HTTP Module

Apa Itu Web Server?

- Sebuah server web adalah perangkat keras atau perangkat lunak yang melayani permintaan dari klien (browser) dan mengirimkan halaman web kepada mereka.
- Web server memainkan peran kunci dalam pengiriman konten web dari server ke klien.

Peran Utama Web Server

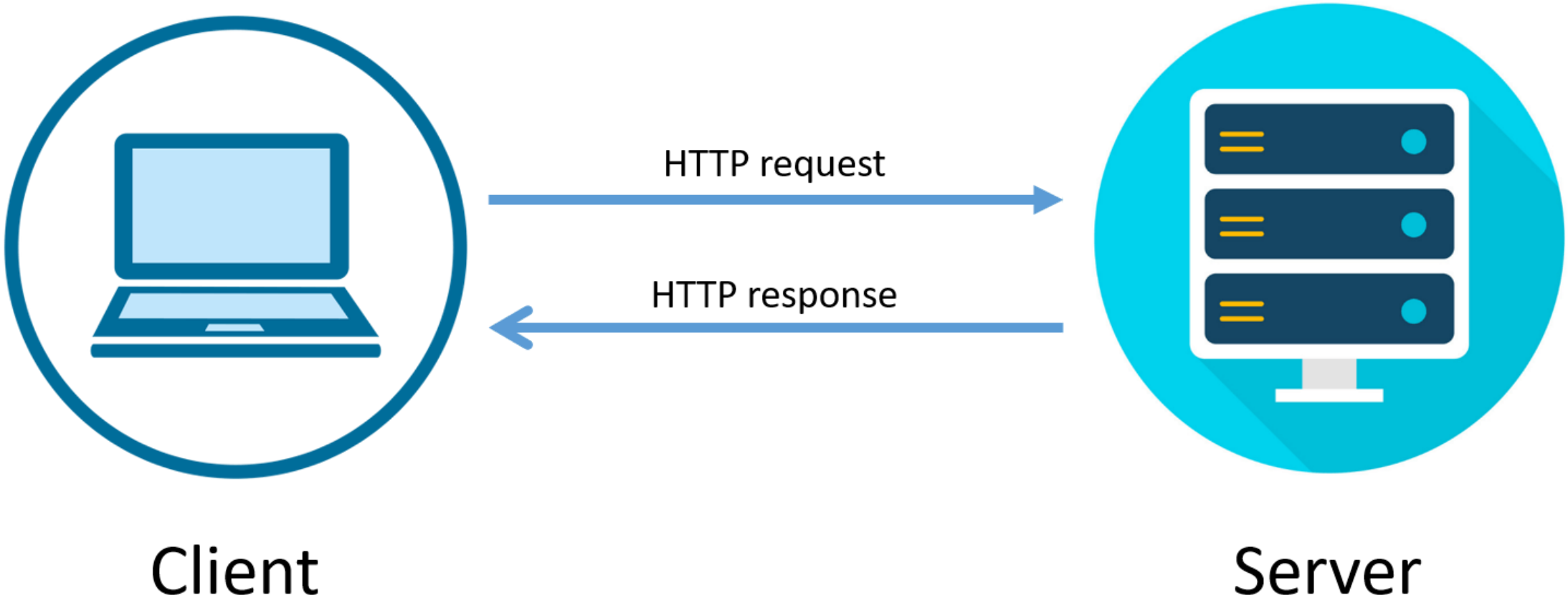
- **Menerima Permintaan (Request):** Web server menerima permintaan dari klien. Permintaan ini biasanya berupa URL atau alamat halaman web.
- **Menanggapi Permintaan (Response):** Web server menanggapi permintaan dengan mengirimkan konten halaman web yang diminta oleh klien.
- **Mengelola Sumber Daya:** Web server mengelola sumber daya seperti berkas, basis data, atau layanan lain yang diperlukan untuk menghasilkan halaman web.
- **Menangani Logika Aplikasi:** Dalam kasus server web yang lebih canggih, server dapat menangani logika aplikasi dan menyediakan data dinamis.

Jenis-jenis Web Server

- Terdapat berbagai jenis server web yang dapat digunakan, seperti Apache, Nginx, IIS (Internet Information Services), dan sebagainya.
- Pilihan server web dapat bergantung pada kebutuhan proyek, konfigurasi, dan preferensi.

Alur Kerja Sederhana

1. Klien (browser) mengirim permintaan HTTP ke server.
2. Web server menerima permintaan dan memprosesnya.
3. Web server mengirim respons HTTP dengan konten yang diminta kembali ke klien.
4. Klien menerima respons dan menampilkan halaman web atau konten tersebut.


HTTP request
HTTP response
Client
Server

Hubungan Antara Web Server dan HTTP

- Protokol HTTP (Hypertext Transfer Protocol) adalah dasar dari komunikasi di web.
- Web server adalah komponen yang menjalankan perangkat lunak untuk melayani permintaan HTTP.
- Protokol HTTP biasanya berjalan di port 80, tetapi dapat diubah sesuai kebutuhan.

HTTP Request

Permintaan HTTP terdiri dari beberapa elemen, termasuk:

- **Method**: Jenis tindakan yang diminta (`GET`, `POST`, `PUT`, `PATCH` `DELETE`).
- **URL**: Alamat sumber daya yang diminta.
- **Header**: Informasi tambahan seperti jenis konten atau cookie.
- **Body**: Data opsional yang dikirim bersama permintaan.

HTTP Response

Respons HTTP juga memiliki elemen-elemen penting, termasuk:

- **Status Code**: Kode numerik yang mengindikasikan apakah respons berhasil atau ada masalah.
- **Header**: Informasi tambahan dari server.
- **Body**: Konten yang dikirimkan kembali kepada klien.

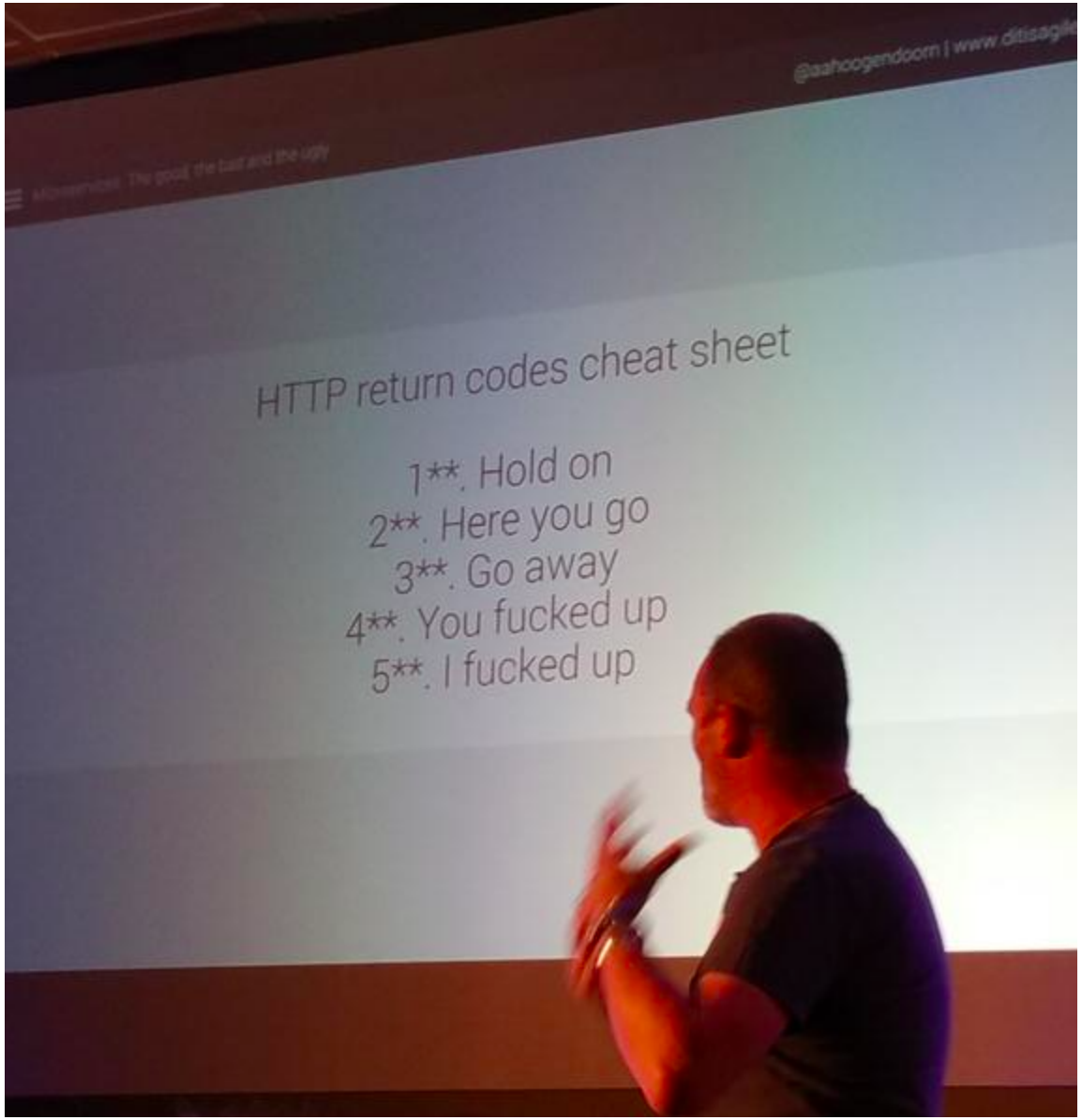
HTTP Status Code

HTTP status code dapat dibagi menjadi 5 kategori:

- 1xx: Informasi
- 2xx: Sukses
- 3xx: Pengalihan
- 4xx: Kesalahan Klien
- 5xx: Kesalahan Server

Referensi:

- https://http.cat/
- https://httpstatuses.io/


@aahoogendoorn | www.ditisagile
HTTP return codes cheat sheet
1**. Hold on
2**. Here you go
3**. Go away
4**. You fucked up
5**. I fucked up

HTTP Content Type

- HTTP Content Type adalah informasi yang dikirimkan oleh server kepada klien tentang jenis konten yang dikirimkan.
- Contoh: `text/html`, `application/json`, `image/png`, dan sebagainya.

Referensi
https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Modul HTTP dalam Node.js

- Node.js memiliki modul HTTP yang memungkinkan Anda untuk membuat server HTTP dan berinteraksi dengan request HTTP.
- Modul ini sangat penting dalam pengembangan aplikasi web dengan Node.js.

Membuat Server HTTP

```
const http = require("http")

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" })
  res.end("Hello World!\n")
})

const port = 3000

server.listen(port, () => {
  console.log(`Server running on http://localhost:${port}/`)
})
```

Dalam contoh, kita membuat server yang mendengarkan request HTTP di port 3000. Ketika request diterima, server akan mengirimkan response berisi teks "Hello World!".

Menangani Request HTTP

Contoh berikut menunjukkan cara menangani request untuk route tertentu:

```
const http = require("http")

const server = http.createServer((req, res) => {
  if (req.url === "/about") {
    res.writeHead(200, { "Content-Type": "text/plain" })
    res.end("About page\n")
  } else {
    res.writeHead(404, { "Content-Type": "text/plain" })
    res.end("Page not found\n")
  }
})

const port = 3000

server.listen(port, () => {
  console.log(`Server running on http://localhost:${port}/`)
})
```

Middleware dalam Node.js

- Anda dapat menggunakan middleware untuk memproses permintaan sebelum sampai ke penanganan rute.
- Middleware adalah fungsi-fungsi yang dieksekusi dalam urutan tertentu.

Middleware dalam Node.js

```
const http = require("http")

const middleware = (req, res, next) => {
  console.log("Middleware dipanggil")
  next()
}

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" })
  res.end("Hello World!\n")
})

server.on("request", middleware)

const port = 3000

server.listen(port, () => {
  console.log(`Server running on http://localhost:${port}/`)
})
```

Latihan

Buat server HTTP yang dapat menangani request untuk menampilkan response dalam format JSON dari sebuah array yang berisi data buku yang melayani beberapa route untuk menampilkan:

- Daftar buku
- Detail buku
- Pencarian buku berdasarkan judul