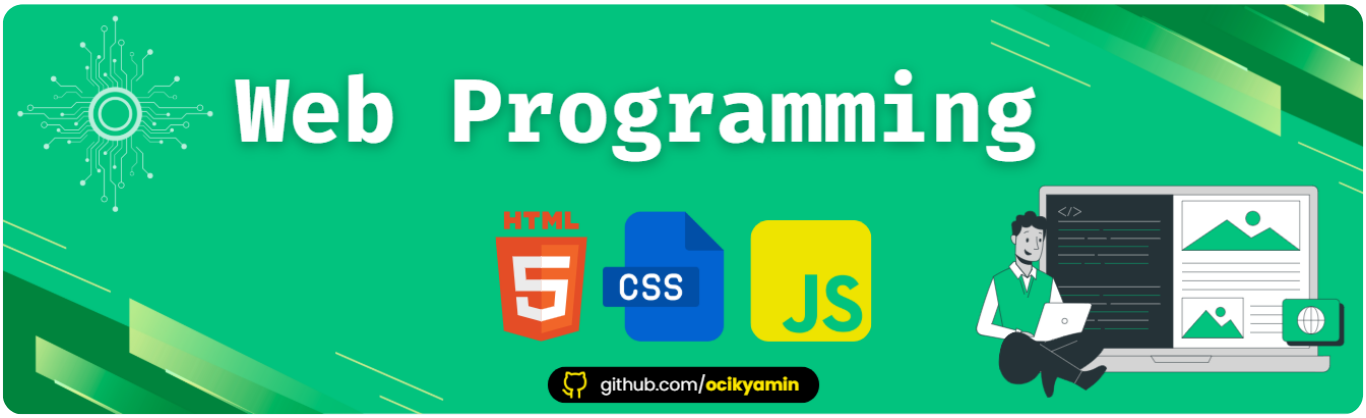

Pengantar

Modul ajar ini dirancang untuk pengembangan web dasar menggunakan HTML, CSS, dan JavaScript, hingga manipulasi DOM. Pemahaman yang kuat terhadap materi dalam modul ini akan menjadi fondasi esensial sebelum Anda melangkah lebih jauh ke pengembangan web sisi server menggunakan bahasa seperti PHP dan interaksi dengan database seperti MySQL, karena semua data yang diproses di backend pada akhirnya akan ditampilkan dan dimanipulasi di sisi klien melalui teknologi yang dibahas di sini. Modul ini akan membimbing Anda melalui 6 sesi praktikum yang berfokus pada konsep-konsep inti dan praktik dalam membangun antarmuka web interaktif. Semoga modul ini bermanfaat bagi Anda dalam mempelajari pengembangan web dasar. Selamat belajar dan bereksperimen!

Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

- Membangun struktur halaman web menggunakan HTML semantik.
- Mendesain tampilan halaman web menggunakan CSS, termasuk responsivitas.
- Mengimplementasikan logika interaktif pada halaman web menggunakan JavaScript.
- Memanipulasi elemen-elemen HTML dan CSS secara dinamis menggunakan Document Object Model (DOM).
- Mengembangkan aplikasi web sederhana yang interaktif dan responsif.

Pertemuan 1: Pengantar HTML (Struktur dan Semantik)

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Memahami struktur dasar dokumen HTML5.
2. Menggunakan elemen-elemen HTML semantik untuk membangun kerangka halaman web.
3. Membuat tautan dan menyisipkan gambar.
4. Membuat daftar dan tabel dasar.
5. Memahami pentingnya validasi HTML.

Materi

HTML (HyperText Markup Language) adalah bahasa markup standar untuk membuat halaman web. HTML mendefinisikan struktur dan konten halaman web. HTML5 adalah versi terbaru dari HTML yang memperkenalkan banyak fitur baru untuk multimedia, grafis, dan semantik.

1. Struktur Dasar Dokumen HTML5

Setiap dokumen HTML5 dimulai dengan deklarasi `<!DOCTYPE html>` dan elemen `<html>`. Di dalamnya terdapat elemen `<head>` (untuk metadata) dan `<body>` (untuk konten yang terlihat oleh pengguna).

```html
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Judul Halaman Saya</title>
</head>
<body>
    <h1>Selamat Datang di Halaman Web Saya</h1>
    <p>Ini adalah paragraf pertama.</p>
</body>
</html>
```

2. Elemen HTML Semantik

HTML5 memperkenalkan elemen semantik yang memberikan makna pada struktur konten, bukan hanya presentasi. Ini membantu mesin pencari dan teknologi asistif memahami halaman Anda dengan lebih baik.

Contoh elemen semantik:

- `<header>`: Bagian pengantar atau navigasi.
- `<nav>`: Bagian navigasi.
- `<main>`: Konten utama dokumen.
- `<article>`: Konten mandiri yang dapat didistribusikan secara independen (misalnya, posting blog).
- `<section>`: Bagian tematik dari dokumen.
- `<aside>`: Konten yang terkait secara tidak langsung dengan konten utama (misalnya, sidebar).
- `<footer>`: Bagian kaki halaman.

3. Tautan dan Gambar

- **Tautan (`<a>`)**: Digunakan untuk menghubungkan satu halaman web ke halaman lain atau sumber daya eksternal.

```
<a href="https://www.google.com">Kunjungi Google</a>
<a href="halaman-lain.html">Pergi ke Halaman Lain</a>
```

- **Gambar (`<img>`)**: Digunakan untuk menyisipkan gambar ke dalam halaman web. Atribut `src` menentukan lokasi gambar, dan `alt` menyediakan teks alternatif untuk aksesibilitas.

```
<img src="gambar.jpg" alt="Deskripsi Gambar">
```

4. Daftar (`<ul>`, `<ol>`, `<li>`)

- **Daftar Tak Berurut (`<ul>`)**: Item daftar ditandai dengan bullet point.

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
</ul>
```

- **Daftar Berurut (`<ol>`)**: Item daftar ditandai dengan angka atau huruf.

```html
<ol>
    <li>Langkah 1</li>
    <li>Langkah 2</li>
</ol>
```

5. Tabel (`<table>`, `<thead>`, `<tbody>`, `<tr>`, `<th>`, `<td>`)

Digunakan untuk menampilkan data dalam format baris dan kolom.

```html
<table>
    <thead>
        <tr>
            <th>Nama</th>
            <th>Usia</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Yamin</td>
            <td>20</td>
        </tr>
        <tr>
            <td>Ani</td>
            <td>22</td>
        </tr>
    </tbody>
</table>
```

Pembahasan

HTML adalah fondasi dari setiap halaman web. Memahami struktur dasar dan penggunaan elemen semantik sangat penting untuk membangun halaman yang mudah diakses, diindeks oleh mesin pencari, dan mudah dipelihara. Praktikum ini akan fokus pada penerapan elemen-elemen dasar HTML untuk membuat kerangka halaman web yang solid.

Contoh Kode

Berikut adalah contoh kode HTML lengkap untuk sebuah halaman profil sederhana:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profil Saya</title>
</head>
<body>
    <header>
        <h1>Profil Singkat</h1>
        <nav>
            <ul>
                <li><a href="#tentang">Tentang Saya</a></li>
                <li><a href="#pendidikan">Pendidikan</a></li>
                <li><a href="#kontak">Kontak</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="tentang">
            <h2>Tentang Saya</h2>
            <p>Halo, nama saya [Nama Anda]. Saya adalah seorang mahasiswa yang
tertarik pada pengembangan web.</p>
            <img src="https://via.placeholder.com/150" alt="Foto Profil">
        </section>

        <section id="pendidikan">
            <h2>Pendidikan</h2>
            <table>
                <thead>
                    <tr>
                        <th>Institusi</th>
                        <th>Jurusan</th>
                        <th>Tahun</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>Universitas XYZ</td>
                        <td>Teknik Informatika</td>
                        <td>2022 - Sekarang</td>
```

```
                    </tr>
                </tbody>
            </table>
        </section>

        <section id="kontak">
            <h2>Kontak</h2>
            <ul>
                <li>Email: <a href="mailto:nama@example.com">nama@example.com</a>
</li>
                <li>LinkedIn: <a href="#">[Profil LinkedIn Anda]</a></li>
            </ul>
        </section>
    </main>

    <footer>
        <p>&copy; 2025 [Nama Anda]. Hak Cipta Dilindungi.</p>
    </footer>
</body>
</html>
```

Langkah-langkah Praktikum

1. Buat folder baru di komputer Anda dengan nama `praktikum_html`.
2. Di dalam folder tersebut, buat file baru bernama `index.html`.
3. Buka `index.html` dengan editor teks favorit Anda (misalnya VS Code).
4. Salin dan tempel kode contoh di atas ke dalam `index.html`.
5. Ganti placeholder `[Nama Anda]`, `nama@example.com`, dan `[Profil LinkedIn Anda]` dengan informasi Anda sendiri.
6. Ganti URL gambar `https://via.placeholder.com/150` dengan URL gambar Anda sendiri atau biarkan untuk sementara.
7. Simpan file `index.html`.
8. Buka file `index.html` di browser web Anda untuk melihat hasilnya.
9. Eksplorasi:
    - Tambahkan paragraf baru di bagian

Tentang Saya `. * Tambahkan baris baru ke tabel` Pendidikan `. * Buat daftar tak berurut baru di bagian` Kontak` untuk menambahkan media sosial lain.

Penugasan

Buatlah halaman web sederhana yang berisi:

1. Informasi pribadi Anda (nama, hobi, minat).
2. Daftar 3-5 hobi atau minat Anda menggunakan daftar tak berurut.
3. Tabel sederhana yang berisi riwayat pendidikan atau pengalaman kerja (minimal 3 baris).
4. Gambar yang relevan dengan salah satu hobi atau minat Anda.
5. Tautan ke profil media sosial Anda atau situs web favorit.
6. Pastikan menggunakan elemen semantik HTML5 yang sesuai (`<header>` , `<main>` , `<section>` , `<footer>` , dll.).

Kumpulkan file `index.html` Anda.

Pertemuan 2: Dasar CSS (Styling dan Layout)

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Memahami konsep dasar CSS (Cascading Style Sheets).
2. Menggunakan berbagai cara untuk menyisipkan CSS ke dalam dokumen HTML.
3. Memilih elemen HTML menggunakan selektor CSS dasar (tipe, kelas, ID).
4. Menerapkan properti CSS untuk mengatur warna, font, ukuran, dan spasi.
5. Memahami model kotak (box model) CSS.
6. Membuat layout sederhana menggunakan Flexbox atau Grid.

Materi

CSS (Cascading Style Sheets) adalah bahasa stylesheet yang digunakan untuk mendeskripsikan presentasi dokumen yang ditulis dalam HTML atau XML. CSS memungkinkan Anda mengontrol tampilan halaman web, seperti warna, font, layout, dan responsivitas.

1. Cara Menyisipkan CSS

Ada tiga cara utama untuk menyisipkan CSS ke dalam dokumen HTML:

- **Inline Style**: Langsung pada elemen HTML menggunakan atribut `style`. (Tidak disarankan untuk proyek besar)

```html
<p style="color: blue; font-size: 16px;">Teks berwarna biru.</p>
```

- **Internal Style Sheet**: Di dalam elemen `<style>` di bagian `<head>` dokumen HTML.

```html
<head>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>
```

- **External Style Sheet**: File `.css` terpisah yang dihubungkan ke dokumen HTML menggunakan elemen `<link>` di bagian `<head>`. (Paling disarankan)

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

2. Selektor CSS Dasar

Selektor digunakan untuk "menemukan" (atau memilih) elemen HTML yang ingin Anda gayakan.

- **Selektor Tipe/Elemen**: Memilih semua elemen dengan nama tag tertentu.

```
p {
    color: black;
}
```

- **Selektor Kelas (`.`)**: Memilih elemen dengan atribut `class` tertentu.

```
<p class="highlight">Teks penting.</p>
```

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

- **Selektor ID (`#`)**: Memilih elemen dengan atribut `id` tertentu. ID harus unik dalam satu halaman.

```
<div id="header">Ini adalah header.</div>
```

```
#header {
    border: 1px solid red;
}
```

3. Properti CSS Umum

- **Warna**: `color` (teks), `background-color` (latar belakang).
- **Font**: `font-family`, `font-size`, `font-weight`, `font-style`.
- **Teks**: `text-align`, `text-decoration`, `line-height`.
- **Ukuran**: `width`, `height`.
- **Spasi**: `margin` (spasi luar), `padding` (spasi dalam).
- **Border**: `border` (ketebalan, gaya, warna).

4. Box Model CSS

Setiap elemen HTML dianggap sebagai kotak. Box model terdiri dari:

- **Content**: Konten aktual elemen (teks, gambar).
- **Padding**: Ruang di antara konten dan border.
- **Border**: Garis di sekitar padding dan konten.
- **Margin**: Ruang di luar border, memisahkan elemen dari elemen lain.

```
|        Margin         |
|  +-----------------+  |
|  |     Border      |  |
|  |  +-----------+  |  |
|  |  |  Padding  |  |  |
|  |  | +-------+ |  |  |
|  |  | |Content| |  |  |
|  |  | +-------+ |  |  |
|  |  |           |  |  |
|  |  +-----------+  |  |
|  +-----------------+  |
+-----------------------+
```

5. Layout dengan Flexbox

Flexbox (Flexible Box Layout) adalah modul layout satu dimensi yang dirancang untuk mendistribusikan ruang di antara item-item dalam sebuah kontainer, bahkan ketika ukurannya tidak diketahui atau dinamis.

- Untuk mengaktifkan Flexbox, atur `display: flex;` pada kontainer induk.
- Properti umum Flexbox: `justify-content`, `align-items`, `flex-direction`, `flex-wrap`.

Pembahasan

CSS adalah kunci untuk membuat halaman web terlihat menarik dan profesional. Dengan menguasai selektor dan properti dasar, serta memahami box model, mahasiswa dapat mulai mendesain layout yang kompleks. Penggunaan external style sheet adalah praktik terbaik untuk menjaga kode tetap rapi dan mudah dikelola. Flexbox akan menjadi alat utama untuk mengatur tata letak elemen secara efisien.

Contoh Kode

Berikut adalah contoh kode HTML dan CSS untuk halaman profil yang dipercantik:

`index.html`

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profil Saya - Dengan CSS</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>Profil Singkat</h1>
        <nav>
            <ul>
                <li><a href="#tentang">Tentang Saya</a></li>
                <li><a href="#pendidikan">Pendidikan</a></li>
                <li><a href="#kontak">Kontak</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="tentang">
            <h2>Tentang Saya</h2>
            <p>Halo, nama saya [Nama Anda]. Saya adalah seorang mahasiswa yang
tertarik pada pengembangan web.</p>
            <img src="https://via.placeholder.com/150" alt="Foto Profil">
        </section>
```

```
        <section id="pendidikan">
            <h2>Pendidikan</h2>
            <table>
                <thead>
                    <tr>
                        <th>Institusi</th>
                        <th>Jurusan</th>
                        <th>Tahun</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>Universitas XYZ</td>
                        <td>Teknik Informatika</td>
                        <td>2022 - Sekarang</td>
                    </tr>
                </tbody>
            </table>
        </section>

        <section id="kontak">
            <h2>Kontak</h2>
            <ul>
                <li>Email: <a href="mailto:nama@example.com">nama@example.com</a>
</li>
                <li>LinkedIn: <a href="#">[Profil LinkedIn Anda]</a></li>
            </ul>
        </section>
    </main>

    <footer>
        <p>&copy; 2025 [Nama Anda]. Hak Cipta Dilindungi.</p>
    </footer>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
```

```
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    color: #333;
}

header {
    background-color: #333;
    color: #fff;
    padding: 1rem 0;
    text-align: center;
}

header h1 {
    margin: 0;
}

nav ul {
    list-style: none;
    padding: 0;
    display: flex;
    justify-content: center;
}

nav ul li {
    margin: 0 15px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
}

main {
    padding: 20px;
    max-width: 800px;
    margin: 20px auto;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

section {
    margin-bottom: 20px;
    padding: 15px;
```

```
    border: 1px solid #ddd;
    border-radius: 5px;
}

img {
    max-width: 100%;
    height: auto;
    display: block;
    margin: 10px 0;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 10px;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 8px;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}

footer {
    text-align: center;
    padding: 20px;
    background-color: #333;
    color: #fff;
    position: relative;
    bottom: 0;
    width: 100%;
}
```

Langkah-langkah Praktikum

1. Lanjutkan dari folder `praktikum_html` yang Anda buat di Pertemuan 1.

2. Di dalam folder `praktikum_html`, buat file baru bernama `style.css`.
3. Buka `style.css` dengan editor teks Anda dan salin/tempel kode CSS di atas ke dalamnya.
4. Buka `index.html` Anda dari Pertemuan 1.
5. Tambahkan baris `<link rel="stylesheet" href="style.css">` di dalam elemen `<head>` `index.html` Anda, di bawah elemen `<title>`.
6. Simpan kedua file.
7. Buka `index.html` di browser Anda dan amati perubahan tampilannya.
8. Eksplorasi:
    - Ubah warna latar belakang `body` menjadi warna lain.
    - Ubah `font-family` untuk `body`.
    - Tambahkan `border-radius` pada gambar profil Anda.
    - Eksperimen dengan `margin` dan `padding` pada elemen `section`.
    - Coba ubah `justify-content` pada `nav ul` menjadi `space-around` atau `flex-start`.

Penugasan

Perbaiki tampilan halaman web pribadi Anda dari Penugasan Pertemuan 1 menggunakan CSS:

1. Buat file `style.css` terpisah dan tautkan ke `index.html` Anda.
2. Terapkan gaya dasar untuk `body`, `header`, `main`, `section`, dan `footer` (misalnya, warna latar belakang, font, padding).
3. Gunakan Flexbox untuk mengatur tata letak navigasi atau bagian lain di halaman Anda.
4. Berikan gaya pada tabel dan daftar agar lebih mudah dibaca.
5. Pastikan gambar Anda memiliki ukuran yang responsif (misalnya, `max-width: 100%; height: auto;`).
6. Kumpulkan file `index.html` dan `style.css` Anda.

Pertemuan 3: JavaScript Dasar (Variabel, Tipe Data, Operator, Kondisional)

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Memahami peran JavaScript dalam pengembangan web interaktif.
2. Mendeklarasikan dan menggunakan variabel dengan benar.
3. Mengenali dan bekerja dengan berbagai tipe data JavaScript.
4. Menggunakan operator aritmatika, perbandingan, dan logika.
5. Mengimplementasikan struktur kontrol kondisional (`if` , `else if` , `else` , `switch`).
6. Memahami konsep dasar input dan output menggunakan `alert()` , `prompt()` , dan `console.log()` .

Materi

JavaScript adalah bahasa pemrograman yang memungkinkan Anda mengimplementasikan hal-hal kompleks pada halaman web. Setiap kali Anda melihat peta interaktif, animasi grafis 2D/3D, atau pembaruan konten yang dinamis, kemungkinan besar JavaScript terlibat.

1. Cara Menyisipkan JavaScript

Ada dua cara utama untuk menyisipkan JavaScript ke dalam dokumen HTML:

- **Internal Script**: Di dalam elemen `<script>` di bagian `<head>` atau `<body>` dokumen HTML. Disarankan di akhir `<body>` agar HTML dimuat terlebih dahulu.

```
<body>
    <!-- Konten HTML lainnya -->
    <script>
        console.log("Halo dari JavaScript internal!");
    </script>
</body>
```

- **External Script**: File `.js` terpisah yang dihubungkan ke dokumen HTML menggunakan elemen `<script>` dengan atribut `src` . (Paling disarankan)

```
<body>
    <!-- Konten HTML lainnya -->
    <script src="script.js"></script>
</body>
```

2. Variabel

Variabel digunakan untuk menyimpan nilai data. JavaScript memiliki tiga kata kunci untuk mendeklarasikan variabel:

- `var`: Deklarasi lama, memiliki cakupan fungsi.
- `let`: Deklarasi modern, memiliki cakupan blok, bisa diubah nilainya.
- `const`: Deklarasi modern, memiliki cakupan blok, tidak bisa diubah nilainya (konstan).

```
let nama = "Yamin";
const umur = 25;
var pekerjaan = "Mahasiswa";

console.log(nama); // Output: Yamin
console.log(umur); // Output: 25
```

3. Tipe Data

JavaScript memiliki beberapa tipe data dasar:

- **Primitive Data Types**:
    - `String`: Teks (misalnya, `

"Halo", "123").
* `Number`: Angka (misalnya, `10`, `3.14`).
* `Boolean`: `true` atau `false`.
* `Undefined`: Variabel yang dideklarasikan tetapi belum diberi nilai.
* `Null`: Nilai yang sengaja tidak ada.
* `Symbol` (ES6):
* `BigInt` (ES11):

- **Non-Primitive Data Types**:
    - `Object`: Kumpulan pasangan key-value (misalnya, `{}`, `[]`).

4. Operator

- **Aritmatika**: `+`, `-`, `*`, `/`, `%` (modulus), `**` (eksponen).
- **Perbandingan**: `==` (sama nilai), `===` (sama nilai dan tipe), `!=` (tidak sama nilai), `!==` (tidak sama nilai atau tipe), `>`, `<`, `>=`, `<=`.
- **Logika**: `&&` (AND), `||` (OR), `!` (NOT).
- **Penugasan**: `=`, `+=`, `-=`, `*=`.

```
let a = 10;
let b = 5;

console.log(a + b); // 15
console.log(a > b); // true
console.log(a === '10'); // false (tipe berbeda)
```

5. Struktur Kontrol Kondisional

- `if`, `else if`, `else`:

```
let nilai = 75;

if (nilai >= 80) {
    console.log("Nilai A");
} else if (nilai >= 70) {
    console.log("Nilai B");
} else {
    console.log("Nilai C");
}
```

- `switch`:

```
let hari = "Senin";

switch (hari) {
    case "Senin":
```

```
            console.log("Hari kerja");
            break;
        case "Minggu":
            console.log("Hari libur");
            break;
        default:
            console.log("Bukan hari Senin atau Minggu");
    }
```

6. Input/Output Dasar

- `alert()`: Menampilkan kotak dialog peringatan.
- `prompt()`: Menampilkan kotak dialog dengan input teks.
- `console.log()`: Menampilkan output di konsol browser (untuk debugging).

Pembahasan

JavaScript adalah bahasa yang dinamis dan fleksibel. Memahami variabel, tipe data, operator, dan struktur kondisional adalah fundamental untuk menulis logika program yang efektif. Praktikum ini akan melatih mahasiswa untuk berpikir secara algoritmik dan menerapkan konsep-konsep dasar JavaScript untuk memecahkan masalah sederhana.

Contoh Kode

Berikut adalah contoh kode JavaScript yang meminta input nama dan menampilkan pesan personalisasi:

`script.js`

```
// Meminta input nama dari pengguna
let namaPengguna = prompt("Masukkan nama Anda:");

// Memeriksa apakah namaPengguna tidak kosong atau null
if (namaPengguna) {
    // Menampilkan pesan sapaan personalisasi
    alert("Halo, " + namaPengguna + "! Selamat datang di dunia JavaScript.");
    console.log("Pengguna bernama " + namaPengguna + " telah masuk.");

    // Contoh penggunaan operator dan kondisional
    let tahunLahir = prompt("Tahun berapa Anda lahir?");
    let tahunSekarang = new Date().getFullYear();
    let umur = tahunSekarang - parseInt(tahunLahir);
```

```
    if (umur >= 17) {
        console.log("Anda sudah cukup umur untuk memiliki KTP.");
    } else {
        console.log("Anda belum cukup umur untuk memiliki KTP.");
    }

} else {
    alert("Nama tidak boleh kosong!");
    console.log("Pengguna membatalkan atau tidak memasukkan nama.");
}
```

`index.html` **(untuk menghubungkan** `script.js` **)**

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Praktikum JavaScript Dasar</title>
</head>
<body>
    <h1>Lihat Konsol Browser untuk Output!</h1>
    <script src="script.js"></script>
</body>
</html>
```

Langkah-langkah Praktikum

1. Buat folder baru dengan nama `praktikum_js_dasar`.
2. Di dalam folder tersebut, buat file `index.html` dan `script.js`.
3. Salin kode `index.html` di atas ke dalam `index.html` Anda.
4. Salin kode `script.js` di atas ke dalam `script.js` Anda.
5. Buka `index.html` di browser Anda. Perhatikan kotak dialog `prompt` dan `alert` yang muncul.
6. Buka konsol browser (biasanya dengan menekan F12 atau klik kanan -> Inspect -> Console) untuk melihat output `console.log()`.
7. Eksplorasi:
    - Ubah pesan `alert` dan `prompt`.
    - Tambahkan variabel baru dengan tipe data berbeda (misalnya, boolean).
    - Buat kondisi `if-else if-else` yang lebih kompleks berdasarkan input pengguna.

- Coba gunakan operator logika (`&&` , `||`) dalam kondisi Anda.

Penugasan

Buatlah program JavaScript sederhana yang:

1. Meminta pengguna memasukkan dua angka.
2. Meminta pengguna memilih operasi aritmatika (`+` , `-` , `*` , `/`).
3. Melakukan perhitungan berdasarkan pilihan pengguna.
4. Menampilkan hasil perhitungan menggunakan `alert()` .
5. Menangani kasus pembagian dengan nol (jika angka kedua adalah nol, tampilkan pesan error).
6. Gunakan `console.log()` untuk menampilkan setiap langkah proses (input, operasi, hasil).

Kumpulkan file `index.html` dan `script.js` Anda.

Pertemuan 4: JavaScript Menengah (Loop, Fungsi, Array, Objek)

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Menggunakan struktur perulangan (`for` , `while` , `do-while` , `for...of` , `for...in`) untuk mengulang blok kode.
2. Mendefinisikan dan memanggil fungsi untuk mengorganisir kode dan menghindari pengulangan.
3. Bekerja dengan array untuk menyimpan koleksi data.
4. Membuat dan memanipulasi objek untuk merepresentasikan entitas kompleks.
5. Memahami konsep dasar fungsi callback dan fungsi panah (arrow functions).

Materi

Setelah menguasai dasar-dasar JavaScript, kita akan melangkah lebih jauh ke konsep-konsep menengah yang esensial untuk membangun aplikasi yang lebih kompleks dan terstruktur.

1. Perulangan (Loops)

Perulangan digunakan untuk mengeksekusi blok kode berulang kali.

- `for` **loop**: Digunakan ketika jumlah iterasi diketahui.

```
for (let i = 0; i < 5; i++) {
    console.log("Iterasi ke-" + i);
}
```

- `while` **loop**: Digunakan ketika jumlah iterasi tidak diketahui, perulangan berlanjut selama kondisi `true` .

```
let count = 0;
while (count < 3) {
    console.log("Hitungan: " + count);
    count++;
}
```

- `do-while` **loop**: Mirip dengan `while`, tetapi blok kode dieksekusi setidaknya sekali.

```
let i = 0;
do {
    console.log("Do-while iterasi: " + i);
    i++;
} while (i < 3);
```

- `for...of` **loop**: Mengulang nilai dari objek yang dapat diulang (seperti array, string).

```
const buah = ["Apel", "Pisang", "Ceri"];
for (const b of buah) {
    console.log(b);
}
```

- `for...in` **loop**: Mengulang properti (kunci) dari sebuah objek.

```
const orang = { nama: "Yamin", usia: 30 };
for (const kunci in orang) {
    console.log(kunci + ": " + orang[kunci]);
}
```

2. Fungsi

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi membuat kode lebih modular, dapat digunakan kembali, dan mudah dipelihara.

- **Deklarasi Fungsi**:

```
function sapa(nama) {
    return "Halo, " + nama + "!";
}
console.log(sapa("Ani")); // Output: Halo, Ani!
```

- **Ekspresi Fungsi**:

```
const hitungJumlah = function(a, b) {
    return a + b;
};
console.log(hitungJumlah(5, 3)); // Output: 8
```

- **Fungsi Panah (Arrow Functions)** (ES6):
  Sintaks yang lebih ringkas, terutama untuk fungsi anonim.

```
const kaliDua = (angka) => angka * 2;
console.log(kaliDua(7)); // Output: 14

const sapaLengkap = (namaDepan, namaBelakang) => {
    return `Halo, ${namaDepan} ${namaBelakang}!`
};
console.log(sapaLengkap("Abdul", "Yamin"));
```

3. Array

Array adalah objek yang digunakan untuk menyimpan koleksi data yang berurutan. Elemen array diakses menggunakan indeks numerik (dimulai dari 0).

- **Deklarasi Array**:

```
const angka = [1, 2, 3, 4, 5];
const namaSiswa = ["Alice", "Bob", "Charlie"];
```

- **Mengakses Elemen**:

```
console.log(angka[0]); // Output: 1
```

```
console.log(namaSiswa[1]); // Output: Bob
```

- **Metode Array Umum**:
  - `push()`: Menambah elemen ke akhir array.
  - `pop()`: Menghapus elemen terakhir dari array.
  - `unshift()`: Menambah elemen ke awal array.
  - `shift()`: Menghapus elemen pertama dari array.
  - `length`: Properti untuk mendapatkan jumlah elemen.
  - `indexOf()`: Mencari indeks elemen.
  - `forEach()`, `map()`, `filter()`, `reduce()`: Untuk iterasi dan transformasi array.

4. Objek

Objek adalah kumpulan properti, di mana setiap properti memiliki nama (kunci) dan nilai. Objek digunakan untuk merepresentasikan entitas dunia nyata.

- **Deklarasi Objek (Object Literal)**:

```
const buku = {
    judul: "Filosofi Teras",
    penulis: "Henry Manampiring",
    tahunTerbit: 2018,
    isAvailable: true
};
```

- **Mengakses Properti**:
  - Dot notation: `obj.property`
  - Bracket notation: `obj['property']`

```
console.log(buku.judul); // Output: Filosofi Teras
console.log(buku["penulis"]); // Output: Henry Manampiring
```

- **Menambah/Mengubah Properti**:

```
buku.penerbit = "Kompas";
buku.tahunTerbit = 2019;
```

- **Menghapus Properti**:

```
delete buku.isAvailable;
```

Pembahasan

Perulangan, fungsi, array, dan objek adalah pilar penting dalam pemrograman JavaScript. Menguasai konsep-konsep ini memungkinkan mahasiswa untuk menulis kode yang lebih efisien, terstruktur, dan dapat diskalakan. Fungsi memungkinkan modularitas, array menangani koleksi data, dan objek merepresentasikan data yang lebih kompleks. Pemahaman mendalam tentang topik ini akan menjadi dasar yang kuat untuk manipulasi DOM dan pengembangan aplikasi web yang lebih interaktif.

Contoh Kode

Berikut adalah contoh kode JavaScript yang mengilustrasikan penggunaan loop, fungsi, array, dan objek:

script.js

```
// 1. Contoh Penggunaan Array dan Loop
const daftarMahasiswa = [
    { nama: "Andi", nilai: 85 },
    { nama: "Yamin", nilai: 70 },
    { nama: "Citra", nilai: 92 },
    { nama: "Dewi", nilai: 60 }
];

console.log("\nDaftar Nilai Mahasiswa:");
for (let i = 0; i < daftarMahasiswa.length; i++) {
    const mhs = daftarMahasiswa[i];
    console.log(`${mhs.nama}: ${mhs.nilai}`);
}

// 2. Contoh Fungsi untuk Menghitung Rata-rata Nilai
```

```
function hitungRataRata(dataMahasiswa) {
    let totalNilai = 0;
    for (const mhs of dataMahasiswa) {
        totalNilai += mhs.nilai;
    }
    return totalNilai / dataMahasiswa.length;
}

const rataRata = hitungRataRata(daftarMahasiswa);
console.log(`\nRata-rata nilai kelas: ${rataRata.toFixed(2)}`);

// 3. Contoh Fungsi Panah dan Filter Array
const mahasiswaLulus = daftarMahasiswa.filter(mhs => mhs.nilai >= 75);
console.log("\nMahasiswa yang Lulus (nilai >= 75):");
mahasiswaLulus.forEach(mhs => {
    console.log(`${mhs.nama} (${mhs.nilai})`);
});

// 4. Contoh Objek dengan Metode
const kalkulator = {
    tambah: (a, b) => a + b,
    kurang: (a, b) => a - b,
    kali: (a, b) => a * b,
    bagi: (a, b) => {
        if (b === 0) {
            return "Error: Pembagian dengan nol!";
        } else {
            return a / b;
        }
    }
};

console.log("\nOperasi Kalkulator:");
console.log(`5 + 3 = ${kalkulator.tambah(5, 3)}`);
console.log(`10 - 4 = ${kalkulator.kurang(10, 4)}`);
console.log(`6 * 7 = ${kalkulator.kali(6, 7)}`);
console.log(`10 / 2 = ${kalkulator.bagi(10, 2)}`);
console.log(`10 / 0 = ${kalkulator.bagi(10, 0)}`);
```

`index.html` **(untuk menghubungkan** `script.js` **)**

```html
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Praktikum JavaScript Menengah</title>
</head>
<body>
    <h1>Buka Konsol Browser untuk Melihat Output JavaScript!</h1>
    <script src="script.js"></script>
</body>
</html>
```

Langkah-langkah Praktikum

1. Buat folder baru dengan nama `praktikum_js_menengah`.
2. Di dalam folder tersebut, buat file `index.html` dan `script.js`.
3. Salin kode `index.html` di atas ke dalam `index.html` Anda.
4. Salin kode `script.js` di atas ke dalam `script.js` Anda.
5. Buka `index.html` di browser Anda.
6. Buka konsol browser (F12 atau klik kanan -> Inspect -> Console) untuk melihat output dari `console.log()`.
7. Eksplorasi:
    - Tambahkan lebih banyak mahasiswa ke `daftarMahasiswa`.
    - Buat fungsi baru yang menerima array angka dan mengembalikan angka terbesar.
    - Buat objek baru yang merepresentasikan sebuah produk (nama, harga, stok) dan coba akses serta ubah propertinya.
    - Gunakan `for...in` loop untuk mengulang properti objek `kalkulator`.

Penugasan

Buatlah program JavaScript yang mengelola daftar tugas (To-Do List) sederhana:

1. Buat sebuah array bernama `daftarTugas` yang berisi beberapa objek tugas. Setiap objek tugas harus memiliki properti `deskripsi` (string) dan `selesai` (boolean).
   Contoh:

```javascript
const daftarTugas = [
```

```
    { deskripsi: "Belajar HTML", selesai: true },
    { deskripsi: "Mengerjakan CSS", selesai: false },
    { deskripsi: "Mulai JavaScript", selesai: false }
];
```

2. Buat fungsi bernama `tambahTugas(deskripsiTugas)` yang menambahkan tugas baru ke `daftarTugas` dengan `selesai: false` secara default.
3. Buat fungsi bernama `tandaiSelesai(indeksTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
4. Buat fungsi bernama `tampilkanTugas()` yang mengulang `daftarTugas` dan menampilkan setiap tugas ke konsol, dengan format: ☑ Deskripsi Tugas jika sudah selesai, atau ☐ Deskripsi Tugas jika belum selesai.
5. Panggil fungsi-fungsi tersebut untuk:
    - Menambahkan 2-3 tugas baru.
    - Menandai salah satu tugas yang belum selesai menjadi selesai.
    - Menampilkan semua tugas setelah perubahan.

Kumpulkan file `index.html` dan `script.js` Anda.

Pertemuan 5: Pengenalan DOM (Document Object Model) dan Manipulasi Dasar

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Memahami apa itu DOM dan perannya dalam pengembangan web interaktif.
2. Mengakses elemen HTML menggunakan berbagai metode seleksi DOM.
3. Memanipulasi konten teks dan atribut elemen HTML.
4. Mengubah gaya (CSS) elemen secara dinamis menggunakan JavaScript.
5. Membuat dan menghapus elemen HTML baru.

Materi

DOM (Document Object Model) adalah antarmuka pemrograman untuk dokumen HTML dan XML. Ini merepresentasikan halaman web sebagai struktur pohon (tree structure) di mana setiap node adalah bagian dari dokumen (elemen, atribut, teks, dll.). JavaScript dapat menggunakan DOM untuk mengakses dan memanipulasi struktur, konten, dan gaya halaman web secara dinamis.

1. Apa itu DOM?

Bayangkan halaman HTML Anda sebagai sebuah pohon keluarga. Setiap tag HTML (seperti `<html>`, `<body>`, `<p>`, `<div>`) adalah anggota keluarga (node). Hubungan antara tag-tag ini (induk, anak, saudara) membentuk struktur pohon. DOM adalah cara JavaScript untuk "berbicara" dengan pohon ini, memungkinkan kita untuk:

- Menemukan elemen HTML.
- Mengubah konten HTML.
- Mengubah gaya CSS.
- Menambah atau menghapus elemen HTML.
- Bereaksi terhadap peristiwa (events) pengguna.

2. Mengakses Elemen HTML (Seleksi DOM)

Ada beberapa metode untuk memilih elemen dari DOM:

- `document.getElementById('id')`: Memilih elemen berdasarkan atribut `id` yang unik.

```
const judul = document.getElementById('main-title');
console.log(judul.textContent); // Mengambil teks konten
```

- `document.getElementsByClassName('class')`: Memilih semua elemen dengan nama kelas tertentu. Mengembalikan HTMLCollection (mirip array).

```
const itemDaftar = document.getElementsByClassName('list-item');
for (let i = 0; i < itemDaftar.length; i++) {
    console.log(itemDaftar[i].textContent);
}
```

- `document.getElementsByTagName('tagname')`: Memilih semua elemen dengan nama tag tertentu. Mengembalikan HTMLCollection.

```
const semuaParagraf = document.getElementsByTagName('p');
```

- `document.querySelector('selector')`: Memilih elemen pertama yang cocok dengan selektor CSS yang diberikan. (Paling sering digunakan)

```
const pertamaParagraf = document.querySelector('p');
const elemenDenganKelas = document.querySelector('.highlight');
```

- `document.querySelectorAll('selector')`: Memilih semua elemen yang cocok dengan selektor CSS yang diberikan. Mengembalikan NodeList (mirip array).

```
const semuaLink = document.querySelectorAll('a');
semuaLink.forEach(link => {
    console.log(link.href);
});
```

3. Memanipulasi Konten dan Atribut

Setelah elemen dipilih, Anda dapat memanipulasinya:

- **Mengubah Konten Teks**:

- `element.textContent`: Mengatur atau mendapatkan konten teks dari elemen (mengabaikan HTML).
- `element.innerHTML`: Mengatur atau mendapatkan konten HTML dari elemen (termasuk tag HTML).

```
judul.textContent = "Judul Baru";
document.getElementById('konten').innerHTML = '<b>Teks tebal baru</b>';
```

- **Mengubah Atribut**:
  - `element.setAttribute('nama-atribut', 'nilai')`: Mengatur nilai atribut.
  - `element.getAttribute('nama-atribut')`: Mendapatkan nilai atribut.
  - `element.removeAttribute('nama-atribut')`: Menghapus atribut.

```
const gambar = document.querySelector('img');
gambar.setAttribute('src', 'gambar_baru.jpg');
console.log(gambar.getAttribute('alt'));
```

4. Mengubah Gaya (CSS) Dinamis

Anda dapat mengubah properti CSS elemen menggunakan properti `style`:

```
const kotak = document.getElementById('myBox');
kotak.style.backgroundColor = 'red';
kotak.style.fontSize = '20px';

// Menambah/menghapus kelas CSS
kotak.classList.add('active');
kotak.classList.remove('hidden');
kotak.classList.toggle('highlight'); // Menambah jika tidak ada, menghapus jika ada
```

5. Membuat dan Menghapus Elemen

- **Membuat Elemen Baru**:

- `document.createElement('tagname')`: Membuat elemen HTML baru.
- `document.createTextNode('text')`: Membuat node teks.

```
const newDiv = document.createElement('div');
newDiv.textContent = "Ini adalah div baru.";
document.body.appendChild(newDiv); // Menambahkan ke body
```

- **Menambahkan Elemen ke DOM**:
  - `parentNode.appendChild(childNode)`: Menambahkan node sebagai anak terakhir.
  - `parentNode.insertBefore(newNode, referenceNode)`: Menambahkan node sebelum node referensi.
- **Menghapus Elemen**:
  - `element.remove()`: Menghapus elemen itu sendiri.
  - `parentNode.removeChild(childNode)`: Menghapus anak dari induknya.

```
const elemenUntukDihapus = document.getElementById('oldElement');
elemenUntukDihapus.remove();
```

Pembahasan

Manipulasi DOM adalah inti dari interaktivitas web modern. Dengan DOM, JavaScript dapat merespons tindakan pengguna, memperbarui konten secara real-time, dan menciptakan pengalaman pengguna yang dinamis tanpa perlu memuat ulang halaman. Praktikum ini akan memberikan dasar yang kuat untuk berinteraksi dengan struktur halaman web menggunakan JavaScript.

Contoh Kode

Berikut adalah contoh sederhana yang menunjukkan manipulasi DOM:

`index.html`

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Praktikum DOM Dasar</title>
    <style>
        #kotak {
            width: 100px;
            height: 100px;
            background-color: lightblue;
            margin: 20px;
            border: 1px solid blue;
            display: flex;
            justify-content: center;
            align-items: center;
            font-size: 1.2em;
            transition: background-color 0.3s;
        }
        .highlight {
            background-color: yellow;
            border-color: orange;
        }
    </style>
</head>
<body>
    <h1 id="judul">Selamat Datang di DOM!</h1>
    <p class="teks-info">Ini adalah paragraf informasi.</p>
    <button id="ubahJudul">Ubah Judul</button>
    <button id="tambahParagraf">Tambah Paragraf</button>
    <button id="hapusParagraf">Hapus Paragraf Terakhir</button>
    <button id="ubahWarna">Ubah Warna Kotak</button>
    <button id="toggleHighlight">Toggle Highlight</button>

    <div id="kotak">Kotak Saya</div>

    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.addEventListener('DOMContentLoaded', function() {
    const judulElement = document.getElementById('judul');
    const ubahJudulBtn = document.getElementById('ubahJudul');
    const tambahParagrafBtn = document.getElementById('tambahParagraf');
```

```
    const hapusParagrafBtn = document.getElementById('hapusParagraf');
    const ubahWarnaBtn = document.getElementById('ubahWarna');
    const kotakElement = document.getElementById('kotak');
    const toggleHighlightBtn = document.getElementById('toggleHighlight');

    // Mengubah judul
    ubahJudulBtn.addEventListener('click', function() {
        judulElement.textContent = 'Judul Telah Diubah oleh DOM!';
    });

    // Menambah paragraf baru
    tambahParagrafBtn.addEventListener('click', function() {
        const newParagraph = document.createElement('p');
        newParagraph.textContent = 'Paragraf baru ditambahkan secara dinamis.';
        newParagraph.classList.add('teks-info');
        document.body.insertBefore(newParagraph, tambahParagrafBtn.nextSibling);
// Tambah setelah tombol
    });

    // Menghapus paragraf terakhir
    hapusParagrafBtn.addEventListener('click', function() {
        const semuaParagrafInfo = document.querySelectorAll('.teks-info');
        if (semuaParagrafInfo.length > 0) {
            semuaParagrafInfo[semuaParagrafInfo.length - 1].remove();
        } else {
            alert('Tidak ada paragraf untuk dihapus!');
        }
    });

    // Mengubah warna kotak
    ubahWarnaBtn.addEventListener('click', function() {
        const randomColor = '#' + Math.floor(Math.random()*16777215).toString(16);
        kotakElement.style.backgroundColor = randomColor;
    });

    // Toggle kelas highlight pada kotak
    toggleHighlightBtn.addEventListener('click', function() {
        kotakElement.classList.toggle('highlight');
    });
});
```

Langkah-langkah Praktikum

1. Buat folder baru dengan nama `praktikum_dom_dasar`.
2. Di dalam folder tersebut, buat file `index.html` dan `script.js`.
3. Salin kode `index.html` di atas ke dalam `index.html` Anda.
4. Salin kode `script.js` di atas ke dalam `script.js` Anda.
5. Buka `index.html` di browser Anda.
6. Klik tombol-tombol yang tersedia dan amati perubahan pada halaman.
7. Buka konsol browser untuk melihat jika ada error atau output `console.log()`.
8. Eksplorasi:
    - Ubah teks tombol.
    - Coba ubah atribut `id` atau `class` dari elemen yang sudah ada.
    - Buat tombol baru yang mengubah ukuran font dari `h1`.
    - Buat tombol yang menambahkan gambar baru ke halaman.

Penugasan

Buatlah halaman web sederhana dengan fungsionalitas berikut menggunakan manipulasi DOM:

1. Sebuah `<div>` dengan `id="counter"` yang menampilkan angka `0`.
2. Dua tombol: satu dengan `id="incrementBtn"` dan satu lagi dengan `id="decrementBtn"`.
3. Ketika `incrementBtn` diklik, angka di dalam `div#counter` bertambah 1.
4. Ketika `decrementBtn` diklik, angka di dalam `div#counter` berkurang 1.
5. Pastikan angka tidak bisa kurang dari 0.
6. Jika angka mencapai 10, ubah warna teks di `div#counter` menjadi hijau. Jika kurang dari 10, kembalikan ke warna hitam.

Kumpulkan file `index.html` dan `script.js` Anda.

Pertemuan 6: DOM Lanjutan (Event Handling, Form Validation, Integrasi)

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Mengimplementasikan penanganan peristiwa (event handling) pada elemen DOM.
2. Memahami berbagai jenis peristiwa (klik, submit, keypress, dll.).
3. Melakukan validasi formulir sisi klien (client-side form validation).
4. Mengintegrasikan JavaScript dengan HTML dan CSS untuk membangun antarmuka pengguna yang dinamis.
5. Membangun aplikasi web sederhana yang interaktif dan responsif.

Materi

Pada pertemuan terakhir ini, kita akan menggabungkan semua pengetahuan yang telah diperoleh tentang HTML, CSS, dan JavaScript, dengan fokus pada penanganan peristiwa dan validasi formulir untuk menciptakan pengalaman pengguna yang lebih kaya.

1. Penanganan Peristiwa (Event Handling)

Peristiwa adalah sinyal bahwa sesuatu telah terjadi di browser (misalnya, halaman selesai dimuat, tombol diklik, input diubah). JavaScript memungkinkan kita untuk "mendengarkan" peristiwa ini dan menjalankan fungsi sebagai respons.

- **Metode** `addEventListener()`: Cara paling modern dan disarankan untuk mendaftarkan event handler.

```
const tombol = document.getElementById("myButton");
tombol.addEventListener("click", function() {
    alert("Tombol diklik!");
});

// Dengan fungsi panah
tombol.addEventListener("mouseover", () => {
    console.log("Mouse di atas tombol");
});
```

- **Jenis Peristiwa Umum**:

    - **Mouse Events**: `click`, `dblclick`, `mouseover`, `mouseout`, `mousedown`, `mouseup`.
    - **Keyboard Events**: `keydown`, `keyup`, `keypress`.
    - **Form Events**: `submit`, `change`, `focus`, `blur`, `input`.
    - **Document/Window Events**: `DOMContentLoaded`, `load`, `resize`, `scroll`.

- **Objek Event**: Fungsi event handler menerima objek `event` sebagai argumen, yang berisi informasi tentang peristiwa yang terjadi.

```
document.addEventListener("click", function(event) {
    console.log("Elemen yang diklik:", event.target);
    console.log("Koordinat X:", event.clientX);
});
```

- `event.preventDefault()`: Mencegah perilaku default dari suatu peristiwa (misalnya, mencegah formulir dikirim saat `submit`).

2. Validasi Formulir Sisi Klien

Validasi formulir adalah proses memastikan bahwa input pengguna memenuhi kriteria tertentu sebelum data dikirim ke server. Validasi sisi klien memberikan umpan balik instan kepada pengguna.

- **Mengakses Nilai Input**: Gunakan properti `value` dari elemen input.

```
const inputNama = document.getElementById("nama");
const nilaiNama = inputNama.value;
```

- **Pengecekan Kondisi**: Gunakan pernyataan `if/else` untuk memeriksa apakah input kosong, formatnya benar (misalnya, email), atau panjangnya sesuai.
- **Memberikan Umpan Balik**: Ubah gaya elemen input (misalnya, border merah), tampilkan pesan error di dekat input, atau gunakan `alert()`.

3. Integrasi HTML, CSS, dan JavaScript

Untuk membangun aplikasi web yang kohesif, penting untuk memahami bagaimana ketiga teknologi ini bekerja sama:

- **HTML**: Menyediakan struktur dan konten dasar.
- **CSS**: Mengatur presentasi dan layout, termasuk responsivitas.
- **JavaScript**: Menambahkan interaktivitas, memanipulasi DOM, dan menangani peristiwa.

Contoh integrasi:

- JavaScript mengubah kelas CSS elemen untuk mengubah tampilannya (misalnya, `element.classList.add("error")`).
- JavaScript membuat elemen HTML baru dan menyisipkannya ke DOM.
- JavaScript mengambil data dari formulir HTML dan memprosesnya.

Pembahasan

Penanganan peristiwa adalah fondasi dari setiap aplikasi web interaktif. Dengan menguasai event handling, mahasiswa dapat membuat halaman web yang responsif terhadap tindakan pengguna. Validasi formulir sisi klien adalah praktik penting untuk meningkatkan pengalaman pengguna dan mengurangi beban server. Pertemuan ini akan menyatukan semua konsep yang telah dipelajari untuk membangun aplikasi web yang lebih fungsional dan dinamis.

Contoh Kode

Berikut adalah contoh aplikasi kalkulator sederhana dengan validasi formulir dan penanganan peristiwa:

`index.html`

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Kalkulator Interaktif</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            background-color: #f0f2f5;
            margin: 0;
        }
        .container {
            background-color: #fff;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
            text-align: center;
```

```
            width: 300px;
        }
        h1 {
            color: #333;
            margin-bottom: 20px;
        }
        input, select, button {
            width: calc(100% - 20px);
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
        }
        button {
            background-color: #007bff;
            color: white;
            border: none;
            cursor: pointer;
            font-size: 16px;
            transition: background-color 0.3s ease;
        }
        button:hover {
            background-color: #0056b3;
        }
        #result {
            margin-top: 20px;
            font-size: 1.5em;
            font-weight: bold;
            color: #28a745;
        }
        .error-message {
            color: red;
            font-size: 0.9em;
            margin-top: -10px;
            margin-bottom: 10px;
            text-align: left;
        }
        input.invalid {
            border-color: red;
        }
    </style>
</head>
<body>
```

```
    <div class="container">
        <h1>Kalkulator Sederhana</h1>
        <form id="calculatorForm">
            <input type="text" id="num1" placeholder="Angka Pertama">
            <div id="num1Error" class="error-message"></div>

            <select id="operator">
                <option value="+">+</option>
                <option value="-">-</option>
                <option value="*">*</option>
                <option value="/">/</option>
            </select>

            <input type="text" id="num2" placeholder="Angka Kedua">
            <div id="num2Error" class="error-message"></div>

            <button type="submit">Hitung</button>
        </form>
        <div id="result"></div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.addEventListener("DOMContentLoaded", function() {
    const calculatorForm = document.getElementById("calculatorForm");
    const num1Input = document.getElementById("num1");
    const num2Input = document.getElementById("num2");
    const operatorSelect = document.getElementById("operator");
    const resultDiv = document.getElementById("result");
    const num1Error = document.getElementById("num1Error");
    const num2Error = document.getElementById("num2Error");

    calculatorForm.addEventListener("submit", function(event) {
        event.preventDefault(); // Mencegah form dari submit default

        // Reset error messages and styles
        num1Error.textContent = "";
```

```
        num2Error.textContent = "";
        num1Input.classList.remove("invalid");
        num2Input.classList.remove("invalid");
        resultDiv.textContent = "";

        let isValid = true;

        const num1 = parseFloat(num1Input.value);
        const num2 = parseFloat(num2Input.value);
        const operator = operatorSelect.value;

        // Validasi input pertama
        if (isNaN(num1)) {
            num1Error.textContent = "Angka pertama harus berupa angka.";
            num1Input.classList.add("invalid");
            isValid = false;
        }

        // Validasi input kedua
        if (isNaN(num2)) {
            num2Error.textContent = "Angka kedua harus berupa angka.";
            num2Input.classList.add("invalid");
            isValid = false;
        }

        if (!isValid) {
            return; // Hentikan eksekusi jika ada error validasi
        }

        let result;
        switch (operator) {
            case "+":
                result = num1 + num2;
                break;
            case "-":
                result = num1 - num2;
                break;
            case "*":
                result = num1 * num2;
                break;
            case "/":
                if (num2 === 0) {
                    resultDiv.textContent = "Error: Pembagian dengan nol!";
                    resultDiv.style.color = "red";
```

```
                    return;
                } else {
                    result = num1 / num2;
                }
                break;
            default:
                resultDiv.textContent = "Operator tidak valid.";
                resultDiv.style.color = "red";
                return;
        }

        resultDiv.textContent = `Hasil: ${result.toFixed(2)}`;
        resultDiv.style.color = "#28a745"; // Kembalikan warna hijau jika berhasil
    });

    // Event listener untuk membersihkan error saat input berubah
    num1Input.addEventListener("input", function() {
        num1Error.textContent = "";
        num1Input.classList.remove("invalid");
    });

    num2Input.addEventListener("input", function() {
        num2Error.textContent = "";
        num2Input.classList.remove("invalid");
    });
});
```

Langkah-langkah Praktikum

1. Buat folder baru dengan nama `praktikum_dom_lanjutan`.
2. Di dalam folder tersebut, buat file `index.html` dan `script.js`.
3. Salin kode `index.html` di atas ke dalam `index.html` Anda (termasuk bagian `<style>`).
4. Salin kode `script.js` di atas ke dalam `script.js` Anda.
5. Buka `index.html` di browser Anda.
6. Coba masukkan angka dan pilih operator, lalu klik "Hitung".
7. Coba masukkan input non-angka atau lakukan pembagian dengan nol dan amati pesan errornya.
8. Eksplorasi:
    - Tambahkan operator lain (misalnya, modulus `%`).
    - Buat validasi tambahan, misalnya, memastikan input tidak kosong.
    - Ubah tampilan pesan error agar lebih menonjol.
    - Tambahkan tombol "Reset" yang akan mengosongkan input dan hasil.

Penugasan

Buatlah halaman web sederhana yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

1. HTML: Buat sebuah input teks untuk item baru, sebuah tombol "Tambah Item", dan sebuah daftar tak berurut (`<ul>`) kosong dengan `id="itemList"` .
2. CSS: Berikan gaya dasar agar tampilan menarik (misalnya, styling untuk input, tombol, dan item daftar).
3. JavaScript:
    - Ketika tombol "Tambah Item" diklik:
        - Ambil nilai dari input teks.
        - Lakukan validasi: jika input kosong, tampilkan pesan error dan jangan tambahkan item.
        - Jika tidak kosong, buat elemen `<li>` baru.
        - Tambahkan teks dari input ke `<li>` .
        - Tambahkan tombol "Hapus" (`<button>Hapus</button>`) di samping setiap item `<li>` yang baru dibuat.
        - Tambahkan `<li>` ini ke `<ul>` dengan `id="itemList"` .
        - Kosongkan input teks setelah item ditambahkan.
    - Ketika tombol "Hapus" di samping item diklik:
        - Hapus item `<li>` yang bersangkutan dari daftar.

Kumpulkan file `index.html` , `style.css` , dan `script.js` Anda.