BAB 4
CSS

4.1 Tujuan

Setelah menyelesaikan bab ini dan mengikuti kegiatan praktikum, mahasiswa diharapkan mampu:

1. Memahami dasar-dasar CSS (*Cascading Style Sheets*), termasuk struktur, properti, dan selektor.
2. Mengimplementasikan CSS dalam pembuatan website untuk mengatur gaya (*style*), latar belakang (*background*), *web fonts*, dan membangun tata letak (*layout*) modern menggunakan CSS Grid.

4.2 Pengantar

Jika HTML adalah kerangka (*skeleton*) dari sebuah halaman web, maka CSS (*Cascading Style Sheets*) adalah kulit, pakaian, dan penampilannya. CSS adalah bahasa yang kita gunakan untuk "menghias" elemen-elemen HTML, mulai dari warna dan ukuran teks, hingga tata letak halaman yang kompleks. Menguasai CSS adalah langkah awal untuk mengubah halaman web yang polos menjadi sebuah halaman yang menarik dan profesional.

4.2.1 Apa itu CSS?

CSS adalah bahasa *stylesheet* yang digunakan untuk mendeskripsikan presentasi visual dari sebuah dokumen HTML. Peran utamanya adalah untuk memisahkan antara konten (HTML) dan visual (CSS).

4.2.2 Sintaks dasar CSS

Aturan CSS (*CSS Rule*) terdiri dari **Selektor** dan **Blok Deklarasi**.

```
/* Ini adalah contoh aturan CSS */
p {
    color: blue;
    font-size: 16px;
}
```

- Selektor (`p`): Menargetkan elemen HTML yang ingin diberi gaya.
- Blok Deklarasi (`{...}`): Berisi satu atau lebih deklarasi gaya.
- Deklarasi (`color: blue;`): Terdiri dari Properti (color) dan Nilai (blue)

4.2.3 Penerapan CSS

Ada tiga metode untuk mengintegrasikan CSS ke dalam dokumen HTML. Memahami ketiganya penting untuk mengetahui kapan harus menggunakan setiap metode.

1. **External CSS**

   Metode ini adalah cara yang paling umum dan sangat direkomendasikan. Anda menulis semua aturan CSS Anda di dalam sebuah file terpisah dengan ekstensi `.css` (misalnya, `style.css`), lalu menghubungkannya ke file HTML menggunakan tag `<link>` di dalam `<head>`.

   Keuntungan:

   1. Kode HTML dan CSS benar-benar terpisah, membuat proyek lebih rapi dan mudah dikelola.
   2. Satu file CSS dapat digunakan oleh banyak halaman HTML. Mengubah satu file ini akan mengubah tampilan semua halaman yang terhubung.

Contoh di `index.html`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Halaman Saya</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Selamat Datang</h1>
    <p>Ini adalah paragraf yang diberi gaya oleh file CSS eksternal.</p>
</body>
</html>
```

Contoh di `style.css`:

```
body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}

h1 {
    color: navy;
}
```

2. **Internal CSS**
   Metode ini dilakukan dengan menulis aturan CSS langsung di dalam file HTML menggunakan tag `<style>` yang ditempatkan di dalam `<head>`. Metode ini berguna jika sebuah halaman memiliki gaya yang sangat unik dan tidak akan digunakan di halaman lain.

Keuntungan:

- Semua yang dibutuhkan untuk satu halaman (HTML dan CSS) ada dalam satu file.

Kerugian:

- Tidak dapat digunakan kembali untuk halaman lain.

Contoh di `index.html`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Halaman dengan Internal CSS</title>
    <style>
        body {
```

```
            background-color: #e6f7ff;
        }
        p {
            color: #005f99;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>Paragraf ini diatur gayanya oleh Internal CSS.</p>
</body>
</html>
```

3. **Inline CSS**
   Metode diterapkan langsung pada sebuah elemen HTML menggunakan atribut `style`. Metode ini tidak disarankan untuk penggunaan umum karena mencampurkan kembali konten dan presentasi, yang bertentangan dengan prinsip dasar CSS. Biasanya hanya digunakan untuk tujuan yang sangat spesifik seperti pengujian cepat atau saat gaya dihasilkan secara dinamis oleh JavaScript.

Kerugian:

- Membuat debuging dan pembaruan desain menjadi sangat sulit.
- Gaya inline akan menimpa gaya dari internal maupun external CSS, yang bisa menyebabkan kebingungan.

Contoh di `index.html`:

```
<body>
    <h1 style="color: red; text-
   align: center;">Judul ini memiliki gaya inline.</h1>
</body>
```

4.2.4 Selektor CSS

Selektor adalah bagian paling kuat dari CSS. Selektor merupakan pola yang kita gunakan untuk "memilih" atau menargetkan elemen HTML yang ingin kita beri gaya.

A. **Selektor Tipe/Elemen (Type Selector):** Menargetkan semua elemen dengan nama tag yang sama. Selektor ini adalah selektor yang paling dasar.

Contoh CSS :

```
/* Semua elemen <h2> akan berwarna hijau */
h2 {
    color: green;
}
```

```css
/* Semua paragraf akan memiliki ukuran font 16px */
p {
    font-size: 16px;
}
```

B. **Selektor Class (Class Selector):** Menargetkan elemen berdasarkan nilai atribut class-nya. Selektor *class* diawali dengan tanda titik (.). Selektor ini adalah selektor yang paling fleksibel dan umum digunakan karena satu *class* dapat diterapkan pada banyak elemen, dan satu elemen dapat memiliki beberapa *class*.

Contoh CSS:

```css
/* Semua elemen dengan class="highlight" akan memiliki latar kuning */
.highlight {
    background-color: yellow;
}
.text-center {
    text-align: center;
}
```

Contoh HTML

```html
<p class="highlight">Paragraf ini akan disorot.</p>
<div class="highlight text-
center">Div ini disorot dan teksnya di tengah.</div>
```

C. **Selektor ID (ID Selector):** Menargetkan satu elemen unik berdasarkan nilai atribut id-nya. Selektor ID diawali dengan tanda pagar (#). Atribut id harus unik dalam satu halaman; tidak boleh ada dua elemen dengan id yang sama.

Contoh CSS:

```css
/* Elemen dengan id="header-utama" akan memiliki border bawah */
#header-utama {
    border-bottom: 2px solid black;
    padding: 20px;
}
```

Contoh HTML

```html
<div id="header-utama">
    <h1>Judul Utama Website</h1>
</div>
```

4.2.5 Blok Layout Div, span, ID, class

Setelah memahami cara menargetkan elemen, kita perlu tahu elemen apa yang paling sering kita gunakan untuk membangun struktur halaman. Di sinilah `<div>` dan `<span>` berperan sebagai "blok pembangun" utama, yang kemudian kita beri "label" menggunakan `id` dan `class` agar bisa ditata dengan CSS.

- **`<div>` :** Anggap `<div>` sebagai sebuah kotak kosong atau kontainer. Secara visual, `<div>` tidak menghasilkan apa-apa. Kegunaan utamanya adalah untuk mengelompokkan elemen-elemen lain menjadi satu bagian logis. `<div>` adalah elemen *block-level*, yang digunakan untuk untuk membangun bagian-bagian besar halaman, seperti header, sidebar, area konten utama, dan footer.
- **`<span>`:** Jika `<div>` adalah kotak besar, `<span>` adalah pembungkus kecil di dalam teks. `<span>` adalah elemen *inline-level*, artinya `<span>` tidak memulai baris baru dan hanya mengambil lebar secukupnya. `<span>` digunakan untuk menargetkan bagian kecil dari teks di dalam elemen lain (seperti paragraf) untuk memberinya gaya khusus, tanpa merusak isi kalimat.

Kapan Menggunakan `id` vs. `class`?

- Gunakan `id` (tanda #) untuk elemen yang unik.
  Sebuah `id` hanya boleh digunakan satu kali per halaman. `id` cocok untuk bagian utama layout yang hanya ada satu, seperti `#header-utama`, `#konten-utama`, atau `#footer`.
- Gunakan `class` (tanda `.`) untuk elemen yang dapat digunakan berulang kali.
  Kita bisa menerapkan `class` yang sama ke banyak elemen. `class` cocok untuk komponen UI yang muncul berkali-kali, seperti `.tombol`, `.kartu-produk`, `.pesan-error`.

Contoh Penerapan:

```
<!-- ID digunakan untuk kontainer utama yang unik -->
<div id="profil-pengguna">

    <!-
    - Class digunakan untuk 'kartu' yang mungkin bisa muncul lagi di tempat l
    ain -->
    <div class="kartu-info">
        <h2>Biodata</h2>
        <p>Nama: Budi Sanjaya</p>
        <p>Email: <span class="teks-
    rahasia">budi@email.com</span></p> <!-
    - Span untuk menata bagian kecil teks -->
    </div>

    <div class="kartu-info">
        <h2>Alamat</h2>
        <p>Jalan Pendidikan No. 123</p>
    </div>

```

```
16. </div>
```

4.2.6 Web Fonts

Secara *default*, browser hanya akan menggunakan font yang terinstal di sistem operasi pengguna. Dengan *web fonts*, kita dapat menggunakan jenis huruf (*font*) kustom dari server eksternal agar desain web kita lebih unik dan konsisten di semua perangkat.

Cara menggunakan Web Fonts dari Google Fonts:

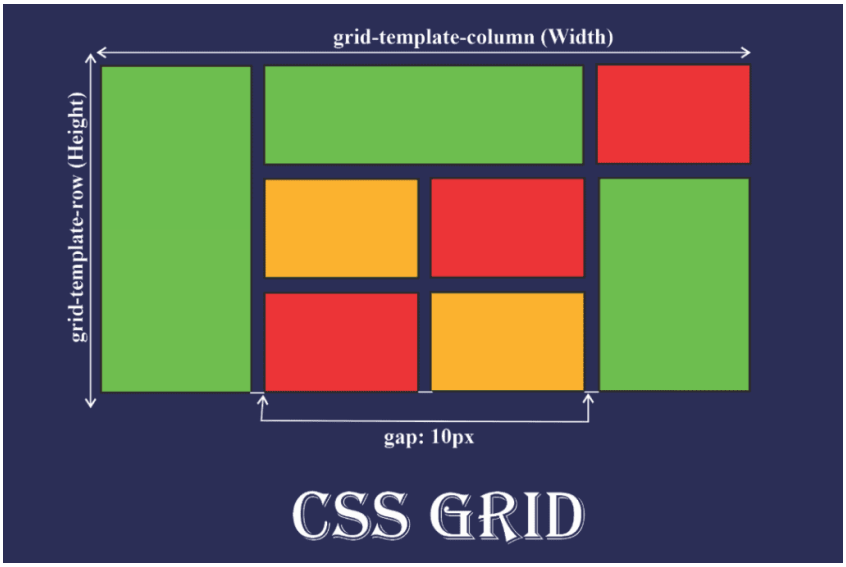
1. Kunjungi https://fonts.google.com.
2. Pilih font yang Anda suka. Pilih Get embed code.
3. Salin kode <link> yang disediakan dan tempelkan di dalam <head> file HTML Anda, atau salin kode @import dan tempelkan di baris paling atas file CSS Anda.
4. Gunakan nama font tersebut dalam properti font-family di CSS Anda.

Contoh di CSS:

```
1. @import url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@
   0,300..800;1,300..800&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,5
   00;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800
   ;1,900&display=swap');
2.
3. body {
4.     font-family: 'Poppins', sans-
   serif; /* Poppins akan digunakan, jika gagal, sans-
   serif akan jadi fallback */
5. }
```

4.2.7 CSS Grid

CSS Grid adalah sistem layout dua dimensi (baris dan kolom) yang sangat fleksibel. CSS Grid adalah cara modern untuk merancang tata letak halaman web, menggantikan teknik lama seperti `float`.



Konsep CSS Grid:

1. **Grid Container:** Elemen induk yang kita beri `display: grid;`.
2. **Grid Items:** Elemen-elemen anak langsung dari *grid container*.
3. **Grid Lines:** Garis-garis pembagi horizontal dan vertikal yang membentuk struktur grid.
4. **Grid Tracks:** Ruang di antara dua *grid lines* (membentuk kolom atau baris).
5. **Grid Cells:** Unit terkecil dari grid, ruang di antara empat *grid lines* yang bersilangan.

Properti Dasar pada Container:

- `display: grid;`: Mengaktifkan layout grid.
- `grid-template-columns`: Mendefinisikan jumlah dan ukuran kolom. Contoh: `grid-template-columns: 200px 1fr 1fr;` (membuat 3 kolom: kolom pertama 200px, dua kolom berikutnya membagi sisa ruang secara merata). Unit `fr` (*fractional unit*) sangat berguna untuk layout fleksibel.
- `grid-template-rows`: Mendefinisikan jumlah dan ukuran baris.
- `gap`: Menentukan jarak (celah) antara baris dan kolom. Contoh: `gap: 20px;`

4.3 Kegiatan Praktikum

Langkah 1: Membuat file CSS

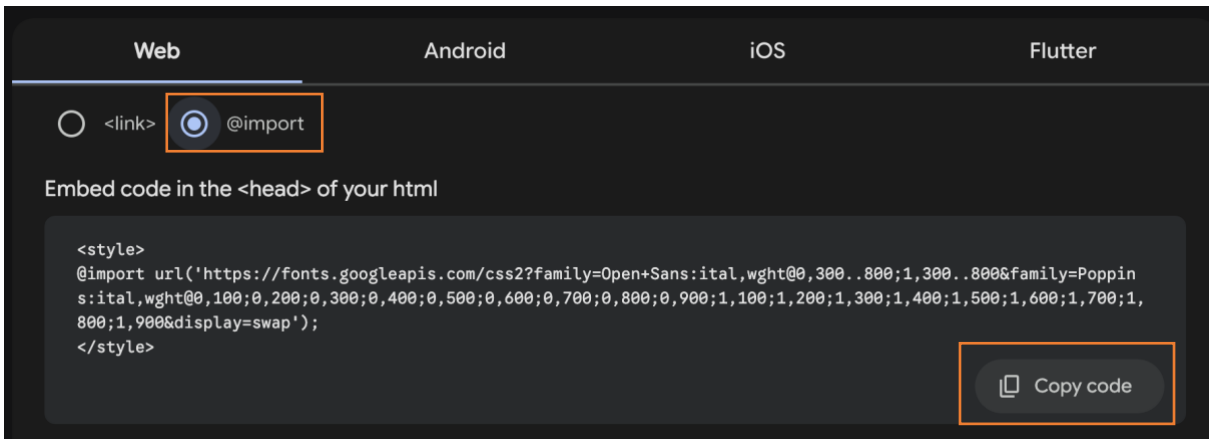
Buka folder `PraktikumHTML` dari pertemuan sebelumnya. Buat file baru bernama `style.css`.
Buka file index.HTML, hubungkan `style.css` ke `index.html` menggunakan tag `<link>` di dalam `<head>`.

`Index.html`

```
1. <!DOCTYPE html>
2. <html lang="en">
3.   <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Biodata Diri</title>
    <link rel="stylesheet" href="style.css">
  </head>
```

Buka `style.css`. Di baris paling atas, impor font "Poppins" dari Google Fonts dengan cara buka https://fonts.google.com/specimen/Poppins Pilih Get Font kemudian pilih Get embed code, pilih Web kemudinan @import, klik copy code



Paste pada file style.css, jangan lupa hapus tag `<style>` dan `</style>` karena tidak diperlukan pada file `style.css`

Style.css

```
@import url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1,300..800&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');
```

Langkah 2: Memberi Style dan Background

Atur gaya untuk `body` dan headline menggunakan font yang baru diimpor dan tambahkan warna latar belakang.

Style.css

```
@import url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1,300..800&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');

body {
    font-family: 'Poppins', sans-serif;
    background-color: #f0f2f5; /* Warna latar sedikit kebiruan */
    color: #333;
    margin: 0; /* Menghilangkan margin default dari body */
    padding: 0;
}
```

```
h1, h2 {
    color: #0d2c54; /* Warna biru tua */
}
```

Langkah 3: Membuat Kontainer Layout

Pada `index.html` bungkus semua konten di dalam `<body>` dengan `<div id="container">`

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
  scale=1.0" />
    <title>Biodata Diri</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div id="container">
    <h1>Biodata Diri</h1>
    ..
    ..kode program
    ..
    </div>
  </body>
</html>
```

Lalu beri gaya `#container` pada CSS

style.css

```
/*kode sebelumnya */

#container {
    width: 90%;
    max-width: 900px;
    margin: 30px auto;
    background-color: #ffffff;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 15px rgba(0,0,0,0.1);
}
```

Langkah 4: Membangun Layout Profil dengan CSS Grid

Pada `index.html`, ubah struktur bagian profil dan deskripsi singkat menjadi seperti berikut. Kita akan membuat class profil-grid, profil-foto dan profil-teks.

Index.html

```
      <h1>Biodata Diri</h1>
      <div class="profil-grid">
```

```
        <div class="profil-foto">
          <img src="foto.jpg" alt="Foto Profil Saya" />
        </div>
        <div class="profil-teks">
          <h2>Deskripsi Singkat</h2>
          <p>
            Perkenalkan, nama saya <strong>[Nama Anda]</strong>. Saya adalah seorang mahasiswa
          </p>
          <em>Pendidikan Teknik Informatika</em> di
          <u>Universitas Muhammadiyah Surakarta</u> Saat ini saya sedang fokus belajar Pemrograman Web.
        </div>
      </div>
```

Tambahkan css untuk `.profil-grid` dan `.profil-foto` pada `style.css`

Style.css

```
/*kode sebelumnya */

.profil-grid {
    display: grid;
    grid-template-columns: 200px 1fr; /* Kolom 1: 200px, Kolom 2: sisa ruang */
    gap: 30px; /* Jarak antara foto dan teks */
    align-items: center; /* Menyelaraskan item secara vertikal di tengah */
    margin-bottom: 30px;
}

.profil-foto img {
    width: 100%;
    border-radius: 50%; /* Membuat foto menjadi bulat */
    box-shadow: 0 2px 8px rgba(0,0,0,0.15);
}
```

Langkah 5: Menata Tabel dan Form

Agar tabel lebih mudah dibaca. Tambahkan kode berikut di `style.css`

style.css

```
/*kode sebelumnya */

table {
    width: 100%;
    border-collapse: collapse; /* Menghilangkan spasi antar border sel */
    margin-top: 20px;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
```

```
}

thead {
    background-color: #0d2c54;
    color: white;
}

tbody tr:nth-child(even) {
    background-color: #f9f9f9; /* Efek 'zebra-stripe' untuk baris genap */
}
```

Atur juga style pada formulir kontak agar terlihat lebih modern.

style.css

```
/*kode sebelumnya */

form {
    margin-top: 20px;
}

input[type="text"],
input[type="email"],
textarea {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box; /* Agar padding tidak menambah lebar elemen */
}

input[type="submit"] {
    background-color: #0d2c54;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: 600;
}

input[type="submit"]:hover {
    background-color: #1e4a8a; /* Warna sedikit lebih terang saat hover */
}
```

4.4 Rangkuman

- CSS memungkinkan kita untuk memperkaya tampilan visual halaman web dengan mengatur background, tipografi menggunakan *web fonts*, dan banyak lagi.

- **CSS Grid** adalah sistem layout yang kuat untuk membuat tata letak dua dimensi (baris dan kolom). Ia adalah pendekatan modern yang menggantikan teknik lama seperti `float`.
- Konsep inti Grid meliputi **Grid Container** (`display: grid`) dan **Grid Items**. Properti seperti `grid-template-columns` dan `gap` adalah kunci untuk mendefinisikan struktur layout.
- Dengan menggabungkan selektor yang tepat dan properti-properti CSS modern, kita dapat mengubah dokumen HTML yang sederhana menjadi halaman web yang terstruktur dengan baik dan menarik secara visual.

4.5 Tugas

Kerjakan tugas-tugas di bawah ini untuk memperkuat pemahaman dan keterampilan Anda.

1. Jelaskan keuntungan utama menggunakan CSS Grid dibandingkan dengan menggunakan `float` untuk membangun tata letak halaman.
2. Modifikasi kegiatan praktikum :

   - Pada file `style.css`, ubah `background-color` pada `body` menjadi `background-image` dengan sebuah gradient linear. Contoh: `background-image: linear-gradient(to top, #accbee, #e7f0fd);`

   - Di bawah bagian "Profil", buatlah sebuah layout grid baru untuk bagian "Hobi" dan "Riwayat Pendidikan" agar keduanya tampil **berdampingan** dalam dua kolom yang sama lebar. (Hint: Bungkus kedua bagian tersebut dalam satu `div` baru, lalu terapkan `display: grid` dan `grid-template-columns: 1fr 1fr;` pada `div` pembungkus tersebut).