By: Hilman Fikry

Backend Development with Node.js

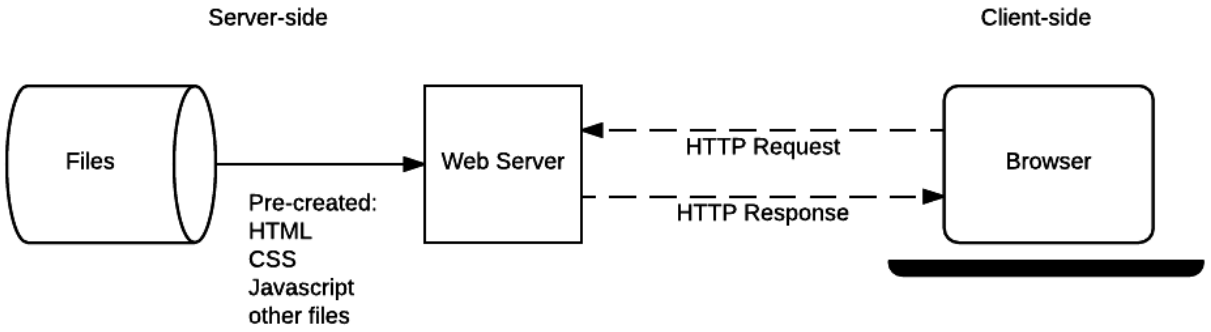
Node.js, Express, Authentication, REST APIs

Backend Development

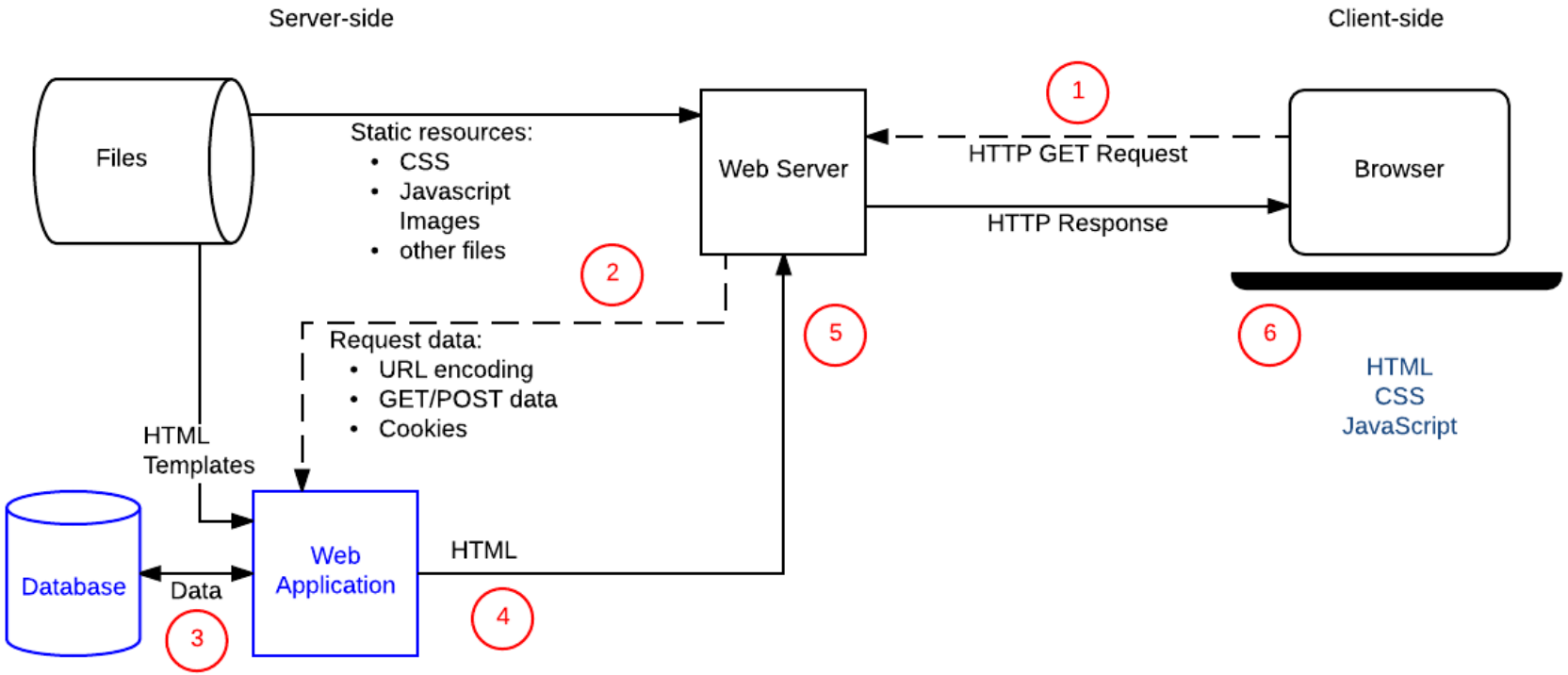
- Jika frontend berkaitan dengan app interface tempat user berinteraksi, maka **backend berkaitan dengan segala sesuatu yang berada di balik layar pada web server**.
- Backend dapat dibuat dengan berbagai bahasa pemrograman.
- Contoh proses backend:
  - memproses web page request yang masuk
  - menjalankan script untuk menghasilkan HTML
  - mengakses data dari database
  - menyimpan atau mengupdate data di database
  - mengenkripsi dan mendekripsi data
  - memproses download dan upload file
  - memproses input pengguna

Static sites vs dynamic sites

Static sites:



Dynamic sites:



Node.js

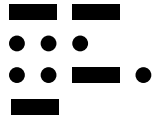
- Apa itu Node.js?
  Berdasarkan Node.js website:
  “*As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications.*”
- Keuntungan Node.js:
  - Bagus untuk prototyping dan agile development
  - Sangat cepat dan scalable
  - JavaScript di mana-mana
  - Codebase lebih bersih dan konsisten
  - Ekosistem open-source libraries yang luas

Contoh sederhana program Node.js

- Contoh program:

```
1   const http = require("node:http");
2
3   http.createServer(function (req, res) {
4     res.writeHead(200, {'Content-Type': 'text/html'});
5     res.end('Hello World!');
6   }).listen(8080);
7
```

- Cara lengkap menggunakan Node.js dapat dilihat di website Node.js: https://nodejs.org/



Express

- Framework bertujuan untuk menyediakan struktur umum sehingga kita tidak perlu membangun semuanya dari awal
- Berdasarkan websitenya, Express adalah framework Node.js yang cepat, unopinionated, dan minimalist
- Install package express di npm:
  `npm install express`

Contoh sederhana program Express

- Contoh program:

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/", (req, res) => res.send("Hello, world!"));
5
6   const PORT = 3000;
7   app.listen(PORT, (error) => {
8     if (error) {
9       throw error;
10    }
11    console.log(`My first Express app - listening on port ${PORT}!`);
12  });
13
```

- Cara lengkap menggunakan Express dapat dilihat di website Express: [Express - Node.js web application framework]

Express Routes

- Contoh struktur route:

```
app.get("/", (req, res) => res.send("Hello, world!"));
app.post("/messages", (req, res) => res.send("This is where you can see any messages."));
```

- Paths

```
// Matches both /message and /messages
"/message{s}"

// Matches both / and /messages
"/{messages}"

// Matches both /foo/baz and /foo/bar/baz
"/foo{/bar}/baz"

// Matches / and /odin as well as /sdds8fjsdifhj98sdfh
"/{*splat}"
```

Route Paths

Route parameters (akan disimpan di req.params)

```
/**
 * GET /odin/messages will have this log
 * { username: "odin" }
 *
 * GET /theodinproject79687378/messages would instead log
 * { username: "theodinproject79687378" }
 */
app.get("/:username/messages", (req, res) => {
  console.log(req.params);
  res.end();
});

/**
 * GET /odin/messages/79687378 will have this log
 * { username: "odin", messageId: "79687378" }
 */
app.get("/:username/messages/:messageId", (req, res) => {
  console.log(req.params);
  res.end();
});
```

Query parameters (akan disimpan di req.params)

```
/**
 * GET /odin/messages?sort=date&direction=ascending will log
 * Params: { username: "odin" }
 * Query: { sort: "date", direction: "ascending" }
 *
 * GET /odin/messages?sort=date&sort=likes&direction=ascending will log
 * Params: { username: "odin" }
 * Query: { sort: ["date", "likes"], direction: "ascending" }
 */
app.get("/:username/messages", (req, res) => {
  console.log("Params:", req.params);
  console.log("Query:", req.query);
  res.end();
});
```

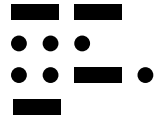
Express Controllers

- Metode umum untuk menangani respons
  - `res.send` – metode serbaguna untuk mengirim respons
  - `res.json` – cara eksplisit untuk merespons dengan json
  - `res.redirect` – meredirect client ke URL lain
  - `res.render` – merender view template
- Middleware (biasanya callback di dalam `router.use` / `router.METHOD`)
  - `req`
  - `res`
  - `next`
- Controller akan bekerja ketika ada request dan route cocok dengan request

Views

- View adalah bagian aplikasi yang berhadapan langsung dengan pengguna, contohnya file HTML
- Untuk membuat view, kita akan menggunakan template engine EJS
- Install package EJS di npm:
  `npm install ejs`
- Contoh syntax EJS:

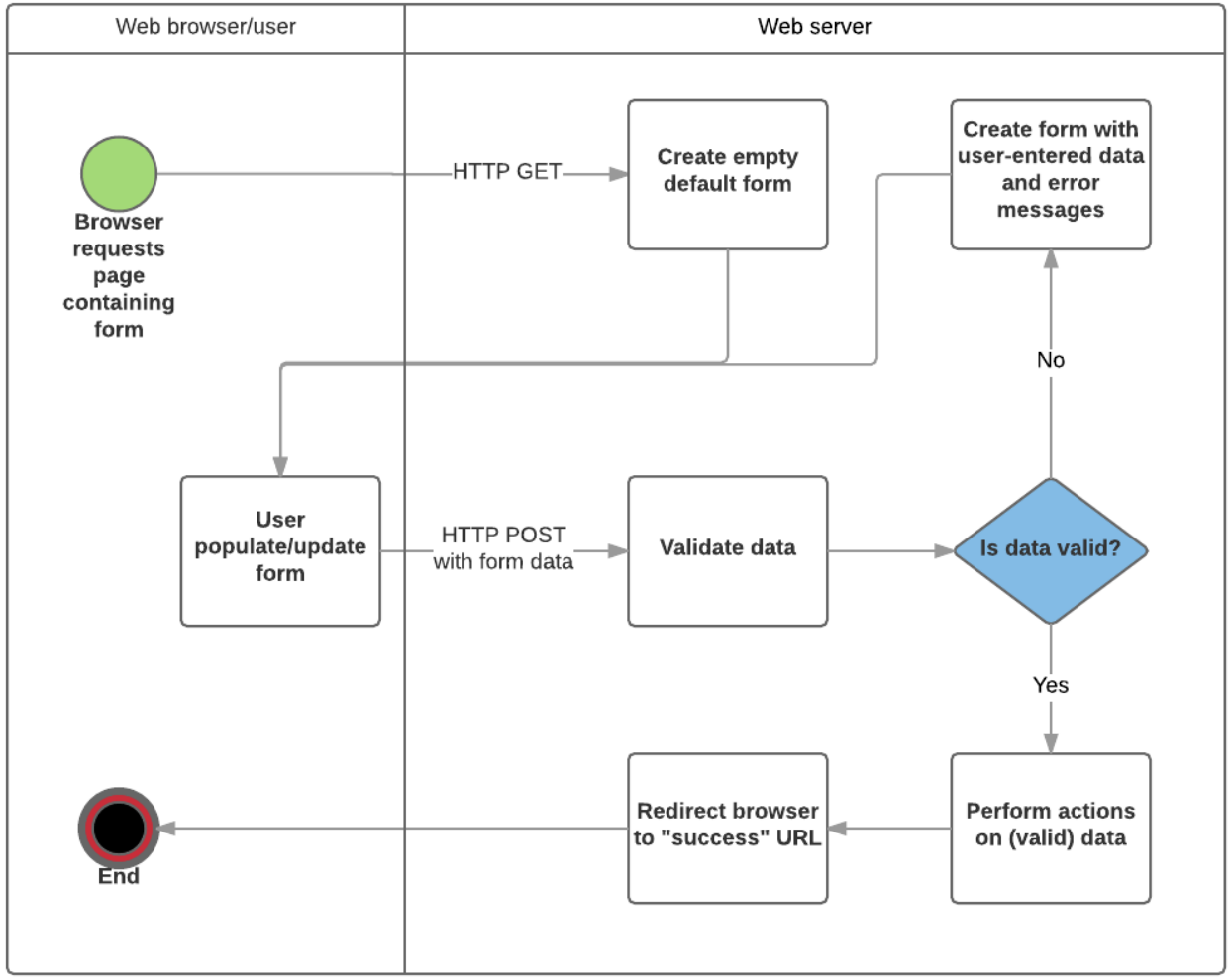
```
<% const animals = ["Cat", "Dog", "Lemur", "Hawk"] %>

<ul>
  <% animals.map((animal) => { %>
    <li><%= animal %>s are cute</li>
  <% }) %>
</ul>
```



Contoh struktur HTML form

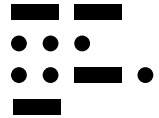
```
<form action="/create" method="POST">
  <label for="fullName">Full Name:</label>
  <input placeholder="John Doe" type="text" name="fullName" id="fullName">
  <button type="submit">Submit</button>
</form>
```

Cara menangani form



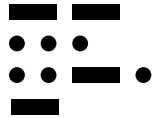
Menggunakan PostgreSQL

- Untuk menggunakan database, kita perlu:
  - menyiapkan db baru di PostgreSQL shell
  - menyiapkan dan melakukan query dengan node-postgres (pg)
  - menggunakan script untuk populate db



Authentication dengan Passport.js

- Website: Passport.js
- Passport.js menggunakan ***Strategies*** untuk mengautentikasi user
- Kita akan fokus menggunakan `LocalStrategy`, yang menggunakan session-based username dan password
- Jangan simpan password di database tanpa di-hash terlebih dahulu (bisa hash dengan package `bcrypt`)



API (Application Programming Interface)

- Cara alternatif dalam mengembangkan aplikasi adalah dengan memisahkan frontend dengan backend dan database
- Frontend dan backend dapat berkomunikasi menggunakan JSON
- Jadi pada dasarnya kita mengirim informasi dengan `res.json()` dibandingkan dengan `res.send()` atau `res.render()`
- Struktur konvensional API adalah REST (Representational State Transfer)

REST APIs

- REST API bersifat resource-based
- Contoh endpoints dalam REST API:

HTTP Verbs Table

Verb	Action	Example
POST	Create	`POST /posts` Creates a new blog post
GET	Read	`GET /posts/:postid` Fetches a single post
PUT	Update	`PUT /posts/:postid` Updates a single post
DELETE	Delete	`DELETE /posts/:postid` Deletes a single post