

# **PENGANTAR PEMROGRAMAN WEB 1**

## **HTML, CSS, dan JavaScript Modern**



**Dosen Pengampu:**

Maulana Ardiansyah S.Kom, M.Kom.

**Disusun Oleh:**

Kelompok : 6 (Enam)

Kelas : 05TPLP017

**Anggota Kelompok:**

Aufa Wafi Dhaifullah (231011401814)

Fadlan Lesmana (231011400735)

Ivandhika Satria Putra Ferdianto (231011401818)

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS PAMULANG  
2025**

## KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan Makalah yang berjudul "Pengantar Pemrograman Web 1: Dasar HTML, CSS, dan JavaScript Modern" ini dengan baik dan tepat waktu.

Makalah ini disusun sebagai salah satu syarat pemenuhan tugas mata kuliah Pengantar Pemrograman Web 1 dan bertujuan memberikan pemahaman mendalam mengenai tiga pilar utama dalam pengembangan web modern, yaitu HTML5, CSS3, dan JavaScript (ES6+). Ketiga teknologi ini merupakan fondasi dasar yang krusial bagi setiap pengembang aplikasi web.

Dalam penyusunan makalah ini, kami membahas secara komprehensif mulai dari konsep dasar arsitektur web, struktur dokumen HTML yang semantik, gaya visual responsif menggunakan CSS (termasuk Flexbox dan Grid), hingga aspek interaktif dan dinamis menggunakan JavaScript modern serta manipulasi DOM.

Kami menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan demi perbaikan di masa yang akan datang. Semoga makalah ini dapat memberikan manfaat dan menambah wawasan bagi pembaca, khususnya dalam memahami dunia pemrograman web.

Tangerang, 07 Januari 2026

Kelompok 6

## DAFTAR ISI

|   |            |
|---|------------|
| <b>KATA PENGANTAR.....</b>  | <b>ii</b>  |
| <b>DAFTAR ISI.....</b>  | <b>iii</b> |
| <b>BAB I PENDAHULUAN PEMROGRaMAN WEB .....</b>                            | <b>1</b>   |
| 1.1    Sejarah dan Perkembangan Pemrograman Web .....                     | 1          |
| 1.2    Perbedaan Pemrograman Desktop, Mobile, dan Web.....                | 3          |
| 1.3    Arsitektur Aplikasi Web Modern .....                               | 4          |
| 1.4    Peran HTML, CSS, dan JavaScript dalam Pengembangan Web .....       | 6          |
| <b>BAB II PENGENALAN DASAR HTML5 .....</b>                                | <b>8</b>   |
| 2.1    Struktur Dasar Dokumen HTML .....                                  | 8          |
| 2.1.1    Deklarasi Dokumen (DOCTYPE) .....                                | 8          |
| 2.1.2    Elemen <html>.....   | 9          |
| 2.1.3    Bagian <head> .....  | 9          |
| 2.1.4    Bagian <body> .....  | 10         |
| 2.1.5    Struktur Dasar Dokumen HTML Secara Lengkap .....                 | 12         |
| 2.2    Elemen dan Tag Penting (Headings, Paragraph, Links, Images).....   | 15         |
| 2.2.1    Elemen Heading (Judul) .....                                     | 15         |
| 2.2.2    Elemen Paragraf (<p>).....                                       | 15         |
| 2.2.3    Elemen Tautan (Links) – <a> .....                                | 16         |
| 2.2.4    Elemen Gambar (<img>) .....                                      | 16         |
| 2.2.5    Peran Elemen-elemen Ini Dalam Struktur Dokumen .....             | 17         |
| 2.3    Elemen Semantik HTML (header, nav, article, section, footer) ..... | 17         |
| 2.3.1    Elemen <header> .....  | 18         |
| 2.3.2    Elemen <nav> .....   | 18         |
| 2.3.3    Elemen <section>.....  | 19         |
| 2.3.4    Elemen <article> .....   | 20         |
| 2.3.5    Elemen <footer> .....  | 20         |
| 2.3.6    Peran Elemen Semantik Dalam Aksesibilitas & SEO .....            | 21         |
| 2.4    Formulir dan Input Modern.....                                     | 21         |
| 2.4.1    Struktur Dasar Formulir (<form>) .....                           | 21         |

|   |   |    |
|---|---|----|
| 2.4.2                                     | Elemen Input Dasar (Classic Inputs) .....                 | 22 |
| 2.4.3                                     | Input Modern dalam HTML .....                             | 22 |
| 2.4.4                                     | Atribut Modern Pada Input .....                           | 25 |
| 2.4.5                                     | Elemen Pendukung Formulir .....                           | 26 |
| 2.4.6                                     | Validasi Formulir Secara Otomatis.....                    | 27 |
| 2.4.7                                     | Contoh Formulir Modern Lengkap .....                      | 27 |
| 2.5                                       | Multimedia (Audio, Video, Canvas) .....                   | 29 |
| 2.5.1                                     | Elemen Audio .....  | 29 |
| 2.5.2                                     | Elemen Video.....   | 30 |
| 2.5.3                                     | Elemen Canvas.....  | 30 |
| <b>BAB III PENGENALAN DASAR CSS3.....</b> | <b>32</b>   |    |
| 3.1                                       | Konsep CSS (Inline, Internal, External Stylesheet) .....  | 32 |
| 3.1.1                                     | CSS Inline .....  | 32 |
| 3.1.2                                     | CSS Internal .....  | 33 |
| 3.1.3                                     | CSS External .....  | 34 |
| 3.2                                       | Selektor, Properti, dan Nilai .....                       | 35 |
| 3.2.1                                     | Selektor .....  | 35 |
| 3.2.2                                     | Properti (Property) .....                                 | 38 |
| 3.2.3                                     | Nilai (Value) .....                                       | 39 |
| 3.2.4                                     | Contoh Penggunaan Selektor, Properti, dan Nilai .....     | 39 |
| 3.3                                       | Warna, Teks, dan Font .....                               | 40 |
| 3.3.1                                     | Pengaturan Warna (Color) .....                            | 40 |
| 3.3.2                                     | Pengaturan Teks (Text Properties).....                    | 42 |
| 3.3.3                                     | Pengaturan Font (Font Properties .....                    | 43 |
| 3.3.4                                     | Contoh Implementasi Warna, Teks, dan Font.....            | 45 |
| 3.4                                       | Box Model (Margin, Padding, Border) .....                 | 46 |
| 3.4.1                                     | Konsep Dasar Box Model.....                               | 46 |
| 3.4.2                                     | Padding .....   | 47 |
| 3.4.3                                     | Border .....  | 48 |
| 3.4.4                                     | Margin.....   | 49 |
| 3.4.5                                     | Perhitungan Ukuran Elemen (Element Size Calculation)..... | 50 |
| 3.4.6                                     | Contoh Implementasi Box Model .....                       | 51 |
| 3.5                                       | Layout Dasar dengan Flexbox & Grid CSS .....              | 51 |
| 3.5.1                                     | Flexbox (Flexible Box Layout).....                        | 52 |

|   |   |            |
|---|---|------------|
| 3.5.2   | CSS Grid Layout.....  | 56         |
| 3.5.3   | Perbandingan Flexbox dan Grid .....                           | 60         |
| 3.6   | Efek Visual (Transisi, Transformasi, Animasi Sederhana).....  | 60         |
| 3.6.1   | CSS Transition (Efek Perubahan yang Halus).....               | 60         |
| 3.6.2   | CSS Transform (Mengubah Bentuk Elemen) .....                  | 62         |
| 3.6.3   | CSS Animation (Animasi Berulang) .....                        | 63         |
| <b>BAB IV PENGENALAN JAVASCRIPT MODERN (ES6+)</b> | .....   | <b>64</b>  |
| 4.1   | Konsep Dasar Pemrograman dengan JavaScript.....               | 64         |
| 4.2   | Variabel (let, const, var) .....                              | 66         |
| 4.3   | Tipe Data dan Operator .....                                  | 67         |
| 4.4   | Struktur Kontrol (if, loop, switch).....                      | 70         |
| 4.5   | Fungsi (Regular Function, Arrow Function).....                | 74         |
| 4.6   | Event Handling Dasar .....                                    | 75         |
| <b>BAB V DOM DAN MANIPULASI ELEMEN</b>            | .....   | <b>79</b>  |
| 5.1   | Konsep Document Object Model (DOM).....                       | 79         |
| 5.2   | Mengakses Elemen HTML dengan getElementById dan querySelector | 80         |
| 5.2.1   | Menggunakan ID (getElementById).....                          | 80         |
| 5.2.2   | Menggunakan Selektor CSS (querySelector) .....                | 81         |
| 5.2.3   | Menggunakan querySelectorAll (Banyak Elemen) .....            | 81         |
| 5.3   | Mengubah Konten dan Atribut Elemen .....                      | 82         |
| 5.3.1   | Manipulasi Konten (innerText vs innerHTML) .....              | 82         |
| 5.3.2   | Manipulasi Atribut (setAttribute, src, href) .....            | 83         |
| 5.3.3   | Manipulasi Style dan Class (Penting!).....                    | 84         |
| 5.4   | Event Listener dan Interaktivitas Halaman .....               | 85         |
| 5.5   | Contoh Sederhana (Studi Kasus).....                           | 87         |
| <b>BAB VI RESPONSIVE WEB DESIGN</b>               | .....   | <b>102</b> |
| 6.1   | Konsep Responsive Design dan Mobile-First.....                | 102        |
| 6.1.1   | Definisi Responsive Web Design (RWD) .....                    | 102        |
| 6.1.2   | Konsep Mobile-First .....                                     | 103        |
| 6.1.3   | Viewport Meta Tag .....                                       | 104        |
| 6.2   | Media Queries CSS .....                                       | 104        |
| 6.2.1   | Sintaksis dan Struktur Logika .....                           | 104        |
| 6.2.2   | Penentuan Titik Henti (Breakpoints) .....                     | 105        |
| 6.2.3   | Fitur Media Lainnya .....                                     | 108        |

|  |   |     |
|--|---|-----|
| 6.3  | Layout Responsif dengan Flexbox dan Grid.....                                       | 109 |
| 6.3.1  | Flexbox Responsif .....   | 109 |
| 6.3.2  | CSS Grid Responsif .....  | 110 |
| 6.4  | Penggunaan Framework CSS Populer .....  | 111 |
| 6.4.1  | Klasifikasi Framework CSS.....  | 111 |
| 6.4.2  | Memilih Framework yang Tepat .....  | 112 |
| <b>BAB VII INTEGRASI HTML, CSS, DAN JAVASCRIPT .....</b> | <b>113</b>  |     |
| 7.1  | Membuat Halaman Web Interaktif Sederhana .....                                      | 113 |
| 7.1.1  | Filosofi "Separation of Concerns" .....   | 113 |
| 7.1.2  | Mekanisme Pemuatan Halaman (Rendering Flow) .....                                   | 114 |
| 7.2  | Form Input dengan Validasi JavaScript .....   | 118 |
| 7.2.1  | Pentingnya Validasi Sisi Klien .....  | 118 |
| 7.2.2  | Teknik Manipulasi Class untuk Validasi .....  | 119 |
| 7.2.3  | Mencegah Aksi Default (preventDefault).....   | 119 |
| 7.3  | Animasi Interaktif Menggunakan CSS + JS .....                                       | 124 |
| 7.3.1  | Konsep "State-Based UI" .....   | 124 |
| 7.3.2  | Keunggulan Kombinasi CSS + JS .....   | 124 |
| 7.4  | Studi Kasus Mini Project: Halaman Landing Page Responsif.....                       | 126 |
| <b>BAB VIII PRAKTIK &amp; PROYEK AKHIR .....</b>         | <b>141</b>  |     |
| 8.1  | Panduan Membangun Website Portofolio Sederhana .....                                | 141 |
| 8.2  | Menerapkan HTML5, CSS3, dan JavaScript Dasar.....                                   | 141 |
| 8.3  | Menambahkan Elemen Interaktif (Menu Navigasi, Slideshow, dan Form Validation) ..... | 159 |
| <b>BAB IX TREN &amp; ARAH PERKEMBANGAN WEB .....</b>     | <b>164</b>  |     |
| 9.1  | Evolusi HTML, CSS, dan JavaScript Terbaru .....                                     | 164 |
| 9.2  | Framework Modern dalam Pengembangan Web .....                                       | 165 |
| 9.3  | Progressive Web Apps (PWA) sebagai Tren Baru .....                                  | 166 |
| <b>DAFTAR PUSTAKA.....</b>                               | <b>167</b>  |     |
| <b>BIOGRAFI PENULIS .....</b>                            | <b>168</b>  |     |

## BAB I

### PENDAHULUAN PEMROGRAMAN WEB

#### 1.1 Sejarah dan Perkembangan Pemrograman Web

Sejarah World Wide Web (WWW) bermula pada tahun 1989 ketika Tim Berners-Lee, seorang ilmuwan di CERN (Organisasi Eropa untuk Riset Nuklir), mengajukan proposal sistem manajemen informasi terdistribusi. Tujuannya kala itu cukup sederhana, yaitu untuk memudahkan pertukaran dokumen dan hasil penelitian antar sesama ilmuwan di berbagai universitas di seluruh dunia. Pada tahun 1991, website pertama di dunia diluncurkan dengan tampilan yang sangat sederhana, hanya berisi teks dan *hyperlink*. Perkembangan teknologi web kemudian dikategorikan ke dalam beberapa fase evolusi utama:

1. **Web 1.0 (The Static Web):** Era ini berlangsung dari awal 1990-an hingga awal 2000-an. Pada masa ini, website bersifat "read-only" atau satu arah. Pengguna hanya bisa membaca informasi yang disajikan oleh pembuat website tanpa bisa memberikan umpan balik, komentar, atau membuat konten sendiri. Halaman web bersifat statis dan dibangun murni menggunakan HTML dasar.



#### Web 1.0

Gambar 1. 1 Web 1.0

2. **Web 2.0 (The Social Web):** Muncul sekitar tahun 2004, era ini menandai revolusi interaksi digital. Website berubah menjadi *platform* "read-write", di mana pengguna tidak hanya menjadi konsumen informasi tetapi juga produsen konten (*User Generated Content*). Contoh nyata dari era ini adalah kemunculan media sosial (Facebook, Twitter), blog, wiki, dan forum diskusi. Teknologi seperti AJAX (*Asynchronous JavaScript and XML*) mulai digunakan untuk membuat website lebih responsif tanpa perlu memuat ulang seluruh halaman.



## Web 2.0

Gambar 1. 2 Web 2.0

3. **Web 3.0 (The Semantic Web):** Saat ini kita sedang berada di masa transisi menuju Web 3.0. Fokus utamanya adalah pada pemrosesan data oleh mesin (*machine readable*) agar komputer dapat memahami konteks informasi selayaknya manusia. Era ini ditandai dengan kecerdasan buatan (AI), personalisasi konten yang ekstrem, desentralisasi data (*blockchain*), serta penggunaan *Single Page Application* (SPA) yang membuat pengalaman menggunakan web terasa secepat aplikasi desktop.



## Web 3.0

Gambar 1. 3 Web 3.0

### 1.2 Perbedaan Pemrograman Desktop, Mobile, dan Web

Dalam rekayasa perangkat lunak, pemilihan *platform* sangat menentukan teknologi dan pendekatan yang digunakan. Berikut adalah analisis perbedaan mendalam antara ketiga *platform* utama: Secara lebih rinci, tujuan pembelajaran Pemrograman Web 1 meliputi:

#### 1.3 Aplikasi Desktop:

- **Karakteristik:** Aplikasi berjalan secara lokal di atas sistem operasi tertentu (Windows, macOS, atau Linux).
- **Instalasi & Distribusi:** Pengguna wajib mengunduh *installer* (.exe, .dmg) dan menginstalnya. Pembaruan (*update*) versi sering kali merepotkan karena pengguna harus mengunduh ulang atau melakukan *patching*.
- **Kelebihan:** Memiliki performa sangat tinggi dan akses penuh ke perangkat keras (seperti RAM, GPU, dan File System).
- **Bahasa Pemrograman:** C++, C#, Java (Swing/JavaFX), Python.

#### 2.3 Aplikasi Mobile:

- **Karakteristik:** Dirancang khusus untuk perangkat genggam (smartphone/tablet) dengan interaksi layar sentuh.
- **Instalasi & Distribusi:** Didistribusikan melalui toko aplikasi resmi (*App Store* atau *Play Store*). Aplikasi mobile terbagi menjadi *Native*

(dibuat spesifik untuk Android/iOS) dan *Hybrid* (menggunakan teknologi web yang dibungkus menjadi aplikasi).

- **Kelebihan:** Integrasi mendalam dengan fitur *hardware* spesifik seperti GPS, Kamera, *Accelerometer*, dan *Push Notifications*.
- **Bahasa Pemrograman:** Swift (iOS), Kotlin/Java (Android), Flutter/React Native.

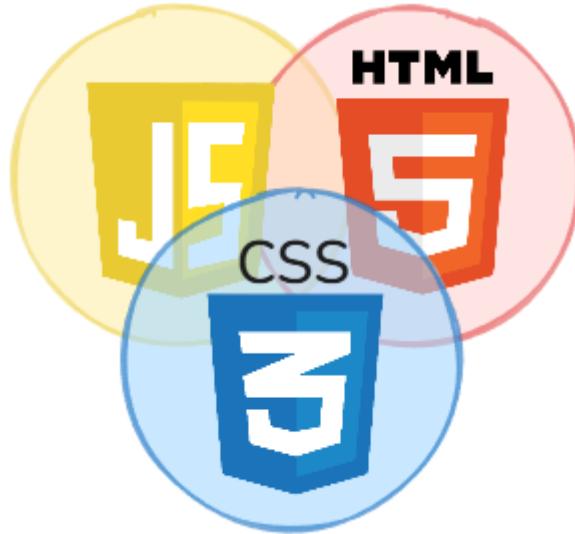
### 3.3 Aplikasi Web:

- **Karakteristik:** Aplikasi yang diakses melalui peramban (*web browser*) dan tidak terikat pada sistem operasi tertentu.
- **Instalasi & Distribusi:** Tidak memerlukan instalasi. Pengguna hanya perlu mengetahui alamat URL (*Uniform Resource Locator*). Pembaruan sistem terjadi di sisi *server*, sehingga pengguna akan selalu mendapatkan versi terbaru setiap kali memuat ulang halaman (*refresh*).
- **Kelebihan:** Jangkauan akses yang sangat luas (*cross-platform*) dan biaya distribusi yang rendah. Dengan teknologi *Progressive Web Apps* (PWA), web kini bahkan bisa berjalan semi-offline.
- **Bahasa Pemrograman:** HTML, CSS, JavaScript, PHP, Ruby, Go.

## 1.3 Arsitektur Aplikasi Web Modern

Aplikasi web modern bekerja berdasarkan arsitektur **Client-Server** yang melibatkan komunikasi data melalui protokol HTTP (*Hypertext Transfer Protocol*). Proses ini dapat diuraikan sebagai berikut:

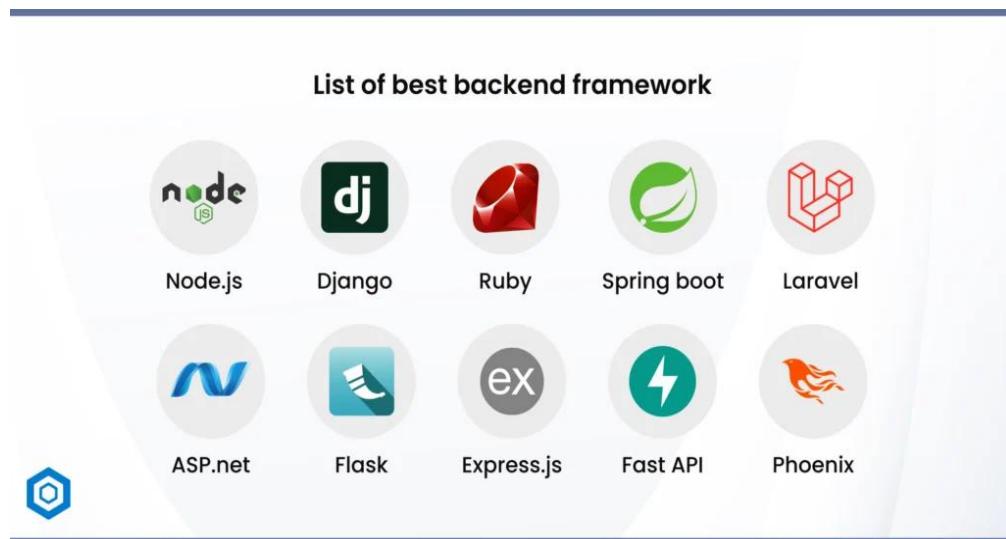
1. **Sisi Client (Frontend):** merupakan antarmuka visual yang berhadapan langsung dengan pengguna. Client biasanya adalah web browser (Chrome, Firefox, Edge) yang bertugas menerjemahkan kode program menjadi tampilan grafis. Ketika pengguna mengetikkan alamat website, browser mengirimkan permintaan (Request) ke komputer server. Teknologi yang bekerja di sisi ini disebut Frontend, yang meliputi:
  - HTML (Struktur)
  - CSS (Desain)
  - JavaScript (Interaksi)



Gambar 1. 4 Html, css, javascript

2. **Sisi Server (Backend):** server adalah komputer pusat yang melayani permintaan Client. *Server* menerima *request*, memprosesnya, dan mengirimkan balasan (*Response*) kembali ke *client*. Bahasa yang digunakan meliputi PHP, Node.js, Python, Java, dan GoLang. Sisi ini menangani logika bisnis yang tidak terlihat oleh pengguna, seperti:

- Mengecek apakah *password* saat login benar atau salah.
- Menghitung total belanjaan di keranjang e-commerce.
- Menyimpan data pendaftaran mahasiswa ke *Database*.



Gambar 1. 5 Best Backend Frameworks

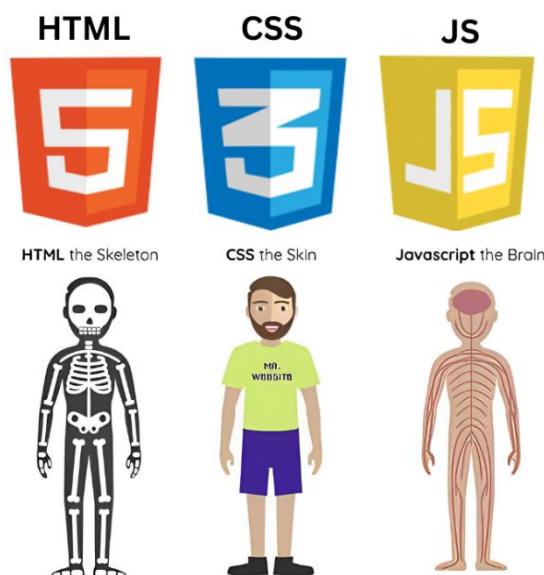
**3. Database (Basis Data):** Merupakan tempat penyimpanan data persisten. Aplikasi web membutuhkan *database* untuk menyimpan data pengguna, produk, artikel, dan informasi lainnya agar tidak hilang saat server dimatikan. Contoh sistem basis data populer adalah MySQL, PostgreSQL, dan MongoDB.



Gambar 1. 6 Database SQL & NoSQL

#### 1.4 Peran HTML, CSS, dan JavaScript dalam Pengembangan Web

Dalam pengembangan *frontend* modern, terdapat tiga pilar teknologi utama yang memiliki peran spesifik namun saling melengkapi. Analogi yang sering digunakan untuk menggambarkan ketiganya adalah "Tubuh Manusia":



Gambar 1. 7 Peran HTML, CSS, dan JS

## **1. HTML (*HyperText Markup Language*):**

HTML berperan sebagai kerangka tulang (*skeleton*) dari sebuah halaman web. HTML mendefinisikan struktur dan makna semantik dari konten, menentukan mana yang merupakan judul (*heading*), paragraf, gambar, tautan, atau tabel. Tanpa HTML, tidak akan ada konten yang bisa ditampilkan. HTML5 adalah standar terbaru yang membawa fitur elemen semantik (seperti <header>, <footer>, <section>) yang memudahkan mesin pencari (SEO) memahami isi website.

## **2. CSS (*Cascading Style Sheets*):**

Jika HTML adalah tulangnya, maka CSS adalah kulit, pakaian, dan riasannya. CSS bertanggung jawab atas aspek visual dan tata letak (layout). CSS mengatur warna, jenis huruf (typography), jarak antar elemen (margin/padding), dan bagaimana elemen tersebut diposisikan di layar. CSS3 juga memungkinkan pembuatan animasi dan desain responsif (Responsive Web Design), yang memastikan tampilan web tetap rapi saat dibuka di layar ponsel yang kecil maupun monitor desktop yang besar.

## **3. JavaScript:**

JavaScript adalah sistem saraf dan otot yang membuat website menjadi hidup. Bahasa ini menangani logika pemrograman di sisi *client*. Dengan JavaScript, halaman web bisa merespons tindakan pengguna secara *real-time* tanpa harus memuat ulang halaman. Contoh perannya meliputi validasi formulir (memastikan email yang dimasukkan benar), membuat menu *dropdown*, menampilkan peta interaktif, hingga mengambil data dari server di latar belakang (*AJAX/Fetch API*).

## BAB II

### PENGENALAN DASAR HTML5

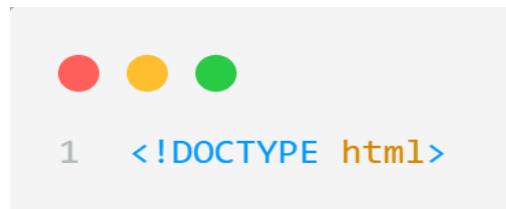
#### 2.1 Struktur Dasar Dokumen HTML

Struktur dasar dokumen HTML merupakan fondasi utama dalam penyusunan halaman web. Dokumen HTML disusun menggunakan elemen-elemen (tags) yang membentuk kerangka sebuah halaman, sehingga browser dapat menampilkan konten secara terstruktur, rapi, dan sesuai standar. Pada dasarnya, setiap dokumen HTML memiliki beberapa elemen wajib yang harus disertakan, yaitu elemen deklarasi dokumen, bagian kepala (*head*), dan bagian isi (*body*). Secara umum, struktur HTML terdiri atas:

##### 2.1.1 Deklarasi Dokumen (DOCTYPE)

Bagian paling atas dari dokumen HTML yang berfungsi memberi tahu browser bahwa halaman tersebut menggunakan standar HTML modern.

Contoh :



```
<!DOCTYPE html>
```

Deklarasi ini membantu browser menampilkan halaman dalam mode standar (*standards mode*) sehingga perilaku elemen HTML mengikuti aturan resmi yang ditetapkan oleh W3C.

### **2.1.2 Elemen <html>**

Elemen ini menjadi pembungkus (root element) dari seluruh konten HTML. Semua elemen lain harus berada di dalam tag <html>.

Contoh :

```
<html>  
....  
</html>
```



Tag <html> juga sering dilengkapi atribut seperti lang untuk menentukan bahasa utama dokumen.

### **2.1.3 Bagian <head>**

Bagian *head* digunakan untuk menyimpan informasi metadata dari halaman seperti:

- Judul halaman (<title>)
- Referensi stylesheet (CSS)
- Meta tag deskripsi
- Charset atau encoding teks
- Script eksternal
- Favicon

Contoh:

**Input:**

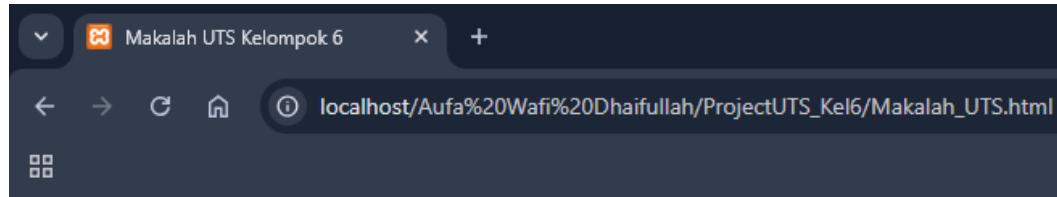
```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Makalah UTS Kelompok 6</title>
    <link rel="stylesheet" href="styles.css">
</head>
```



A screenshot of a code editor window. At the top, there are three colored circular icons: red, yellow, and green. Below them, the code for the head section of an HTML document is displayed. The code is color-coded: blue for tags like <head>, <meta>, <title>, and <link>; orange for attributes like charset, http-equiv, content, and rel; and green for href. The code is numbered from 1 to 7 on the left.

```
1  <head>
2      <meta charset="UTF-8">
3      <meta http-equiv="X-UA-Compatible" content="IE=edge">
4      <meta name="viewport" content="width=device-width, initial-scale=1.0">
5      <title>Makalah UTS Kelompok 6</title>
6      <link rel="stylesheet" href="styles.css">
7  </head>
```

#### Output:



#### 2.1.4 Bagian <body>

Bagian *body* merupakan tempat ditampilkannya seluruh konten halaman kepada pengguna, seperti:

- Teks
- Gambar
- Link
- Formulir
- Tabel
- Video
- Navigasi
- Elemen layout lainnya

Contoh:

**Input:**

```
<body>
  <header>
    <h1>Makalah UTS Kelompok 6</h1>
    <nav>
      <ul>
        <li><a href="#introduction">Pendahuluan</a></li>
        <li><a href="#methodology">Metodologi</a></li>
      </ul>
    </nav>
  </header>
  <section id="introduction">
    <h2>Pendahuluan</h2>
    <p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>
  </section>
  <section id="methodology">
    <h2>Metodologi</h2>
    <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>
  </section>
  <footer>
    <p>&copy; 2024 Kelompok 6. All rights reserved.</p>
  </footer>
</body>
```

```
1 <body>
2   <header>
3     <h1>Makalah UTS Kelompok 6</h1>
4     <nav>
5       <ul>
6         <li><a href="#introduction">Pendahuluan</a></li>
7         <li><a href="#methodology">Metodologi</a></li>
8       </ul>
9     </nav>
10    </header>
11    <section id="introduction">
12      <h2>Pendahuluan</h2>
13      <p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>
14    </section>
15    <section id="methodology">
16      <h2>Metodologi</h2>
17      <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>
18    </section>
19    <footer>
20      <p>© 2024 Kelompok 6. All rights reserved.</p>
21    </footer>
22  </body>
```

### Output:

# Makalah UTS Kelompok 6

- [Pendahuluan](#)
- [Metodologi](#)

## Pendahuluan

Bagian ini berisi latar belakang dan tujuan dari makalah ini.

## Metodologi

Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.

© 2024 Kelompok 6. All rights reserved.

### 2.1.5 Struktur Dasar Dokumen HTML Secara Lengkap

Struktur dasar halaman HTML dapat dituliskan sebagai berikut:

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Makalah UTS Kelompok 6</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Makalah UTS Kelompok 6</h1>
        <nav>
            <ul>
                <li><a href="#introduction">Pendahuluan</a></li>
                <li><a href="#methodology">Metodologi</a></li>
            </ul>
        </nav>
    </header>
    <section id="introduction">
        <h2>Pendahuluan</h2>
        <p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>
    </section>
    <section id="methodology">
        <h2>Metodologi</h2>
        <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>
    </section>
    <footer>
        <p>&copy; 2024 Kelompok 6. All rights reserved.</p>
    </footer>

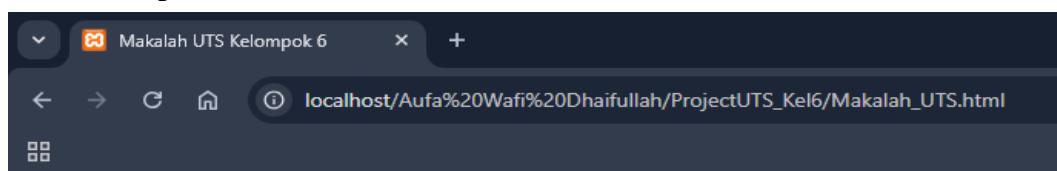
```

```
</footer>  
</body>  
</html>
```



```
● ○ ● ●  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">  
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">  
7      <title>Makalah UTS Kelompok 6</title>  
8      <link rel="stylesheet" href="styles.css">  
9  </head>  
10 <body>  
11     <header>  
12         <h1>Makalah UTS Kelompok 6</h1>  
13         <nav>  
14             <ul>  
15                 <li><a href="#introduction">Pendahuluan</a></li>  
16                 <li><a href="#methodology">Metodologi</a></li>  
17             </ul>  
18         </nav>  
19     </header>  
20     <section id="introduction">  
21         <h2>Pendahuluan</h2>  
22         <p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>  
23     </section>  
24  
25     <section id="methodology">  
26         <h2>Metodologi</h2>  
27         <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>  
28     </section>  
29     <footer>  
30         <p>© 2024 Kelompok 6. All rights reserved.</p>  
31     </footer>  
32 </body>  
33 </html>
```

## Output:



# Makalah UTS Kelompok 6

- [Pendahuluan](#)
- [Metodologi](#)

## Pendahuluan

Bagian ini berisi latar belakang dan tujuan dari makalah ini.

## Metodologi

Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.

© 2024 Kelompok 6. All rights reserved.

Struktur tersebut merupakan kerangka minimal yang harus dimiliki oleh setiap halaman web sehingga dapat tampil secara benar dan mengikuti standar web yang berlaku.

## 2.2 Elemen dan Tag Penting (Headings, Paragraph, Links, Images)

Dalam penyusunan halaman web, HTML menyediakan berbagai elemen dasar yang berfungsi untuk menampilkan informasi secara terstruktur. Elemen-elemen ini meliputi heading, paragraf, tautan, dan gambar. Penggunaan elemen penting tersebut membantu browser memahami hierarki informasi serta memudahkan pengguna dalam membaca konten.

### 2.2.1 Elemen Heading (Judul)

Heading digunakan untuk memberi struktur hierarkis pada teks. HTML menyediakan enam tingkatan heading, mulai dari `<h1>` sebagai tingkat paling tinggi hingga `<h6>` sebagai tingkat paling rendah. Fungsi heading:

- Menandai judul dan subjudul halaman
- Membentuk struktur konten
- Meningkatkan aksesibilitas
- Penting untuk SEO (Search Engine Optimization)

Contoh:

```
<h1>Makalah UTS Kelompok 6</h1>
<h2>Topik: [Masukkan Topik Di Sini]</h2>
<h3>Anggota Kelompok:</h3>
```



```
1 <h1>Makalah UTS Kelompok 6</h1>
2 <h2>Topik: [Masukkan Topik Di Sini]</h2>
3 <h3>Anggota Kelompok:</h3>
```

### 2.2.2 Elemen Paragraf (<p>)

Elemen paragraf adalah elemen dasar untuk menampilkan teks dalam bentuk paragraf. Browser secara otomatis menambahkan jarak (margin) antar paragraf untuk memudahkan pembacaan. Fungsi paragraph adalah

Menyajikan informasi dalam bentuk teks dan Memisahkan ide atau kalimat dalam konten.

Contoh:

```
<p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>
```



```
1 <p>Bagian ini berisi latar belakang dan tujuan dari makalah ini.</p>
```

### 2.2.3 Elemen Tautan (Links) – <a>

Tautan digunakan untuk menghubungkan satu halaman ke halaman lain, baik dalam situs yang sama maupun situs lain di internet. Elemen ini menggunakan atribut href untuk menampung alamat tujuan. Fungsi tautan:

- Menghubungkan halaman internal
- Mengarahkan pengguna ke website eksternal
- Mengunduh file
- Menjalankan fungsi tertentu melalui URL

Contoh:

```
<li><a href="#introduction">Pendahuluan</a></li>
<li><a href="#methodology">Metodologi</a></li>
```



```
1 <li><a href="#introduction">Pendahuluan</a></li>
2 <li><a href="#methodology">Metodologi</a></li>
```

Untuk membuka tautan di tab baru, dapat digunakan atribut target="\_blank".

### 2.2.4 Elemen Gambar (<img>)

Elemen gambar digunakan untuk menampilkan gambar pada halaman web. Elemen <img> bersifat *self-closing*, sehingga tidak memiliki tag penutup. Ada atribut penting yaitu:

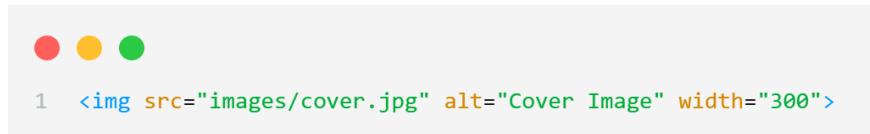
- **src:** lokasi gambar

- **alt**: teks alternatif yang muncul jika gambar gagal dimuat
- **width/height**: ukuran tampilan gambar

Contoh:

```

```



Teks alternatif sangat penting untuk aksesibilitas, terutama bagi pengguna yang menggunakan screen reader.

### 2.2.5 Peran Elemen-elemen Ini Dalam Struktur Dokumen

Elemen heading, paragraf, tautan, dan gambar merupakan komponen fundamental yang membentuk konten utama dalam halaman web. Dengan memanfaatkan elemen-elemen tersebut secara tepat, pengembang dapat:

- Menyajikan informasi secara struktural
- Menghasilkan halaman yang mudah dinavigasi
- Meningkatkan pengalaman pengguna
- Meningkatkan kualitas aksesibilitas dan SEO

## 2.3 Elemen Semantik HTML (header, nav, article, section, footer)

Elemen semantik HTML merupakan elemen yang memiliki makna khusus sesuai fungsinya dalam struktur dokumen. Berbeda dengan elemen non-semantik seperti `<div>`, elemen semantik memberikan konteks mengenai isi yang berada di dalamnya. Hal ini membantu browser, developer, serta mesin pencari memahami tujuan dari suatu bagian halaman.

Penggunaan elemen semantik ditujukan untuk meningkatkan:

- Struktur dokumen
- Aksesibilitas
- SEO (Search Engine Optimization)
- Kemudahan pemeliharaan kode

Elemen-elemen semantik yang paling umum digunakan antara lain: <header>, <nav>, <section>, <article>, dan <footer>.

### 2.3.1 Elemen <header>

Elemen <header> digunakan untuk menampilkan bagian kepala atau pengantar dari suatu halaman atau suatu bagian konten. Fungsi <header>:

- Menampilkan judul halaman
- Menyertakan logo website
- Menampilkan navigasi utama (opsional)
- Menampilkan tagline atau pengantar singkat

Contoh:

```
<header>
    <h1>Makalah UTS Kelompok 6</h1>
    <p>Belajar HTML dan CSS</p>
</header>
```



### 2.3.2 Elemen <nav>

Elemen <nav> digunakan untuk menampung kumpulan tautan navigasi yang digunakan untuk berpindah halaman atau bagian tertentu dalam website. Fungsi <nav>:

- Menampung menu utama
- Menyediakan navigasi sidebar
- Mengelompokkan link navigasi agar lebih terstruktur

Contoh:

```
<nav>
  <ul>
    <li><a href="#introduction">Pendahuluan</a></li>
    <li><a href="#methodology">Metodologi</a></li>
  </ul>
</nav>
```



```
1  <nav>
2    <ul>
3      <li><a href="#introduction">Pendahuluan</a></li>
4      <li><a href="#methodology">Metodologi</a></li>
5    </ul>
6  </nav>
```

### 2.3.3 Elemen <section>

Elemen <section> digunakan untuk mengelompokkan konten dalam satu topik tertentu. Biasanya digunakan sebagai pembagian utama dalam suatu halaman. Fungsi <section>:

- Mengelompokkan bagian-bagian utama konten
- Menyusun struktur bab atau topik
- Menjadi container bagi sub-konten yang berhubungan

Contoh:

```
<section id="methodology">
  <h2>Metodologi</h2>
  <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>
</section>
```



```
1  <section id="methodology">
2    <h2>Metodologi</h2>
3    <p>Bagian ini menjelaskan metode yang digunakan dalam penelitian atau proyek ini.</p>
4  </section>
```

### 2.3.4 Elemen <article>

Elemen <article> digunakan untuk menampung konten mandiri dan berdiri sendiri. Konten yang ada di dalam <article> biasanya bisa dipublikasikan secara terpisah. Contoh konten yang cocok menggunakan <article>:

- Berita
- Postingan blog
- Forum post
- Konten mandiri lainnya

Contoh:

```
<article>
    <h2>Hasil dan Diskusi</h2>
    <p>Bagian ini menyajikan hasil yang diperoleh dan diskusi terkait hasil tersebut.</p>
</article>
```



```
1  <article>
2      <h2>Hasil dan Diskusi</h2>
3      <p>Bagian ini menyajikan hasil yang diperoleh dan diskusi terkait hasil tersebut.</p>
4  </article>
```

### 2.3.5 Elemen <footer>

Elemen <footer> digunakan untuk menampilkan bagian kaki dari halaman atau bagian tertentu dalam dokumen. Fungsi <footer>:

- Menampilkan copyright
- Menampilkan informasi kontak
- Menampilkan link tambahan atau kebijakan
- Dapat muncul di bawah artikel maupun di bawah seluruh halaman

Contoh:

```
<footer>
    <p>&copy; 2025 Kelompok 6. All rights reserved.</p>
</footer>
```



```
1 <footer>
2   <p>&copy; 2025 Kelompok 6. All rights reserved.</p>
3 </footer>
```

### 2.3.6 Peran Elemen Semantik Dalam Aksesibilitas & SEO

Penggunaan elemen semantik memberikan beberapa keuntungan signifikan, antara lain:

1. Aksesibilitas

Screen reader dapat mengidentifikasi bagian halaman dengan mudah, sehingga pengguna berkebutuhan khusus dapat menavigasi konten lebih efisien.

2. SEO

Mesin pencari seperti Google dapat memahami struktur konten dan menentukan bagian penting suatu halaman.

3. Maintainability

Kode lebih mudah dibaca dan dikelola oleh pengembang.

4. Struktur Konten yang Lebih Logis

Memudahkan pemetaan konten dalam proyek web modern.

## 2.4 Formulir dan Input Modern

Formulir merupakan bagian penting dalam pengembangan web karena memungkinkan pengguna mengirimkan data ke server. HTML menyediakan berbagai elemen formulir yang dapat digunakan untuk mengumpulkan informasi seperti nama, alamat email, nomor telepon, pilihan, hingga file. Seiring perkembangan teknologi web, elemen formulir mengalami peningkatan yang signifikan sehingga lebih interaktif, informatif, dan mudah digunakan. Formulir modern kini mendukung berbagai tipe input baru, validasi otomatis, serta atribut tambahan yang membantu meningkatkan pengalaman pengguna.

### 2.4.1 Struktur Dasar Formulir (<form>)

Elemen `<form>` merupakan wadah utama untuk semua elemen input. Form memiliki atribut penting seperti:

- **action**: URL tujuan untuk mengirim data
- **method**: metode pengiriman data (GET atau POST)
- **enctype**: tipe encoding data (untuk upload file)

Contoh:

```
<form action="#" method="post" enctype="multipart/form-data">
    ...
</form>
```



```
1 <form action="#" method="post" enctype="multipart/form-data">
2   ....
3 </form>
```

#### 2.4.2 Elemen Input Dasar (Classic Inputs)

HTML menyediakan elemen input yang sering digunakan dalam formulir, seperti:

- <input type="text"> → teks biasa
- <input type="password"> → sandi
- <input type="radio"> → pilihan tunggal
- <input type="checkbox"> → pilihan jamak
- <textarea> → teks panjang
- <select> → daftar dropdown

#### 2.4.3 Input Modern dalam HTML

HTML modern memperkenalkan berbagai tipe input baru yang membantu meningkatkan akurasi dan kemudahan pengisian data.

Contoh:

## 1. Input Email

```
<input type="email" name="email" placeholder="Masukkan  
email Anda" required/>
```



```
1 <!-- Input email -->  
2 <input type="email" name="email" placeholder="Masukkan email Anda" required>
```

Browser otomatis memeriksa format email.

## 2. Input Number

```
<input type="number" min="1" max="100" name="score"  
placeholder="Masukkan skor Anda (1-100)" required/>
```



```
1 <!-- Input nomor skor antara 1-100 -->  
2 <input type="number" min="1" max="100" name="score" placeholder="Masukkan skor Anda (1-100)" required>
```

Memungkinkan pembatasan nilai.

## 3. Input Date / Time

```
<input type="date" name="date" required/>  
<input type="time" name="time" required/>  
<input type="datetime-local" name="datetime" required/>
```



```
1 <!-- Input tanggal, waktu, dan datetime -->  
2 <input type="date" name="date" required/>  
3 <input type="time" name="time" required/>  
4 <input type="datetime-local" name="datetime" required/>
```

memungkinkan pengguna memilih tanggal, waktu,

## 4. Input URL

```
<input type="url" name="website"  
placeholder="https://example.com" required/>
```



```
1 <!-- Input URL -->
2 <input type="url" name="website" placeholder="https://example.com" required/>
```

Validasi format URL otomatis.

## 5. Input Color

```
<input type="color" name="favcolor" value="#000000"
required/>
```



```
1 <!-- Input warna -->
2 <input type="color" name="favcolor" value="#000000" required/>
```

Memunculkan color picker.

## 6. Input Range

```
<input type="range" name="range" min="0" max="100"
required/>
```



```
1 <!-- Input range -->
2 <input type="range" name="range" min="0" max="100" required/>
```

Menggunakan slider untuk memilih nilai.

## 7. Input File

```
<input type="file" name="fileupload" required/>
```



```
1 <!-- Input file upload -->
2 <input type="file" name="fileupload" required/>
```

Untuk mengunggah dokumen atau gambar.

#### 2.4.4 Atribut Modern Pada Input

HTML modern juga membawa atribut tambahan untuk membuat formulir lebih kuat dan aman.

##### 1. Required

Memastikan input wajib diisi.

```
<input type="text" required/>
```



```
1 <input type="text" required/>
```

##### 2. Placeholder

Teks bantuan dalam kotak input.

```
<input type="text" placeholder="Masukan Nama"/>
```



```
1 <input type="text" placeholder="Masukan Nama"/>
```

##### 3. Pattern

Validasi berdasarkan pola regex.

```
<input type="text" pattern="[a-zA-Z0-9]" />
```



```
1 <input type="text" pattern="[a-zA-Z0-9]" />
```

##### 4. Autocomplete

Mengaktifkan atau menonaktifkan fitur isi otomatis.

```
<input type="email" autocomplete="on"/>
```



```
1 <input type="email" autocomplete="on"/>
```

##### 5. Readonly dan Disabled

- readonly → hanya bisa dibaca

- disabled → tidak bisa diakses

```
<input type="text" readonly/>
<input type="text" disabled/>
```



```
1 <input type="text" readonly/>
2 <input type="text" disabled/>
```

## 6. Multiple

Untuk memilih lebih dari satu file.

```
<input type="file" multiple
```



```
1 <input type="file" multiple
```

### 2.4.5 Elemen Pendukung Formulir

Beberapa elemen penting lainnya antara lain:

#### 5. <label>

Menjelaskan input.

```
<label for="name">Nama:</label>
<input type="text" id="name" name="nama" required/>
```



```
1 <!-- Label -->
2 <label for="name">Nama:</label>
3 <input type="text" id="name" name="nama" required/>
```

#### 6. <fieldset> dan <legend>

Mengelompokkan input.

```
<fieldset>
    <legend>Informasi Tambahan</legend>
    <label for="address">Alamat:</label>
```

```
<input type="text" id="address" name="alamat"  
required/>  
</fieldset>
```



```
1 <!-- Fieldset dan Legend -->  
2 <fieldset>  
3   <legend>Informasi Tambahan</legend>  
4   <label for="address">Alamat:</label>  
5   <input type="text" id="address" name="alamat" required/><br />  
6 </fieldset>
```

## 7. <button>

Tombol untuk aksi.

```
<button type="submit">Kirim</button>  
<button type="reset">Reset</button>
```



```
1 <!-- Button -->  
2 <button type="submit">Kirim</button>  
3 <button type="reset">Reset</button>
```

### 2.4.6 Validasi Formulir Secara Otomatis

HTML modern mendukung *built-in validation* tanpa JavaScript:

- Format email, URL, number
- Input wajib (required)
- Batas min–max
- Pola tertentu (pattern)

Browser akan otomatis menampilkan pesan jika input tidak sesuai.

### 2.4.7 Contoh Formulir Modern Lengkap

**Input:**

```
<form action="#" method="post" enctype="multipart/form-data">  
  
<h2>Formulir Modern Lengkap</h2>  
<label for="fullname">Nama Lengkap:</label>
```

```

<input type="text" id="fullname" name="fullname"
placeholder="Masukkan nama lengkap" required/><br />

<label for="emailmod">Email:</label>
<input type="email" id="emailmod" name="emailmod"
placeholder="Masukkan email" required/><br />

<label for="password">Kata Sandi:</label>
<input type="password" id="password"
name="password" placeholder="Masukkan kata sandi"
required/><br />

<label for="birthdate">Tanggal Lahir:</label>
<input type="date" id="birthdate" name="birthdate"
required/><br />

<label for="profilepic">Foto Profil:</label>
<input type="file" id="profilepic"
name="profilepic" accept="image/*" required/><br />

<button type="submit">Submit</button>
<button type="reset">Reset</button>
</form>

```

● ● ●

```

1 <!-- Contoh formulir modern Lengkap -->
2 <form action="#" method="post" enctype="multipart/form-data">
3 <h2>Formulir Modern Lengkap</h2>
4 <label for="fullname">Nama Lengkap:</label>
5 <input type="text" id="fullname" name="fullname" placeholder="Masukkan nama lengkap" required/><br />
6
7 <label for="emailmod">Email:</label>
8 <input type="email" id="emailmod" name="emailmod" placeholder="Masukkan email" required/><br />
9
10 <label for="password">Kata Sandi:</label>
11 <input type="password" id="password" name="password" placeholder="Masukkan kata sandi" required/><br />
12
13 <label for="birthdate">Tanggal Lahir:</label>
14 <input type="date" id="birthdate" name="birthdate" required/><br />
15
16 <label for="profilepic">Foto Profil:</label>
17 <input type="file" id="profilepic" name="profilepic" accept="image/*" required/><br />
18
19 <button type="submit">Submit</button>
20 <button type="reset">Reset</button>
21 </form>

```

**Output:**

## **Formulir Modern Lengkap**

**Nama Lengkap:**

**Email:**

**Kata Sandi:**

**Tanggal Lahir:**

**Foto Profil:**  Tidak ada file yang dipilih

### **2.5 Multimedia (Audio, Video, Canvas)**

Multimedia dalam pemrograman web mengacu pada penggunaan elemen-elemen seperti suara, video, dan grafis dinamis untuk memperkaya pengalaman pengguna. HTML5 menyediakan elemen bawaan yang memungkinkan pengembang menampilkan serta mengontrol media tanpa memerlukan plugin tambahan. Tiga elemen multimedia utama dalam HTML5 adalah <audio>, <video>, dan <canvas>, masing-masing memiliki fungsi dan karakteristik teknis tersendiri.

#### **2.5.1 Elemen Audio**

Elemen <audio> digunakan untuk memutar suara secara langsung pada halaman web. HTML5 menyediakan atribut seperti controls untuk menampilkan tombol play/pause, autoplay untuk memulai otomatis, serta loop untuk memutar berulang. File audio umumnya menggunakan format seperti .mp3, .wav, dan .ogg.

Selain itu, JavaScript memungkinkan kontrol lebih lanjut seperti mengatur volume, durasi, dan event audio (misalnya saat audio selesai diputar). Elemen ini banyak digunakan dalam website edukasi, podcast, game berbasis web, serta lingkungan interaktif lainnya.

Contoh:

```
<audio controls>
    <source src="audio/sample.mp3" type="audio/mpeg">
</audio>
```

```
1 <!-- Audio -->
2 <audio controls>
3   <source src="audio/sample.mp3" type="audio/mpeg">
4 </audio>
```

### 2.5.2 Elemen Video

Elemen `<video>` berfungsi menampilkan dan memutar video dalam halaman web tanpa perlu plugin eksternal seperti Flash. HTML5 mendukung atribut seperti `controls`, `autoplay`, `width`, `height`, `loop`, dan `poster` (gambar sebelum video diputar).

Format umum video antara lain .mp4, webm, dan .ogg. Dengan bantuan JavaScript, pengembang dapat membuat fitur tambahan seperti custom player, pengaturan kecepatan playback, serta integrasi dengan API streaming.

Contoh:

```
<video controls width="300">
  <source src="video/sample.mp4" type="video/mp4">
</video>
```

```
1 <!-- Video -->
2 <video controls width="300">
3   <source src="video/sample.mp4" type="video/mp4">
4 </video>
```

### 2.5.3 Elemen Canvas

Elemen `<canvas>` memungkinkan pengembang menggambar grafik secara dinamis menggunakan JavaScript. Canvas bekerja dengan sistem gambar berbasis pixel (bitmap) dan mendukung pembuatan bentuk 2D maupun animasi. Elemen ini sangat populer dalam pembuatan grafik data, game 2D, simulasi, efek visual, dan editor gambar berbasis web.

Semua gambar dalam canvas tidak bersifat DOM element, melainkan dirender langsung ke area gambar sehingga manipulasi grafis menjadi lebih cepat dan fleksibel.

Contoh:

```
<canvas id="myCanvas" width="300"
height="200"></canvas>

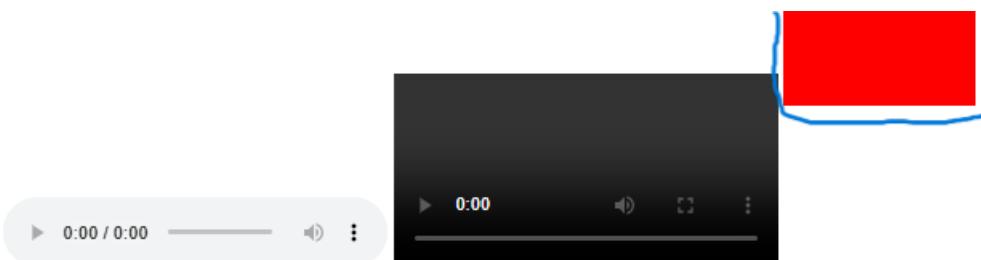
<script>
    const canvas = document.getElementById('myCanvas');
    const ctx = canvas.getContext('2d');
    ctx.fillStyle = '#FF0000';
    ctx.fillRect(0, 0, 150, 75);
</script>
```



The screenshot shows a browser window with three window control buttons (red, yellow, green) at the top. The main content area displays a black rectangular background with a red rectangle centered on it. The red rectangle has a bounding box of approximately [150, 75] pixels. A blue hand-drawn bracket highlights the red rectangle.

```
1  <!-- Canvas + JavaScript -->
2  <canvas id="myCanvas" width="300" height="200"></canvas>
3
4  <script>
5      const canvas = document.getElementById('myCanvas');
6      const ctx = canvas.getContext('2d');
7      ctx.fillStyle = '#FF0000';
8      ctx.fillRect(0, 0, 150, 75);
9  </script>
```

Output:



## BAB III

### PENGENALAN DASAR CSS3

#### 3.1 Konsep CSS (Inline, Internal, External Stylesheet)

Cascading Style Sheets (CSS) adalah bahasa pemrograman yang digunakan untuk mengatur tampilan dan gaya pada halaman web. CSS3 merupakan versi terbaru yang menawarkan fitur lebih lengkap seperti transition, transform, animation, dan fitur modular yang memudahkan pengembangan desain modern. CSS memungkinkan pemisahan antara struktur dokumen HTML dan desain visual, sehingga mempermudah pengelolaan, pemeliharaan, serta konsistensi tampilan pada suatu website.

Dengan CSS3, pengembang dapat mengontrol layout, warna, tipografi, jarak, hingga efek animasi yang memperkaya pengalaman pengguna. CSS dapat diterapkan pada dokumen HTML dengan tiga cara utama: inline, internal, dan external stylesheet. Ketiga metode ini memiliki karakteristik dan kegunaan masing-masing sesuai kebutuhan pengembangan.

#### 3.1.1 CSS Inline

CSS inline adalah metode penulisan CSS langsung pada elemen HTML menggunakan atribut `style`. Cara ini hanya memengaruhi elemen tempat CSS diterapkan. Inline style biasanya digunakan untuk perubahan cepat atau penyesuaian khusus pada satu elemen.

Contoh:

```
<p style="color: blue; font-size: 20px;">Ini adalah  
teks berwarna biru dengan gaya inline CSS.</p>
```



The screenshot shows a code editor window with three colored status icons (red, yellow, green) at the top. Below them, the code for a paragraph element is displayed. The first line is a comment: `1 //-- Css Inline -->`. The second line contains the HTML element with inline styles: `2 <p style="color: blue; font-size: 20px;">Ini adalah teks berwarna biru dengan gaya inline CSS.</p>`.

##### 1. Kelebihan:

- Mudah diterapkan untuk perubahan kecil.
- Tidak memerlukan file tambahan.

2. Kekurangan:

- Tidak efisien jika digunakan pada banyak elemen.
- Menyulitkan pemeliharaan karena gaya bercampur dengan struktur HTML.
- Tidak mendukung konsep pemisahan konten dan presentasi.

### 3.1.2 CSS Internal

CSS internal ditulis dalam tag `<style>` di dalam bagian `<head>` pada dokumen HTML. Metode ini digunakan ketika style hanya diperlukan untuk satu halaman saja.

Contoh:

```
<style>
p {
    color: green;
    font-size: 18px;
}
</style>
```



The screenshot shows a code editor window with a light gray background. At the top left, there are three colored circular icons: red, yellow, and green. Below them, the code is displayed in a monospaced font. The code consists of seven numbered lines, each starting with a number from 1 to 7 followed by a space and a code snippet. Lines 1 and 2 are comments indicating the start of internal CSS. Lines 3 through 6 define a style for the 'p' element, setting its color to green and font size to 18px. Line 7 marks the end of the style block.

```
1  <!-- Css Internal -->
2  <style>
3      p {
4          color: green;
5          font-size: 18px;
6      }
7  </style>
```

1. Kelebihan:

- Tidak memerlukan file luar.
- Mudah digunakan untuk halaman tunggal.
- Struktur tetap lebih bersih dibanding inline.

2. Kekurangan:

- Tidak efektif untuk banyak halaman.

- Memperbesar ukuran file HTML jika style terlalu banyak.
- Style tidak dapat digunakan ulang pada halaman lain.

### 3.1.3 CSS External

CSS external menggunakan file .css yang diautakan ke dokumen HTML dengan tag <link>. Metode ini adalah yang paling umum digunakan pada proyek web profesional karena mendukung pemisahan konten dan tampilan.

Contoh:

Link file style.css ke file .html:

```
<link rel="stylesheet" href="Makalah_UTS.css"/>
```



```
1  <!-- Link External CSS di HTML -->
2  <link rel="stylesheet" href="Makalah_UTS.css" />
```

Isi file style.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 20px;
    background-color: #f4f4f4;
}

h1 {
    color: #333;
    font-size: 24px;
}
```

```
1  /* Isi file CSS Makalah_UTS */
2  body {
3      font-family: Arial, sans-serif;
4      margin: 20px;
5      background-color: #f4f4f4;
6  }
7  h1 {
8      color: #333;
9      font-size: 24px;
10 }
```

### 1. Kelebihan:

- Style dapat digunakan ulang pada banyak halaman.
- Struktur HTML lebih bersih.
- Pengelolaan dan pemeliharaan lebih mudah.
- Ukuran file HTML menjadi lebih ringan.

### 2. Kekurangan:

- Membutuhkan file tambahan.
- Tampilan mungkin tidak langsung muncul jika terjadi gangguan koneksi atau file CSS gagal dimuat.

## 3.2 Selektor, Properti, dan Nilai

Dalam CSS, tampilan elemen HTML diatur melalui kombinasi antara selektor, properti, dan nilai. Ketiga komponen ini merupakan dasar dalam penulisan aturan CSS. Selektor menentukan elemen mana yang akan diberi gaya, properti menentukan aspek visual yang akan diubah, dan nilai menjelaskan pengaturan atau konfigurasi pada properti tersebut. Pemahaman ketiga konsep ini menjadi fondasi utama dalam perancangan antarmuka web yang efektif, konsisten, dan terstruktur.

### 3.2.1 Selektor

Selektor adalah pola yang digunakan untuk memilih elemen HTML yang akan diberi style. CSS menyediakan berbagai jenis selektor untuk kebutuhan yang beragam.

## 1. Selektor Elemen (Element Selector)

Memilih elemen berdasarkan nama tag HTML. Contoh:

```
p {  
    color: #666;  
    line-height: 1.6;  
}
```



## 2. Selektor Class

Menggunakan atribut class pada HTML. Ditandai dengan tanda titik (.). Contoh:

```
.highlight {  
    background-color: yellow;  
}
```



## 3. Selektor ID

Menggunakan atribut id. Ditandai dengan tanda pagar (#). Setiap id hanya boleh digunakan satu kali per halaman. Contoh:

```
#main-content {  
    margin: 20px;  
}
```



```
1 /* Contoh Selector ID */
2 #main-content {
3     margin: 20px;
4 }
```

#### 4. Selektor Atribut

Memilih elemen berdasarkan nilai atau keberadaan atribut tertentu.

Contoh:

```
input[type="text"] {
    border: 1px solid #ccc;
    padding: 5px;
}
```



```
1 /* Contoh Selector Atribut */
2 input[type="text"] {
3     border: 1px solid #ccc;
4     padding: 5px;
5 }
```

#### 5. Selektor Descendant (Turunan)

Digunakan untuk menargetkan elemen yang berada di dalam elemen lain. Contoh:

```
nav ul li {
    list-style: none;
    display: inline;
    margin-right: 10px;
}
```



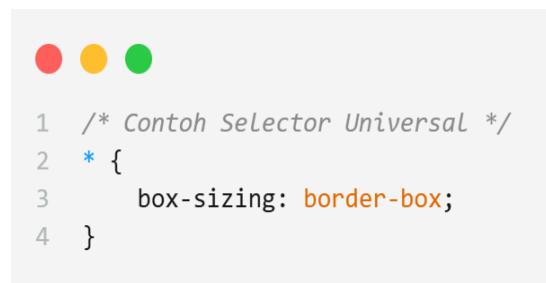
```
1 /* Contoh Selector Descendant*/
2 nav ul li {
3     list-style: none;
4     display: inline;
5     margin-right: 10px;
6 }
```

## 6. Selektor Universal

Menargetkan semua elemen HTML menggunakan tanda bintang \*.

Contoh:

```
* {  
    box-sizing: border-box;  
}
```



### 3.2.2 Properti (Property)

Properti adalah aspek tampilan yang ingin diatur pada elemen HTML. CSS memiliki ratusan properti, namun beberapa kategori utama yang sering digunakan antara lain:

#### a. Properti Warna dan Latar Belakang

- Color
- background-color
- background-image

#### b. Properti Teks dan Tipografi

- font-size
- font-family
- text-align
- font-weight

#### c. Properti Layout dan Box Model

- Margin
- Padding
- Border
- width, height

d. Properti Tampilan (Display)

- display: block;
- display: inline;
- display: flex;

e. Properti Visual Modern

- box-shadow
- border-radius
- opacity

### 3.2.3 Nilai (Value)

Nilai merupakan isi atau konfigurasi dari sebuah properti. Nilai dapat berupa angka, warna, kata kunci, atau kombinasi tertentu. Contoh jenis nilai:

| Jenis Nilai    | Contoh                      | Keterangan                         |
|----------------|-----------------------------|------------------------------------|
| Warna          | red, #ff0000, rgb (255,0,0) | Mendefinisikan warna               |
| Satuan panjang | px, em, rem, %              | Mengatur ukuran                    |
| Kata kunci     | center, bold, flex          | Nilai yang telah ditentukan CSS    |
| URL            | url("image.jpg")            | Biasanya untuk gambar latar        |
| Angka          | 1, 0.5                      | Dipakai pada opacity, z-index, dll |

### 3.2.4 Contoh Penggunaan Selektor, Properti, dan Nilai

```
p.intro{  
    font-size: 18px;  
    color: #333;  
    line-height: 1.6;  
}
```

```
1  /* Contoh penggunaan Selector, property, dan nilai */
2  p.intro{
3      font-size: 18px;
4      color: #333;
5      line-height: 1.6;
6  }
```

Penjelasan:

- p.intro → selector
- font-size, color, line-height → property
- 18px, #333, 1.6 → nilai

### 3.3 Warna, Teks, dan Font

Dalam perancangan antarmuka web, pengaturan warna, teks, dan font berperan penting dalam meningkatkan estetika, keterbacaan, dan pengalaman pengguna. CSS menyediakan berbagai properti yang dapat digunakan untuk mengontrol aspek visual dari elemen teks, mulai dari warna tulisan, jarak antarkarakter, hingga pemilihan jenis dan ukuran font. Pemahaman dasar mengenai properti-properti ini menjadi kemampuan fundamental bagi seorang pengembang web, terutama dalam membangun halaman yang menarik sekaligus mudah dipahami.

#### 3.3.1 Pengaturan Warna (Color)

Warna merupakan elemen visual penting dalam desain web. CSS menyediakan beberapa format nilai warna yang dapat digunakan sesuai kebutuhan dan preferensi perancang.

- a) Format Penulisan Warna
  1. Nama warna standar (color names)



```
1 color: red;
```

## 2. Hexadecimal

Nilai 6 digit atau 3 digit.



```
1 color: #323232;
```

## 3. RGB (Red, Green, Blue)



```
1 color: rgb(51, 51, 51);
```

## 4. RGBA (RGB + Alpha)

Menambahkan tingkat transparansi.



```
1 color: rgba(51, 51, 51, 1);
```

## 5. HSL (Hue, Saturation, Lightness)



```
1 color: hsl(240, 100%, 50%);
```

### b) Properti Warna yang Sering Digunakan

- color → warna teks
- background-color → warna latar belakang
- border-color → warna garis tepi.

### 3.3.2 Pengaturan Teks (Text Properties)

Properti teks digunakan untuk mengatur tampilan paragraf, heading, dan elemen tekstual lainnya.

#### 1. text-align

Mengatur perataan teks:

```
● ● ●  
1 text-align: center;  
2 text-align: justify;  
3 text-align: left;  
4 text-align: right;
```

#### 2. text-decoration

Menambah garis seperti underline atau strikethrough.

```
● ● ●  
1 text-decoration: underline;  
2 text-decoration: line-through;  
3 text-decoration: overline;  
4 text-decoration: none;
```

#### 3. text-transform

Mengatur kapitalisasi:

```
● ● ●  
1 text-transform: uppercase;  
2 text-transform: lowercase;  
3 text-transform: capitalize;  
4 text-transform: none;
```

#### 4. line-height

Mengatur jarak antarbaris:

```
● ● ●  
1 line-height: 1.5;
```

#### 5. letter-spacing & word-spacing

Mengatur jarak antar huruf dan kata:



```
1 letter-spacing: 2px;  
2 word-spacing: 5px;
```

## 6. text-shadow

Membuat efek bayangan teks:



```
1 text-shadow: gray 1px 1px 2px;
```

### 3.3.3 Pengaturan Font (Font Properties)

CSS menyediakan berbagai properti untuk mengontrol tipografi sehingga teks lebih menarik dan mudah dibaca.

#### 1. font-family

Menentukan jenis font yang digunakan. Disarankan menuliskan *fallback* font.



```
1 font-family: Arial, sans-serif;
```

#### 2. font-size

Mengatur ukuran teks. Satuan yang sering digunakan:

- px (pixel)
- em (relatif terhadap ukuran font elemen induk)
- rem (relatif terhadap root/HTML)
- %

Contoh:



```
1 font-size: 24px;  
2 font-size: 24rem;  
3 font-size: 24em;
```

### 3. font-weight

Mengatur ketebalan teks:



```
1  font-weight: 600;
2  font-weight: bold;
3  font-weight: normal;
```

### 4. font-style

Mengatur gaya font:



```
1  font-style: italic;
```

### 5. font-variant

Digunakan untuk small-caps:



```
1  font-variant: small-caps;
```

### 6. @font-face

Mengimport font custom lokal maupun eksternal. Contoh:



```
1  @font-face {
2      font-family: 'CustomFont';
3      src: url('fonts/CustomFont.woff2') format('woff2');
4  }
```

### 7. Menggunakan Google Fonts

Google Fonts umum digunakan di banyak projek mahasiswa.

Contoh di HTML:



```
1  <!-- Menggunakan font dari google fonts -->
2  <link rel="preconnect" href="https://fonts.googleapis.com">
3  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
```

Di CSS:

```
● ○ ●  
1 /* Menggunakan font dari google fonts */  
2 body {  
3   font-family: 'CustomFont', Arial, sans-serif;  
4 }  
5
```

### 3.3.4 Contoh Implementasi Warna, Teks, dan Font

```
body {  
  color: #444;  
  background-color: #e0e0e0;  
  font-family: 'Roboto', sans-serif;  
}  
  
h1, h2, h3 {  
  color: #222;  
  text-align: center;  
  text-transform: capitalize;  
}  
  
p {  
  font-size: 16px;  
  line-height: 1.6;  
  text-align: justify;  
}
```

```
● ○ ●  
1 /* Contoh Implementasi warna, teks, dan font */  
2 body {  
3   color: #444;  
4   background-color: #e0e0e0;  
5   font-family: 'Roboto', sans-serif;  
6 }  
7 h1, h2, h3 {  
8   color: #222;  
9   text-align: center;  
10  text-transform: capitalize;  
11 }  
12 p {  
13   font-size: 16px;  
14   line-height: 1.6;  
15   text-align: justify;  
16 }
```

## Output:

The screenshot shows a web page with the title "Makalah UTS Kelompok 6". The page includes a navigation menu with links to "Belajar HTML dan CSS", "Pendahuluan", "Metodologi", and "Cover Image". Below the menu, there are sections for "Pendahuluan", "Metodologi", and "Hasil Dan Diskusi". A "Formulir Modern Lengkap" section contains a form with fields for Name, Email, Password, Birth Date, and Profile Photo, along with "Submit" and "Reset" buttons. A video player is also present. The page footer includes a copyright notice: "© 2025 Kelompok 6. All rights reserved.".

### 3.4 Box Model (Margin, Padding, Border)

Dalam CSS, seluruh elemen pada halaman web direpresentasikan menggunakan *box model*. Konsep ini sangat penting karena menentukan bagaimana suatu elemen ditampilkan, diatur jaraknya, serta berinteraksi dengan elemen lainnya di dalam layout halaman. Box model terdiri atas empat bagian utama: content, padding, border, dan margin. Pemahaman struktur ini membantu pengembang mengontrol ruang, ukuran, serta tata letak elemen secara lebih efektif.

#### 3.4.1 Konsep Dasar Box Model

Secara visual, *CSS Box Model* dapat dijelaskan sebagai lapisan-lapisan yang menyelimuti konten

Komponen utama:

##### 1. Content

Isi dari elemen, seperti teks, gambar, atau komponen lain.

##### 2. Padding

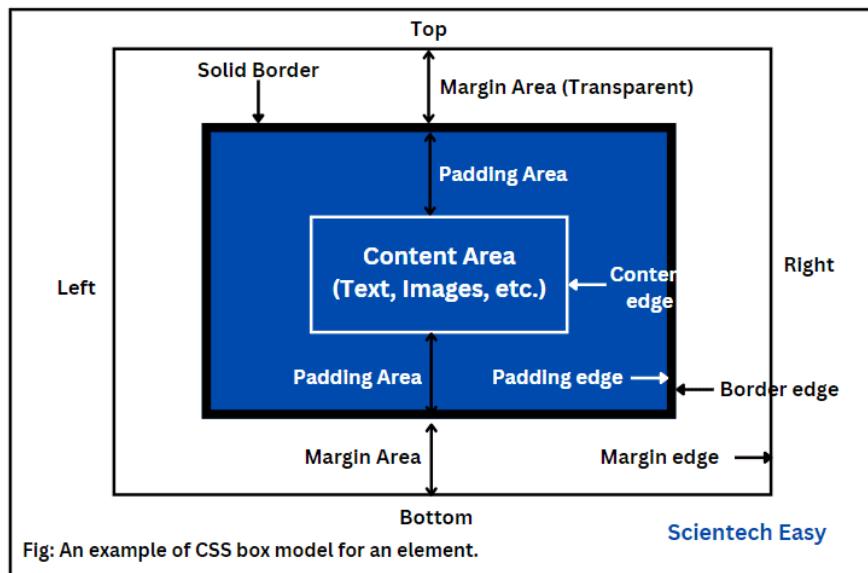
Jarak antara content dan border memberikan ruang dalam elemen.

##### 3. Border

Garis yang mengelilingi padding dan content.

#### 4. Margin

Ruang luar yang memisahkan elemen dari elemen lainnya.



#### 3.4.2 Padding

Padding menambah ruang di dalam sebuah elemen—antara *content* dan *border*. Properti ini tidak mempengaruhi elemen lain secara langsung, tetapi menambah ukuran keseluruhan elemen. Contoh:

```
P {  
    padding: 10px 15px;  
}
```

```
● ● ●  
1 /* Padding */  
2 p {  
3     padding: 10px 15px;  
4 }
```

Bentuk penulisan shorthand:

- padding: 10px; → semua sisi
- padding: 10px 20px; → vertical horizontal
- padding: 10px 20px 30px; → top horizontal bottom

- padding: 10px 20px 30px 40px; → top right bottom left

### 3.4.3 Border

Border adalah garis tepi yang mengelilingi elemen. Border memiliki tiga komponen: width, style, dan color. Contoh:

```
P {
    border: 1px solid #ccc;
}
```



```
1 /* Border */
2 p {
3     border: 1px solid #ccc;
4 }
```



```
1 border: 1px solid #ccc;
2 border-radius: 5px;
3 border-width: 2px;
4 border-color: #888;
5 border-style: dashed;
```

Macam-macam border-style:

- solid
- dashed
- dotted
- double
- ridge
- groove
- none

Border-radius

Untuk membuat sudut melengkung:



```
1 border-radius: 5px;
```

### 3.4.4 Margin

Margin adalah ruang di luar border, digunakan untuk memberi jarak antar elemen. Margin dapat bernilai positif maupun negatif. Contoh:

```
P {  
    margin: 10px 15px;  
}
```



```
1 /* Margin */  
2 p {  
3     margin: 10px 15px;  
4 }
```

Bentuk shorthand:

- margin: 10px;
- margin: 10px 20px;
- margin: 10px 20px 30px;
- margin: 10px 20px 30px 40px;

Auto margin → sering untuk *centering*:

```
P {  
    margin: 0 auto;  
    width: 80%;  
}
```



```
1 /* Auto margin */
2 p {
3     margin: 0 auto;
4     width: 80%;
5 }
```

### 3.4.5 Perhitungan Ukuran Elemen (Element Size Calculation)

Secara default, width dan height hanya menghitung *content*, tidak termasuk padding, border, dan margin. Rumus:



```
1 /* Perhitungan ukuran elemen:
2 Total Width = width + padding-left + padding-right + border-left + border-right
3 Total Height = height + padding-top + padding-bottom + border-top + border-bottom
4 */
```

Menggunakan box-sizing: border-box;. Mengubah perhitungan sehingga padding dan border sudah termasuk ke dalam width dan height.

```
* {
    box-sizing: border-box;
}
```

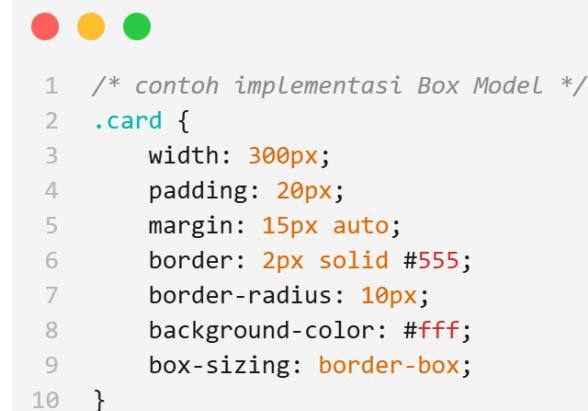


```
1 /* Box-sizing */
2 * {
3     box-sizing: border-box;
4 }
```

Ini sangat umum digunakan pada desain modern karena memudahkan pengaturan layout.

### 3.4.6 Contoh Implementasi Box Model

```
.card {  
    width: 300px;  
    padding: 20px;  
    margin: 15px auto;  
    border: 2px solid #555;  
    border-radius: 10px;  
    background-color: #fff;  
    box-sizing: border-box;  
}
```



```
1 /* contoh implementasi Box Model */  
2 .card {  
3     width: 300px;  
4     padding: 20px;  
5     margin: 15px auto;  
6     border: 2px solid #555;  
7     border-radius: 10px;  
8     background-color: #fff;  
9     box-sizing: border-box;  
10 }
```

Pada contoh di atas:

- Padding → memberi ruang dalam
- Margin → memberi jarak antar elemen
- Border → memberi batas visual
- Box-sizing → memudahkan perhitungan ukuran

## 3.5 Layout Dasar dengan Flexbox & Grid CSS

Dalam pengembangan antarmuka web modern, pengaturan tata letak (layout) menjadi faktor penting untuk memastikan tampilan halaman yang responsif, rapi, dan mudah dikelola. CSS menyediakan dua teknik layout yang sangat efektif dan kini menjadi standar industri, yaitu Flexbox dan Grid Layout.

Keduanya dirancang untuk mempermudah proses penyusunan komponen halaman agar lebih fleksibel dan dapat beradaptasi dengan berbagai ukuran layar perangkat.

### 3.5.1 Flexbox (Flexible Box Layout)

Flexbox merupakan model layout satu dimensi yang digunakan untuk mengatur elemen dalam satu baris (row) atau kolom (column). Flexbox mempermudah penataan elemen yang sebelumnya sulit dilakukan dengan float atau inline-block, terutama ketika diperlukan distribusi ruang yang dinamis dan rapi.

#### A. Konsep Dasar Flexbox

Flexbox bekerja melalui dua komponen utama:

- Flex Container = elemen pembungkus, diberi properti `display: flex;`
- Flex Items = elemen-elemen di dalam container yang terkena pengaturan flex.

#### B. Properti Utama Flex Container

##### 1. `display: flex`

Mengaktifkan mode Flexbox.

```
.container {  
    display: flex;  
}
```



```
1  /* Properti flex container */  
2  .container {  
3      display: flex;  
4  }
```

##### 2. `flex-direction`

Menentukan arah elemen (horizontal/vertical).

```
.container {  
    flex-direction: row;  
    flex-direction: column;  
}
```



```
1 flex-direction: row;  
2 flex-direction: column;
```

### 3. justify-content

Mengatur perataan elemen secara horizontal (main axis).

Contoh:

- flex-start
- center
- flex-end
- space-between
- space-around

```
.container {  
    justify-content: space-around;  
}
```



```
1 justify-content: space-around;
```

### 4. align-items

Mengatur perataan vertikal (cross axis).

```
.container {  
    align-items: center;  
}
```



```
1 align-items: center;
```

### 5. flex-wrap

Mengatur apakah elemen akan turun ke baris berikutnya.

```
.container {  
    flex-wrap: wrap;  
}
```



### C. Properti Flex Item

- **flex-grow**

Mengatur apakah elemen boleh melebar

```
.item {  
    flex-grow: 1;  
}
```



- **flex-shrink**

Mengatur apakah elemen boleh menyusut.

```
.item {  
    flex-shrink: 1;  
}
```



- **flex-basis**

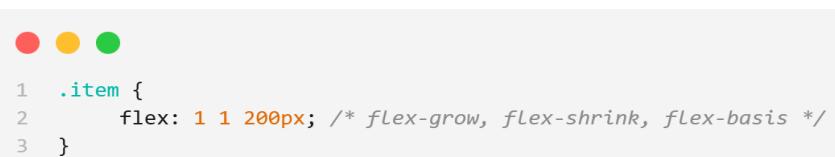
Menentukan ukuran awal item.

```
.item {  
    flex-basis: 30%;  
}
```



- Shorthand: flex

```
.item {  
    flex: 1 1 200px;  
}
```



#### D. Contoh Implementasi Flexbox

```
.container {  
    display: flex;  
    flex-direction: row;  
    justify-content: space-between;  
    align-items: center;  
    padding: 20px;  
}  
  
.box {  
    width: 150px;  
    height: 150px;  
    background-color: #4CAF50;  
    color: white;  
    display: flex;  
    justify-content: center;
```

```
    align-items: center;  
    margin: 10px;  
}
```



```
1  /* Contoh Implementasi Flexbox */  
2  .container {  
3      display: flex;  
4      flex-direction: row;  
5      justify-content: space-between;  
6      align-items: center;  
7      padding: 20px;  
8  }  
9  .box {  
10     width: 150px;  
11     height: 150px;  
12     background-color: #4CAF50;  
13     color: white;  
14     display: flex;  
15     justify-content: center;  
16     align-items: center;  
17     margin: 10px;  
18 }
```

### 3.5.2 CSS Grid Layout

CSS Grid adalah sistem layout dua dimensi yang memungkinkan pengaturan elemen berdasarkan baris (rows) dan kolom (columns). Ini membuat Grid sangat cocok digunakan untuk membuat template halaman web seperti dashboard, galeri, atau layout kompleks lainnya.

#### a) Konsep Dasar Grid

Grid menggunakan dua elemen utama:

1. Grid Container → diberi *display: grid;*
2. Grid Items → elemen - elemen di dalam grid.

#### b) Properti Utama Grid Container

1. *grid-template-columns & grid-template-rows*

Menentukan jumlah dan ukuran kolom serta baris grid.

```
.grid-container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: auto;  
}
```



```
1 .grid-container {  
2     display: grid;  
3     grid-template-columns: repeat(3, 1fr);  
4     grid-template-rows: auto;  
5 }
```

## 2. gap

Jarak antar-elemen grid.

```
.grid-container {  
    display: grid;  
    gap: 20px;  
}
```



```
1 .grid-container {  
2     display: grid;  
3     gap: 20px;  
4 }
```

## 3. justify-items & align-items

Mengatur perataan isi dalam setiap grid cell.

```
.grid-container {  
    display: grid;  
    justify-items: center;  
    align-items: center;  
}
```



```
1 .grid-container {  
2     display: grid;  
3     justify-items: center;  
4     align-items: center;  
5 }
```

#### 4. grid-template-areas

Membuat layout visual yang lebih kompleks dan mudah dipahami.

```
.grid-container {  
    display: grid;  
    grid-template-areas:  
        "header header header"  
        "sidebar main aside"  
        "footer footer footer";  
}
```



```
1 .grid-container {  
2     display: grid;  
3     grid-template-areas:  
4         "header header header"  
5         "sidebar main aside"  
6         "footer footer footer";  
7 }
```

#### c) Properti Grid Item

- grid-column & grid-row

Menentukan posisi item dalam grid.

```
.item1 {  
    grid-column: 1 / 3; /* Dari kolom 1 sampai 3 */  
    grid-row: 1 / 2; /* Dari baris 1 sampai 2 */  
}
```



```
1 /* Properti Grid Item */
2 .item1 {
3     grid-column: 1 / 3; /* Dari kolom 1 sampai 3 */
4     grid-row: 1 / 2;    /* Dari baris 1 sampai 2 */
5 }
```

- place-self

Mengatur perataan item individual.

```
.item1 {
    grid-column: 1 / 3; /* Dari kolom 1 sampai 3 */
    place-self: center;
}
```



```
1 .item1 {
2     grid-column: 1 / 3; /* Dari kolom 1 sampai 3 */
3     place-self: center;
4 }
5
```

#### d) Contoh Implementasi Grid

```
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
    padding:
}

.grid-item {
    background-color: #2196F3;
    color: white;
    border: 1px solid #000;
    padding: 20px;
    font-size: 30px;
    text-align: center;
}
```

```

1  /* Contoh Implementasi Grid */
2  .grid-container {
3      display: grid;
4      grid-template-columns: repeat(3, 1fr);
5      gap: 10px;
6      padding: 10px;
7  }
8  .grid-item {
9      background-color: #2196F3;
10     color: white;
11     border: 1px solid #000;
12     padding: 20px;
13     font-size: 30px;
14     text-align: center;
15 }

```

### 3.5.3 Perbandingan Flexbox dan Grid

| Aspek            | Flexbox                        | Grid                                   |
|------------------|--------------------------------|--|
| Dimensi          | Satu dimensi (row/column)      | Dua dimensi (row & column)             |
| Kegunaan         | Menu, navbar, card horizontal  | Layout halaman, galeri, dashboard      |
| Kemudahan        | Sederhana, fleksibel           | Lebih kompleks namun lebih terstruktur |
| Distribusi ruang | Sangat baik untuk item dinamis | Sangat baik untuk layout terencana     |

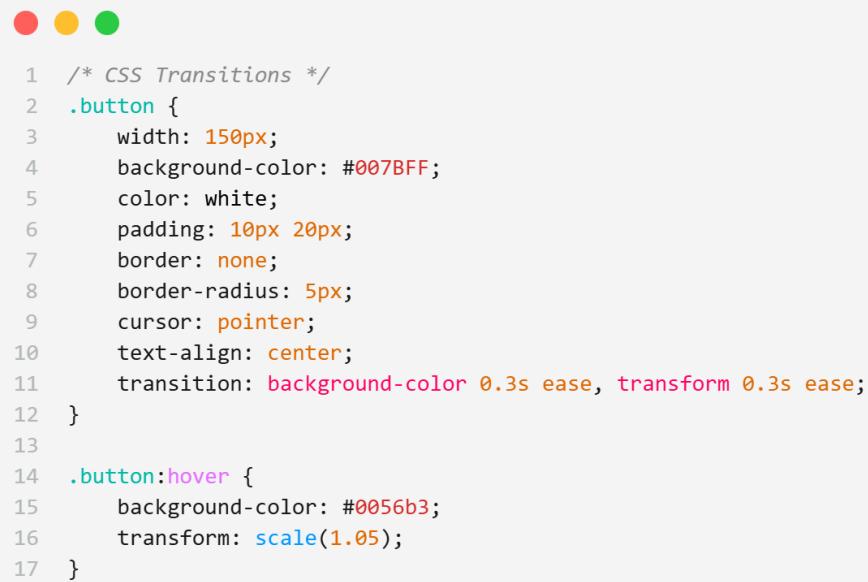
## 3.6 Efek Visual (Transisi, Transformasi, Animasi Sederhana)

CSS3 menyediakan berbagai efek visual yang membuat tampilan website lebih menarik, interaktif, dan modern. Tiga fitur penting dalam efek visual adalah transition, transform, dan animation.

### 3.6.1 CSS Transition (Efek Perubahan yang Halus)

Transition membuat perubahan properti CSS menjadi lebih mulus (smooth) dengan durasi tertentu. Contoh:

```
.button {  
    width: 150px;  
    background-color: #007BFF;  
    color: white;  
    padding: 10px 20px;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
    text-align: center;  
    transition: background-color 0.3s ease, transform  
0.3s ease;  
}  
.button:hover {  
    background-color: #0056b3;  
    transform: scale(1.05);  
}
```



```
/* CSS Transitions */  
.button {  
    width: 150px;  
    background-color: #007BFF;  
    color: white;  
    padding: 10px 20px;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
    text-align: center;  
    transition: background-color 0.3s ease, transform 0.3s ease;  
}  
.button:hover {  
    background-color: #0056b3;  
    transform: scale(1.05);  
}
```

Ketika hover, warna background berubah pelan dan ukuran membesar halus.

Properti penting transition:

- transition-property → properti apa yang berubah
- transition-duration → durasi (ms atau s)
- transition-timing-function → gaya percepatan (ease, linear, ease-in, dll)
- transition-delay → jeda sebelum animasi dimulai

### 3.6.2 CSS Transform (Mengubah Bentuk Elemen)

Transform digunakan untuk memutar (rotate), memperbesar (scale), menggeser (translate), atau memiringkan (skew) elemen. Contoh:

```
.box {
    width: 200px;
    height: 200px;
    background-color: #28a745;
    transition: transform 0.3s ease;
}

.box:hover {
    transform: rotate(15deg) scale(1.1) translateX(20px);
}
```



```
1 /* CSS Transform */
2 .box {
3     width: 200px;
4     height: 200px;
5     background-color: #28a745;
6     transition: transform 0.3s ease;
7 }
8
9 .box:hover {
10    transform: rotate(15deg) scale(1.1) translateX(20px);
11 }
```

Jenis transformasi:

- scale() → memperbesar/memperkecil
- rotate() → memutar elemen
- translate() → menggeser posisi
- skew() → memiringkan
- matrix() → kombinasi transformasi lanjutan

### 3.6.3 CSS Animation (Animasi Berulang)

Animation memungkinkan animasi yang lebih kompleks dengan *keyframes*.

```
@keyframes bergerak {  
    0% { transform: translateY(0); }  
    50% { transform: translateY(-20px); }  
    100% { transform: translateY(0); }  
}  
.bola {  
    width: 100px;  
    height: 100px;  
    background-color: #ffc107;  
    border-radius: 50%;  
    animation: bergerak 2s infinite;  
}
```



```
/* CSS Animations */  
@keyframes bergerak {  
    0% { transform: translateY(0); }  
    50% { transform: translateY(-20px); }  
    100% { transform: translateY(0); }  
}  
.bola {  
    width: 100px;  
    height: 100px;  
    background-color: #ffc107;  
    border-radius: 50%;  
    animation: bergerak 2s infinite;  
}
```

Elemen bergerak ke kanan, lalu kembali, dan berulang terus. Properti penting animation:

- **animation-name** → nama keyframes
- **animation-duration** → lamanya animasi
- **animation-iteration-count** → jumlah perulangan (1, 2, infinite)
- **animation-timing-function** → percepatan (ease, linear, dll)
- **animation-delay** → jeda awal
- **animation-direction** → normal, reverse, alternate

## BAB IV

### PENGENALAN JAVASCRIPT MODERN (ES6+)

#### 4.1 Konsep Dasar Pemrograman dengan JavaScript

JavaScript merupakan bahasa pemrograman yang berjalan di sisi klien (client-side) dan berfungsi untuk menambahkan interaktivitas pada halaman web. Berbeda dengan HTML yang mengatur struktur dan CSS yang mengatur tampilan, JavaScript berperan dalam mengendalikan logika, perilaku elemen, serta respons halaman web terhadap tindakan pengguna. Konsep dasar pemrograman dengan JavaScript meliputi pemahaman variabel, tipe data, operator, struktur kontrol, fungsi, serta interaksi dengan Document Object Model (DOM).

Pertama, variabel digunakan untuk menyimpan data yang dapat diproses atau dimodifikasi selama program berjalan. JavaScript modern menggunakan let dan const untuk deklarasi variabel yang lebih aman dibanding var. Selanjutnya, tipe data dalam JavaScript mencakup tipe primitif seperti string, number, boolean, null, undefined, serta tipe kompleks seperti object dan array. Operator digunakan untuk melakukan operasi aritmatika, logika, maupun perbandingan.

Selain itu, JavaScript menyediakan struktur kontrol seperti if, else, switch, serta pengulangan for dan while untuk mengatur alur eksekusi program. Fungsi berperan sebagai blok kode yang dapat dipanggil berulang kali, baik dalam bentuk deklarasi fungsi biasa maupun arrow function versi modern. Konsep dasar ini menjadi fondasi bagi pengembangan logika yang lebih kompleks.

Terakhir, salah satu aspek penting dalam JavaScript adalah kemampuannya untuk berinteraksi dengan DOM (Document Object Model). Melalui DOM, JavaScript dapat mengubah konten HTML, mengubah gaya CSS, menangani event seperti klik atau input, serta menciptakan elemen baru secara dinamis. Dengan memahami konsep dasar tersebut, mahasiswa dapat membangun aplikasi web yang lebih interaktif, responsif, dan fungsional.

Contoh Code: Kode JavaScript dapat ditulis langsung di dalam file HTML menggunakan tag <script>

```

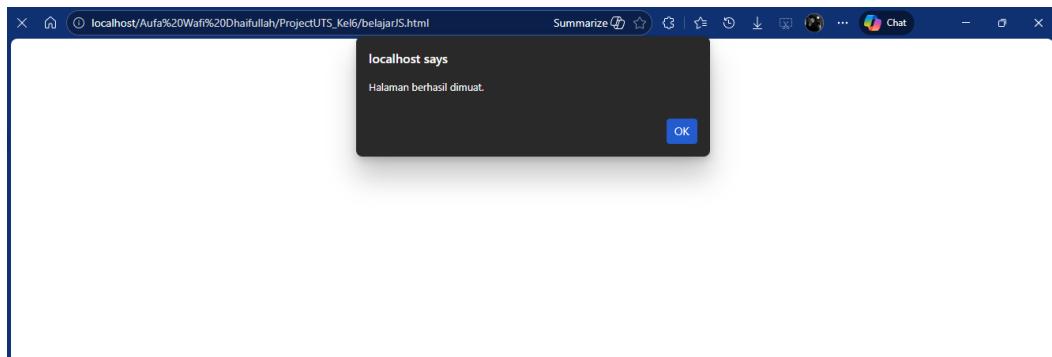
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Belajar JS</title>
</head>
<body>
    <h1>Hallo JavaScript</h1>
    <script>
        console.log("Selamat Datang di Dunia JavaScript!");
        alert("Halaman berhasil dimuat.");
    </script>
</body>
</html>

```

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Belajar JS</title>
7  </head>
8  <body>
9      <h1>Hallo JavaScript</h1>
10     <script>
11         console.log("Selamat Datang di Dunia JavaScript!");
12         alert("Halaman berhasil dimuat.");
13     </script>
14 </body>
15 </html>

```



## 4.2 Variabel (**let**, **const**, **var**)

Variabel dalam JavaScript berfungsi sebagai wadah untuk menyimpan data yang dapat digunakan, dimodifikasi, atau diproses selama program berjalan. JavaScript modern menyediakan tiga cara utama untuk mendeklarasikan variabel, yaitu var, let, dan const. Meskipun ketiganya memiliki fungsi dasar yang sama, ketiganya memiliki karakteristik berbeda terkait cakupan (scope), kemampuan perubahan nilai (mutability), serta konteks penggunaannya dalam pengembangan aplikasi web modern.

Pertama, **var** merupakan cara deklarasi variabel yang telah digunakan sejak versi awal JavaScript. Variabel yang dideklarasikan menggunakan var memiliki *function scope* dan bukan *block scope*, sehingga cakupan variabel tidak terbatas pada blok tertentu, misalnya pada kurung kurawal {}. Hal ini sering menimbulkan masalah seperti *hoisting* dan konflik nilai variabel, sehingga penggunaannya semakin jarang direkomendasikan dalam JavaScript modern.

Kedua, **let** merupakan bentuk deklarasi variabel yang diperkenalkan pada standar ECMAScript 2015 (ES6). Variabel let memiliki *block scope*, sehingga nilai dan keberadaannya hanya berlaku dalam blok tempat variabel tersebut dideklarasikan. let memungkinkan perubahan nilai setelah deklarasi, sehingga cocok digunakan untuk variabel yang bersifat dinamis atau akan dimodifikasi selama proses program.

Ketiga, **const** digunakan untuk mendeklarasikan variabel yang nilainya bersifat konstan atau tidak dapat diubah setelah pertama kali diberikan. Sama seperti let, const juga memiliki *block scope*. Meskipun demikian, pada variabel yang berupa objek atau array, struktur internalnya masih dapat dimodifikasi; yang tidak dapat diubah adalah referensinya. const umumnya dipilih ketika nilai variabel dianggap bersifat tetap atau digunakan sebagai konfigurasi.

Secara umum, pengembangan JavaScript modern lebih menganjurkan penggunaan let dan const karena keduanya memberikan kontrol yang lebih baik terhadap cakupan variabel serta mengurangi potensi kesalahan logika. Pemahaman mengenai perbedaan ketiga jenis deklarasi variabel ini menjadi dasar penting dalam menulis kode JavaScript yang lebih aman, terstruktur, dan mudah dikelola.

Contoh Code:

```
// Deklarasi Variabel dalam JavaScript
// Var (cara lama - Hindari jika memungkinkan)
var nama = "Aufa Wafi Dhaifullah";

// Let (cara modern - Gunakan ini untuk variabel yang nilainya bisa berubah)
let kelas = "05TPLP017";
kelas = "05TPLP018"; // Mengubah nilai variabel kelas

// Const (cara modern - Gunakan ini untuk variabel yang nilainya tetap)
const NIM = "231011401814";
// NIM = "231011401815"; // Ini akan menyebabkan error karena NIM adalah konstanta

console.log(nama, kelas, NIM);
```



```
1 // Var (cara Lama - Hindari jika memungkinkan)
2 var nama = "Aufa Wafi Dhaifullah";
3
4 // Let (cara modern - Gunakan ini untuk variabel yang nilainya bisa berubah)
5 let kelas = "05TPLP017";
6 kelas = "05TPLP018"; // Mengubah nilai variabel kelas
7
8 // Const (cara modern - Gunakan ini untuk variabel yang nilainya tetap)
9 const NIM = "231011401814";
10 // NIM = "231011401815"; // Ini akan menyebabkan error karena NIM adalah konstanta
11
12 console.log(nama, kelas, NIM);
```

### 4.3 Tipe Data dan Operator

JavaScript merupakan bahasa pemrograman yang bersifat *loosely typed* dan *dynamic*, artinya variabel tidak harus dideklarasikan dengan tipe data tertentu dan tipe data dapat berubah selama eksekusi program. Pemahaman mengenai tipe data serta operator sangat penting untuk membangun logika yang benar dan menghindari

kesalahan dalam manipulasi nilai. Secara umum, tipe data JavaScript dibagi menjadi dua kategori, yaitu tipe data primitif dan tipe data non-primitif (objek).

Tipe data primitif meliputi:

1. String, yaitu teks yang ditulis di antara tanda kutip tunggal, ganda, atau backticks.
2. Number, yang mencakup bilangan bulat, desimal, maupun nilai khusus seperti NaN dan Infinity.
3. Boolean, yang hanya memiliki dua nilai: true atau false.
4. Null, yaitu nilai yang secara eksplisit menyatakan "kosong".
5. Undefined, yaitu nilai yang muncul ketika variabel belum diberikan nilai.
6. BigInt, digunakan untuk menyimpan bilangan sangat besar yang melebihi batas tipe Number.
7. Symbol, digunakan untuk membuat identifier unik pada objek.

Sementara itu, tipe data non-primitif umumnya berbentuk Object, termasuk struktur seperti array, function, dan objek literal. Objek dapat menyimpan banyak nilai dalam bentuk pasangan *key-value* sehingga sering digunakan untuk menyusun data yang kompleks dalam aplikasi web.

Selain tipe data, JavaScript juga menyediakan berbagai operator untuk melakukan proses perhitungan maupun pengolahan logika. Operator aritmatika meliputi +, -, \*, /, %, dan \*\* untuk eksponensial. Operator perbandingan, seperti ==, ===, !=, !==, <, >, <=, dan >=, digunakan untuk membandingkan dua nilai. Dalam praktik pengembangan modern, === lebih dianjurkan karena melakukan perbandingan nilai sekaligus tipe data. JavaScript juga memiliki operator logika, yaitu && (AND), || (OR), dan! (NOT) sehingga memungkinkan pengambilan keputusan yang kompleks. Selain itu, terdapat operator penugasan seperti =, +=, -=, dan operator khusus seperti ternary (condition? value1: value2) yang mempersingkat penulisan kondisi.

Pemahaman terhadap tipe data dan operator ini sangat penting karena menjadi dasar dalam menyusun algoritma, mengelola data, serta membangun struktur logika pada aplikasi web. Kesalahan pemahaman terhadap tipe data atau

operator sering menyebabkan *bug* yang sulit dilacak, sehingga penguasaan konsep ini merupakan langkah awal dalam menjadi pengembang JavaScript yang efektif.

Contoh Code:

```
// Tipe data dan operator dalam JavaScript

// Tipe data dasar

let Nim = 231011401814; // Number

let Nama = "Aufa Wafi Dhaifullah"; // String

let isMahasiswa = true; // Boolean

let hobi = ["Membaca", "Menulis", "Coding"]; // Array

let alamat = { // Object

    jalan: "Jl. Merdeka No.1",
    kota: "Bekasi",
    kodePos: 10110
};

let kosong = null; // Null

let tidakTerdefinisi; // Undefined


// Operator Aritmatika dan Perbandingan

let hasilTambah = 10 + 5; // 15

let hasilKurang = 10 - 5; // 5

let hasilKali = 10 * 5; // 50

let hasilBagi = 10 / 5; // 2

let umur = 20;

let cekumur = umur > 18; // true jika umur lebih dari 18


// Menampilkan hasil di console

console.log('Nama saya ${Nama}, NIM saya ${Nim}, saya berumur ${umur} tahun.');

console.log('Hobi saya adalah ${hobi.join(", ")}.');
```

```

console.log('Alamat saya: ${alamat.jalan},
${alamat.kota}, ${alamat.kodePos}.');

console.log('Apakah saya mahasiswa? ${isMahasiswa}');

console.log('Hasil penjumlahan: ${hasilTambah},
pengurangan: ${hasilKurang}, perkalian: ${hasilKali},
pembagian: ${hasilBagi}.');

console.log('Apakah umur saya lebih dari 18?
${cekumur}');

```



```

1 // Tipe data dan operator dalam JavaScript
2 // Tipe data dasar
3 let Nim = 231011401814; // Number
4 let Nama = "Aufa Wafi Dhaifullah"; // String
5 let isMahasiswa = true; // Boolean
6 let hobis = ["Membaca", "Menulis", "Coding"]; // Array
7 let alamat = { // Object
8   jalan: "Jl. Merdeka No.1",
9   kota: "Bekasi",
10  kodePos: 10110
11 };
12 let kosong = null; // Null
13 let tidakTerdefinisi; // Undefined
14
15 // Operator Aritmatika dan Perbandingan
16 let hasilTambah = 10 + 5; // 15
17 let hasilKurang = 10 - 5; // 5
18 let hasilKali = 10 * 5; // 50
19 let hasilBagi = 10 / 5; // 2
20 let umur = 20;
21 let cekumur = umur > 18; // true jika umur Lebih dari 18
22
23 // Menampilkan hasil di console
24 console.log('Nama saya ${Nama}, NIM saya ${Nim}, saya berumur ${umur} tahun.');
25 console.log('Hobi saya adalah ${hobis.join(", ")}.');
26 console.log('Alamat saya: ${alamat.jalan}, ${alamat.kota}, ${alamat.kodePos}.');
27 console.log('Apakah saya mahasiswa? ${isMahasiswa}');
28 console.log('Hasil penjumlahan: ${hasilTambah}, pengurangan: ${hasilKurang}, perkalian: ${hasilKali}, pembagian: ${hasilBagi}.');
29 console.log('Apakah umur saya lebih dari 18? ${cekumur}');

```

#### 4.4 Struktur Kontrol (if, loop, switch)

Struktur kontrol merupakan komponen penting dalam pemrograman JavaScript karena menentukan alur eksekusi dari sebuah program. Dengan menggunakan struktur kontrol, program dapat mengambil keputusan, melakukan pengulangan, serta menentukan percabangan logika sesuai kondisi tertentu. Tiga struktur kontrol yang paling umum digunakan dalam JavaScript adalah *if statement*, *looping*, dan *switch statement*.

Pertama, **struktur kontrol if** memungkinkan program mengeksekusi blok kode tertentu jika suatu kondisi terpenuhi. Kondisi tersebut dievaluasi dalam bentuk ekspresi boolean, yang menghasilkan nilai true atau false. Variasi dari struktur ini mencakup if, if-else, serta if-else if yang digunakan ketika terdapat lebih dari satu kondisi yang perlu diuji. Dengan if, programmer dapat mengontrol perilaku

program berdasarkan situasi yang berbeda, sehingga logika aplikasi dapat berjalan secara dinamis.

Contoh Code:

```
// Percabangan (If-Else)
let nilai = 85;

if (nilai >= 90) { //jika nilai lebih dari atau sama dengan 90, maka Grade A
    console.log("Grade A");

} else if (nilai >= 80) { //jika nilai lebih dari atau sama dengan 80, maka Grade B
    console.log("Grade B");

} else if (nilai >= 70) { //jika nilai lebih dari atau sama dengan 70, maka Grade C
    console.log("Grade C");

} else { //jika nilai kurang dari 70, maka Grade D
    console.log("Grade D");
}
```



```
1 // Percabangan (If-Else)
2 let nilai = 85;
3 if (nilai >= 90) { //jika nilai lebih dari atau sama dengan 90, maka Grade A
4     console.log("Grade A");
5 } else if (nilai >= 80) { //jika nilai lebih dari atau sama dengan 80, maka Grade B
6     console.log("Grade B");
7 } else if (nilai >= 70) { //jika nilai lebih dari atau sama dengan 70, maka Grade C
8     console.log("Grade C");
9 } else { //jika nilai kurang dari 70, maka Grade D
10    console.log("Grade D");
11 }
```

Kedua, **struktur pengulangan (loop)** digunakan ketika program perlu mengeksekusi suatu blok kode secara berulang. JavaScript menyediakan beberapa jenis loop seperti for, while, dan do...while. Loop for digunakan ketika jumlah iterasi sudah diketahui, sedangkan loop while akan berjalan selama kondisi bernilai benar. Loop do - while memastikan blok kode dijalankan setidaknya sekali sebelum kondisi diperiksa. Selain itu, JavaScript modern juga menyediakan for - of untuk iterasi pada array dan for - in untuk iterasi pada properti objek. Struktur

pengulangan ini penting untuk mengelola data dalam jumlah besar dan menjalankan proses berulang secara efisien.

Contoh Code:

```
// Perulangan (For Loop)
for (let i = 1; i <= 5; i++) {
    console.log("Perulangan ke-" + i);
}
```



```
1 // Perulangan (For Loop)
2 for (let i = 1; i <= 5; i++) {
3     console.log("Perulangan ke-" + i);
4 } // Hasilnya akan menampilkan Perulangan ke-1 sampai Perulangan ke-5
```

Ketiga, **struktur kontrol switch** berfungsi sebagai alternatif dari penggunaan banyak kondisi if-else if. Struktur ini cocok digunakan ketika terdapat beberapa kemungkinan nilai tetap yang harus diuji. Dalam switch, nilai ekspresi dibandingkan dengan beberapa *case*, dan setiap *case* memiliki blok kode yang akan dijalankan jika cocok. Penggunaan break penting agar eksekusi tidak berlanjut ke *case* lain, kecuali memang diperlukan (*fall-through*). Struktur ini meningkatkan keterbacaan kode, khususnya untuk pilihan nilai yang banyak namun konstan.

Secara keseluruhan, struktur kontrol seperti if, loop, dan switch merupakan dasar pengambilan keputusan dalam JavaScript yang memungkinkan program berjalan secara logis, efisien, dan fleksibel. Penguasaan terhadap struktur kontrol ini sangat penting karena hampir semua fungsi interaktif pada aplikasi web modern bergantung pada kemampuan untuk memproses kondisi dan pengulangan secara tepat.

Contoh Code:

```
// Switch Case
let hari = 3;
switch (hari) {
    case 1:
```

```
    console.log("Hari Senin"); break;
  case 2:
    console.log("Hari Selasa"); break;
  case 3:
    console.log("Hari Rabu"); break;
  case 4:
    console.log("Hari Kamis"); break;
  case 5:
    console.log("Hari Jumat"); break;
  case 6:
    console.log("Hari Sabtu"); break;
  case 7:
    console.log("Hari Minggu"); break;
  default:
    console.log("Hari tidak valid");
}
// Jika hari tidak antara 1-7, maka tampilkan pesan ini
```



```
1 // Switch Case
2 let hari = 3;
3 switch (hari) {
4   case 1:
5     console.log("Hari Senin"); break;
6   case 2:
7     console.log("Hari Selasa"); break;
8   case 3:
9     console.log("Hari Rabu"); break;
10  case 4:
11    console.log("Hari Kamis"); break;
12  case 5:
13    console.log("Hari Jumat"); break;
14  case 6:
15    console.log("Hari Sabtu"); break;
16  case 7:
17    console.log("Hari Minggu"); break;
18  default:
19    console.log("Hari tidak valid");
20
21 } // Jika hari tidak antara 1-7, maka tampilkan pesan ini
```

#### 4.5 Fungsi (Regular Function, Arrow Function)

Fungsi merupakan salah satu elemen fundamental dalam JavaScript karena memungkinkan pengembang untuk mengelompokkan serangkaian perintah ke dalam satu kesatuan logis yang dapat dipanggil kapan saja. Fungsi digunakan untuk mengurangi redundansi kode, mempermudah pemeliharaan program, serta memberikan struktur yang lebih jelas dalam pengembangan aplikasi web. Dalam JavaScript modern, terdapat dua bentuk umum dari fungsi, yaitu *regular function* dan *arrow function*, yang masing-masing memiliki karakteristik serta kegunaan yang berbeda.

Pertama, **regular function** adalah bentuk fungsi tradisional yang telah digunakan sejak versi awal JavaScript. Regular function dapat dideklarasikan menggunakan kata kunci `function`, baik sebagai *function declaration* maupun *function expression*. Kelebihan dari fungsi ini adalah dukungan penuh terhadap konsep *hoisting*, sehingga fungsi dapat dipanggil sebelum lokasi deklarasinya dalam kode. Regular function juga memiliki konteks `this` tersendiri yang bersifat dinamis, artinya nilai `this` bergantung pada bagaimana fungsi tersebut dipanggil. Hal ini menjadikan regular function fleksibel dalam berbagai situasi, terutama ketika digunakan sebagai method dalam sebuah objek.

Kedua, **arrow function** merupakan fitur yang diperkenalkan dalam ECMAScript 2015 (ES6) dan dirancang untuk memberikan sintaks yang lebih ringkas dan mudah dibaca. Arrow function ditulis menggunakan tanda panah `=>` dan biasanya digunakan dalam fungsi pendek atau ekspresi callback. Salah satu perbedaan utama antara arrow function dan regular function adalah bahwa arrow function tidak memiliki konteks `this` sendiri; melainkan mewarisi konteks `this` dari lingkup tempat fungsi tersebut didefinisikan (*lexical this*). Hal ini membuat arrow function sangat berguna dalam penulisan kode yang melibatkan callback pada event atau operasi asynchronous. Namun, arrow function tidak dapat digunakan sebagai constructor dan tidak dapat mengakses objek arguments secara langsung.

Dalam praktik pengembangan aplikasi web modern, pemilihan antara regular function dan arrow function bergantung pada kebutuhan konteks dan gaya penulisan kode. Arrow function lebih efisien untuk fungsi singkat dan manipulasi

DOM sederhana, sementara regular function lebih sesuai untuk fungsi kompleks yang memerlukan pengelolaan konteks this. Pemahaman mengenai kedua jenis fungsi ini penting agar pengembang dapat menulis kode yang lebih terstruktur, bersih, serta mudah di-maintain.

Contoh Code:

```
// Regular Function (cara lama)
function tambah(a, b) {
    return a + b;
}
console.log("Hasil penjumlahan: " + tambah(5, 4));
// Hasil penjumlahan: 9

// Arrow Function (cara modern)
const kali = (a, b) => a * b;
console.log("Hasil perkalian: " + kali(5, 4));
// Hasil perkalian: 20
```



```
1 // Fungsi dalam JavaScript (Regular Function dan Arrow Function)
2 // Regular Function (cara Lama)
3 function tambah(a, b) {
4     return a + b;
5 }
6 console.log("Hasil penjumlahan: " + tambah(5, 4)); // Hasil penjumlahan: 9
7
8 // Arrow Function (cara modern)
9 const kali = (a, b) => a * b;
10 console.log("Hasil perkalian: " + kali(5, 4)); // Hasil perkalian: 20
```

## 4.6 Event Handling Dasar

Event handling merupakan salah satu konsep penting dalam JavaScript yang memungkinkan pengembang menciptakan interaktivitas pada halaman web. Event dalam konteks pemrograman web merujuk pada setiap tindakan atau kejadian yang terjadi pada halaman, seperti klik, ketikan pengguna, pergerakan mouse, pemuatan halaman, atau perubahan nilai pada sebuah elemen. Melalui event handling,

JavaScript dapat merespons peristiwa tersebut dengan menjalankan kode tertentu, sehingga aplikasi web menjadi dinamis dan responsif.

Secara umum, terdapat tiga cara utama dalam menangani event di JavaScript. Pertama, event dapat ditangani secara inline, yaitu dengan menuliskan pemanggilan fungsi langsung pada atribut HTML seperti onclick. Meskipun mudah digunakan, pendekatan ini dianggap kurang efisien dan tidak dianjurkan pada pengembangan modern karena mencampurkan logika JavaScript dengan struktur HTML. Kedua, event dapat ditangani menggunakan event handler properti, yaitu dengan menetapkan fungsi pada properti event suatu elemen, contohnya element.onclick = function() { ... }. Cara ini lebih baik dari inline, namun tetap terbatas karena hanya memungkinkan satu event handler untuk setiap jenis event pada suatu elemen.

Metode yang paling direkomendasikan dalam pengembangan modern adalah penggunaan addEventListener(), yang memungkinkan penambahan lebih dari satu handler untuk jenis event yang sama serta memberikan fleksibilitas yang lebih besar. Dengan addEventListener(), pengembang dapat menentukan jenis event dan fungsi yang akan dijalankan ketika event tersebut terjadi. Selain itu, metode ini juga mendukung fitur-fitur seperti *event bubbling*, *event capturing*, dan opsi konfigurasi lanjutan.

Contoh event yang paling sering digunakan antara lain click, keyup, change, mouseover, dan submit. Sebagai contoh, ketika pengguna menekan tombol pada formulir, event submit dapat digunakan untuk memvalidasi input sebelum data dikirimkan. Sementara itu, event click dapat digunakan untuk membuat tombol navigasi atau elemen interaktif lainnya. Dengan memanfaatkan event handling, halaman web tidak hanya berfungsi sebagai tampilan statis, tetapi dapat bereaksi secara real-time terhadap aksi pengguna.

Secara keseluruhan, pemahaman tentang event handling dasar sangat penting dalam membangun pengalaman pengguna yang lebih interaktif dan intuitif. Kemampuan JavaScript dalam merespons berbagai peristiwa memungkinkan pengembang menciptakan antarmuka yang adaptif, menarik, serta sesuai kebutuhan aplikasi web modern.

Contoh Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Belajar JS</title>

</head>

<body>

    <h1>Hallo JavaScript</h1>

    <!-- Event Handling Dasar -->

    <button onclick="sapaUser()">Klik Disini!</button>

<script>

    function sapaUser() {

        alert("Hallo User!. Terima kasih sudah mengklik
tombol ini.");

    }

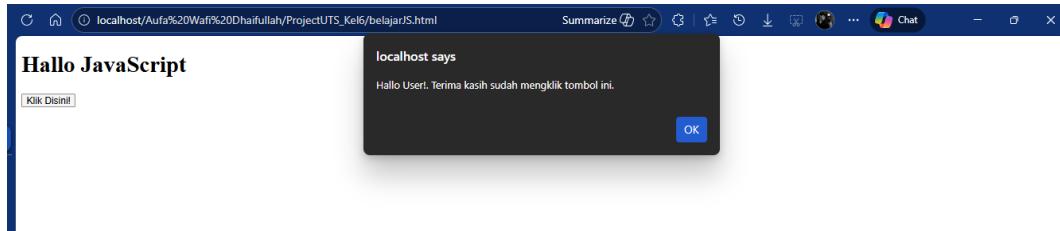
</script>

</body>

</html>
```

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Belajar JS</title>
7  </head>
8  <body>
9      <h1>Hallo JavaScript</h1>
10
11     <!-- Event Handling Dasar -->
12     <button onclick="sapaUser()">Klik Disini!</button>
13
14     <script>
15         function sapaUser() {
16             alert("Hallo User!. Terima kasih sudah mengklik tombol ini.");
17         }
18     </script>
19
20 </body>
</html>
```

Outputnya:



## BAB V

### DOM DAN MANIPULASI ELEMEN

Pada bab sebelumnya, kita telah mempelajari dasar-dasar sintaks JavaScript seperti variabel, tipe data, dan fungsi. Namun, kode JavaScript tersebut hanya berjalan di konsol dan belum benar-benar berinteraksi dengan tampilan halaman web. Di bab ini, kita akan masuk ke inti dari pengembangan *Front-End*, yaitu DOM (Document Object Model). DOM adalah jembatan yang menghubungkan JavaScript dengan HTML dan CSS, memungkinkan kita untuk membuat halaman web yang dinamis, interaktif, dan responsif terhadap tindakan pengguna.

#### 5.1 Konsep Document Object Model (DOM)

Document Object Model atau disingkat DOM adalah representasi terstruktur dari dokumen HTML yang dibuat oleh browser saat halaman web dimuat. Sederhananya, DOM adalah "peta" atau model dari seluruh elemen yang ada di halaman web Anda. Browser membaca kode HTML baris demi baris dan mengubahnya menjadi serangkaian objek yang tersusun dalam struktur pohon (*tree structure*).

Struktur Pohon DOM (DOM Tree), dalam struktur pohon ini, setiap bagian dari dokumen disebut sebagai Node (simpul).

1. Document Node Akar utama dari pohon, mewakili seluruh halaman web.
2. Element Node Mewakili tag HTML (seperti `<body>`, `<h1>`, `<div>`, `<p>`).
3. Text Node Mewakili isi teks di dalam elemen (misalnya tulisan "Halo Dunia" di dalam `<h1>`).
4. Attribute Node Mewakili atribut pada tag HTML (seperti `id`, `class`, `src`, `href`).

Hubungan antar node ini digambarkan seperti silsilah keluarga:

- Parent (Induk): Elemen yang membungkus elemen lain. Contoh: `<body>` adalah parent dari `<h1>`.
- Child (Anak): Elemen yang berada di dalam elemen lain.
- Sibling (Saudara): Elemen-elemen yang berada di dalam parent yang sama.

Dengan memahami konsep ini, JavaScript dapat "berjalan-jalan" menelusuri pohon DOM untuk menemukan elemen tertentu, lalu mengubah, menghapus, atau menambahkannya secara dinamis tanpa perlu memuat ulang (*reload*) halaman.

## 5.2 Mengakses Elemen HTML dengan getElementById dan querySelector

Langkah pertama dalam memanipulasi DOM adalah "menangkap" atau menyeleksi elemen yang ingin kita ubah. JavaScript menyediakan beberapa metode untuk melakukan hal ini, mulai dari cara klasik hingga cara modern yang lebih fleksibel.

### 5.2.1 Menggunakan ID (getElementById)

Ini adalah metode paling cepat dan spesifik. Karena aturan HTML mengharuskan setiap ID bersifat unik (hanya boleh ada satu ID yang sama dalam satu halaman), metode ini akan selalu mengembalikan satu elemen tunggal.

Contoh Code:

```
<h1 id = "judul-utama">Selamat Datang</h1>
<script>
//Menangkap elemen dengan id "judul-utama"
const judul = document.getElementById("judul-utama");

//Mengecek di console apakah elemen berhasil ditangkap
console.log(judul);
</script>
```



```
1 <h1 id = "judul-utama">Selamat Datang</h1>
2
3 <script>
4 //Menangkap elemen dengan id "judul-utama"
5 const judul = document.getElementById("judul-utama");
6
7 //Mengecek di console apakah elemen berhasil ditangkap
8 console.log(judul);
9 </script>
```

### 5.2.2 Menggunakan Selektor CSS (querySelector)

Ini adalah metode modern yang sangat *powerful*. querySelector memungkinkan kita memilih elemen menggunakan sintaks yang sama persis dengan CSS Selector (bisa berdasarkan ID, Class, Tag, atau kombinasi atribut). Metode ini akan mengembalikan elemen pertama yang ditemukan jika ada banyak elemen yang cocok.

Contoh Code:

```
//Menggunakan Selector CSS
//Memilih berdasarkan id (Menggunakan #)
const sidebar = document.querySelector("#judul-utama");

//Memilih berdasarkan class (Menggunakan .)
const tombollogin = document.querySelector(".tombol-login");

//memilih elemen <p> pertama yang ada di dalam <div
//class="content">
const paragrafPertama =
document.querySelector(".content p");
```



```
1 //Menggunakan Selector CSS
2 //Memilih berdasarkan id (Menggunakan #)
3 const sidebar = document.querySelector("#judul-utama");
4
5 //Memilih berdasarkan class (Menggunakan .)
6 const tombollogin = document.querySelector(".tombol-login");
7
8 //memilih elemen <p> pertama yang ada di dalam <div class="content">
9 const paragrafPertama = document.querySelector(".content p");
```

### 5.2.3 Menggunakan querySelectorAll (Banyak Elemen)

Jika kita ingin memilih semua elemen yang cocok dengan kriteria tertentu (misalnya memilih semua tombol di halaman), kita menggunakan querySelectorAll. Hasilnya bukan satu elemen, melainkan sebuah daftar (*NodeList*) yang mirip dengan Array.

Contoh Code:

```
//Menggunakan querySelectorAll  
//mengambil semua elemen dengan class 'menu-item'  
const menuItems = document.querySelectorAll(".menu-item");  
  
//Karena hasilnya banyak, kita bisa menggunakan looping  
menuItems.forEach((menu) => {  
    console.log('Menu Ditemukan: ' + menu.textContent);  
});
```



```
1 //Menggunakan querySelectorAll  
2 //mengambil semua elemen dengan class 'menu-item'  
3 const menuItems = document.querySelectorAll(".menu-item");  
4  
5 //Karena hasilnya banyak, kita bisa menggunakan Looping  
6 menuItems.forEach((menu) => {  
    console.log('Menu Ditemukan: ' + menu.textContent);  
});
```

### 5.3 Mengubah Konten dan Atribut Elemen

Setelah berhasil menangkap elemen, langkah selanjutnya adalah memanipulasinya. Kita bisa mengubah teks, menyisipkan HTML baru, mengganti gambar, atau bahkan mengubah gaya visual (CSS) secara langsung.

#### 5.3.1 Manipulasi Konten (innerText vs innerHTML)

- `innerText` atau `textContent` digunakan untuk mengubah teks polos di dalam elemen. Ini aman dari serangan XSS (*Cross Site Scripting*) karena tidak mengeksekusi tag HTML.
- `innerHTML` digunakan jika kita ingin menyisipkan elemen HTML baru (seperti menambahkan `<b>tebal</b>` atau tag `<span>`).

Contoh Code:

```
<p id = "deskripsi">Teks lama</p>
<script>

    const p = document.getElementById("deskripsi");

    //mengubah teks biasa
    p.innerText = "Teks ini telah diubah menggunakan
JavaScript.';

    //Menyisipkan HTML (Teks menjadi tebal)
    p.innerHTML = "Teks ini <b>TEBAL</b> karena
disisisipkan innerHTML.';

</script>
```



```
1  <p id = "deskripsi">Teks lama</p>
2  <script>
3      const p = document.getElementById("deskripsi");
4
5      //mengubah teks biasa
6      p.innerText = "Teks ini telah diubah menggunakan JavaScript.';
7
8      //Menyisipkan HTML (Teks menjadi tebal)
9      p.innerHTML = "Teks ini <b>TEBAL</b> karena disisisipkan innerHTML.';
10 </script>
```

### 5.3.2 Manipulasi Atribut (setAttribute, src, href)

Kita sering perlu mengubah atribut elemen, misalnya mengganti gambar pada slider, atau mengubah tujuan link.

Contoh Code:

```
//manipulasi Atribut
const gambar = document.querySelector("img");
const link = document.querySelector("a");

//mengganti sumber gambar
gambar.src = "gambar_baru.jpg";
```

```

//Atau secara eksplisit
gambar.setAttribute("src", "gambar_baru.jpg");

//mengubah tujuan link
link.href = "https://www.example.com";
link.setAttribute("href", "https://www.example.com");

```



```

1 //manipulasi Atribut
2 const gambar = document.querySelector("img");
3 const link = document.querySelector("a");
4
5 //mengganti sumber gambar
6 gambar.src = "gambar_baru.jpg";
7
8 //Atau secara eksplisit
9 gambar.setAttribute("src", "gambar_baru.jpg");
10
11 //mengubah tujuan Link
12 link.href = "https://www.example.com";
13 link.setAttribute("href", "https://www.example.com");

```

### 5.3.3 Manipulasi Style dan Class (Penting!)

Mengubah tampilan elemen bisa dilakukan dengan dua cara:

1. style property, untuk menulis CSS inline (tidak disarankan untuk perubahan besar).
2. classList, untuk menambah atau menghapus *class* CSS yang sudah didefinisikan di file .css. Ini adalah cara terbaik (*Best Practice*) karena memisahkan logika (JS) dan gaya (CSS).

Contoh Code:

```

//manipulasi Style dan class
const kotak = document.getElementById("box");

//cara 1: ubah style langsung (kurang rapih)

```

```

kotak.style.backgroundColor = "blue";
kotak.style.border = "2px solid black";
kotak.style.width = "200px";
kotak.style.height = "200px";
kotak.style.display = "inline-block";

//cara 2: ubah class (lebih rapih)
kotak.classList.add("aktif"); //menambahkan class
kotak.classList.remove("aktif"); //menghapus class
kotak.classList.toggle("dark-mode"); //Saklar (Jika ada
hapus, jika tidak ada tambahkan)

```



```

1 //manipulasi Style dan class
2 const kotak = document.getElementById("box");
3
4 //cara 1: ubah style Langsung (kurang rapih)
5 kotak.style.backgroundColor = "blue";
6 kotak.style.border = "2px solid black";
7 kotak.style.width = "200px";
8 kotak.style.height = "200px";
9 kotak.style.display = "inline-block";
10
11 //cara 2: ubah class (Lebih rapih)
12 kotak.classList.add("aktif"); //menambahkan class
13 kotak.classList.remove("aktif"); //menghapus class
14 kotak.classList.toggle("dark-mode"); //Saklar (Jika ada hapus, jika tidak ada tambahkan)

```

## 5.4 Event Listener dan Interaktivitas Halaman

Halaman web statis itu kurang menarik, kita ingin halaman bereaksi ketika pengguna melakukan sesuatu. Di sinilah Event berperan, Event adalah kejadian yang terjadi di browser, seperti pengguna mengklik mouse, menekan tombol keyboard, menggerakkan mouse (*hover*), atau mengirim formulir (*submit*).

Menggunakan `addEventListener`, cara modern dan standar untuk menangani event adalah menggunakan metode `addEventListener`. Metode ini lebih unggul dibandingkan atribut HTML kuno (seperti `onclick="..."`) karena memungkinkan kita memisahkan kode JavaScript dari HTML sepenuhnya. Sintaks dasarnya yaitu adalah `elemen.addEventListener('jenis_event', fungsi_callback);`

### Jenis-Jenis Event Populer:

- Mouse Events: click, dblclick, mouseover, mouseout.
- Keyboard Events: keydown, keyup, keypress.
- Form Events: submit, change, focus, blur.
- Window Events: load, resize, scroll.

### Contoh Code:

```
<button id="btn-sapa">Klik disini</button>

<input type="text" id="input-nama"
placeholder="Masukkan nama Anda">

<script>

const tombol = document.getElementById('btn-sapa');
const input = document.getElementById('input-nama');

//Event klik mouse
tombol.addEventListener('click', function() {
    alert('Tombol telah diklik!');
    tombol.style.backgroundColor = 'lightblue'; // Mengubah
    warna tombol saat diklik
});

//Event Keyboard
input.addEventListener('keydown', function(event) {
    console.log("User mengetik: " + event.target.value);
});

</script>
```

```

1  <button id="btn-sapa">Klik disini</button>
2  <input type="text" id="input-nama" placeholder="Masukkan nama Anda">
3  <script>
4      const tombol = document.getElementById('btn-sapa');
5      const input = document.getElementById('input-nama');
6
7      //Event klik mouse
8      tombol.addEventListener('click', function() {
9          alert('Tombol telah diklik!');
10         tombol.style.backgroundColor = 'lightblue'; // Mengubah warna tombol saat diklik
11     });
12
13     //Event Keyboard
14     input.addEventListener('keydown', function(event) {
15         console.log("User mengetik: " + event.target.value);
16     });
17 </script>

```

## 5.5 Contoh Sederhana (Studi Kasus)

Sebagai penutup bab ini, mari kita gabungkan semua konsep di atas (Seleksi, Manipulasi, dan Event) untuk membuat dua fitur umum: Toggle Menu Navigasi (untuk tampilan responsif) dan Form Validasi Sederhana.

**Studi Kasus:** Toggle Menu (Navigasi Mobile)

Fitur ini umum digunakan pada desain web responsif. Saat di layar kecil, menu disembunyikan dan hanya muncul tombol garis tiga. Ketika tombol diklik, menu muncul.

**Kode html:**

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Studi Kasus bab 5</title>

    <link rel="stylesheet" href="belajarJS.css">

</head>

<body>

    <nav class ="navbar">

```

```

<div class="logo">Web Saya</div>
<button id="btn-menu" class="tombol-menu">
    Menu
</button>
<ul id="menu-list" class="menu-items">
    <li><a href="#">Home</a></li>
    <li><a href="#">Profile</a></li>
    <li><a href="#">Gallery</a></li>
    <li><a href="#">Contact</a></li>
</ul>
</nav>
<hr>
<div class="container-form">
    <h3>Belajar JavaScript</h3>
    <p>Silahkan Masukan username (Min. 5 karakter) dan password (Min. 8 karakter)</p>
    <form id="form-login">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" placeholder="Ketik Username" required>
            <small id="err-user" class="error-message">Username minimal 5 karakter</small>
        </div>
        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" placeholder="Ketik Password" required>
            <small id="err-pass" class="error-message">Password minimal 8 karakter</small>
        </div>
        <div class="form-group">

```

```

        <input type="checkbox" id="show-pass"
onclick="togglePassword()"> Tampilkan Password
    </div>

    <button type="submit">Login</button>
</form>
</div>
<script src="belajarJS.js"></script>
</body>
</html>

```



```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Studi Kasus bab 5</title>
7      <link rel="stylesheet" href="belajarJS.css">
8  </head>
9  <body>
10     <nav class ="navbar">
11         <div class="logo">Web Saya</div>
12         <button id="btn-menu" class="tombol-menu">
13             Menu
14         </button>
15         <ul id="menu-list" class="menu-items">
16             <li><a href="#">Home</a></li>
17             <li><a href="#">Profile</a></li>
18             <li><a href="#">Gallery</a></li>
19             <li><a href="#">Contact</a></li>
20         </ul>
21     </nav>
22     <hr>

```

```

1   <div class="container-form">
2     <h3>Belajar JavaScript</h3>
3     <p>Silahkan Masukan username (Min. 5 karakter) dan password (Min. 8 karakter)</p>
4     <form id="form-login">
5       <div class="form-group">
6         <label for="username">Username:</label>
7         <input type="text" id="username" name="username" placeholder="Ketik Username" required>
8         <small id="err-user" class="error-message">Username minimal 5 karakter</small>
9       </div>
10      <div class="form-group">
11        <label for="password">Password:</label>
12        <input type="password" id="password" name="password" placeholder="Ketik Password" required>
13        <small id="err-pass" class="error-message">Password minimal 8 karakter</small>
14      </div>
15      <div class ="form-group">
16        <input type="checkbox" id="show-pass" onclick="togglePassword()"> Tampilkan Password
17      </div>
18      <button type="submit">Login</button>
19    </form>
20  </div>
21  <script src="belajarJS.js"></script>
22 </body>
23 </html>

```

## Kode CSS:

```

/* belajarCSS.css */
body {
  font-family: Arial, sans-serif;
  background: #f4f4f4;
  margin: 0;
  padding: 0;
}

.navbar {
  background: #2d3e50;
  color: #fff;
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 10px 30px;
}

.logo {

```

```
    font-size: 1.5em;
    font-weight: bold;
}

.tombol-menu {
    background: #4caf50;
    color: #fff;
    border: none;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1em;
    display: none;
}

.menu-items {
    list-style: none;
    display: flex;
    gap: 20px;
    margin: 0;
    padding: 0;
}

.menu-items li a {
    color: #fff;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.2s;
}

.menu-items li a:hover {
    color: #4caf50;
}
```

```
hr {  
    margin: 0;  
    border: none;  
    border-top: 1px solid #ddd;  
}  
  
.container-form {  
    background: #fff;  
    max-width: 400px;  
    margin: 40px auto;  
    padding: 30px 25px 20px 25px;  
    border-radius: 8px;  
    box-shadow: 0 2px 8px rgba(0,0,0,0.08);  
}  
  
.container-form h3 {  
    margin-top: 0;  
    color: #2d3e50;  
}  
  
.form-group {  
    margin-bottom: 18px;  
}  
  
label {  
    display: block;  
    margin-bottom: 6px;  
    color: #333;  
}
```

```
input[type="text"],  
input[type="password"] {  
    width: 100%;  
    padding: 8px 10px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
    font-size: 1em;  
    box-sizing: border-box;  
}  
  
input[type="checkbox"] {  
    margin-right: 6px;  
}  
  
button[type="submit"] {  
    background: #2d3e50;  
    color: #fff;  
    border: none;  
    padding: 10px 20px;  
    border-radius: 4px;  
    font-size: 1em;  
    cursor: pointer;  
    transition: background 0.2s;  
}  
button[type="submit"]:hover {  
    background: #4caf50;  
}  
  
.error-message {  
    color: #e74c3c;  
    font-size: 0.95em;
```

```
    display: none;
    margin-top: 4px;
}

@media (max-width: 600px) {
    .navbar {
        flex-direction: column;
        align-items: flex-start;
        padding: 10px 15px;
    }
    .tombol-menu {
        display: block;
        margin-top: 10px;
    }
    .menu-items {
        flex-direction:column;
        gap: 10px;
        width: 100%;
        display: none;
        margin-top: 10px;
    }
    .menu-items.active {
        display: flex;
    }
}
```

```
1  /* belajarCSS.css */
2  body {
3      font-family: Arial, sans-serif;
4      background: #f4f4f4;
5      margin: 0;
6      padding: 0;
7  }
8
9  .navbar {
10     background: #2d3e50;
11     color: #fff;
12     display: flex;
13     align-items: center;
14     justify-content: space-between;
15     padding: 10px 30px;
16 }
17
18 .logo {
19     font-size: 1.5em;
20     font-weight: bold;
21 }
22
23 .tombol-menu {
24     background: #4caf50;
25     color: #fff;
26     border: none;
27     padding: 8px 16px;
28     border-radius: 4px;
29     cursor: pointer;
30     font-size: 1em;
31     display: none;
32 }
33
34 .menu-items {
35     list-style: none;
36     display: flex;
37     gap: 20px;
38     margin: 0;
39     padding: 0;
40 }
41 .menu-items li a {
42     color: #fff;
43     text-decoration: none;
44     font-weight: 500;
45     transition: color 0.2s;
46 }
47 .menu-items li a:hover {
48     color: #4caf50;
49 }
```



```
1  hr {
2    margin: 0;
3    border: none;
4    border-top: 1px solid #ddd;
5  }
6
7  .container-form {
8    background: #fff;
9    max-width: 400px;
10   margin: 40px auto;
11   padding: 30px 25px 20px 25px;
12   border-radius: 8px;
13   box-shadow: 0 2px 8px rgba(0,0,0,0.08);
14 }
15
16 .container-form h3 {
17   margin-top: 0;
18   color: #2d3e50;
19 }
20
21 .form-group {
22   margin-bottom: 18px;
23 }
24
25 label {
26   display: block;
27   margin-bottom: 6px;
28   color: #333;
29 }
30
31 input[type="text"],
32 input[type="password"] {
33   width: 100%;
34   padding: 8px 10px;
35   border: 1px solid #ccc;
36   border-radius: 4px;
37   font-size: 1em;
38   box-sizing: border-box;
39 }
40
41 input[type="checkbox"] {
42   margin-right: 6px;
43 }
```



```
1 button[type="submit"] {  
2     background: #2d3e50;  
3     color: #fff;  
4     border: none;  
5     padding: 10px 20px;  
6     border-radius: 4px;  
7     font-size: 1em;  
8     cursor: pointer;  
9     transition: background 0.2s;  
10 }  
11 button[type="submit"]:hover {  
12     background: #4caf50;  
13 }  
14  
15 .error-message {  
16     color: #e74c3c;  
17     font-size: 0.95em;  
18     display: none;  
19     margin-top: 4px;  
20 }  
21  
22 @media (max-width: 600px) {  
23     .navbar {  
24         flex-direction: column;  
25         align-items: flex-start;  
26         padding: 10px 15px;  
27     }  
28     .tombol-menu {  
29         display: block;  
30         margin-top: 10px;  
31     }  
32     .menu-items {  
33         flex-direction: column;  
34         gap: 10px;  
35         width: 100%;  
36         display: none;  
37         margin-top: 10px;  
38     }  
39     .menu-items.active {  
40         display: flex;  
41     }  
42 }
```

### Kode JavaScript:

```
// belajarJS.JS

document.addEventListener('DOMContentLoaded',
function() {

    const form = document.getElementById('form-login');

    const username = document.getElementById('username');

    const password = document.getElementById('password');

    const errUser = document.getElementById('err-user');

    const errPass = document.getElementById('err-pass');

    const showPass = document.getElementById('show-
pass');

    // Hide error messages initially
    errUser.style.display = 'none';
    errPass.style.display = 'none';

    form.addEventListener('submit', function(e) {

        let valid = true;

        // Username validation
        if (username.value.length < 5) {
            errUser.style.display = 'block';
            valid = false;
        } else {
            errUser.style.display = 'none';
        }

        // Password validation
        if (password.value.length < 8) {
            errPass.style.display = 'block';
            valid = false;
        } else {
            errPass.style.display = 'none';
        }
    });
});
```

```

        }

        if (!valid) {
            e.preventDefault();
        }
    }) ;



// Show/hide password
showPass.addEventListener('change', function() {
    if (showPass.checked) {
        password.type = 'text';
    } else {
        password.type = 'password';
    }
}) ;




// Responsive menu
const btnMenu = document.getElementById('btn-menu');
const menuList = document.getElementById('menu-list');

btnMenu.addEventListener('click', function() {
    menuList.classList.toggle('active');
}) ;




// Function for inline onclick (optional, for compatibility)
function togglePassword() {
    var pass = document.getElementById('password');
    var show = document.getElementById('show-pass');
    pass.type = show.checked ? 'text' : 'password';
}

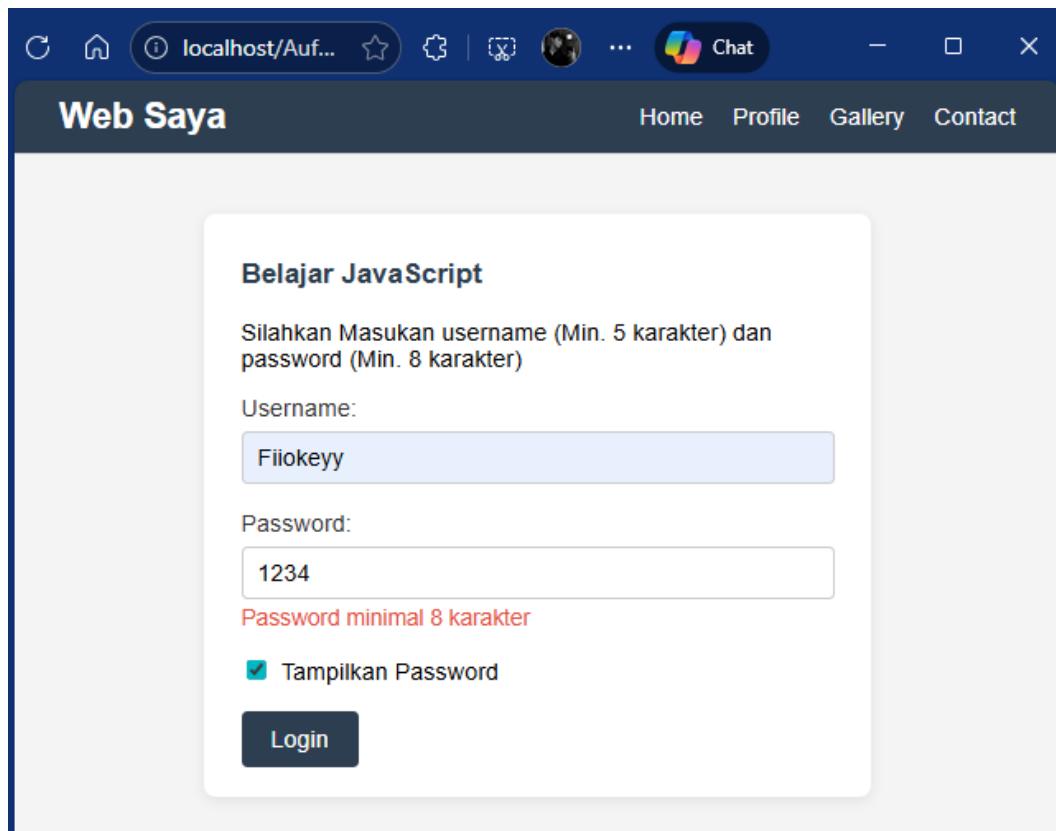
```

```

1 // belajarJS.js
2
3 document.addEventListener('DOMContentLoaded', function() {
4   const form = document.getElementById('form-login');
5   const username = document.getElementById('username');
6   const password = document.getElementById('password');
7   const errUser = document.getElementById('err-user');
8   const errPass = document.getElementById('err-pass');
9   const showPass = document.getElementById('show-pass');
10
11  // Hide error messages initially
12  errUser.style.display = 'none';
13  errPass.style.display = 'none';
14
15  form.addEventListener('submit', function(e) {
16    let valid = true;
17    // Username validation
18    if (username.value.length < 5) {
19      errUser.style.display = 'block';
20      valid = false;
21    } else {
22      errUser.style.display = 'none';
23    }
24    // Password validation
25    if (password.value.length < 8) {
26      errPass.style.display = 'block';
27      valid = false;
28    } else {
29      errPass.style.display = 'none';
30    }
31    if (!valid) {
32      e.preventDefault();
33    }
34  });
35
36  // Show/hide password
37  showPass.addEventListener('change', function() {
38    if (showPass.checked) {
39      password.type = 'text';
40    } else {
41      password.type = 'password';
42    }
43  });
44
45  // Responsive menu
46  const btnMenu = document.getElementById('btn-menu');
47  const menuList = document.getElementById('menu-list');
48  btnMenu.addEventListener('click', function() {
49    menuList.classList.toggle('active');
50  });
51});
52
53 // Function for inline onclick (optional, for compatibility)
54 function togglePassword() {
55  var pass = document.getElementById('password');
56  var show = document.getElementById('show-pass');
57  pass.type = show.checked ? 'text' : 'password';
58}

```

## Output:



## BAB VI

### RESPONSIVE WEB DESIGN

Perkembangan teknologi perangkat bergerak (*mobile devices*) telah mengubah lanskap internet secara drastis. Jika satu dekade lalu mayoritas pengguna mengakses situs web melalui komputer *desktop*, kini statistik global menunjukkan bahwa akses melalui perangkat seluler seperti ponsel pintar (*smartphone*) dan tablet telah mendominasi lalu lintas internet. Perubahan perilaku ini menuntut para pengembang web untuk meninggalkan metode desain statis tradisional dan beralih ke pendekatan yang lebih adaptif. Bab ini akan mengupas tuntas konsep Responsive Web Design (RWD), sebuah metodologi desain yang memungkinkan satu situs web untuk tampil optimal di berbagai ukuran layar, mulai dari jam tangan pintar hingga layar monitor berukuran besar.

#### 6.1 Konsep Responsive Design dan Mobile-First

##### 6.1.1 Definisi Responsive Web Design (RWD)

Secara fundamental, *Responsive Web Design* (RWD) bukanlah sebuah teknologi tunggal, melainkan sekumpulan teknik dan prinsip desain yang bekerja bersama-sama. Istilah ini pertama kali diperkenalkan oleh Ethan Marcotte pada tahun 2010. Inti dari RWD adalah fluiditas; elemen-elemen web seperti tata letak (*layout*), gambar, dan tipografi dirancang agar fleksibel dan mampu menyesuaikan diri dengan "wadah" (*viewport*) tempat mereka ditampilkan.

Sebelum adanya RWD, pengembang web sering kali harus membuat dua versi situs web yang berbeda: satu untuk desktop (misalnya [www.website.com](http://www.website.com)) dan satu lagi sub-domain khusus untuk seluler (misalnya [m.website.com](http://m.website.com)). Pendekatan ini memiliki banyak kelemahan, seperti duplikasi konten, upaya pemeliharaan (*maintenance*) yang ganda, dan masalah SEO (*Search Engine Optimization*). Dengan RWD, kita hanya memelihara satu basis kode HTML dan CSS yang cerdas untuk melayani semua perangkat.

Tiga pilar teknis utama dalam RWD adalah:

- Grid Cair (*Fluid Grids*): Sistem tata letak yang menggunakan satuan relatif (seperti persentase % atau unit fr) alih-alih satuan absolut (seperti px). Ini memastikan kolom tidak kaku dan bisa melebar atau menyempit.
- Media Fleksibel (*Flexible Media*): Gambar dan video yang dibatasi agar tidak pernah melebihi lebar wadahnya (biasanya menggunakan max-width: 100%).
- Media Queries: Fitur CSS3 yang memungkinkan penerapan gaya yang berbeda berdasarkan kondisi spesifik perangkat, seperti lebar layar, orientasi, atau resolusi.

### 6.1.2 Konsep Mobile-First

Dalam mengimplementasikan RWD, terdapat dua strategi alur kerja utama:

- Desktop-First (Graceful Degradation): Pendekatan tradisional di mana pengembang mendesain versi desktop yang lengkap terlebih dahulu, kemudian "memangkas" atau menyesuaikan elemen-elemen tersebut agar muat di layar yang lebih kecil. Kelemahan pendekatan ini adalah sering kali kode CSS menjadi berat karena memuat gaya desktop yang tidak diperlukan oleh perangkat seluler, yang pada akhirnya memperlambat waktu muat (*loading time*) di ponsel.
- Mobile-First (Progressive Enhancement): Ini adalah standar industri modern saat ini. Desain dimulai dari layar terkecil (ponsel) dengan fitur-fitur esensial saja. Kode CSS dasar ditulis untuk tampilan seluler tanpa menggunakan *Media Query*. Kemudian, seiring bertambahnya lebar layar (misalnya ke tablet atau desktop), kita menambahkan kompleksitas desain dan fitur tambahan menggunakan *Media Query* (`min-width`).

Keunggulan Mobile-First:

- Kinerja Lebih Cepat: Perangkat seluler dengan koneksi data terbatas hanya mengunduh aset dan kode yang benar-benar mereka butuhkan.

- Fokus pada Konten: Ruang layar yang sempit memaksa desainer untuk memprioritaskan konten yang paling penting bagi pengguna.
- Skalabilitas: Lebih mudah membangun kompleksitas ke atas (menambah fitur) daripada mencoba menyederhanakan sistem yang sudah rumit ke bawah.

### 6.1.3 Viewport Meta Tag

Salah satu komponen teknis terpenting dalam RWD yang sering terlupakan oleh pemula adalah tag meta viewport. Tanpa tag ini, browser pada perangkat seluler akan berperilaku seolah-olah mereka adalah layar desktop lebar, lalu melakukan *zoom-out* hingga seluruh halaman muat dalam layar kecil, membuat teks menjadi sangat kecil dan tidak terbaca.



```
1 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Tag berikut wajib disisipkan di dalam elemen `<head>` HTML:

Penjelasan Atribut:

- `width=device-width`: Menginstruksikan browser untuk mengatur lebar halaman web agar sama dengan lebar fisik layar perangkat.
- `initial-scale=1.0`: Mengatur tingkat zoom awal ke 100% (ukuran normal) saat halaman pertama kali dimuat.

## 6.2 Media Queries CSS

Media Queries adalah mekanisme CSS3 yang memungkinkan konten ditampilkan secara berbeda tergantung pada perangkat yang merender konten tersebut. Ini adalah "otak" dari desain responsif yang menentukan kapan tata letak harus berubah bentuk. Media query diawali dengan `@media`, diikuti oleh kondisi breakpoint.

### 6.2.1 Sintaksis dan Struktur Logika

Sebuah blok Media Query terdiri dari tipe media (opsional) dan satu atau lebih ekspresi kondisi (seperti lebar layar). Jika kondisi bernilai true, maka aturan CSS di dalam blok tersebut akan diterapkan.

Struktur dasarnya adalah:

```
● ● ●  
1 @media tipe-media and (kondisi media) {  
2   /* aturan CSS di sini */  
3 }
```

- `@media`: Kata kunci untuk memulai query.
- `screen`: Tipe media yang paling umum, menargetkan layar komputer, tablet, dan ponsel.
- `and`: Operator logika untuk menggabungkan kondisi.
- `min-width: ...`: Fitur media yang paling sering digunakan dalam pendekatan *Mobile-First*. Ini berarti "terapkan gaya ini jika lebar layar MINIMAL sekian".

Contoh Penulisan:

```
● ● ●  
1 @media screen and (max-width: 768px) {  
2   .container {  
3     padding: 10px;  
4   }  
5   .container-form {  
6     max-width: 100%;  
7     padding: 20px 15px 15px 15px;  
8   }  
9 }
```

### 6.2.2 Penentuan Titik Henti (Breakpoints)

*Breakpoint* adalah titik piksel tertentu di mana tampilan web akan berubah drastis (misalnya dari 1 kolom menjadi 2 kolom). Meskipun tidak ada standar baku karena variasi perangkat yang tak terbatas, *breakpoint* umum yang sering digunakan dalam industri didasarkan pada kategori perangkat berikut:

- Perangkat Seluler (Ponsel): Lebar < 768px (Biasanya gaya default tanpa media query).

- Tablet (Portrait): Lebar  $\geq$  768px.
- Laptop/Desktop Kecil: Lebar  $\geq$  992px.
- Desktop Layar Lebar: Lebar  $\geq$  1200px.

Contoh Implementasi Detail CSS:

```
/* 1. Basis gaya (Mobile) - Tidak perlu media query */
body {
    font-size: 14px;
    background-color: #ffffff; /* Warna latar belakang Putih */
}
.sidebar {
    display: none;
}

/* 2. Tampilan Tablet */
@media screen and (min-width: 768px) {
    body {
        font-size: 16px;
        background-color: #f9f9f9; /* Warna latar belakang Abu-abu terang */
    }
    .sidebar {
        display: block; /* Tampilkan sidebar di tablet */
        width: 30%; /* Untuk memastikan sidebar menempati 30% lebar layar */
        background-color: #e0e0e0; /* Warna latar belakang Abu-abu */
        position: fixed; /* Untuk memastikan sidebar tetap di tempat saat scroll */
        height: 100%; /* Untuk memastikan sidebar menempati seluruh tinggi layar */
        float: left; /* Untuk memastikan sidebar berada di sebelah kiri */
    }
}
```

```

}

.konten-utama {
    margin-left: 30%; /* Memberi ruang untuk sidebar */
    width: 70%; /* Konten utama menempati 70% lebar
    layar */
    float: left; /* Untuk memastikan konten utama
    berada di sebelah kanan sidebar */
}

/* 3. Tampilan Desktop */

@media screen and (min-width: 1024px) {
    .container {
        width: 960px; /* Lebar konten utama di desktop */
        margin: 0 auto; /* Pusatkan konten utama */
    }
    .sidebar {
        width: 25%; /* Lebar sidebar di desktop */
    }
    .konten-utama {
        margin-left: 25%; /* Memberi ruang untuk sidebar */
        width: 75%; /* Konten utama menempati 75% lebar
        layar */
    }
}

```

```

1  /* 1. Basis gaya (Mobile) - Tidak perlu media query */
2  body {
3    font-size: 14px;
4    background-color: #ffffff; /* Warna Latar belakang Putih */
5  }
6  .sidebar {
7    display: none;
8  }
9
10 /* 2. Tampilan Tablet */
11 @media screen and (min-width: 768px) {
12   body {
13     font-size: 16px;
14     background-color: #f9f9f9; /* Warna Latar belakang Abu-abu terang */
15   }
16   .sidebar {
17     display: block; /* Tampilkan sidebar di tablet */
18     width: 30%; /* Untuk memastikan sidebar menempati 30% Lebar Layar */
19     background-color: #e0e0e0; /* Warna Latar belakang Abu-abu */
20     position: fixed; /* Untuk memastikan sidebar tetap di tempat saat scroll */
21     height: 100%; /* Untuk memastikan sidebar menempati seluruh tinggi Layar */
22     float: left; /* Untuk memastikan sidebar berada di sebelah kiri */
23   }
24   .konten-utama {
25     margin-left: 30%; /* Memberi ruang untuk sidebar */
26     width: 70%; /* Konten utama menempati 70% Lebar Layar */
27     float: left; /* Untuk memastikan konten utama berada di sebelah kanan sidebar */
28   }
29 }
30
31 /* 3. Tampilan Desktop */
32 @media screen and (min-width: 1024px) {
33   .container {
34     width: 960px; /* Lebar konten utama di desktop */
35     margin: 0 auto; /* Pusatkan konten utama */
36   }
37   .sidebar {
38     width: 25%; /* Lebar sidebar di desktop */
39   }
40   .konten-utama {
41     margin-left: 25%; /* Memberi ruang untuk sidebar */
42     width: 75%; /* Konten utama menempati 75% Lebar Layar */
43   }
44 }

```

### 6.2.3 Fitur Media Lainnya

Selain width, Media Queries juga bisa mendeteksi fitur lain yang berguna untuk UX:

- **Orientation:** Mendeteksi apakah perangkat dalam mode portrait (tegak) atau landscape (mendatar).

```

1  /* fitur media Lainnya */
2  /* Orientasi */
3  @media screen and (orientation: landscape) {
4    /* Gaya khusus saat Handphone dimiringkan */
5    body {
6      font-size: 18px;
7      background-color: #f0f0f0; /* Warna Latar belakang Abu-abu terang */
8    }
9  }

```

- **Resolution:** Mendeteksi kepadatan piksel layar (berguna untuk layar Retina/High DPI) untuk menyajikan gambar beresolusi tinggi.

```

● ● ●
1 /* Resolusi */
2 @media screen and (min-resolution: 2dppx) {
3   /* Gaya khusus untuk Layar dengan resolusi tinggi */
4   body {
5     font-size: 18px;
6     background-color: #f0f0f0; /* Warna Latar belakang Abu-abu terang */
7   }
8 }
```

## 6.3 Layout Responsif dengan Flexbox dan Grid

Sebelum kehadiran CSS3 modern, pembuatan tata letak responsif sangat bergantung pada properti float dan teknik *clearing* yang rumit dan rentan kesalahan (*buggy*). Saat ini, CSS menyediakan dua sistem tata letak yang sangat kuat: Flexbox (untuk 1 dimensi) dan Grid (untuk 2 dimensi).

### 6.3.1 Flexbox Responsif

Flexbox dirancang untuk mengatur distribusi ruang antar elemen dalam sebuah antarmuka dan memiliki kemampuan penyelarasan (*alignment*) yang sangat baik. Flexbox sangat ideal untuk komponen baris tunggal seperti *Navbar*, daftar produk horizontal, atau penyusunan tombol.

Properti Kunci untuk Responsivitas:

- *flex-wrap: wrap;*: Ini adalah properti "ajaib" untuk responsivitas. Secara default, Flexbox akan memaksa semua elemen berada dalam satu baris. Dengan *wrap*, elemen yang tidak muat dalam lebar layar akan otomatis turun ke baris berikutnya.
- *flex-direction*: Memungkinkan kita mengubah arah tata letak dengan mudah. Pada seluler kita bisa menggunakan column (vertikal), dan pada desktop kita ubah menjadi row (horizontal) hanya dengan satu baris kode di dalam *Media Query*.
- *flex-grow & flex-shrink*: Mengizinkan item untuk membesar mengisi ruang kosong atau mengecil agar muat, tanpa perlu hitungan piksel yang kaku.

Contoh Kasus Flexbox:



```
1 /* Flexbox dan Grid */
2 .galeri-foto {
3   display: flex;
4   flex-wrap: wrap; /*Kunci responsive */
5   gap: 10px;
6 }
7 .foto-item{
8   flex: 1 1 300px;
9   /*
10  flex-grow: 1; (bisa membesar)
11  flex-shrink: 1; (bisa mengecil)
12  flex-basis : 300px; (ukuran dasar 300px)
13 */
14 }
```

### 6.3.2 CSS Grid Responsif

CSS Grid adalah sistem tata letak dua dimensi (baris dan kolom) yang paling komprehensif di web saat ini. Berbeda dengan Flexbox yang menangani konten satu dimensi, Grid dirancang untuk menangani tata letak halaman secara keseluruhan.

Fitur Kunci untuk Responsivitas:

- *repeat(auto-fit, ...)*: Fungsi ini memungkinkan pembuatan kolom sebanyak mungkin yang muat dalam container tanpa perlu mendefinisikan jumlah kolom secara eksplisit.
- *minmax()*: Menetapkan batas minimal dan maksimal ukuran kolom.
- *Unit fr (Fraction)*: Unit fleksibel yang membagi ruang sisa yang tersedia.

Contoh Kekuatan CSS Grid:



```
1 /* CSS Grid responsive */
2 .layout-halaman {
3   display: grid;
4   /* membuat grid responsif tanpa menggunakan media query */
5   grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
6   gap: 15px;
7 }
```

## 6.4 Penggunaan Framework CSS Populer

Dalam ekosistem pengembangan web profesional, efisiensi waktu adalah hal yang krusial. Meskipun memahami CSS murni (Native) adalah fondasi wajib, banyak pengembang memilih menggunakan Framework CSS untuk mempercepat proses pembuatan situs web responsif. Framework menyediakan kumpulan kode CSS (dan kadang JavaScript) yang telah diuji lintas browser dan perangkat.

### 6.4.1 Klasifikasi Framework CSS

Framework CSS umumnya dibagi menjadi dua paradigma utama:

- Component-Based Framework (Contoh: Bootstrap, Foundation, Bulma) Jenis ini menyediakan komponen UI siap pakai yang lengkap dan matang secara visual, seperti tombol, *navbar*, *modal*, *card*, dan *carousel*. Pengembang cukup menyalin struktur HTML dan menambahkan *class* tertentu.
  - Kelebihan: Sangat cepat untuk membuat prototipe atau situs admin; desain konsisten.
  - Kekurangan: Situs cenderung terlihat "mirip Bootstrap" jika tidak dikustomisasi; ukuran file bawaan bisa cukup besar karena memuat banyak gaya yang mungkin tidak dipakai.
  - Sistem Grid Bootstrap: Menggunakan sistem 12 kolom yang sangat populer. Contoh: col-md-6 artinya elemen mengambil 6 dari 12 kolom (setengah lebar) pada layar medium.
- Utility-First Framework (Contoh: Tailwind CSS, Windi CSS) Paradigma modern yang tidak menyediakan komponen jadi (seperti tombol siap pakai), melainkan menyediakan kelas-kelas utilitas tingkat rendah yang merepresentasikan properti CSS tunggal.
  - Kelebihan: Fleksibilitas desain tanpa batas (tidak terikat gaya bawaan); ukuran file CSS produksi sangat kecil (karena fitur *Tree Shaking* membuang kelas yang tidak dipakai); tidak perlu berpindah-pindah antara file HTML dan CSS.

- Kekurangan: Kode HTML menjadi terlihat "kotor" dan panjang karena banyaknya nama kelas (contoh: `class="bg-blue-500 text-white p-4 rounded shadow-lg hover:bg-blue-600"`).
- Responsivitas di Tailwind: Menggunakan prefix `sm:`, `md:`, `lg:`, `xl:` langsung pada kelas utilitas. Contoh: `w-full md:w-1/2` berarti lebar penuh di seluler, dan lebar setengah di desktop.

#### **6.4.2 Memilih Framework yang Tepat**

Pemilihan antara CSS Native, Bootstrap, atau Tailwind bergantung pada kebutuhan proyek:

- Gunakan CSS Native jika Anda sedang belajar dasar-dasar web, membutuhkan kontrol penuh atas setiap piksel, atau membuat situs sederhana yang sangat ringan.
- Gunakan Bootstrap jika Anda membutuhkan komponen UI standar dengan cepat, bekerja dalam tim besar dengan *deadline* ketat, dan desain unik bukan prioritas utama.
- Gunakan Tailwind CSS jika Anda menginginkan desain kustom yang unik, performa tinggi, dan bersedia mempelajari sintaks utilitas baru.

Pemahaman mendalam tentang dasar-dasar CSS (seperti Box Model, Flexbox, dan Position) tetap menjadi prasyarat mutlak sebelum menggunakan framework apa pun, karena framework hanyalah alat bantu untuk menulis CSS standar dengan lebih efisien.

## BAB VII

### INTEGRASI HTML, CSS, DAN JAVASCRIPT

Setelah mempelajari ketiga pilar utama pengembangan web—HTML untuk struktur, CSS untuk presentasi visual, dan JavaScript untuk logika interaktif—secara terpisah pada bab-bab sebelumnya, kini saatnya menyatukan komponen-komponen tersebut. Bab ini akan membahas secara mendalam bagaimana ketiga teknologi ini berkolaborasi membentuk sebuah ekosistem web yang utuh. Integrasi yang baik tidak hanya sekadar "menempelkan" kode, melainkan memahami bagaimana peramban (*browser*) memproses interaksi antara DOM (*Document Object Model*) dan CSSOM (*CSS Object Model*) untuk menciptakan pengalaman pengguna yang dinamis.

#### 7.1 Membuat Halaman Web Interaktif Sederhana

##### 7.1.1 Filosofi "Separation of Concerns"

Dalam pengembangan web modern, prinsip utama yang dipegang adalah *Separation of Concerns* (Pemisahan Tanggung Jawab). Prinsip ini menekankan bahwa setiap teknologi harus fokus pada tugasnya masing-masing namun tetap terintegrasi dengan baik:

- HTML (Content Layer): Hanya bertanggung jawab atas semantik dan hierarki informasi. HTML tidak boleh mengandung atribut gaya (seperti style="..." atau bgcolor) ataupun logika baris program.
- CSS (Presentation Layer): Bertanggung jawab penuh atas tata letak, warna, tipografi, dan responsivitas. CSS tidak boleh mengubah konten HTML.
- JavaScript (Behavior Layer): Bertanggung jawab menangani interaksi, manipulasi data, dan komunikasi dengan server.

Integrasi dilakukan dengan cara "mengaitkan" lapisan-lapisan ini menggunakan *selectors* (untuk CSS) dan *DOM hooks* seperti ID atau Class (untuk JavaScript).

### 7.1.2 Mekanisme Pemuatan Halaman (Rendering Flow)

Memahami urutan eksekusi sangat penting agar halaman web berjalan lancar tanpa *error*.

1. Parsing HTML: Browser membaca dokumen HTML dari atas ke bawah.
2. Fetching Resources: Saat menemukan tag `<link rel="stylesheet">`, browser memuat CSS secara paralel. Namun, saat menemukan tag `<script>`, browser secara default akan menghentikan rendering HTML untuk mengeksekusi script tersebut (kecuali menggunakan atribut defer atau async).
3. Best Practice Integrasi: Oleh karena itu, skrip JavaScript yang memanipulasi elemen visual sebaiknya diletakkan tepat sebelum tag penutup `</body>`. Hal ini memastikan seluruh elemen HTML sudah dimuat ke dalam DOM sebelum JavaScript mencoba mengaksesnya, mencegah error umum seperti "*Cannot read property of null*".

Contoh Implementasi Sederhana, Fitur Dark Mode Toggle Fitur ini mengubah tema warna website secara instan dengan memanipulasi class pada elemen body.

Kode HTML (index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Intergrasi Dasar</title>
    <link rel="stylesheet" href="rendering.css">
</head>
<body>
    <div class="container">
```

```

<h1>Dark Mode & Light Mode</h1>

<p>Tekan tombol di bawah untuk mengganti mode tampilan halaman.</p>

<button id="btn-mode">Ganti Mode</button>

</div>

<script src="rendering.js"></script>

</body>

</html>

```



```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Intergrasi Dasar</title>
7      <link rel="stylesheet" href="rendering.css">
8  </head>
9  <body>
10     <div class="container">
11         <h1>Dark Mode & Light Mode</h1>
12         <p>Tekan tombol di bawah untuk mengganti mode tampilan halaman.</p>
13         <button id="btn-mode">Ganti Mode</button>
14     </div>
15     <script src="rendering.js"></script>
16 </body>
17 </html>

```

### Kode CSS (style.css)

```

body {
    background-color: #ffffff;
    color: #333333;
    transition: background-color 0.3s, color 0.3s;
    font-family: Arial, Helvetica, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

```

```
body.dark-mode {  
    background-color: #121212;  
    color: #f0f0f0;  
}  
  
button {  
    padding: 10px 20px;  
    font-size: 16px;  
    cursor: pointer;  
    border: none;  
    border-radius: 5px;  
    background-color: #4caf50;  
    color: white;  
    transition: background-color 0.3s;  
}
```



```
1 body {  
2     background-color: #ffffff;  
3     color: #333333;  
4     transition: background-color 0.3s, color 0.3s;  
5     font-family: Arial, Helvetica, sans-serif;  
6     display: flex;  
7     justify-content: center;  
8     align-items: center;  
9     height: 100vh;  
10    margin: 0;  
11 }  
12  
13 body.dark-mode {  
14     background-color: #121212;  
15     color: #f0f0f0;  
16 }  
17  
18 button {  
19     padding: 10px 20px;  
20     font-size: 16px;  
21     cursor: pointer;  
22     border: none;  
23     border-radius: 5px;  
24     background-color: #4caf50;  
25     color: white;  
26     transition: background-color 0.3s;
```

Kode JavaScript (script.js):

```
const tombol = document.getElementById('btn-mode');

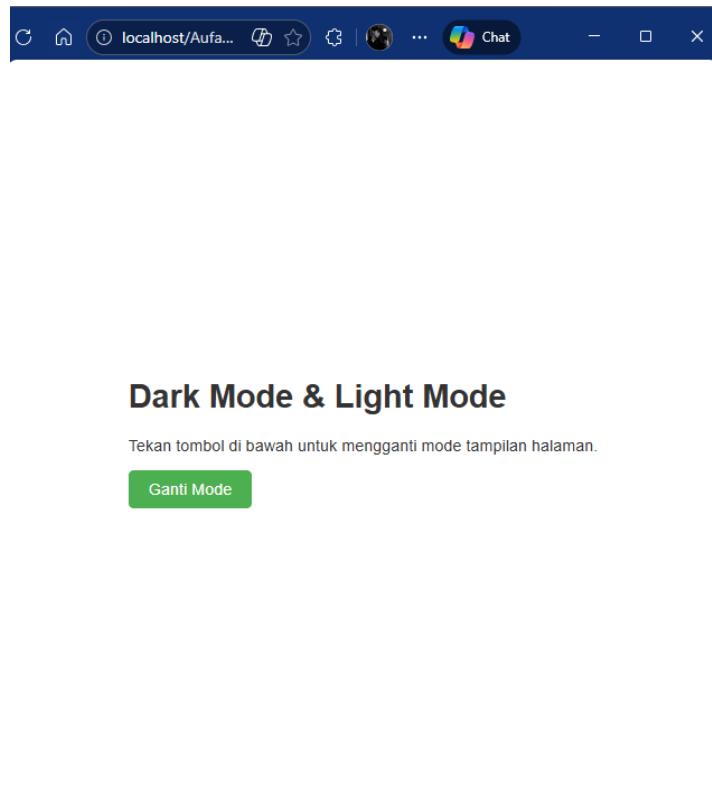
tombol.addEventListener('click', function() {
    document.body.classList.toggle('dark-mode');
});
```



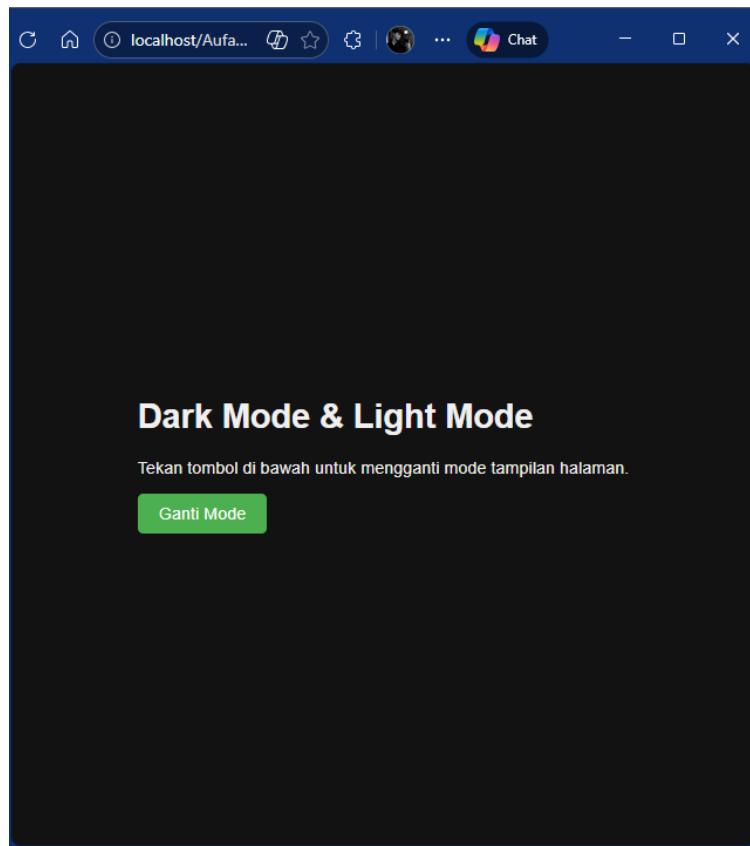
```
1 const tombol = document.getElementById('btn-mode');
2 tombol.addEventListener('click', function() {
3     document.body.classList.toggle('dark-mode');
4});
```

Output:

Light Mode (Default)



Dark Mode (Setelah klik button “Ganti Mode”)



## 7.2 Form Input dengan Validasi JavaScript

Formulir adalah titik interaksi utama antara pengguna dan sistem. Sebelum data dikirim ke server untuk diproses, data tersebut wajib divalidasi di sisi klien (*Client-Side Validation*) untuk memastikan integritas dan kelengkapannya.

### 7.2.1 Pentingnya Validasi Sisi Klien

Validasi menggunakan JavaScript memberikan umpan balik instan kepada pengguna tanpa perlu memuat ulang halaman (*page reload*). Ini meningkatkan *User Experience (UX)* secara signifikan.

- Efisiensi Server: Mengurangi beban server karena data yang tidak valid (misalnya email tanpa simbol '@' atau password kosong) ditolak sejak awal di browser.
- Kecepatan: Pengguna langsung mengetahui letak kesalahan mereka segera setelah mengetik atau menekan tombol kirim.

### 7.2.2 Teknik Manipulasi Class untuk Validasi

Alih-alih mengubah gaya elemen secara langsung menggunakan JavaScript (contoh: `input.style.borderColor = 'red'`), pendekatan yang lebih profesional adalah dengan memanipulasi Class CSS.

- CSS menyediakan class khusus, misalnya `.error` (border merah) atau `.success` (border hijau).
- JavaScript hanya bertugas menambahkan atau menghapus class tersebut menggunakan `element.classList.add('error')` atau `element.classList.remove('error')`.
- Pendekatan ini menjaga kode JavaScript tetap bersih dari urusan desain visual.

### 7.2.3 Mencegah Aksi Default (`preventDefault()`)

Secara default, tombol `submit` pada form akan mengirim data dan menyegarkan halaman. JavaScript menggunakan metode `event.preventDefault()` untuk membatalkan perilaku standar ini, memberikan waktu bagi skrip untuk memeriksa validitas data terlebih dahulu.

Contoh Implementasi: Validasi Form Login

Kode HTML (form.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Form Login</title>
    <link rel="stylesheet" href="form.css">
</head>
<body>
    <form id="form-login">
```

```

<div class="grup-input">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" placeholder="Min. 5 karakter" required>
        <span class="pesan-error" id="err-user">Username minimal 5 karakter</span>
</div>
<div class="grup-input">
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" placeholder="Min. 8 karakter" required>
        <span class="pesan-error" id="err-pass">Password minimal 8 karakter</span>
</div>
<button type="submit">Login</button>
</form>
<script src="form.js"></script>
</body>
</html>

```



```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Form Login</title>
7      <link rel="stylesheet" href="form.css">
8  </head>
9  <body>
10     <form id="form-login">
11         <div class="grup-input">
12             <label for="username">Username:</label>
13             <input type="text" id="username" name="username" placeholder="Min. 5 karakter" required>
14             <span class="pesan-error" id="err-user">Username minimal 5 karakter</span>
15         </div>
16         <div class="grup-input">
17             <label for="password">Password:</label>
18             <input type="password" id="password" name="password" placeholder="Min. 8 karakter" required>
19             <span class="pesan-error" id="err-pass">Password minimal 8 karakter</span>
20         </div>
21         <button type="submit">Login</button>
22     </form>
23     <script src="form.js"></script>
24 </body>
25 </html>

```

Kode CSS (style-form.css):

```
.grup-input {  
    margin-bottom: 15px;  
}  
  
input {  
    padding: 8px;  
    width: 100%;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
    box-sizing: border-box;  
}  
  
/* Style validasi */  
  
input:invalid {  
    border-color: #e74c3c;  
}  
  
input:valid {  
    border-color: #2ecc71;  
}  
  
.pesan-error {  
    color: #e74c3c;  
    font-size: 0.9em;  
    display: none;  
}
```



```
1 .grup-input {
2   margin-bottom: 15px;
3 }
4 input {
5   padding: 8px;
6   width: 100%;
7   border: 1px solid #ccc;
8   border-radius: 4px;
9   box-sizing: border-box;
10 }
11 /* Style validasi */
12 input:invalid {
13   border-color: #e74c3c;
14 }
15 input:valid {
16   border-color: #2ecc71;
17 }
18 .pesan-error {
19   color: #e74c3c;
20   font-size: 0.9em;
21   display: none;
22 }
```

Kode JavaScript (validasi.js):

```
const form = document.getElementById("form-login");

const username = document.getElementById("username");
const password = document.getElementById("password");
const errUser = document.getElementById("err-user");
const errPass = document.getElementById("err-pass");
const showPass = document.getElementById("show-pass");

// Hide error messages initially
errUser.style.display = "none";
errPass.style.display = "none";

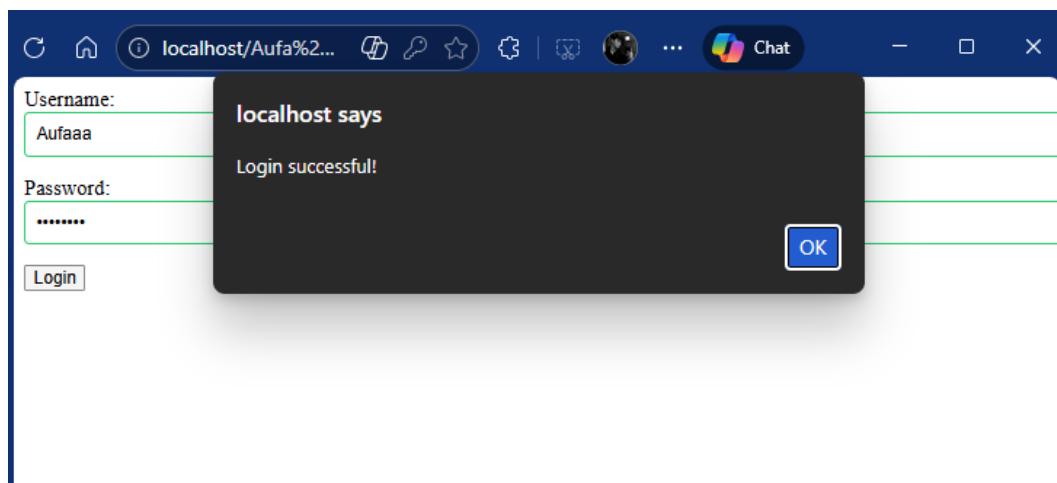
form.addEventListener("submit", function (e) {
  e.preventDefault();
  if (username.value.length < 5) {
    username.classList.add("is-invalid");
    username.classList.remove("is-valid");
    errUser.style.display = "block";
  } else {
```

```
username.classList.add("is-valid");
username.classList.remove("is-invalid");
errUser.style.display = "none";
alert("Login successful!");
});
```

```
● ● ●

1 const form = document.getElementById("form-login");
2 const username = document.getElementById("username");
3 const password = document.getElementById("password");
4 const errUser = document.getElementById("err-user");
5 const errPass = document.getElementById("err-pass");
6 const showPass = document.getElementById("show-pass");
7
8 // Hide error messages initially
9 errUser.style.display = "none";
10 errPass.style.display = "none";
11
12 form.addEventListener("submit", function (e) {
13   e.preventDefault();
14   if (username.value.length < 5) {
15     username.classList.add("is-invalid");
16     username.classList.remove("is-valid");
17     errUser.style.display = "block";
18   } else {
19     username.classList.add("is-valid");
20     username.classList.remove("is-invalid");
21     errUser.style.display = "none";
22     alert("Login successful!");
23   }
24 });
```

Output:



### 7.3 Animasi Interaktif Menggunakan CSS + JS

Animasi modern tidak lagi dibuat sepenuhnya dengan JavaScript yang berat. Sebaliknya, kita menggabungkan performa tinggi dari CSS Transitions dengan logika kontrol dari JavaScript.

#### 7.3.1 Konsep "State-Based UI"

Dalam pola ini, JavaScript tidak menghitung animasi frame-demi-frame. Tugas JavaScript hanyalah mengubah "status" (*state*) elemen, sedangkan CSS yang menangani transisi visualnya.

Contoh Alur Logika:

1. **CSS:** Mendefinisikan status awal (misalnya *opacity: 0*) dan status akhir (misalnya *class .muncul* dengan *opacity: 1*), serta properti *transition: all 0.5s*.
2. **JavaScript:** Mendeteksi kejadian (misalnya *scroll* atau *click*), lalu menyematkan class *.muncul* ke elemen target.
3. **Hasil:** Elemen berubah secara halus karena CSS menangani interpolasi perubahan nilai tersebut.

#### 7.3.2 Keunggulan Kombinasi CSS + JS

1. **Performa Hardware Acceleration:** Animasi CSS (terutama *transform* dan *opacity*) sering kali diproses oleh GPU (*Graphics Processing Unit*), sehingga jauh lebih mulus dibandingkan animasi JS murni yang membebani CPU.
2. **Kode Lebih Ringkas:** Logika animasi visual tetap berada di file CSS, menjaga file JavaScript tetap fokus pada logika bisnis.

Contoh Implementasi: Animasi Fade-In Saat Scroll

Kode CSS:

```
.kotak-info {  
    opacity: 0.9;  
    transform: translateY(20px);  
}
```

```

        transition: opacity 0.6s ease-out, transform 0.6s
        ease-out;
    }

/* Class ini akan ditambahkan JS saat elemen terlihat */

.kotak-info.tampil {
    opacity: 1;
    transform: translateY(0);
}

```



```

1 .kotak-info {
2     opacity: 0.9;
3     transform: translateY(20px);
4     transition: opacity 0.6s ease-out, transform 0.6s ease-out;
5 }
6 /* Class ini akan ditambahkan JS saat elemen terlihat */
7 .kotak-info.tampil {
8     opacity: 1;
9     transform: translateY(0);
10}
11

```

Kode JavaScript:

```

window.addEventListener('scroll', function() {
    const kotak = document.querySelector('.kotak-info');
    const posisiKotak =
    kotak.getBoundingClientRect().top;
    const posisiLayar = window.innerHeight;

    // Jika elemen masuk ke dalam layar
    if (posisiKotak < posisiLayar * 0.75) {
        kotak.classList.add('tampil');
    }
})
;
```



```
1 window.addEventListener('scroll', function() {
2   const kotak = document.querySelector('.kotak-info');
3   const posisiKotak = kotak.getBoundingClientRect().top;
4   const posisiLayar = window.innerHeight;
5
6   // Jika elemen masuk ke dalam Layar
7   if (posisiKotak < posisiLayar * 0.75) {
8     kotak.classList.add('tampil');
9   }
10});
```

## 7.4 Studi Kasus Mini Project: Halaman Landing Page Responsif

Sebagai implementasi akhir dari bab ini, kita akan membangun sebuah Landing Page sederhana untuk produk teknologi. Proyek ini menggabungkan struktur semantik HTML5, tata letak responsif CSS3, dan fitur interaktif JavaScript (Navbar Mobile & Smooth Scroll).

**Kode HTML (index.html):** Struktur dokumen menggunakan elemen semantik seperti `<header>`, `<section>`, dan `<footer>`.

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,
initial-scale=1.0">

  <title>Landing Page - Intergrasi Web</title>

  <link rel="stylesheet" href="LandingPage.css">

</head>

<body>

  <header id="beranda">

    <nav class="navbar">

      <div class="logo">TechFuture</div>

      <ul class="nav-links" id="nav-links">

        <li><a href="#beranda"
class="activate">Beranda</a></li>

        <li><a href="#tentang">Tentang</a></li>
```

```

        <li><a href="#kontak">Kontak</a></li>
    </ul>

    <div class="hamburger" id="hamburger">
        <span class="bar"></span>
        <span class="bar"></span>
        <span class="bar"></span>
    </div>
</nav>

<div class="hero-section">
    <h1>Selamat Datang di TechFuture</h1>
    <p>Menghubungkan Anda dengan Teknologi Masa Depan</p>
    <a href="#tentang" class="btn-hero">Pelajari Lebih Lanjut</a>
</header>

<section id="tentang" class="tentang-section">
    <h2>Tentang Kami</h2>
    <p>TechFuture adalah platform inovatif yang menyediakan informasi terkini tentang teknologi terbaru, tren digital, dan solusi teknologi untuk kehidupan sehari-hari.</p>
    <div class="tentang-grid">
        <div class="card">
            <h3>Berita Teknologi</h3>
            <p>Dapatkan update harian tentang perkembangan teknologi terbaru di seluruh dunia.</p>
        </div>
        <div class="card">
            <h3>Ulasan Produk</h3>
            <p>Baca ulasan mendalam tentang gadget dan perangkat teknologi terbaru.</p>
        </div>
        <div class="card">
            <h3>Tips & Trik</h3>

```

```
<p>Pelajari tips dan trik untuk memaksimalkan penggunaan teknologi dalam kehidupan Anda.</p>
</div>
</div>
</section>
<section id="kontak" class="kontak-section">
    <h2>Kontak Kami</h2>
    <p>Ingin tahu lebih banyak? Hubungi kami melalui formulir di bawah ini:</p>
    <form id="kontak-form" class="form-box">
        <div class="input-group">
            <label>Email Anda</label>
            <input type="email" id="email-input" placeholder="contoh@gmail.com">
            <small id="email-error" class="error-message">Masukkan email yang valid</small>
        </div>
        <div class="input-group">
            <label>Pesan Anda</label>
            <textarea id="message-input" placeholder="Tulis pesan Anda di sini..."></textarea>
            <small id="message-error" class="error-message">Pesan tidak boleh kosong</small>
        </div>
        <button id="submit-btn" type="submit">Kirim Pesan</button>
    </form>
    <footer class="footer">
        <p>© 2025 TechFuture Web Programming 1. All rights reserved.</p>
    </footer>
    <script src="LandingPage.js"></script>
</body>
</html>
```

**Kode CSS (style.css):** Menggunakan CSS Native dengan Flexbox dan Media Queries untuk responsivitas.

```
/* Reset dasar */

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Arial, sans-serif;
    background: #f6f8fa;
    color: #222;
    min-height: 100vh;
}

header {
    background: linear-gradient(120deg, #66a6ff 0%,
#89f7fe 100%);
    padding-bottom: 40px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.04);
}

.navbar {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 18px 8vw;
    background: transparent;
}

.logo {
    font-size: 1.7em;
    font-weight: bold;
    color: #fff;
```

```
letter-spacing: 1px;
}

.nav-links {
  display: flex;
  gap: 28px;
  list-style: none;
}

.nav-links li a {
  color: #fff;
  text-decoration: none;
  font-size: 1.08em;
  font-weight: 500;
  transition: color 0.2s;
  padding: 4px 10px;
  border-radius: 4px;
}

.nav-links li a.activate,
.nav-links li a:hover {
  background: #fff;
  color: #66a6ff;
}

.hamburger {
  display: none;
  flex-direction: column;
  cursor: pointer;
  gap: 4px;
}

.bar {
  width: 26px;
  height: 3px;
  background: #fff;
```

```
border-radius: 2px;
transition: 0.3s;
}

.hero-section {
    text-align: center;
    margin-top: 40px;
}

.hero-section h1 {
    color: #fff;
    font-size: 2.5em;
    margin-bottom: 12px;
    letter-spacing: 1px;
}

.hero-section p {
    color: #f0f8ff;
    font-size: 1.2em;
    margin-bottom: 22px;
}

.btn-hero {
    display: inline-block;
    background: #fff;
    color: #66a6ff;
    padding: 10px 28px;
    border-radius: 24px;
    font-size: 1.1em;
    font-weight: 600;
    text-decoration: none;
    box-shadow: 0 2px 8px rgba(0,0,0,0.07);
    transition: background 0.2s, color 0.2s;
}

.btn-hero:hover {
```

```
background: #66a6ff;
color: #fff;
}

.tentang-section {
padding: 60px 8vw 40px 8vw;
background: #fff;
}

.tentang-section h2 {
text-align: center;
font-size: 2em;
margin-bottom: 18px;
color: #2d3a4b;
}

.tentang-section p {
text-align: center;
margin-bottom: 32px;
color: #444;
}

.tentang-grid {
display: flex;
gap: 24px;
justify-content: center;
flex-wrap: wrap;
}

.card {
background: #f6f8fa;
border-radius: 10px;
box-shadow: 0 2px 8px rgba(0,0,0,0.06);
padding: 28px 22px;
flex: 1 1 220px;
min-width: 220px;
```

```
max-width: 320px;
text-align: center;
transition: transform 0.2s, box-shadow 0.2s;
}

.card:hover {
    transform: translateY(-6px) scale(1.03);
    box-shadow: 0 6px 18px rgba(102,166,255,0.13);
}

.card h3 {
    color: #66a6ff;
    margin-bottom: 10px;
}

.kontak-section {
    padding: 60px 8vw 40px 8vw;
    background: #fafdff;
}

.kontak-section h2 {
    text-align: center;
    font-size: 2em;
    margin-bottom: 18px;
    color: #2d3a4b;
}

.kontak-section p {
    text-align: center;
    margin-bottom: 28px;
    color: #444;
}

.form-box {
    max-width: 420px;
    margin: 0 auto;
    background: #fff;
```

```
border-radius: 10px;  
box-shadow: 0 2px 8px rgba(0,0,0,0.07);  
padding: 28px 22px 18px 22px;  
}  
.input-group {  
margin-bottom: 18px;  
display: flex;  
flex-direction: column;  
}  
.input-group label {  
margin-bottom: 6px;  
color: #2d3a4b;  
font-weight: 500;  
}  
.input-group input,  
.input-group textarea {  
padding: 10px 12px;  
border: 1.5px solid #ccc;  
border-radius: 5px;  
font-size: 1em;  
transition: border-color 0.2s;  
resize: none;  
}  
.input-group input:focus,  
.input-group textarea:focus {  
border-color: #66a6ff;  
outline: none;  
}  
.error-message {  
color: #e74c3c;  
font-size: 0.92em;
```

```
margin-top: 3px;  
display: none;  
}  
  
#submit-btn {  
width: 100%;  
padding: 10px 0;  
background: linear-gradient(90deg, #66a6ff 0%,  
#89f7fe 100%);  
color: #fff;  
border: none;  
border-radius: 5px;  
font-size: 1.1em;  
font-weight: 600;  
cursor: pointer;  
transition: background 0.2s;  
margin-top: 8px;  
}  
  
#submit-btn:hover {  
background: linear-gradient(90deg, #89f7fe 0%,  
#66a6ff 100%);  
}  
  
.footer {  
background: #222;  
color: #fff;  
text-align: center;  
padding: 18px 0 12px 0;  
font-size: 1em;  
margin-top: 0;  
}  
  
/* Responsive */  
@media (max-width: 900px) {  
.navbar {
```

```
    padding: 18px 3vw;
}

.tentang-section, .kontak-section {
    padding: 50px 3vw 30px 3vw;
}

}

@media (max-width: 700px) {

    .tentang-grid {
        flex-direction: column;
        gap: 18px;
        align-items: center;
    }

    .navbar {
        flex-wrap: wrap;
    }
}

@media (max-width: 600px) {

    .navbar {
        padding: 14px 2vw;
    }

    .logo {
        font-size: 1.2em;
    }

    .nav-links {
        position: absolute;
        top: 60px;
        right: 2vw;
        background: #66a6ff;
        flex-direction: column;
        gap: 0;
        width: 160px;
    }
}
```

```
border-radius: 8px;  
box-shadow: 0 2px 8px rgba(0,0,0,0.13);  
display: none;  
z-index: 10;  
}  
.nav-links.open {  
display: flex;  
}  
.nav-links li {  
border-bottom: 1px solid #fff2;  
width: 100%;  
}  
.nav-links li:last-child {  
border-bottom: none;  
}  
.nav-links li a {  
display: block;  
padding: 12px 18px;  
color: #fff;  
background: none;  
border-radius: 0;  
}  
.hamburger {  
display: flex;  
}  
}
```

**Kode JavaScript (main.js):** Menangani logika Hamburger Menu (Mobile) dan Validasi Form Kontak.

```
// Hamburger menu logic

const hamburger = document.getElementById('hamburger');
const navLinks = document.getElementById('nav-links');

hamburger.addEventListener('click', function () {
    navLinks.classList.toggle('open');
});

// Kontak form validation

const kontakForm = document.getElementById('kontak-form');

const emailInput = document.getElementById('email-input');

const messageInput = document.getElementById('message-input');

const emailError = document.getElementById('email-error');

const messageError = document.getElementById('message-error');

function validateEmail(email) {
    // Simple email regex
    return /^[^@\s]+@[^\s@]+\.\[^@\s]+\$/ .test(email);
}

kontakForm.addEventListener('submit', function (e) {
    let valid = true;
    // Email validation
    if (!validateEmail(emailInput.value.trim())) {
        emailError.style.display = 'block';
        emailInput.style.borderColor = '#e74c3c';
    }
})
```

```
    valid = false;

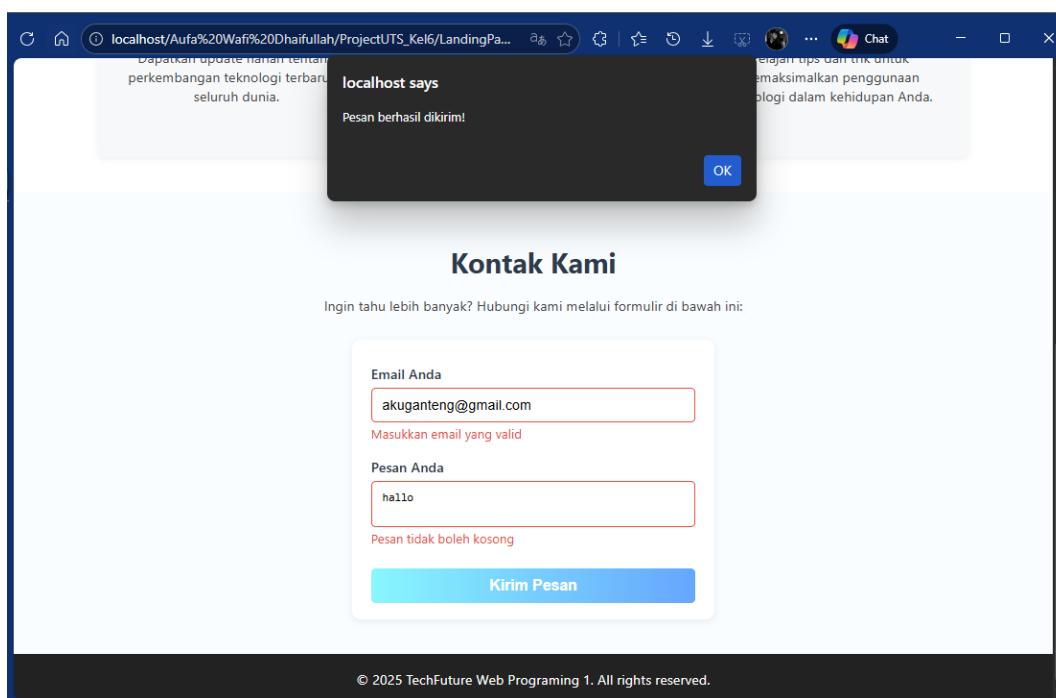
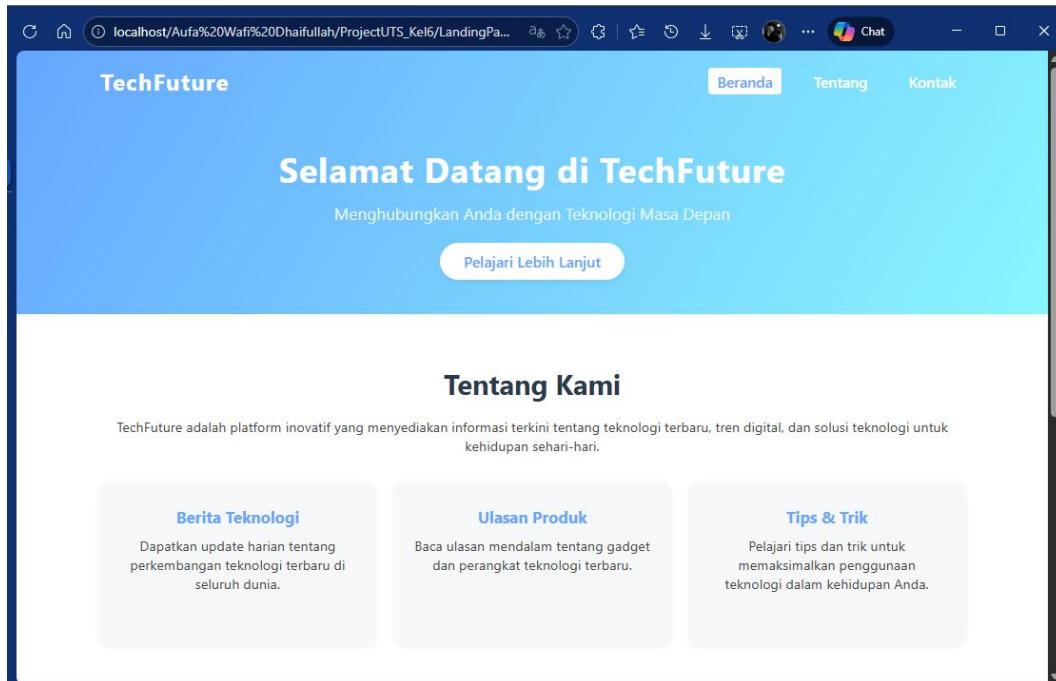
} else {
    emailError.style.display = 'none';
    emailInput.style.borderColor = '#66a6ff';
}

// Message validation

if (messageInput.value.trim() === '') {
    messageError.style.display = 'block';
    messageInput.style.borderColor = '#e74c3c';
    valid = false;
} else {
    messageError.style.display = 'none';
    messageInput.style.borderColor = '#66a6ff';
}

if (!valid) {
    e.preventDefault();
} else {
    alert('Pesan berhasil dikirim!');
    // kontakForm.submit(); // Uncomment jika ingin submit beneran
    e.preventDefault();
    emailInput.value = '';
    messageInput.value = '';
    emailInput.style.borderColor = '#ccc';
    messageInput.style.borderColor = '#ccc';
}
}) ;
```

## Output:



Output landing page ini adalah halaman utama web yang responsif, menampilkan menu navigasi, informasi tentang layanan, dan form kontak dengan validasi otomatis, serta tampilan modern yang nyaman di semua perangkat.

## BAB VIII

### PRAKTIK & PROYEK AKHIR

#### 8.1 Panduan Membangun Website Portofolio Sederhana

Pada subbab ini dibahas panduan dalam membangun sebuah website portofolio sederhana menggunakan HTML5, CSS3, dan JavaScript dasar. Website portofolio dipilih sebagai proyek akhir karena mampu merepresentasikan kemampuan dasar pengembangan web statis sekaligus mengintegrasikan seluruh materi yang telah dipelajari pada bab-bab sebelumnya.

Website portofolio sederhana umumnya terdiri dari beberapa bagian utama, yaitu halaman beranda (home), profil atau tentang saya (about), daftar proyek atau karya (projects), serta halaman kontak (contact). Struktur halaman dirancang menggunakan elemen semantik HTML5 agar dokumen lebih terstruktur, mudah dipahami, serta sesuai dengan standar pengembangan web modern.

Tahapan pembangunan website portofolio sederhana dapat dilakukan sebagai berikut:

1. Menentukan struktur halaman dan navigasi website
2. Membuat kerangka HTML menggunakan elemen semantik
3. Mengatur tampilan dan layout menggunakan CSS3
4. Menambahkan interaktivitas menggunakan JavaScript dasar

Pada tahap awal, fokus utama adalah membangun website statis yang rapi dan mudah digunakan. Website tidak harus memiliki desain yang kompleks, namun mampu menampilkan informasi dengan jelas dan konsisten. Prinsip separation of concerns juga diterapkan, yaitu memisahkan struktur (HTML), tampilan (CSS), dan logika interaksi (JavaScript) agar kode mudah dipelihara dan dikembangkan.

#### 8.2 Menerapkan HTML5, CSS3, dan JavaScript Dasar

Berdasarkan panduan pada subbab 8.1, tahap selanjutnya adalah melakukan penerapan atau implementasi kode program untuk membangun website portofolio sederhana. Implementasi ini mencakup pembuatan struktur HTML5, pengaturan

tampilan dengan CSS3, serta penggunaan JavaScript dasar untuk mendukung fungsi halaman.

### a. Implementasi Struktur HTML5

Struktur dasar website dibuat menggunakan elemen semantik HTML5 seperti <header>, <nav>, <section>, dan <footer>. Elemen-elemen ini membantu membagi halaman menjadi bagian-bagian yang jelas dan terstruktur.

Contoh implementasi HTML:

```
<!DOCTYPE html>

<html lang="id">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
        <title>Portofolio Aufa Wafi Dhaifullah</title>
        <link rel="stylesheet" href="style.css" />
    </head>
    <body>
        <header>
            <div class="container header-content">
                <h1 class="logo">PortoTech.</h1>
                <nav>
                    <ul id="menu-list">
                        <li><a href="#home">Home</a></li>
                        <li><a href="#about">Tentang</a></li>
                        <li><a href="#portfolio">Portofolio</a></li>
                        <li><a href="#contact">Kontak</a></li>
                    </ul>
                    <div class="menu-icon" id="hamburger-
menu">#9776;</div>
                </nav>
            </div>
        </header>
    </body>

```

```
</div>

</header>

<section id="home" class="hero">
  <div class="container">
    <h2>Halo, Saya Aufa Wafi Dhaifullah</h2>
    <p>Mahasiswa Teknik Informatika yang tertarik pada Web Development & Machine Learning.</p>
    <a href="#contact" class="btn">Hubungi Saya</a>
  </div>
</section>

<section id="about" class="about">
  <div class="container about-content">
    
    <div class="about-text">
      <h3>Tentang Saya</h3>
      <p>Saya adalah mahasiswa semester 5 Teknik Informatika di Universitas Pamulang (NIM: 231011401814). Saya memiliki minat pada pengembangan web (Front-end) dan implementasi Machine Learning seperti metode YOLO.</p>
      <h4 class="skills-title">Tech Stack & Tools</h4>
      <div class="skills-container">
        <div class="skill-group">
          <p class="skill-label">Editor & Tools</p>
          <div class="skills-list">
            <span class="skill-tag">VSCode</span>
            <span class="skill-tag">Git/GitHub</span>
            <span class="skill-tag">Firebase Console</span>
            <span class="skill-tag">Supabase</span>
          </div>
        </div>
        <div class="skill-group">
```

```
<p class="skill-label">Frontend</p>
<div class="skills-list">
    <span class="skill-tag">HTML</span>
    <span class="skill-tag">CSS</span>
    <span class="skill-
tag">JavaScript</span>
    <span class="skill-
tag">TypeScript</span>
</div>
</div>
<div class="skill-group">
    <p class="skill-label">Backend &
Database</p>
    <div class="skills-list">
        <span class="skill-tag">MongoDB</span>
        <span class="skill-tag">MySQL</span>
        <span class="skill-
tag">PostgreSQL</span>
        <span class="skill-tag">Firebase</span>
    </div>
</div>
</div>
</div>
</div>
</section>
<section id="portfolio" class="portfolio">
    <div class="container">
        <h3>Proyek Saya</h3>
        <div class="slideshow-wrapper">
            <div class="slideshow-container">
                <div class="slide fade">
                    <div class="project-card">
```

```

        <div class="project-image">
            
        </div>
        <div class="project-content">
            <h4>LAPOR AMAN</h4>
            <p>Web App pelaporan perundungan di
SMP Gelora Depok.</p>
        </div>
        </div>
        <div class="slide fade">
            <div class="project-card">
                <div class="project-image">
                    
                </div>
                <div class="project-content">
                    <h4>E-Ticketing RS</h4>
                    <p>Aplikasi web booking appointment rumah sakit.</p>
                </div>
                </div>
            </div>
            <button class="prev-btn"
id="prevBtn">#10094;</button>
            <button class="next-btn"
id="nextBtn">#10095;</button>
            <div class="dots-container">
                <span class="dot active"
onclick="currentSlide(1)"></span>

```

```

        <span class="dot"
onclick="currentSlide(2)"></span>
    </div>
</div>
</div>
</section>
<section id="contact" class="contact">
    <div class="container">
        <h3>Hubungi Saya</h3>
        <form id="contactForm">
            <input type="text" id="name"
placeholder="Nama Anda" />
            <input type="email" id="email"
placeholder="Email Anda" />
            <textarea id="message" rows="5"
placeholder="Pesan"></textarea>
            <button type="submit" class="btn">Kirim
Pesan</button>
        </form>
    </div>
</section>
<footer>
    <p>&copy; 2025 PortoTech - All Rights
Reserved</p>
</footer>
<script src="script.js"></script>
</body>
</html>

```

### **b. Implementasi CSS3 untuk Tampilan Website**

Setelah struktur HTML selesai, CSS3 digunakan untuk mengatur tampilan visual dan layout halaman agar lebih menarik dan mudah dibaca.

### Code CSS:

```
:root {  
    --primary: #2c3e50;  
    --secondary: #425e79;  
    --light: #f5f5f5;  
    --shadow: 0 4px 15px rgba(0, 0, 0, 0.1); }  
  
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box; }  
  
html {  
    scroll-behavior: smooth; }  
  
body {  
    background: var(--light);  
    color: #333;  
    line-height: 1.6;  
    font-family: sans-serif; }  
  
.container {  
    width: 90%;  
    max-width: 1100px;  
    margin: 0 auto; }  
  
header {background: var(--secondary); box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1); }  
  
.header-content {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    padding: 15px 0; }  
  
.logo {  
    color: #fff; }  
  
nav ul {
```

```
list-style: none;
display: flex;
gap: 25px; }

nav ul li a {
color: #fff;
text-decoration: none;
font-weight: bold;
padding: 5px 10px;
border-radius: 4px; }

nav ul li a:hover {
background: rgba(255, 255, 255, 0.2); }

.menu-icon {
display: none;
font-size: 28px;
color: #fff;
cursor: pointer; }

.hero {
background: linear-gradient(135deg, var(--primary), #34495e);
color: #fff;
text-align: center;
padding: 120px 20px;
min-height: 60vh;
display: flex;
align-items: center;
justify-content: center; }

.hero h2 {
font-size: 48px;
font-weight: 700;
margin-bottom: 20px; }

.hero p {
```

```
max-width: 600px;
margin: 15px auto 30px;
font-size: 18px; }

.btn {
background: var(--secondary);
padding: 12px 30px;
color: #fff;
border: none;
border-radius: 6px;
font-weight: bold;
cursor: pointer;
display: inline-block;
text-decoration: none; }

.btn:hover {
background: var(--primary); }

.about {
padding: 100px 20px;
background: #fff; }

.about-content {
display: flex;
align-items: flex-start;
gap: 50px;
justify-content: center;
flex-wrap: wrap; }

.profile-img {
width: 210px;
height: 360px;
border-radius: 30px;
border: 3px solid var(--primary);
object-fit: cover;
box-shadow: var(--shadow); }
```

```
flex-shrink: 0; }

.about-text {
  max-width: 600px;
  flex: 1;
  min-width: 300px; }

.about-text h3 {
  font-size: 32px;
  color: var(--primary);
  margin: 0 0 15px 0; }

.about-text p {
  font-size: 16px;
  color: #555;
  margin-bottom: 25px; }

.skills-title {
  font-size: 20px;
  color: var(--primary);
  margin: 25px 0 15px 0;
  font-weight: bold; }

.skills-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 25px; }

.skill-group {
  background: #f9f9f9;
  padding: 18px;
  border-radius: 8px;
  border-left: 4px solid var(--secondary); }

.skill-label {
  font-size: 14px;
  font-weight: bold;
```

```
color: var(--secondary);
margin: 0 0 12px 0;
display: block;
text-transform: uppercase;
letter-spacing: 0.5px; }

.skills-list {
  display: flex;
  flex-wrap: wrap;
  gap: 8px; }

.skill-tag {
  background: #fff;
  color: var(--primary);
  padding: 6px 12px;
  border: 1px solid #ddd;
  border-radius: 20px;
  font-size: 13px;
  display: inline-block; }

.skill-tag:hover {
  background: var(--secondary);
  color: #fff;
  border-color: var(--secondary); }

.portfolio {
  padding: 100px 20px;
  background: var(--light);
  text-align: center; }

.portfolio h3 {
  font-size: 36px;
  color: var(--primary);
  margin-bottom: 50px; }

.slideshow-wrapper {
  position: relative;
```

```
max-width: 700px;  
margin: 0 auto;}  
.slideshow-container {  
position: relative;  
width: 100%;}  
.slide {  
display: none;  
animation: fade 0.5s;}  
.slide.active {  
display: block;}  
@keyframes fade {  
from {  
opacity: 0.4;  
}  
to {  
opacity: 1;}}  
.prev-btn,  
.next-btn {  
position: absolute;  
top: 50%;  
transform: translateY(-50%);  
background: rgba(0, 0, 0, 0.5);  
color: #fff;  
border: none;  
padding: 12px 16px;  
font-size: 24px;  
cursor: pointer;  
border-radius: 4px;  
z-index: 10;}  
.prev-btn:hover,  
.next-btn:hover {
```

```
background: rgba(0, 0, 0, 0.8);  
.prev-btn {  
    left: 10px; }  
.next-btn {  
    right: 10px; }  
.dots-container {  
    text-align: center;  
    padding: 20px 0; }  
.dot {  
    height: 12px;  
    width: 12px;  
    margin: 0 8px;  
    background-color: #bbb;  
    border-radius: 50%;  
    display: inline-block;  
    cursor: pointer; }  
.dot.active {  
    background-color: var(--primary); }  
.project-card {  
    background: #fff;  
    border-radius: 12px;  
    overflow: hidden;  
    box-shadow: var(--shadow); }  
.project-card:hover {  
    transform: translateY(-8px);  
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15); }  
.project-image {  
    width: 100%;  
    height: 300px;  
    overflow: hidden;  
    background: #f0f0f0; }
```

```
.project-image img {  
    width: 100%;  
    height: 100%;  
    object-fit: cover;}  
  
.project-card:hover .project-image img {  
    transform: scale(1.05);}  
  
.project-content {  
    padding: 25px 20px;  
    text-align: left;}  
  
.project-content h4 {  
    font-size: 22px;  
    color: var(--primary);  
    margin-bottom: 10px;  
    font-weight: bold;}  
  
.project-content p {  
    color: #666;  
    font-size: 14px;}  
  
.contact {  
    padding: 100px 20px;  
    background: #fff;}  
  
.contact h3 {  
    text-align: center;  
    margin-bottom: 40px;  
    font-size: 36px;  
    color: var(--primary);}  
  
form {  
    max-width: 600px;  
    margin: 0 auto;}  
  
form input,  
form textarea {  
    width: 100%;
```

```
padding: 15px;  
margin-bottom: 20px;  
border: 1px solid #ddd;  
border-radius: 8px;  
font-size: 16px;  
font-family: inherit;  
background: #f9f9f9; }  
  
form input:focus,  
form textarea:focus {  
    outline: none;  
    border-color: var(--primary);  
    background: #fff; }  
  
form textarea {  
    resize: vertical;  
    min-height: 150px; }  
  
footer {  
    background: var(--primary);  
    color: #fff;  
    text-align: center;  
    padding: 30px 20px;  
    font-size: 14px; }  
  
@media (max-width: 768px) {  
    nav ul {  
        display: none;  
        flex-direction: column;  
        background: var(--primary);  
        position: absolute;  
        top: 60px;  
        right: 20px;  
        padding: 15px;  
        border-radius: 8px;
```

```
min-width: 180px;
box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);}

nav ul.active {
    display: flex;}

nav ul li a {
    padding: 10px 15px;
    display: block; }

.menu-icon {
    display: block; }

.hero h2 {
    font-size: 32px; }

.hero p {
    font-size: 16px; }

.about-content {
    gap: 30px;
    flex-direction: column;
    align-items: center; }

.profile-img {
    width: 220px;
    height: 220px; }

.about-text {
    width: 100%; }

.about-text h3,
.portfolio h3,
.contact h3 {
    font-size: 28px; }

.project-image {
    height: 250px; }

.prev-btn,
.next-btn {
    padding: 10px 12px; }
```

```
        font-size: 20px; }

.skills-container {
    grid-template-columns: 1fr;
}

@media (max-width: 480px) {
    .hero {
        padding: 80px 15px;
        min-height: 50vh;
    }

    .hero h2 {
        font-size: 24px;
    }

    .hero p {
        font-size: 15px;
    }

    .btn {
        padding: 10px 20px;
        font-size: 14px;
    }

    .about {
        padding: 60px 15px;
    }

    .about-content {
        gap: 20px;
    }

    .profile-img {
        width: 160px;
        height: 160px;
    }

    .about-text h3,
    .portfolio h3,
    .contact h3 {
        font-size: 24px;
    }

    .about-text p,
    .skill-label {
        font-size: 14px;
    }

    .portfolio,
    .contact {

```

```
    padding: 60px 15px; }

.portfolio h3 {
    margin-bottom: 30px; }

.project-image {
    height: 200px; }

.project-content {
    padding: 20px 15px; }

.project-content h4 {
    font-size: 18px; }

.contact h3 {
    margin-bottom: 25px; }

form input,
form textarea {
    padding: 12px;
    margin-bottom: 15px;
    font-size: 14px; }

.skills-container {
    grid-template-columns: 1fr;
    gap: 15px; }

.skills-title {
    font-size: 18px; }

.skill-group {
    padding: 15px; }

.skill-tag {
    font-size: 12px;
    padding: 5px 10px; }

.prev-btn,
.next-btn {
    padding: 8px 10px;
    font-size: 18px; }

.dot {
```

```
    width: 10px;  
    height: 10px;  
    margin: 0 6px; }  
}
```

### 8.3 Menambahkan Elemen Interaktif (Menu Navigasi, Slideshow, dan Form Validation)

Setelah website portofolio statis berhasil dibuat pada subbab 8.2, tahap berikutnya adalah menambahkan elemen interaktif menggunakan JavaScript dasar. Penambahan interaktivitas bertujuan untuk meningkatkan pengalaman pengguna serta menunjukkan pemahaman mahasiswa terhadap konsep DOM dan event handling.

#### a. Menu Navigasi Interaktif

Menu navigasi memungkinkan pengguna berpindah antarbagian halaman dengan mudah. JavaScript dapat digunakan untuk menambahkan efek interaktif sederhana seperti menampilkan notifikasi saat menu diklik.

```
// 1. Menu Navigasi Responsif (Hamburger Menu)  
  
const menuIcon = document.getElementById("hamburger-menu");  
  
const menuList = document.getElementById("menu-list");  
menuIcon.addEventListener("click", () => {  
    menuList.classList.toggle("active");  
});  
  
// Tutup menu saat link diklik  
  
document.querySelectorAll("#menu-list a").forEach((link) => {  
    link.addEventListener("click", () => {  
        menuList.classList.remove("active");  
    });  
});
```

## b. Slideshow Gambar Sederhana

Slideshow digunakan untuk menampilkan proyek atau dokumentasi secara bergantian.

```
// 2. Slideshow Portofolio

let currentSlideIndex = 1;

showSlide(currentSlideIndex);

document.getElementById("prevBtn").addEventListener("click", () => {changeSlide(-1);});

document.getElementById("nextBtn").addEventListener("click", () => {changeSlide(1);});

function changeSlide(n) {showSlide((currentSlideIndex += n));}

function currentSlide(n) {showSlide((currentSlideIndex = n));}

function showSlide(n) {

    const slides = document.querySelectorAll(".slide");

    const dots = document.querySelectorAll(".dot");

    if (n > slides.length) {currentSlideIndex = 1;}

    if (n < 1) {currentSlideIndex = slides.length;}

    slides.forEach((slide) =>
    slide.classList.remove("active"));

    dots.forEach((dot) =>
    dot.classList.remove("active"));

    slides[currentSlideIndex - 1].classList.add("active");

    dots[currentSlideIndex - 1].classList.add("active");
}
```

## c. Validasi Formulir Kontak

Validasi formulir bertujuan memastikan data yang diinput pengguna tidak kosong sebelum dikirim.

```
// 3. Validasi Formulir

const contactForm =
document.getElementById("contactForm");
```

```
contactForm.addEventListener("submit", function (e) {
    e.preventDefault();

    const name =
document.getElementById("name").value.trim();

    const email =
document.getElementById("email").value.trim();

    const message =
document.getElementById("message").value.trim();

    // Validasi input kosong
    if (name === "" || email === "" || message === "") {
        alert("⚠ Semua kolom wajib diisi!");
        return;
    }

    // Validasi email
    const emailRegex = /^[^@\s]+@[^\s]+\.\w+$/;
    if (!emailRegex.test(email)) {
        alert("⚠ Mohon masukkan alamat email yang valid.");
        return;
    }

    // Jika validasi lolos
    alert("✓ Terima kasih, " + name + "! Pesan Anda telah dikirim.");
    contactForm.reset();
}) ;
```

## Output:

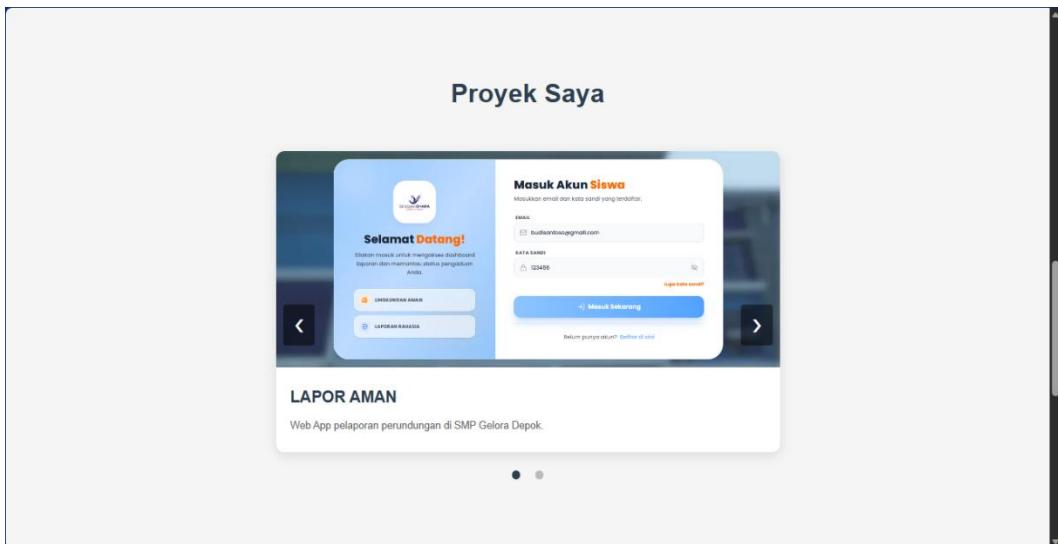
- **Home:**



- **Tentang:**

A screenshot of the "Tentang" (About) page of the website. On the left is a portrait photo of a young man. To the right of the photo is the heading "Tentang Saya". Below this is a paragraph of text: "Saya adalah mahasiswa semester 5 Teknik Informatika di Universitas Pamulang (NIM: 231011401814). Saya memiliki minat pada pengembangan web (Front-end) dan implementasi Machine Learning seperti metode YOLO." Below the text is a section titled "Tech Stack &amp; Tools" which is divided into three categories: "EDITOR &amp; TOOLS", "FRONTEND", and "BACKEND &amp; DATABASE". Each category contains a list of tools represented by small circular icons.

- Portofolio:



- Kontak:

The screenshot shows a messaging interface. At the top, a modal window displays the message "localhost says" and the confirmation "✓ Terima kasih, Wafid! Pesan Anda telah dikirim." (Thank you, Wafid! Your message has been sent.) with an "OK" button. Below the modal, there is a conversation history:

- Wafi (Message input field)
- wafidhaifullah678@gmail.com (Recipient input field)
- Haloo, salam kenal yall!
- Kirim Pesan (Send Message button)

At the bottom of the screen, a dark footer bar contains the copyright notice "© 2025 PortoTech - All Rights Reserved".

## BAB IX

### TREN & ARAH PERKEMBANGAN WEB

#### 9.1 Evolusi HTML, CSS, dan JavaScript Terbaru

Perkembangan teknologi web terus mengalami kemajuan yang signifikan seiring dengan meningkatnya kebutuhan akan aplikasi web yang interaktif, responsif, dan mudah diakses di berbagai perangkat. HTML, CSS, dan JavaScript sebagai teknologi inti pengembangan web juga terus berevolusi untuk mendukung kebutuhan tersebut.

HTML telah berkembang hingga versi HTML5 yang membawa berbagai peningkatan penting, seperti elemen semantik, dukungan multimedia bawaan, serta API tambahan yang memungkinkan pembuatan aplikasi web yang lebih kaya tanpa memerlukan plugin pihak ketiga. Elemen semantik seperti `<header>`, `<nav>`, `<section>`, dan `<footer>` membantu pengembang dalam menyusun struktur halaman yang lebih terorganisir, mudah dipahami, serta mendukung optimasi mesin pencari (SEO) dan aksesibilitas.

CSS juga mengalami perkembangan pesat melalui CSS3 dan fitur-fitur modern seperti Flexbox, Grid Layout, media queries, serta animasi dan transisi. Fitur-fitur ini memungkinkan pengembang menciptakan tampilan website yang responsif dan adaptif terhadap berbagai ukuran layar, mulai dari desktop hingga perangkat mobile. Selain itu, konsep desain modern seperti mobile-first design dan responsive web design semakin banyak diterapkan dalam pengembangan web saat ini.

JavaScript berkembang dari bahasa scripting sederhana menjadi bahasa pemrograman yang sangat powerful. Dengan standar ECMAScript terbaru, JavaScript mendukung fitur-fitur modern seperti arrow function, module, asynchronous programming, serta manipulasi DOM yang lebih efisien. Perkembangan ini menjadikan JavaScript sebagai fondasi utama dalam pengembangan aplikasi web interaktif dan dinamis.

## 9.2 Framework Modern dalam Pengembangan Web

Perkembangan aplikasi web modern yang semakin kompleks mendorong penggunaan framework JavaScript untuk mempermudah proses pengembangan. Framework modern membantu pengembang dalam membangun antarmuka pengguna yang terstruktur, modular, dan mudah dipelihara. Beberapa framework dan platform JavaScript yang banyak digunakan saat ini antara lain React, Vue, Svelte, dan Next.js.

- **React**, merupakan library JavaScript yang dikembangkan oleh Facebook dan banyak digunakan untuk membangun antarmuka pengguna berbasis komponen. React menggunakan pendekatan component-based architecture, di mana setiap bagian antarmuka dibangun sebagai komponen terpisah yang dapat digunakan kembali. Konsep Virtual DOM pada React memungkinkan pembaruan tampilan dilakukan secara efisien, sehingga meningkatkan performa aplikasi web.
- **Vue.js**, adalah framework JavaScript progresif yang dirancang agar mudah dipelajari dan diintegrasikan ke dalam proyek web. Vue menggabungkan keunggulan pendekatan component-based seperti pada React dengan sintaks yang sederhana dan mudah dipahami. Vue banyak digunakan pada pengembangan aplikasi web skala kecil hingga menengah karena fleksibilitas dan dokumentasinya yang lengkap.
- **Svelte**, merupakan framework JavaScript modern yang memiliki pendekatan berbeda dibandingkan framework lainnya. Svelte melakukan proses kompilasi pada saat build time, sehingga kode JavaScript yang dikirim ke browser menjadi lebih ringan dan efisien. Dengan pendekatan ini, Svelte mampu menghasilkan aplikasi web dengan performa tinggi dan ukuran file yang lebih kecil.
- **Next.js**, adalah framework berbasis React yang digunakan untuk membangun aplikasi web dengan performa dan optimasi yang lebih baik. Next.js mendukung fitur server-side rendering dan static site generation, yang memungkinkan halaman web dimuat lebih cepat dan ramah terhadap mesin pencari (SEO). Framework ini banyak digunakan untuk membangun website modern seperti landing page, blog, dan aplikasi web berskala besar.

Meskipun framework-framework tersebut berada di luar cakupan pembelajaran mata kuliah Pemrograman Web 1, pengenalan singkat terhadap React, Vue, Svelte, dan Next.js penting bagi mahasiswa untuk memahami arah perkembangan teknologi web serta mempersiapkan diri dalam mempelajari pengembangan web tingkat lanjut.

### **9.3 Progressive Web Apps (PWA) sebagai Tren Baru**

Progressive Web Apps (PWA) merupakan salah satu pendekatan pengembangan web modern yang menggabungkan kelebihan website dan aplikasi mobile. PWA dirancang untuk memberikan pengalaman pengguna yang menyerupai aplikasi native, seperti tampilan yang responsif, waktu muat yang cepat, serta kemampuan berjalan dengan baik di berbagai perangkat dan sistem operasi. Dengan konsep ini, aplikasi web tidak lagi hanya berfungsi sebagai media informasi, tetapi juga sebagai aplikasi yang interaktif dan andal.

Teknologi utama yang mendukung PWA antara lain service worker, web app manifest, dan mekanisme caching. Service worker berperan dalam mengelola permintaan jaringan dan penyimpanan cache, sehingga aplikasi web tetap dapat diakses meskipun koneksi internet terbatas atau tidak stabil. Sementara itu, web app manifest digunakan untuk mendefinisikan identitas aplikasi, seperti nama, ikon, dan tampilan saat aplikasi ditambahkan ke layar utama perangkat pengguna.

Keunggulan lain dari PWA adalah kemampuannya dalam meningkatkan performa dan kenyamanan pengguna. Dengan memanfaatkan caching, konten website dapat dimuat lebih cepat pada kunjungan berikutnya, sehingga meningkatkan efisiensi penggunaan data dan waktu akses. Selain itu, PWA juga mendukung fitur notifikasi push yang memungkinkan interaksi langsung antara aplikasi web dan pengguna, mirip dengan aplikasi native pada perangkat mobile.

Bagi mahasiswa mata kuliah Pemrograman Web 1, pemahaman konsep PWA penting untuk mengetahui arah perkembangan teknologi web di masa depan. Meskipun implementasi PWA secara penuh berada di luar cakupan pembelajaran dasar, pengenalan terhadap konsep dan manfaat PWA dapat menjadi bahan awal dalam mempelajari pengembangan aplikasi web yang lebih lanjut dan kompleks.

## **DAFTAR PUSTAKA**

Duckett, J. (2014). HTML & CSS: Design and build websites. Wiley.

Duckett, J. (2015). JavaScript and jQuery: Interactive front-end web development. Wiley.

Google Developers. (2024). Progressive web apps.  
<https://developers.google.com/web/progressive-web-apps>

Mozilla Developer Network. (2025, February 10). JavaScript guide. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>

Mozilla Developer Network. (2025, January 15). HTML: HyperText Markup Language. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTML>

Next.js. (2024). Next.js documentation. <https://nextjs.org/docs>

React. (2024). React documentation. <https://react.dev>

Svelte. (2024). Svelte tutorial. <https://svelte.dev/tutorial>

Vue.js. (2024). Vue.js guide. <https://vuejs.org/guide>

W3Schools. (2025). CSS tutorial. <https://www.w3schools.com/css/>

## BIOGRAFI PENULIS



**Aufa Wafi Dhaifullah**, mahasiswa Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pamulang. Terdaftar sebagai mahasiswa aktif dengan Nomor Induk Mahasiswa (NIM) 231011401814. Penulis memiliki minat dalam bidang pengembangan web (Full-Stack Development), infrastuktur Cloud, dan teknologi Machine Learning. Melalui penyusunan makalah ini, penulis bertujuan untuk memperdalam pemahaman mengenai dasar-dasar pemrograman web menggunakan HTML, CSS, dan JavaScript sebagai fondasi pengembangan web modern.



**Fadlan Lesmana** adalah mahasiswa Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pamulang dengan (NIM) 231011400735. Penulis memiliki ketertarikan pada bidang teknologi informasi, pengembangan aplikasi web (Front-End Development), dan aplikasi Mobile (Native). Melalui makalah ini, penulis berupaya mengembangkan kemampuan akademik dan teknis dalam memahami pemrograman web dasar.



**Ivandhika Satria Putra Ferdianto**, mahasiswa Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pamulang dengan (NIM) 231011401818. Penulis aktif mempelajari teknologi pemrograman web (Back-End Development) serta konsep dasar pengembangan aplikasi berbasis web. Kontribusinya dalam penyusunan makalah ini diharapkan dapat menambah wawasan dan pemahaman mengenai pengembangan web modern.