunakan untuk pengembangan aplikasi server, seperti aplikasi web dan API.
- **Server Environment**: Didesain untuk menggantikan server tradisional, seperti Apache HTTP Server.

> Non-blocking: Node.js tidak akan menunggu operasi I/O selesai, melainkan akan melanjutkan ke operasi berikutnya.

> Event-driven: Node.js menggunakan event loop untuk menangani proses yang akan dieksekusi.

Karakteristik Utama Node.js

- **JavaScript Everywhere**: Memungkinkan penggunaan bahasa JavaScript di kedua sisi, server dan klien.
- **Modular dan Ekosistem Besar**: Memiliki ekosistem modul yang besar melalui npm (Node Package Manager).
- **Pemrograman Asinkron**: Cocok untuk aplikasi real-time dan berkinerja tinggi.

Contoh Penggunaan Node.js

- Aplikasi web server dengan framework seperti Express.js.
- Aplikasi real-time seperti chat dan game online.
- Aplikasi berbasis mikrokontroler dan IoT.
- Alat pengembangan (tools) seperti Vite dan Webpack.
- Aplikasi berbasis command line (CLI).

Mengapa Node.js?

- Performa Tinggi: Kemampuan operasi I/O non-blocking dan event-driven.
- Fleksibel: Cocok untuk berbagai jenis aplikasi, dari web hingga aplikasi real-time.
- Komunitas yang Kuat: Node.js memiliki komunitas pengembang yang besar yang terus berkembang.
- Ekosistem Modul: Dapat menggunakan ribuan modul pihak ketiga melalui `npm`.

Node Package Manager `npm`

- Package manager adalah tools yang digunakan untuk mengelola modul atau package yang digunakan dalam suatu proyek.
- `npm` menyediakan akses ke ribuan modul pihak ketiga yang dapat digunakan dalam proyek Node.js.
- `npm` juga dapat digunakan untuk mengelola proyek Node.js, seperti menginisialisasi proyek, menginstal modul, dan menjalankan skrip.
- `npm` telah disertakan dalam instalasi default Node.js.

Package Manager Alternatives

`npm` bukan satu-satunya package manager yang dapat digunakan dalam proyek Node.js. Ada package manager alternatif yang dapat digunakan, seperti:

- `yarn` (https://yarnpkg.com/)
- `pnpm` (https://pnpm.io/)

Node.js Project Initialization

- Inisialisasi proyek Node.js adalah langkah pertama dalam memulai pengembangan aplikasi Node.js.
- Gunakan perintah `npm init` untuk membuat berkas `package.json`.
- Berkas `package.json` berisi informasi proyek dan dependensi.

Node.js Modules

- Modul Node.js dapat berupa modul bawaan, modul custom, maupun modul pihak ketiga.
- Modul adalah berkas JavaScript yang berisi kode yang dapat digunakan kembali.
- Node.js memungkinkan Anda untuk memisahkan kode menjadi modul-modul terpisah.
- Ada 2 cara untuk mendefinisikan dan menggunakan modul Node.js:
    - Menggunakan sintaks commonJS.
    - Menggunakan sintaks ES6.

Node.js Modules

Sintaks CommonJS

- Modul Node.js diimpor menggunakan fungsi `require()`.
- Modul Node.js diekspor menggunakan properti `exports`.

Sintaks ES6

- Modul Node.js diimpor menggunakan sintaks `import`.
- Modul Node.js diekspor menggunakan sintaks `export`.

Contoh Modul Node.js

```
// Modul 'myModule.js'
module.exports = {
  greet: function (name) {
    return `Halo, ${name}!`
  },
}

// Menggunakan modul
// index.js
const myModule = require('./myModule.js')
console.log(myModule.greet('John'))
```

Node.js Core Module

- Node.js memiliki sejumlah modul inti yang telah disertakan dalam instalasi standar.
- Modul inti mencakup modul untuk mengelola sistem berkas, jaringan, HTTP, dll.
- Daftar lengkap modul inti dapat dilihat di https://nodejs.org/dist/latest-v20.x/docs/api/.

Contoh Core Module: `fs`

- Modul `fs` (File System) digunakan untuk melakukan operasi pada berkas dan direktori.
- Contoh: Membaca isi berkas dan menulis ke berkas.

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

// Membaca berkas
fs.readFile('data.txt', 'utf8', (err, data) => {
  if (err) throw err
  console.log(data)
})
```

Contoh Core Module: `fs`

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

// Menulis berkas
fs.writeFile('output.txt', 'Teks yang akan ditulis.', (err) => {
  if (err) throw err
  console.log('Berkas telah ditulis.')
})
```

Contoh Core Module: `fs`

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

// list berkas dan direktori
fs.readdir('.', (err, files) => {
  if (err) throw err
  console.log(files)
})
```

Contoh Core Module: `readline`

- Modul `readline` adalah salah satu modul inti Node.js yang digunakan untuk membaca input dari pengguna melalui terminal.
- Modul ini berguna dalam pengembangan aplikasi berbasis teks atau command-line.

Contoh Core Module: `readline`

```
const readline = require('readline')

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
})

rl.question('Siapa nama Anda? ', (nama) => {
  console.log(`Halo, ${nama}! Selamat datang.`)
  rl.close()
})
```

Contoh Core Module: `readline`

```
const readline = require('readline')

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
})

function showMenu() {
  console.log('Pilih menu:')
  console.log('1. Lihat profil')
  console.log('2. Ubah kata sandi')
  console.log('3. Keluar')
}

showMenu()

rl.question('Pilihan Anda: ', (pilihan) => {
  if (pilihan === '1') {
    console.log('Profil pengguna...')
  } else if (pilihan === '2') {
    console.log('Ubah kata sandi...')
  } else if (pilihan === '3') {
    console.log('Anda telah keluar.')
  } else {
    console.log('Pilihan tidak valid.')
  }
  rl.close()
})
```

Contoh Core Module: `process`

- Modul process memberikan akses ke informasi dan pengendalian proses Node.js.
- Contoh: Mengakses argumen baris perintah dan menghentikan proses.

```
// Mengakses argumen baris perintah
const args = process.argv
console.log('Argumen baris perintah:')
console.log(args)

// Menghentikan proses
process.exit(0)
```

Creating Your Own Module

```
// Modul 'myMath.js'
module.exports = {
  addNumbers: function (a, b) {
    return a + b
  },
  subtractNumbers: function (a, b) {
    return a - b
  },
}

// Menggunakan modul
const myMath = require('./myMath.js')
console.log(myMath.addNumbers(5, 3))
console.log(myMath.subtractNumbers(10, 4))
```

Creating Your Own Modules

```
// Modul 'formatter.js'
module.exports = {
  moneyFormat: function (amount) {
    return `Rp ${amount.toFixed(2)}`
  },
  dateFormat: function (date) {
    return date.toDateString()
  },
}

// Menggunakan modul
const formatter = require('./formatter.js')
console.log(formatter.moneyFormat(1000.5))
console.log(formatter.dateFormat(new Date()))
```

Creating Your Own Module

```
// Modul 'formatter.js'
export function moneyFormat(amount) {
  return `Rp ${amount.toFixed(2)}`
}

export function dateFormat(date) {
  return date.toDateString()
}

// Menggunakan modul
import * as formatter from './formatter.js'
console.log(formatter.moneyFormat(1000.5))
console.log(formatter.dateFormat(new Date()))
```

Penggunaan Modul Pihak Ketiga

- Dalam pengembangan Node.js, Anda seringkali akan menggunakan modul pihak ketiga yang dapat mempermudah pekerjaan Anda.
- Sebagai contoh, kita akan melihat penggunaan modul `date-fns` untuk pengelolaan tanggal dan waktu.
- Anda dapat menginstalnya melalui npm dengan perintah: `npm install date-fns`.

Contoh Penggunaan `date-fns`

```
const { format, addDays } = require('date-fns')

const today = new Date()
const tomorrow = addDays(today, 1)

const formattedToday = format(today, 'dd/MM/yyyy')
const formattedTomorrow = format(tomorrow, 'dd/MM/yyyy')

console.log(`Hari ini: ${formattedToday}`)
console.log(`Besok: ${formattedTomorrow}`)
```

Dalam contoh ini, kita menggunakan `date-fns` untuk memformat tanggal dan menambahkan satu hari.

Keuntungan Modul Pihak Ketiga

- Modul pihak ketiga seperti `date-fns` dapat menghemat waktu Anda dalam proses pengembangan sistem aplikasi.
- Menyediakan fungsi-fungsi yang umum digunakan sehingga Anda tidak perlu menulis ulang kode yang serupa.
- Modul ini seringkali memiliki dokumentasi yang baik.

Latihan

- Tampilkan data buku dari berkas `books.json` menggunakan modul `fs` dalam format tabel dengan module `cli-table3`.

Latihan

Buat sebuah aplikasi quiz sederhana dengan ketentuan

- Dapat menampilkan pertanyaan yang diambil dari sebuah file JSON
- Memberikan feedback jika jawaban benar/salah
- Menampilkan skor akhir.

Latihan

Buat sebuah aplikasi todo list sederhana dengan ketentuan

- Dapat menampilkan daftar todo.
- Dapat menambahkan todo.
- Dapat menghapus todo.
- Dapat menandai todo sebagai selesai.
- Keluar dari aplikasi.