

Modul Ajar: Pengembangan Web Dasar (HTML, CSS, JavaScript, DOM)

Untuk Mahasiswa Semester 5 - Persiapan PHP/MySQL

Pengantar

Modul ajar ini dirancang untuk mahasiswa semester 5 yang telah memiliki dasar pemrograman dan ingin mendalami 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 terbaik dalam membangun antarmuka web interaktif.

Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

- Membangun struktur halaman web menggunakan HTML semantik dan formulir untuk interaksi database.
- Mendesain tampilan halaman web menggunakan CSS dengan pemisahan presentasi dari data.
- Mengimplementasikan logika interaktif pada halaman web menggunakan JavaScript dengan fokus pada tipe data dan konversi.
- Memanipulasi elemen-elemen HTML dan CSS secara dinamis menggunakan Document Object Model (DOM) untuk pengambilan dan pengiriman data.
- Memahami konsep komunikasi asinkron (AJAX/Fetch API) untuk interaksi dengan backend.
- Mengembangkan aplikasi web sederhana yang interaktif, responsif, dan siap untuk integrasi dengan backend PHP/MySQL.

Struktur Modul

Modul ini terbagi menjadi 6 pertemuan praktikum, masing-masing dengan tujuan, materi, pembahasan, contoh kode, langkah-langkah praktikum, dan penugasan yang disesuaikan untuk persiapan pengembangan web full-stack.

Pertemuan 1: HTML Semantik dan Formulir untuk Database

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 formulir HTML yang siap untuk interaksi dengan database.
4. Memahami berbagai jenis input dan validasi HTML5.
5. Membuat tabel untuk menampilkan data dari database.
6. Memahami pentingnya atribut `name`, `action`, dan `method` dalam formulir.

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. Dalam konteks pengembangan web dengan PHP dan MySQL, HTML berfungsi sebagai antarmuka utama untuk menampilkan data dari database dan mengumpulkan input pengguna.

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).



```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistem Manajemen Mahasiswa</title>
</head>
<body>
    <h1>Selamat Datang di Sistem Manajemen Mahasiswa</h1>
    <p>Sistem ini akan terhubung dengan database MySQL melalui PHP.</p>
</body>
</html>
```

2. Elemen HTML Semantik

HTML5 memperkenalkan elemen semantik yang memberikan makna pada struktur konten. Dalam aplikasi web yang terhubung dengan database, struktur semantik membantu dalam organisasi data dan memudahkan pemeliharaan kode.

- `<header>` : Bagian pengantar atau navigasi sistem.
- `<nav>` : Menu navigasi untuk berbagai fitur aplikasi.
- `<main>` : Konten utama aplikasi (formulir, tabel data).

- <article> : Konten mandiri seperti detail record database.
- <section> : Bagian tematik seperti form input atau tampilan data.
- <aside> : Informasi tambahan seperti statistik atau filter.
- <footer> : Informasi sistem atau copyright.

3. Formulir HTML untuk Interaksi Database

Formulir adalah elemen kunci dalam aplikasi web yang berinteraksi dengan database. Setiap input dalam formulir akan menjadi data yang dikirim ke server untuk diproses oleh PHP dan disimpan ke MySQL.



```
<form action="proses_mahasiswa.php" method="POST">
    <!-- Input teks untuk data string -->
    <label for="nama">Nama Lengkap:</label>
    <input type="text" id="nama" name="nama" required maxlength="100">

    <!-- Input untuk NIM dengan pattern validation -->
    <label for="nim">NIM:</label>
    <input type="text" id="nim" name="nim" required pattern="[0-9]{10}" title="NIM harus 10 digit angka">

    <!-- Input email dengan validasi -->
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <!-- Input password -->
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required minlength="8">

    <!-- Input date untuk tanggal lahir -->
    <label for="tanggal_lahir">Tanggal Lahir:</label>
    <input type="date" id="tanggal_lahir" name="tanggal_lahir" required>

    <!-- Select dropdown untuk data referensi -->
    <label for="jurusan">Jurusan:</label>
    <select id="jurusan" name="jurusan" required>
        <option value="">Pilih Jurusan</option>
        <option value="ti">Teknik Informatika</option>
        <option value="si">Sistem Informasi</option>
        <option value="mi">Manajemen Informatika</option>
    </select>

    <!-- Textarea untuk data teks panjang -->
```

```
<label for="alamat">Alamat:</label>
<textarea id="alamat" name="alamat" rows="3" maxlength="255"></textarea>

<!-- Radio button untuk pilihan tunggal --&gt;
&lt;fieldset&gt;
    &lt;legend&gt;Jenis Kelamin:&lt;/legend&gt;
    &lt;input type="radio" id="laki" name="jenis_kelamin" value="L" required&gt;
    &lt;label for="laki"&gt;Laki-laki&lt;/label&gt;
    &lt;input type="radio" id="perempuan" name="jenis_kelamin" value="P" required&gt;
    &lt;label for="perempuan"&gt;Perempuan&lt;/label&gt;
&lt;/fieldset&gt;

<!-- Checkbox untuk pilihan multiple --&gt;
&lt;fieldset&gt;
    &lt;legend&gt;Minat/Hobi:&lt;/legend&gt;
    &lt;input type="checkbox" id="programming" name="hobi[]" value="programming"&gt;
    &lt;label for="programming"&gt;Programming&lt;/label&gt;
    &lt;input type="checkbox" id="design" name="hobi[]" value="design"&gt;
    &lt;label for="design"&gt;Design&lt;/label&gt;
    &lt;input type="checkbox" id="networking" name="hobi[]" value="networking"&gt;
    &lt;label for="networking"&gt;Networking&lt;/label&gt;
&lt;/fieldset&gt;

<!-- Hidden input untuk data sistem --&gt;
&lt;input type="hidden" name="created_at" value="&lt;?php echo date('Y-m-d H:i:s'); ?&gt;"&gt;

&lt;button type="submit" name="submit"&gt;Simpan Data Mahasiswa&lt;/button&gt;
&lt;button type="reset"&gt;Reset Form&lt;/button&gt;
&lt;/form&gt;</pre>
```

Konsep Penting untuk PHP/MySQL:

- **Atribut** `name` : Akan menjadi kunci array `$_POST` atau `$_GET` di PHP.
- **Atribut** `action` : Menentukan file PHP yang akan memproses data.
- **Atribut** `method` : `POST` untuk data sensitif/besar, `GET` untuk query sederhana.
- **Validasi HTML5**: Membantu mengurangi beban validasi di sisi server.
- **Tipe Input**: Sesuaikan dengan tipe data di database MySQL.

4. Tabel untuk Menampilkan Data Database

Tabel HTML digunakan untuk menampilkan data yang diambil dari database MySQL melalui PHP.

```
<section id="data-mahasiswa">
    <h2>Data Mahasiswa</h2>
    <table>
        <thead>
            <tr>
                <th>NIM</th>
                <th>Nama</th>
                <th>Email</th>
                <th>Jurusan</th>
                <th>Jenis Kelamin</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody>
            <!-- Data ini akan diisi oleh PHP dari database MySQL -->
            <tr>
                <td>2023001001</td>
                <td>Ahmad Budi</td>
                <td>ahmad.budi@email.com</td>
                <td>Teknik Informatika</td>
                <td>Laki-laki</td>
                <td>
                    <a href="edit.php?id=1">Edit</a> |
                    <a href="delete.php?id=1" onclick="return confirm('Yakin hapus?')">Hapus</a>
                </td>
            </tr>
            <tr>
                <td>2023001002</td>
                <td>Siti Aminah</td>
                <td>siti.aminah@email.com</td>
                <td>Sistem Informasi</td>
                <td>Perempuan</td>
                <td>
                    <a href="edit.php?id=2">Edit</a> |
                    <a href="delete.php?id=2" onclick="return confirm('Yakin hapus?')">Hapus</a>
                </td>
            </tr>
        </tbody>
    </table>
</section>
```

Pembahasan

HTML adalah fondasi dari setiap aplikasi web. Dalam konteks pengembangan dengan PHP dan MySQL, HTML berfungsi sebagai jembatan antara pengguna dan database. Formulir HTML mengumpulkan input pengguna yang akan diproses oleh PHP dan disimpan ke MySQL, sementara tabel HTML menampilkan data yang diambil dari database. Pemahaman mendalam tentang struktur HTML semantik dan formulir yang baik akan memudahkan integrasi dengan backend dan meningkatkan pengalaman pengguna.

Contoh Kode

Berikut adalah contoh halaman lengkap sistem manajemen mahasiswa:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistem Manajemen Mahasiswa</title>
</head>
<body>
    <header>
        <h1>Sistem Manajemen Mahasiswa</h1>
        <nav>
            <ul>
                <li><a href="#form-mahasiswa">Input Data</a></li>
                <li><a href="#data-mahasiswa">Lihat Data</a></li>
                <li><a href="#statistik">Statistik</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="form-mahasiswa">
            <h2>Form Input Data Mahasiswa</h2>
            <form action="proses_mahasiswa.php" method="POST">
                <fieldset>
                    <legend>Data Pribadi</legend>

                    <label for="nama">Nama Lengkap:</label>
                    <input type="text" id="nama" name="nama" required
maxlength="100">
```

```
<label for="nim">NIM:</label>
<input type="text" id="nim" name="nim" required pattern="[0-9]{10}" title="NIM harus 10 digit angka">

<label for="email">Email:</label>
<input type="email" id="email" name="email" required>

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

<label>Jenis Kelamin:</label>
<input type="radio" id="laki" name="jenis_kelamin" value="L" required>
    <label for="laki">Laki-laki</label>
    <input type="radio" id="perempuan" name="jenis_kelamin" value="P" required>
        <label for="perempuan">Perempuan</label>
    </fieldset>

<fieldset>
    <legend>Data Akademik</legend>

    <label for="jurusan">Jurusan:</label>
    <select id="jurusan" name="jurusan" required>
        <option value="">Pilih Jurusan</option>
        <option value="ti">Teknik Informatika</option>
        <option value="si">Sistem Informasi</option>
        <option value="mi">Manajemen Informatika</option>
    </select>

    <label for="semester">Semester:</label>
    <input type="number" id="semester" name="semester" min="1" max="8" required>
</fieldset>

<fieldset>
    <legend>Informasi Tambahan</legend>

    <label for="alamat">Alamat:</label>
    <textarea id="alamat" name="alamat" rows="3" maxlength="255">
</textarea>
```

```
<label>Minat/Hobi:</label>
<input type="checkbox" id="programming" name="hobi[]"
value="programming">
    <label for="programming">Programming</label>
    <input type="checkbox" id="design" name="hobi[]"
value="design">
    <label for="design">Design</label>
    <input type="checkbox" id="networking" name="hobi[]"
value="networking">
        <label for="networking">Networking</label>
</fieldset>

<button type="submit" name="submit">Simpan Data</button>
<button type="reset">Reset Form</button>
</form>
</section>

<section id="data-mahasiswa">
    <h2>Data Mahasiswa Terdaftar</h2>
    <table>
        <thead>
            <tr>
                <th>No</th>
                <th>NIM</th>
                <th>Nama</th>
                <th>Email</th>
                <th>Jurusان</th>
                <th>Semester</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>2023001001</td>
                <td>Ahmad Budi Santoso</td>
                <td>ahmad.budi@email.com</td>
                <td>Teknik Informatika</td>
                <td>5</td>
                <td>
                    <a href="detail.php?id=1">Detail</a> |
                    <a href="edit.php?id=1">Edit</a> |
                    <a href="delete.php?id=1" onclick="return

```

```
confirm('Yakin hapus?')>Hapus</a>
                                </td>
                            </tr>
                            <tr>
                                <td>2</td>
                                <td>2023001002</td>
                                <td>Siti Aminah</td>
                                <td>siti.aminah@email.com</td>
                                <td>Sistem Informasi</td>
                                <td>3</td>
                                <td>
                                    <a href="detail.php?id=2">Detail</a> |
                                    <a href="edit.php?id=2">Edit</a> |
                                    <a href="delete.php?id=2" onclick="return
confirm('Yakin hapus?')>Hapus</a>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </section>
            </main>

        <footer>
            <p>&copy; 2025 Sistem Manajemen Mahasiswa. Dikembangkan dengan HTML, CSS, JavaScript, PHP, dan MySQL.</p>
        </footer>
    </body>
</html>
```

Langkah-langkah Praktikum

1. Buat folder baru di komputer Anda dengan nama **praktikum_html_database**.
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. Sesuaikan data dalam tabel dengan informasi mahasiswa yang relevan.
6. Simpan file **index.html**.
7. Buka file **index.html** di browser web Anda untuk melihat hasilnya.
8. Eksplorasi:
 - Coba isi formulir dan perhatikan validasi HTML5 yang bekerja.
 - Tambahkan field baru seperti "No. Telepon" dengan validasi pattern.
 - Tambahkan lebih banyak data mahasiswa ke dalam tabel.

- Eksperimen dengan berbagai jenis input HTML5.

Penugasan

Buatlah halaman web sistem manajemen perpustakaan sederhana yang berisi:

1. Formulir input buku baru dengan field: Judul Buku, Pengarang, ISBN (dengan pattern validation), Tahun Terbit (number input), Kategori (select dropdown), Deskripsi (textarea), dan Status Ketersediaan (radio button: Tersedia/Dipinjam).
2. Tabel yang menampilkan daftar buku dengan kolom: No, Judul, Pengarang, Kategori, Tahun Terbit, Status, dan Aksi (Edit/Hapus).
3. Gunakan elemen semantik HTML5 yang sesuai (`<header>`, `<main>`, `<section>`, `<footer>`, dll.).
4. Tambahkan validasi HTML5 yang sesuai untuk setiap input.
5. Simulasikan minimal 5 data buku dalam tabel.
6. Pastikan atribut `name` pada setiap input sesuai dengan nama field yang akan digunakan di database.

Kumpulkan file `index.html` Anda.

Pertemuan 2: CSS untuk Pemisahan Presentasi dan Data

Tujuan Pembelajaran

Pada akhir pertemuan ini, mahasiswa diharapkan mampu:

1. Memahami konsep dasar CSS dan pentingnya pemisahan presentasi dari data.
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 tampilan formulir dan tabel database.
5. Memahami model kotak (box model) CSS untuk layout yang konsisten.
6. Membuat layout responsif menggunakan Flexbox atau Grid untuk aplikasi web.

Materi

CSS (Cascading Style Sheets) adalah bahasa stylesheet yang digunakan untuk mendeskripsikan presentasi dokumen HTML. Dalam pengembangan web dengan PHP dan MySQL, CSS memainkan peran penting dalam memisahkan tampilan (presentasi) dari logika bisnis dan data. Data yang diambil dari MySQL oleh PHP akan disajikan dalam HTML, dan CSS bertanggung jawab penuh atas bagaimana data tersebut terlihat oleh pengguna.

1. Pentingnya Pemisahan Presentasi dan Data

Ketika Anda bekerja dengan PHP dan MySQL, PHP akan mengambil data dari database dan "mencetak" HTML yang berisi data tersebut. CSS kemudian akan mengambil alih untuk menata HTML tersebut. Ini adalah prinsip dasar dari arsitektur Model-View-Controller (MVC):

- **Model (MySQL)**: Menyimpan dan mengelola data.
- **Controller (PHP)**: Mengambil data dari Model, memprosesnya, dan mengirimkannya ke View.
- **View (HTML/CSS)**: Menampilkan data kepada pengguna.

Dengan menjaga CSS di file terpisah (.css), Anda dapat:

- **Mempermudah Pemeliharaan**: Perubahan desain tidak memerlukan perubahan pada kode PHP yang menangani data.
- **Meningkatkan Keterbacaan Kode**: Kode HTML/PHP tetap bersih dari styling.
- **Meningkatkan Kinerja**: File CSS dapat di-cache oleh browser, mengurangi waktu muat halaman.
- **Memungkinkan Desain Responsif**: CSS memungkinkan Anda menyesuaikan tampilan data untuk berbagai ukuran layar tanpa mengubah struktur data atau logika backend.

2. Cara Menyisipkan CSS

Untuk aplikasi web yang terhubung dengan database, selalu gunakan external style sheet:

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

3. Selektor CSS untuk Aplikasi Database

```
/* Selektor untuk elemen formulir */
form {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
}

/* Selektor untuk tabel data */
table.data-table {
    width: 100%;
    border-collapse: collapse;
}
```

```
/* Selektor untuk status data */
.status-active {
    color: green;
    font-weight: bold;
}

.status-inactive {
    color: red;
    font-weight: bold;
}

/* Selektor untuk pesan sistem */
#success-message {
    background-color: #d4edda;
    color: #155724;
    padding: 10px;
    border-radius: 4px;
}

#error-message {
    background-color: #f8d7da;
    color: #721c24;
    padding: 10px;
    border-radius: 4px;
}
```

4. Styling Formulir untuk Input Database

```
/* Container formulir */
.form-container {
    background-color: #ffffff;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    margin-bottom: 30px;
}

/* Fieldset styling */
fieldset {
    border: 2px solid #e9ecef;
```

```
border-radius: 6px;
padding: 20px;
margin-bottom: 20px;
}

legend {
    font-weight: bold;
    color: #495057;
    padding: 0 10px;
}

/* Input styling */
input[type="text"],
input[type="email"],
input[type="password"],
input[type="number"],
input[type="date"],
select,
textarea {
    width: 100%;
    padding: 12px;
    border: 1px solid #ced4da;
    border-radius: 4px;
    font-size: 16px;
    margin-bottom: 15px;
    box-sizing: border-box;
}

/* Focus state untuk input */
input:focus,
select:focus,
textarea:focus {
    outline: none;
    border-color: #007bff;
    box-shadow: 0 0 0 2px rgba(0, 123, 255, 0.25);
}

/* Label styling */
label {
    display: block;
    margin-bottom: 5px;
    font-weight: 500;
    color: #495057;
}
```

```
/* Button styling */
button[type="submit"] {
    background-color: #007bff;
    color: white;
    padding: 12px 24px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    margin-right: 10px;
}

button[type="submit"]:hover {
    background-color: #0056b3;
}

button[type="reset"] {
    background-color: #6c757d;
    color: white;
    padding: 12px 24px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

/* Validation styling */
input:invalid {
    border-color: #dc3545;
}

input:valid {
    border-color: #28a745;
}
```

5. Styling Tabel untuk Data Database

```
/* Container tabel */
.table-container {
    background-color: #ffffff;
```

```
border-radius: 8px;
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
overflow: hidden;
margin-bottom: 30px;
}

/* Tabel styling */
.data-table {
    width: 100%;
    border-collapse: collapse;
    font-size: 14px;
}

.data-table thead {
    background-color: #343a40;
    color: white;
}

.data-table th,
.data-table td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #dee2e6;
}

.data-table th {
    font-weight: 600;
    text-transform: uppercase;
    font-size: 12px;
    letter-spacing: 0.5px;
}

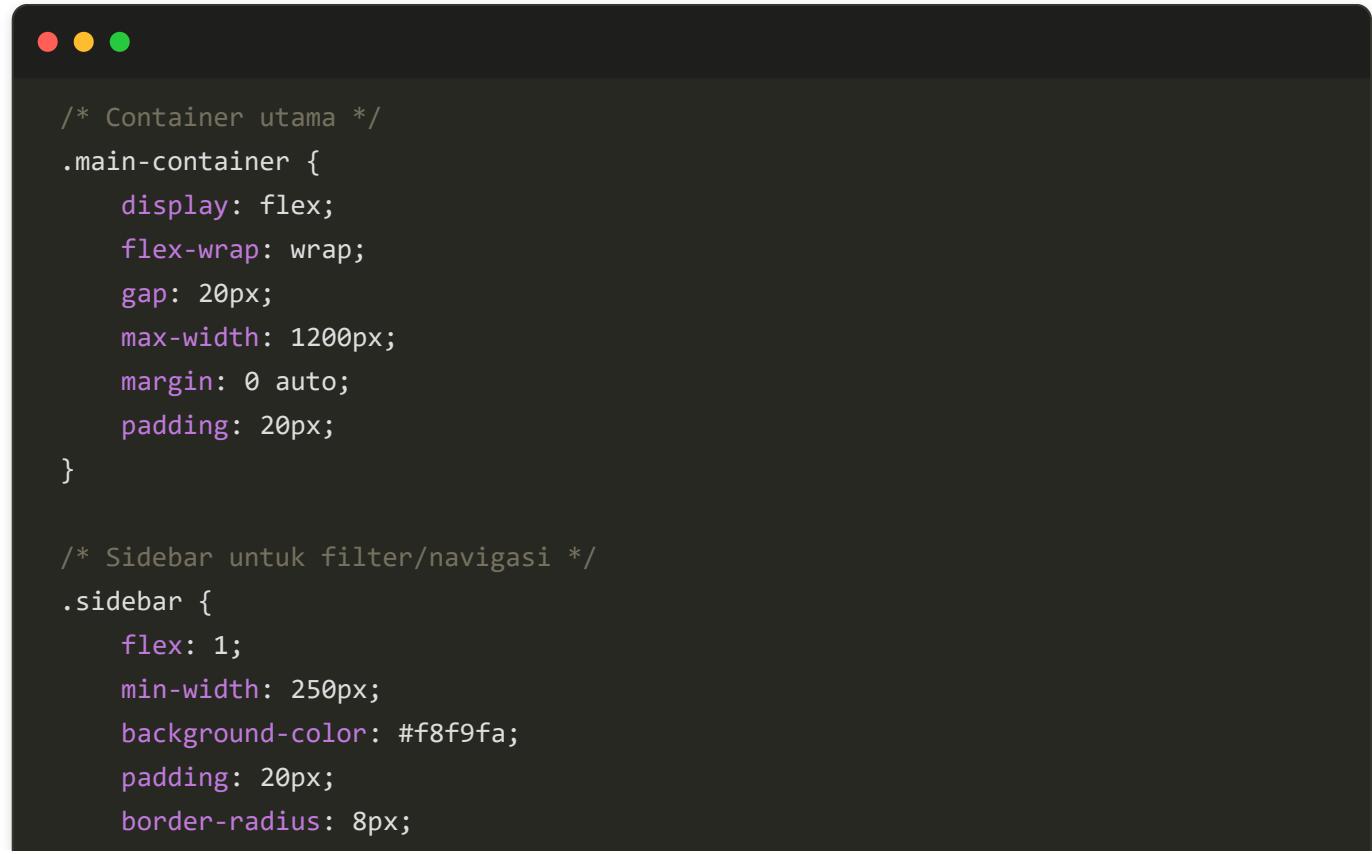
/* Zebra striping untuk tabel */
.data-table tbody tr:nth-child(even) {
    background-color: #f8f9fa;
}

.data-table tbody tr:hover {
    background-color: #e9ecef;
}

/* Action buttons dalam tabel */
.action-buttons a {
    display: inline-block;
```

```
padding: 4px 8px;  
margin: 0 2px;  
text-decoration: none;  
border-radius: 3px;  
font-size: 12px;  
}  
  
.btn-edit {  
background-color: #ffc107;  
color: #212529;  
}  
  
.btn-delete {  
background-color: #dc3545;  
color: white;  
}  
  
.btn-view {  
background-color: #17a2b8;  
color: white;  
}
```

6. Layout Responsif dengan Flexbox



```
/* Container utama */  
.main-container {  
display: flex;  
flex-wrap: wrap;  
gap: 20px;  
max-width: 1200px;  
margin: 0 auto;  
padding: 20px;  
}  
  
/* Sidebar untuk filter/navigasi */  
.sidebar {  
flex: 1;  
min-width: 250px;  
background-color: #f8f9fa;  
padding: 20px;  
border-radius: 8px;
```

```
}

/* Content area untuk formulir dan tabel */
.content-area {
    flex: 3;
    min-width: 300px;
}

/* Responsive design */
@media (max-width: 768px) {
    .main-container {
        flex-direction: column;
    }

    .sidebar {
        order: 2;
    }

    .content-area {
        order: 1;
    }

    .data-table {
        font-size: 12px;
    }

    .data-table th,
    .data-table td {
        padding: 8px 10px;
    }
}
```

Pembahasan

CSS adalah kunci untuk membuat aplikasi web database terlihat profesional dan mudah digunakan. Dengan memisahkan styling dari logika PHP dan struktur data MySQL, Anda dapat dengan mudah mengubah tampilan aplikasi tanpa mempengaruhi fungsionalitas backend. Styling yang konsisten untuk formulir dan tabel akan meningkatkan pengalaman pengguna dan membuat aplikasi terlihat lebih kredibel.

Contoh Kode

Berikut adalah contoh lengkap HTML dan CSS untuk sistem manajemen mahasiswa:

index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistem Manajemen Mahasiswa</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header class="main-header">
        <div class="container">
            <h1>Sistem Manajemen Mahasiswa</h1>
            <nav class="main-nav">
                <ul>
                    <li><a href="#form-section">Input Data</a></li>
                    <li><a href="#data-section">Lihat Data</a></li>
                    <li><a href="#statistics">Statistik</a></li>
                </ul>
            </nav>
        </div>
    </header>

    <main class="main-container">
        <aside class="sidebar">
            <h3>Filter Data</h3>
            <form class="filter-form">
                <label for="filter-jurusan">Jurusan:</label>
                <select id="filter-jurusan" name="filter_jurusan">
                    <option value="">Semua Jurusan</option>
                    <option value="ti">Teknik Informatika</option>
                    <option value="si">Sistem Informasi</option>
                    <option value="mi">Manajemen Informatika</option>
                </select>

                <label for="filter-semester">Semester:</label>
                <select id="filter-semester" name="filter_semester">
                    <option value="">Semua Semester</option>
                    <option value="1">Semester 1</option>
                    <option value="2">Semester 2</option>
                    <option value="3">Semester 3</option>
                    <option value="4">Semester 4</option>
                    <option value="5">Semester 5</option>
                </select>
            </form>
        </aside>
    </main>
</body>
```

```
</select>

    <button type="submit" class="btn-filter">Filter</button>
  </form>
</aside>

<div class="content-area">
  <section id="form-section" class="form-container">
    <h2>Form Input Data Mahasiswa</h2>
    <div id="success-message" style="display: none;">
      Data berhasil disimpan!
    </div>
    <div id="error-message" style="display: none;">
      Terjadi kesalahan saat menyimpan data!
    </div>

    <form action="proses_mahasiswa.php" method="POST" class="student-form">
      <fieldset>
        <legend>Data Pribadi</legend>

        <label for="nama">Nama Lengkap:</label>
        <input type="text" id="nama" name="nama" required
maxlength="100">

        <label for="nim">NIM:</label>
        <input type="text" id="nim" name="nim" required pattern="
[0-9]{10}"
          title="NIM harus 10 digit angka">

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>

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

        <div class="radio-group">
          <label>Jenis Kelamin:</label>
          <div class="radio-options">
            <input type="radio" id="laki" name="jenis_kelamin"
value="L" required>
            <label for="laki">Laki-laki</label>
            <input type="radio" id="perempuan"
value="P" required>
          </div>
        </div>
      </fieldset>
    </form>
  </section>
</div>
```

```
name="jenis_kelamin" value="P" required>
    <label for="perempuan">Perempuan</label>
</div>
</div>
</fieldset>

<fieldset>
    <legend>Data Akademik</legend>

    <label for="jurusan">Jurusan:</label>
    <select id="jurusan" name="jurusan" required>
        <option value="">Pilih Jurusan</option>
        <option value="ti">Teknik Informatika</option>
        <option value="si">Sistem Informasi</option>
        <option value="mi">Manajemen Informatika</option>
    </select>

    <label for="semester">Semester:</label>
    <input type="number" id="semester" name="semester" min="1"
max="8" required>
</fieldset>

<div class="form-actions">
    <button type="submit" name="submit">Simpan Data</button>
    <button type="reset">Reset Form</button>
</div>
</form>
</section>

<section id="data-section" class="table-container">
    <h2>Data Mahasiswa Terdaftar</h2>
    <table class="data-table">
        <thead>
            <tr>
                <th>No</th>
                <th>NIM</th>
                <th>Nama</th>
                <th>Email</th>
                <th>Jurusan</th>
                <th>Semester</th>
                <th>Status</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody>
```

```
<tbody>
    <tr>
        <td>1</td>
        <td>2023001001</td>
        <td>Ahmad Budi Santoso</td>
        <td>ahmad.budi@email.com</td>
        <td>Teknik Informatika</td>
        <td>5</td>
        <td><span class="status-active">Aktif</span></td>
        <td class="action-buttons">
            <a href="detail.php?id=1" class="btn-view">Detail</a>
            <a href="edit.php?id=1" class="btn-edit">Edit</a>
            <a href="delete.php?id=1" class="btn-delete" onclick="return confirm('Yakin hapus?')">Hapus</a>
        </td>
    </tr>
    <tr>
        <td>2</td>
        <td>2023001002</td>
        <td>Siti Aminah</td>
        <td>siti.aminah@email.com</td>
        <td>Sistem Informasi</td>
        <td>3</td>
        <td><span class="status-active">Aktif</span></td>
        <td class="action-buttons">
            <a href="detail.php?id=2" class="btn-view">Detail</a>
            <a href="edit.php?id=2" class="btn-edit">Edit</a>
            <a href="delete.php?id=2" class="btn-delete" onclick="return confirm('Yakin hapus?')">Hapus</a>
        </td>
    </tr>
</tbody>
</table>
</section>
</div>
</main>

<footer class="main-footer">
    <div class="container">
        <p>&copy; 2025 Sistem Manajemen Mahasiswa. Dikembangkan dengan HTML,

```

```
CSS, JavaScript, PHP, dan MySQL.</p>
    </div>
</footer>
</body>
</html>
```

style.css

```
/* Reset and base styling */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f8f9fa;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

/* Header styling */
.main-header {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    padding: 1rem 0;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.main-header h1 {
    margin-bottom: 10px;
    font-size: 2rem;
}
```

```
.main-nav ul {
    list-style: none;
    display: flex;
    gap: 30px;
}

.main-nav a {
    color: white;
    text-decoration: none;
    font-weight: 500;
    transition: opacity 0.3s;
}

.main-nav a:hover {
    opacity: 0.8;
}

/* Main container */
.main-container {
    display: flex;
    gap: 30px;
    max-width: 1200px;
    margin: 30px auto;
    padding: 0 20px;
}

/* Sidebar */
.sidebar {
    flex: 1;
    min-width: 250px;
    background-color: white;
    padding: 25px;
    border-radius: 10px;
    box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);
    height: fit-content;
}

.sidebar h3 {
    margin-bottom: 20px;
    color: #495057;
    border-bottom: 2px solid #e9ecf;
    padding-bottom: 10px;
}
```

```
.filter-form label {  
    display: block;  
    margin-bottom: 5px;  
    font-weight: 500;  
    color: #495057;  
}  
  
.filter-form select {  
    width: 100%;  
    padding: 10px;  
    border: 1px solid #ced4da;  
    border-radius: 5px;  
    margin-bottom: 15px;  
    font-size: 14px;  
}  
  
.btn-filter {  
    width: 100%;  
    background-color: #28a745;  
    color: white;  
    padding: 10px;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
    font-size: 14px;  
    font-weight: 500;  
}  
  
.btn-filter:hover {  
    background-color: #218838;  
}  
  
/* Content area */  
.content-area {  
    flex: 3;  
    min-width: 300px;  
}  
  
/* Form container */  
.form-container {  
    background-color: white;  
    padding: 30px;  
    border-radius: 10px;  
    box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);
```

```
    margin-bottom: 30px;
}

.form-container h2 {
    margin-bottom: 25px;
    color: #495057;
    border-bottom: 3px solid #007bff;
    padding-bottom: 10px;
}

/* Fieldset styling */
fieldset {
    border: 2px solid #e9ecef;
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 25px;
}

legend {
    font-weight: 600;
    color: #495057;
    padding: 0 15px;
    font-size: 1.1rem;
}

/* Form inputs */
input[type="text"],
input[type="email"],
input[type="password"],
input[type="number"],
input[type="date"],
select,
textarea {
    width: 100%;
    padding: 12px 15px;
    border: 2px solid #e9ecef;
    border-radius: 6px;
    font-size: 16px;
    margin-bottom: 20px;
    transition: border-color 0.3s, box-shadow 0.3s;
}

input:focus,
select:focus,
```

```
textarea:focus {
    outline: none;
    border-color: #007bff;
    box-shadow: 0 0 0 3px rgba(0, 123, 255, 0.1);
}

/* Labels */
label {
    display: block;
    margin-bottom: 8px;
    font-weight: 500;
    color: #495057;
}

/* Radio group */
.radio-group {
    margin-bottom: 20px;
}

.radio-options {
    display: flex;
    gap: 20px;
    margin-top: 10px;
}

.radio-options input[type="radio"] {
    width: auto;
    margin-right: 8px;
    margin-bottom: 0;
}

.radio-options label {
    margin-bottom: 0;
    cursor: pointer;
}

/* Form actions */
.form-actions {
    display: flex;
    gap: 15px;
    margin-top: 30px;
}

button[type="submit"] {
```

```
background: linear-gradient(135deg, #007bff 0%, #0056b3 100%);  
color: white;  
padding: 12px 30px;  
border: none;  
border-radius: 6px;  
cursor: pointer;  
font-size: 16px;  
font-weight: 500;  
transition: transform 0.2s, box-shadow 0.2s;  
}  
  
button[type="submit"]:hover {  
    transform: translateY(-2px);  
    box-shadow: 0 4px 15px rgba(0, 123, 255, 0.3);  
}  
  
button[type="reset"] {  
    background-color: #6c757d;  
    color: white;  
    padding: 12px 30px;  
    border: none;  
    border-radius: 6px;  
    cursor: pointer;  
    font-size: 16px;  
    font-weight: 500;  
    transition: background-color 0.3s;  
}  
  
button[type="reset"]:hover {  
    background-color: #545b62;  
}  
  
/* Messages */  
#success-message {  
    background-color: #d4edda;  
    color: #155724;  
    padding: 15px;  
    border-radius: 6px;  
    border-left: 4px solid #28a745;  
    margin-bottom: 20px;  
}  
  
#error-message {  
    background-color: #f8d7da;
```

```
color: #721c24;
padding: 15px;
border-radius: 6px;
border-left: 4px solid #dc3545;
margin-bottom: 20px;
}

/* Table container */
.table-container {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);
    overflow: hidden;
}

.table-container h2 {
    padding: 25px 30px 0;
    margin-bottom: 20px;
    color: #495057;
    border-bottom: 3px solid #007bff;
    padding-bottom: 10px;
    margin-left: 30px;
    margin-right: 30px;
}

/* Data table */
.data-table {
    width: 100%;
    border-collapse: collapse;
    font-size: 14px;
}

.data-table thead {
    background: linear-gradient(135deg, #343a40 0%, #495057 100%);
    color: white;
}

.data-table th {
    padding: 15px 20px;
    text-align: left;
    font-weight: 600;
    text-transform: uppercase;
    font-size: 12px;
    letter-spacing: 1px;
```

```
}

.data-table td {
    padding: 15px 20px;
    border-bottom: 1px solid #e9ecef;
}

.data-table tbody tr:nth-child(even) {
    background-color: #f8f9fa;
}

.data-table tbody tr:hover {
    background-color: #e3f2fd;
    transition: background-color 0.3s;
}

/* Status styling */
.status-active {
    background-color: #28a745;
    color: white;
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 12px;
    font-weight: 500;
}

.status-inactive {
    background-color: #dc3545;
    color: white;
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 12px;
    font-weight: 500;
}

/* Action buttons */
.action-buttons {
    display: flex;
    gap: 8px;
}

.action-buttons a {
    padding: 6px 12px;
    text-decoration: none;
```

```
border-radius: 4px;
font-size: 12px;
font-weight: 500;
transition: transform 0.2s, box-shadow 0.2s;
}

.action-buttons a:hover {
    transform: translateY(-1px);
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
}

.btn-view {
    background-color: #17a2b8;
    color: white;
}

.btn-edit {
    background-color: #ffc107;
    color: #212529;
}

.btn-delete {
    background-color: #dc3545;
    color: white;
}

/* Footer */
.main-footer {
    background-color: #343a40;
    color: white;
    text-align: center;
    padding: 20px 0;
    margin-top: 50px;
}

/* Responsive design */
@media (max-width: 768px) {
    .main-container {
        flex-direction: column;
        gap: 20px;
    }

    .sidebar {
        order: 2;
    }
}
```

```
    min-width: auto;
}

.content-area {
    order: 1;
}

.main-nav ul {
    flex-direction: column;
    gap: 10px;
}

.data-table {
    font-size: 12px;
}

.data-table th,
.data-table td {
    padding: 10px 12px;
}

.action-buttons {
    flex-direction: column;
    gap: 4px;
}

.form-actions {
    flex-direction: column;
}

.radio-options {
    flex-direction: column;
    gap: 10px;
}

}

@media (max-width: 480px) {
    .container,
    .main-container {
        padding: 0 15px;
    }

    .form-container,
    .table-container {
```

```
padding: 20px 15px;  
}  
  
.main-header h1 {  
    font-size: 1.5rem;  
}  
}
```

Langkah-langkah Praktikum

1. Lanjutkan dari folder `praktikum_html_database` yang Anda buat di Pertemuan 1.
2. Di dalam folder tersebut, 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.
6. Sesuaikan struktur HTML Anda agar sesuai dengan contoh di atas (tambahkan class dan id yang diperlukan).
7. Simpan kedua file.
8. Buka `index.html` di browser Anda dan amati perubahan tampilannya.
9. Eksplorasi:
 - Ubah warna tema utama aplikasi dengan mengubah gradient di header.
 - Eksperimen dengan box-shadow untuk memberikan efek depth yang berbeda.
 - Coba ubah layout menggunakan CSS Grid sebagai alternatif Flexbox.
 - Tambahkan animasi CSS untuk transisi yang lebih smooth.

Penugasan

Perbaiki tampilan sistem manajemen perpustakaan dari Penugasan Pertemuan 1 menggunakan CSS:

1. Buat file `style.css` terpisah dan tautkan ke `index.html` Anda.
2. Implementasikan desain yang konsisten dengan sistem manajemen database:
 - Header dengan gradient background dan navigasi yang jelas.
 - Formulir input yang rapi dengan fieldset dan styling yang profesional.
 - Tabel data yang mudah dibaca dengan zebra striping dan hover effects.
 - Sidebar untuk filter atau navigasi tambahan.
3. Gunakan Flexbox atau Grid untuk layout yang responsif.
4. Tambahkan styling untuk status buku (Tersedia/Dipinjam) dengan warna yang berbeda.
5. Implementasikan responsive design dengan media queries untuk mobile.
6. Berikan styling khusus untuk action buttons (Edit/Hapus) dalam tabel.

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:

- **Inline 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 = "Budi";
const umur = 25;
var pekerjaan = "Mahasiswa";

console.log(nama); // Output: Budi
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 (F12 atau klik kanan -> Inspect -> Console) untuk melihat output dari `console.log()`.
7. Eksplorasi:
 - o Ubah pesan `alert` dan `prompt`.
 - o Tambahkan variabel baru dengan tipe data berbeda (misalnya, boolean).
 - o Buat kondisi `if-else if-else` yang lebih kompleks berdasarkan input pengguna.
 - o 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: "Budi", 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("Budi", "Santoso"));
```

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: "Budi", 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)

The screenshot shows a code editor window with two tabs: 'index.html' and 'script.js'. The 'index.html' tab is active, displaying the following HTML code:

```
<!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>
```

The 'script.js' tab is visible at the bottom of the editor window.

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 propertiannya.
 - Gunakan `for...in` loop untuk mengulang properti objek `kalkulator`.

Penugasan

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

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

Contoh:

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

- Buat fungsi bernama `tambahTugas(deskripsiTugas)` yang menambahkan tugas baru ke `daftarTugas` dengan `selesai: false` secara default.
- Buat fungsi bernama `tandaiSelesai(indexTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
- Buat fungsi bernama `cariTugas(keyword)` yang mengembalikan array tugas yang deskripsinya mengandung `keyword`.
- Simulasikan "menyimpan" dan "mengambil" data dari backend dengan menggunakan `JSON.stringify()` saat "menyimpan" `daftarTugas` dan `JSON.parse()` saat "mengambil"nya. Tampilkan string JSON yang dihasilkan ke konsol.
- Tampilkan hasil pencarian tugas ke konsol.

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:

- Memahami apa itu DOM dan perannya dalam pengembangan web interaktif.
- Mengakses elemen HTML menggunakan berbagai metode seleksi DOM.
- Memanipulasi konten teks dan atribut elemen HTML.
- Mengubah gaya (CSS) elemen secara dinamis menggunakan JavaScript.
- 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");
semuanya.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
```

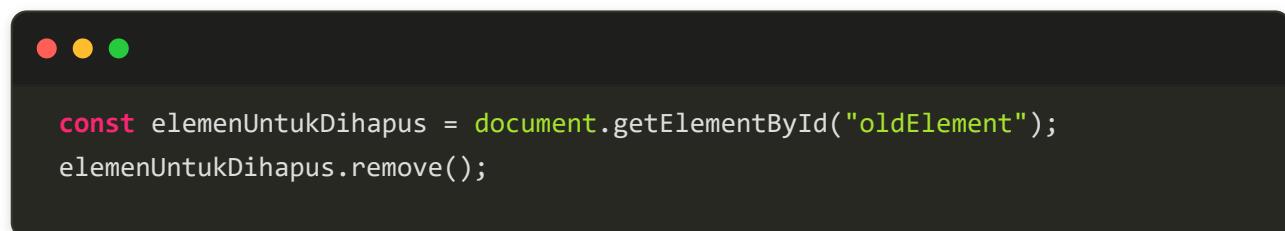
- **Menambahkan Elemen ke DOM:**

- `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();
```

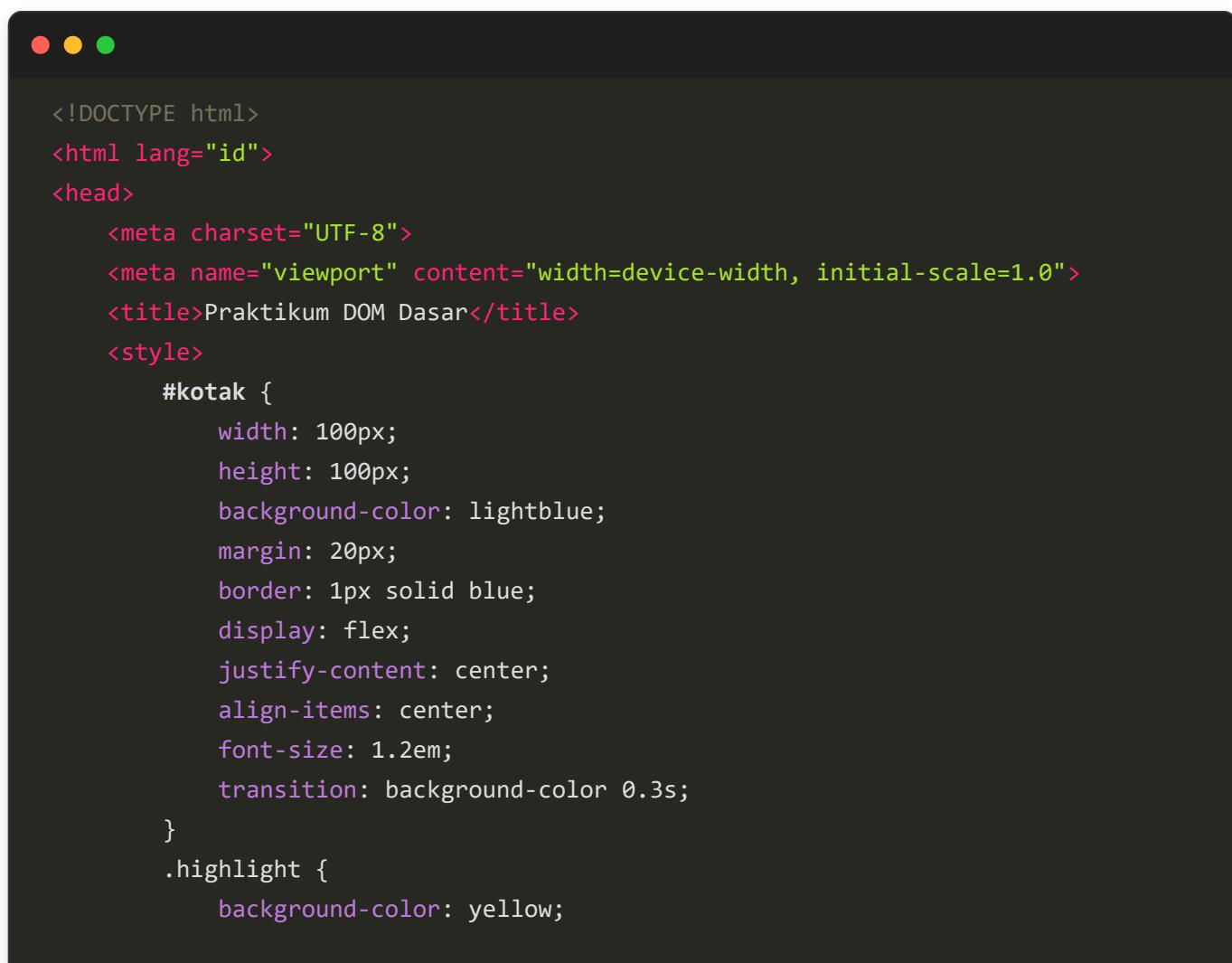
Pembahasan

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

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:** `click`, `dblclick`, `mouseover`, `mouseout`, `mousedown`, `mouseup`, `keydown`, `keyup`, `keypress`, `submit`, `change`, `focus`, `blur`, `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.

- **Pentingnya untuk Backend:** Mengurangi jumlah permintaan yang tidak valid yang mencapai server, sehingga menghemat sumber daya server dan mengurangi beban database.
- **Teknik Validasi Lanjutan:**
 - **Pola Regex:** Menggunakan ekspresi reguler untuk memvalidasi format input (misalnya, email, nomor telepon).
 - **Pengecekan Panjang:** Memastikan input memiliki panjang minimum atau maksimum.
 - **Pengecekan Angka:** Memastikan input adalah angka dan berada dalam rentang tertentu.
 - **Pengecekan Kecocokan:** Memastikan dua input cocok (misalnya, password dan konfirmasi password).
- **Umpan Balik Pengguna:** Selain mengubah border input, tampilkan pesan error yang spesifik di dekat setiap input yang bermasalah.

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 peristiwi:

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.
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 program JavaScript yang mengelola daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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 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:

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



```
<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)

```
❶ <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).

```
❶ let nama = "Budi";
❷ const umur = 25;
❸ var pekerjaan = "Mahasiswa";

❹ console.log(nama); // Output: Budi
❺ 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 (F12 atau klik kanan -> Inspect -> Console) untuk melihat output dari `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: "Budi", 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("Budi", "Santoso"));
```

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: "Budi", 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)

```
<!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 propertiannya.
 - Gunakan `for...in` loop untuk mengulang properti objek `kalkulator`.

Penugasan

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

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

Contoh:

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

- Buat fungsi bernama `tambahTugas(deskripsiTugas)` yang menambahkan tugas baru ke `daftarTugas` dengan `selesai: false` secara default.
- Buat fungsi bernama `tandaiSelesai(indexTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
- Buat fungsi bernama `cariTugas(keyword)` yang mengembalikan array tugas yang deskripsinya mengandung `keyword`.
- Simulasikan "menyimpan" dan "mengambil" data dari backend dengan menggunakan `JSON.stringify()` saat "menyimpan" `daftarTugas` dan `JSON.parse()` saat "mengambil"nya. Tampilkan string JSON yang dihasilkan ke konsol.
- Tampilkan hasil pencarian tugas ke konsol.

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:

- Memahami apa itu DOM dan perannya dalam pengembangan web interaktif.
- Mengakses elemen HTML menggunakan berbagai metode seleksi DOM.
- Memanipulasi konten teks dan atribut elemen HTML.
- Mengubah gaya (CSS) elemen secara dinamis menggunakan JavaScript.
- 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");
semuanya.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();
```

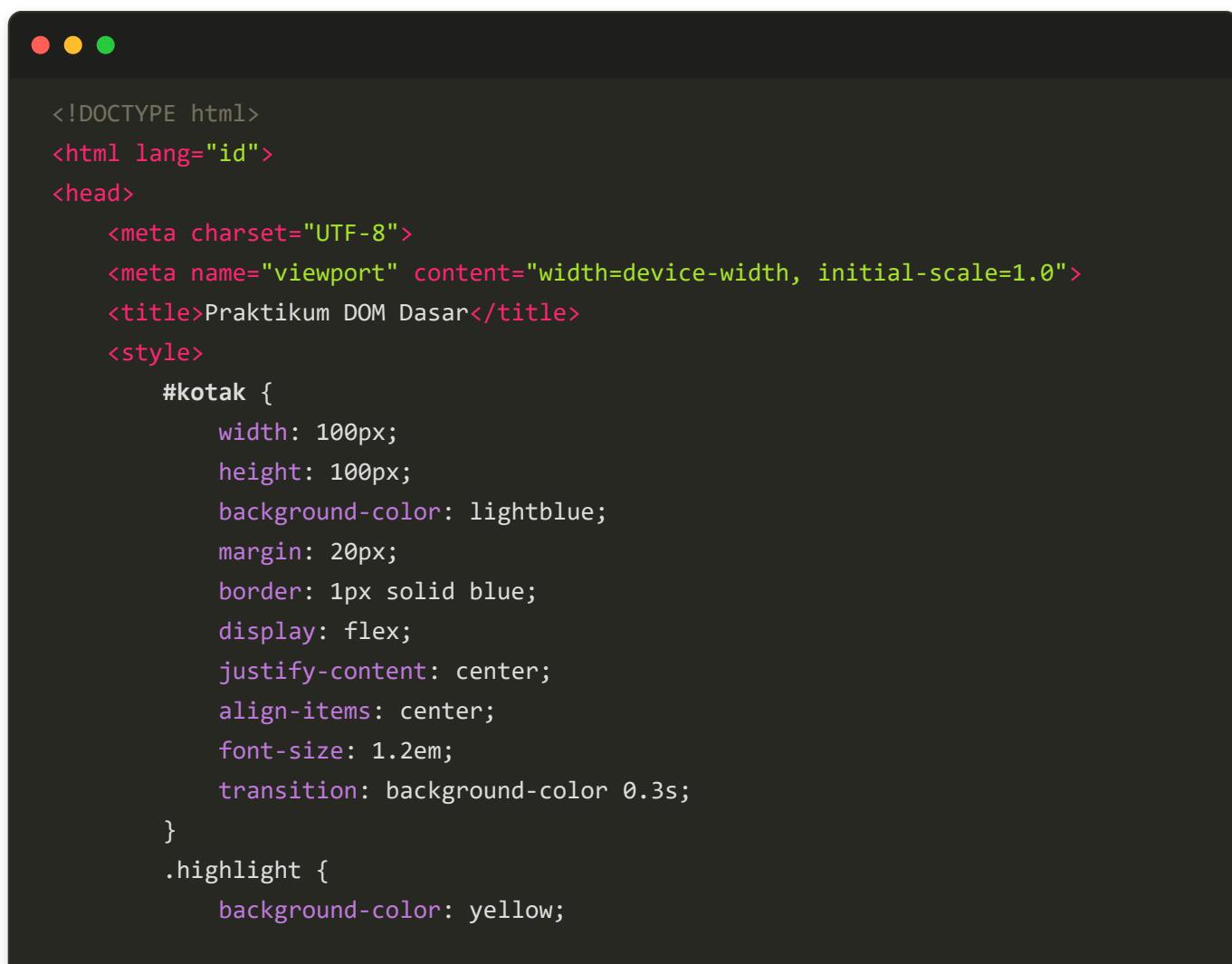
Pembahasan

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

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 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).

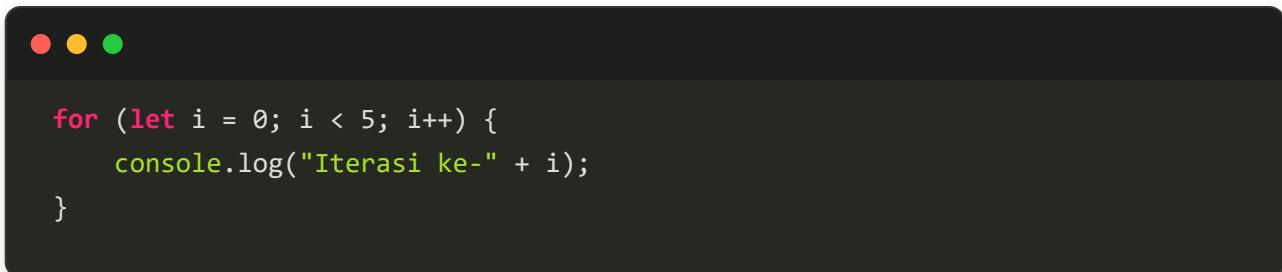
Materi

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: "Budi", 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("Budi", "Santoso"));
```

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: "Budi", 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)}`);

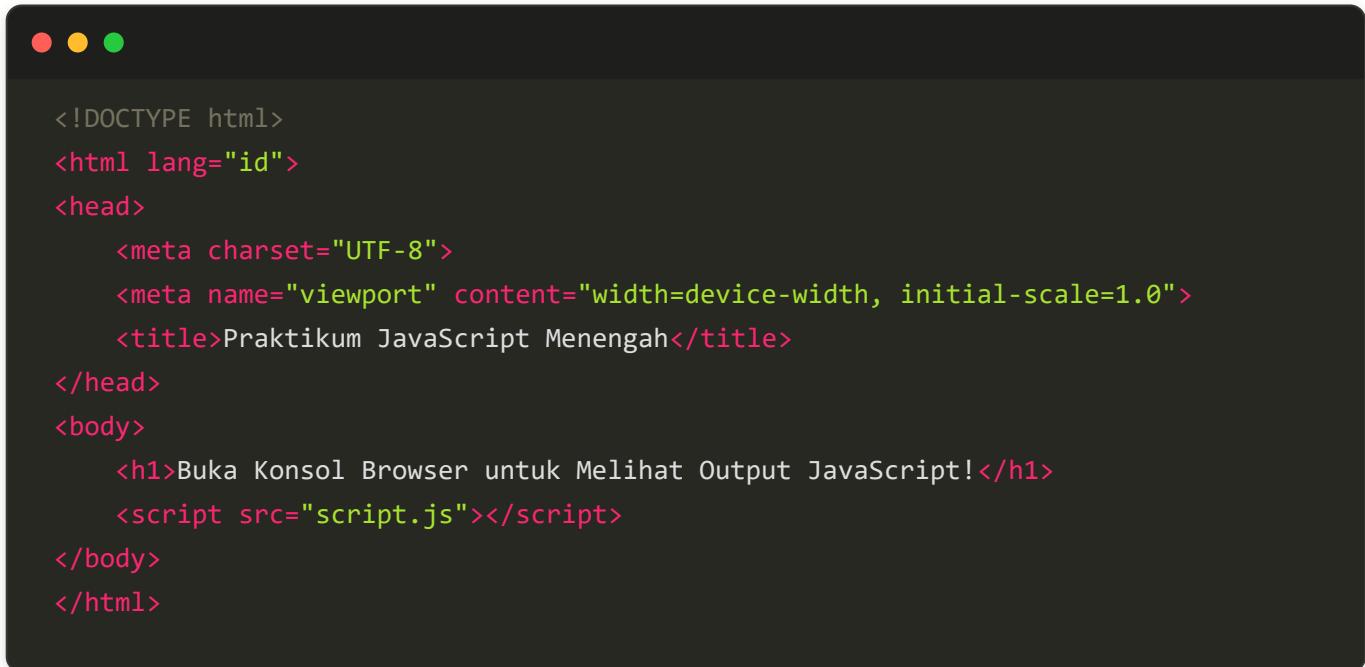
// 3. Contoh Fungsi Panah dan Filter Array
const mahasiswaLulus = daftarMahasiswa.filter(mhs => mhs.nilai >= 75);
console.log("\nMahasiswa yang Lulus (nilai >= 75):");
mahasiswaLulus.forEach(mhs => {
    console.log(` ${mhs.nama} (${mhs.nilai})`);
});

// 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)



```
<!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 propertiannya.
 - Gunakan `for...in` loop untuk mengulang properti objek `kalkulator`.

Penugasan

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

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

Contoh:

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

- Buat fungsi bernama `tambahTugas(deskripsiTugas)` yang menambahkan tugas baru ke `daftarTugas` dengan `selesai: false` secara default.
- Buat fungsi bernama `tandaiSelesai(indexTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
- Buat fungsi bernama `cariTugas(keyword)` yang mengembalikan array tugas yang deskripsinya mengandung `keyword`.
- Simulasikan "menyimpan" dan "mengambil" data dari backend dengan menggunakan `JSON.stringify()` saat "menyimpan" `daftarTugas` dan `JSON.parse()` saat "mengambil"nya. Tampilkan string JSON yang dihasilkan ke konsol.
- Tampilkan hasil pencarian tugas ke konsol.

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:

- Memahami apa itu DOM dan perannya dalam pengembangan web interaktif.
- Mengakses elemen HTML menggunakan berbagai metode seleksi DOM.
- Memanipulasi konten teks dan atribut elemen HTML.
- Mengubah gaya (CSS) elemen secara dinamis menggunakan JavaScript.
- 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");
semuanya.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();
```

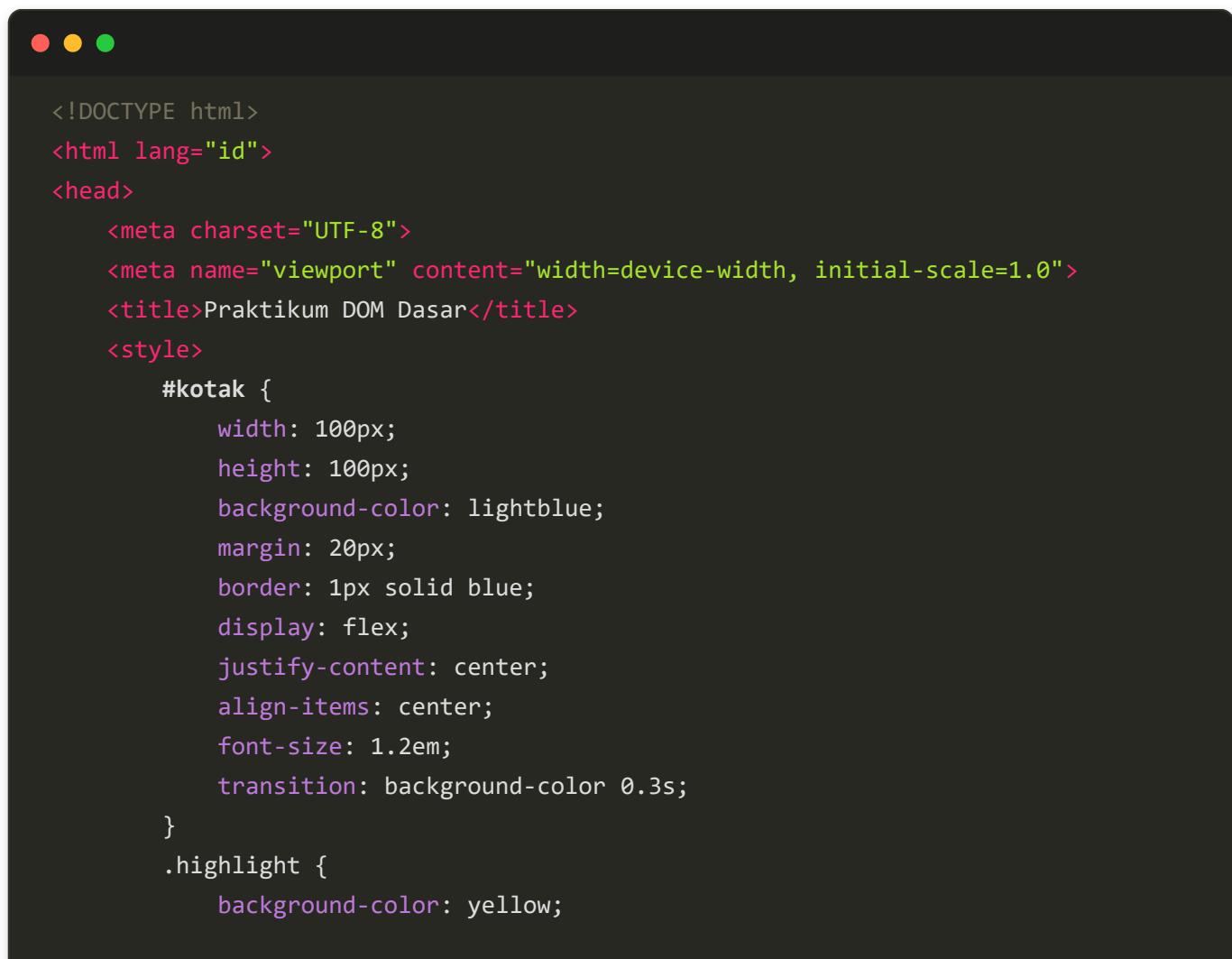
Pembahasan

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



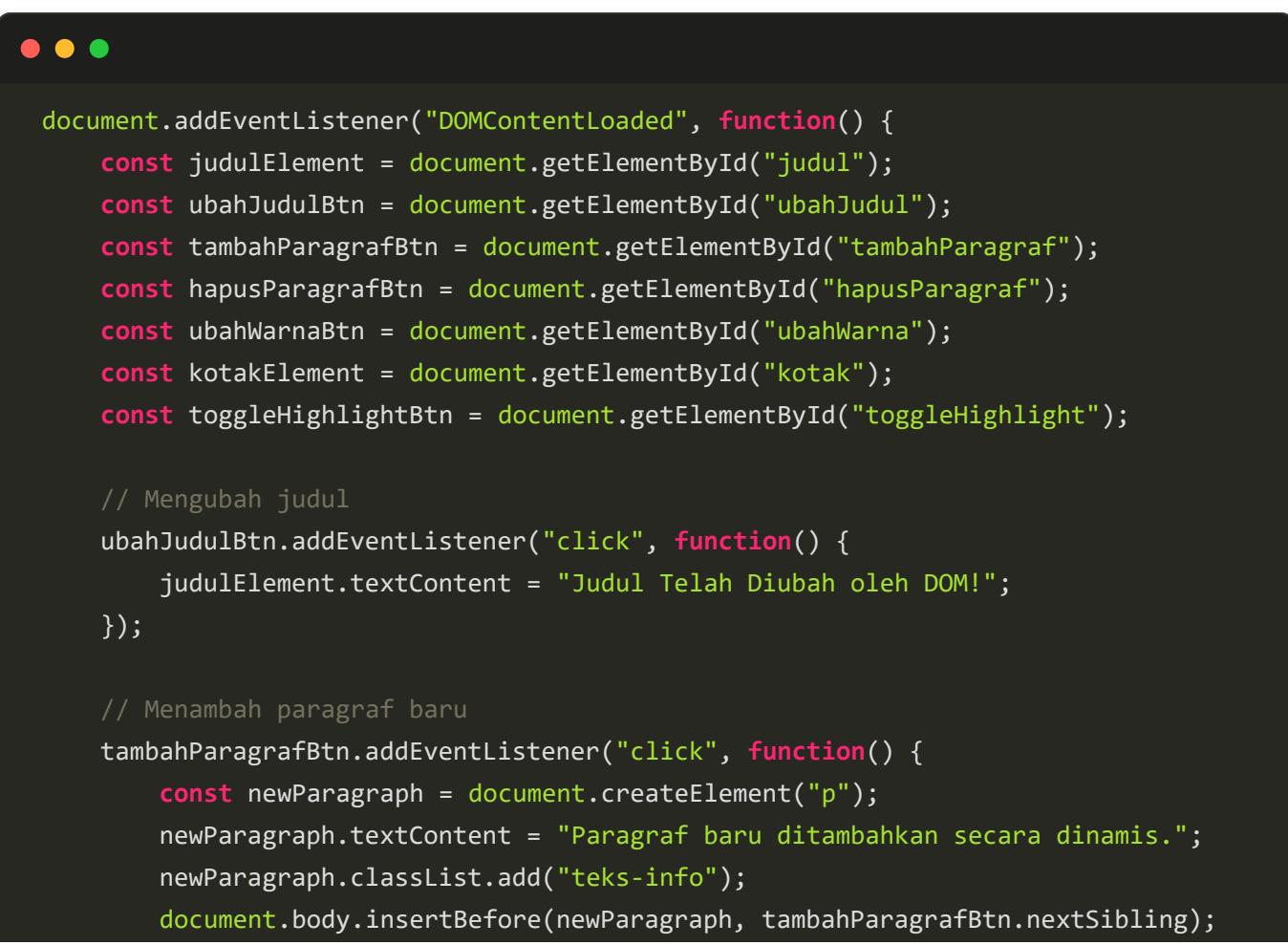
```
<!DOCTYPE html>
<html lang="id">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Praktikum DOM Dasar</title>
<style>
#kotak {
    width: 100px;
    height: 100px;
    background-color: lightblue;
    margin: 20px;
    border: 1px solid blue;
    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 yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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 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()`.

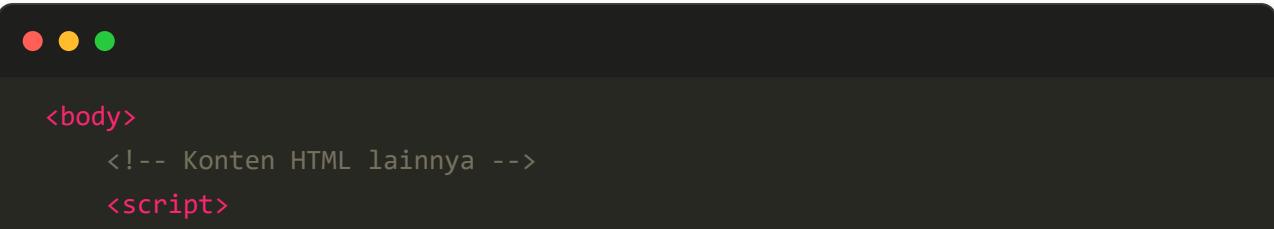
Materi

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:

- **Inline 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 = "Budi";
const umur = 25;
var pekerjaan = "Mahasiswa";

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

3. Tipe Data

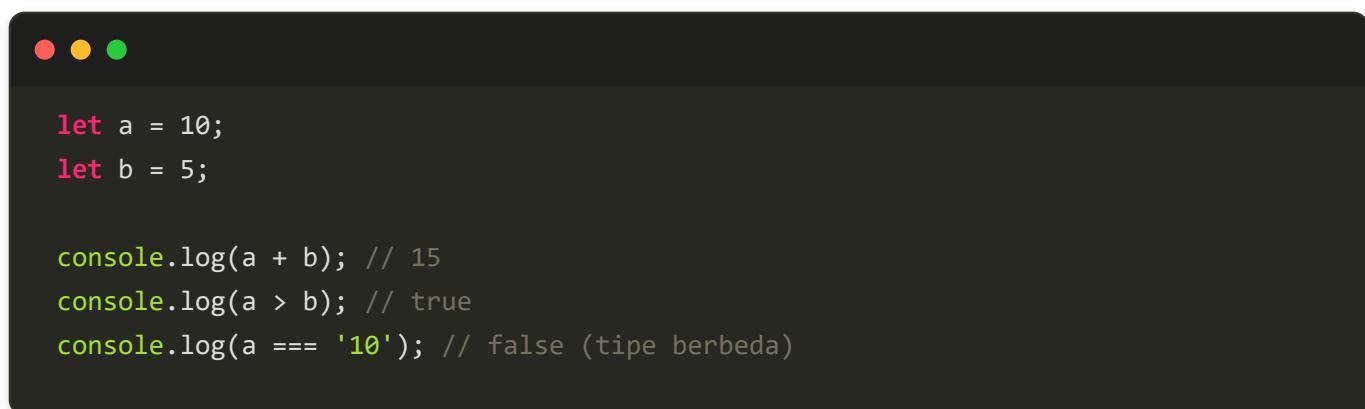
JavaScript memiliki beberapa tipe data dasar:

- **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 (F12 atau klik kanan -> Inspect -> Console) untuk melihat output dari `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: "Budi", 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("Budi", "Santoso"));
```

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: "Budi", 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) {
            console.log("Error: Peny犁i tidak boleh nol!");
            return null;
        }
        return a / b;
    }
};
```

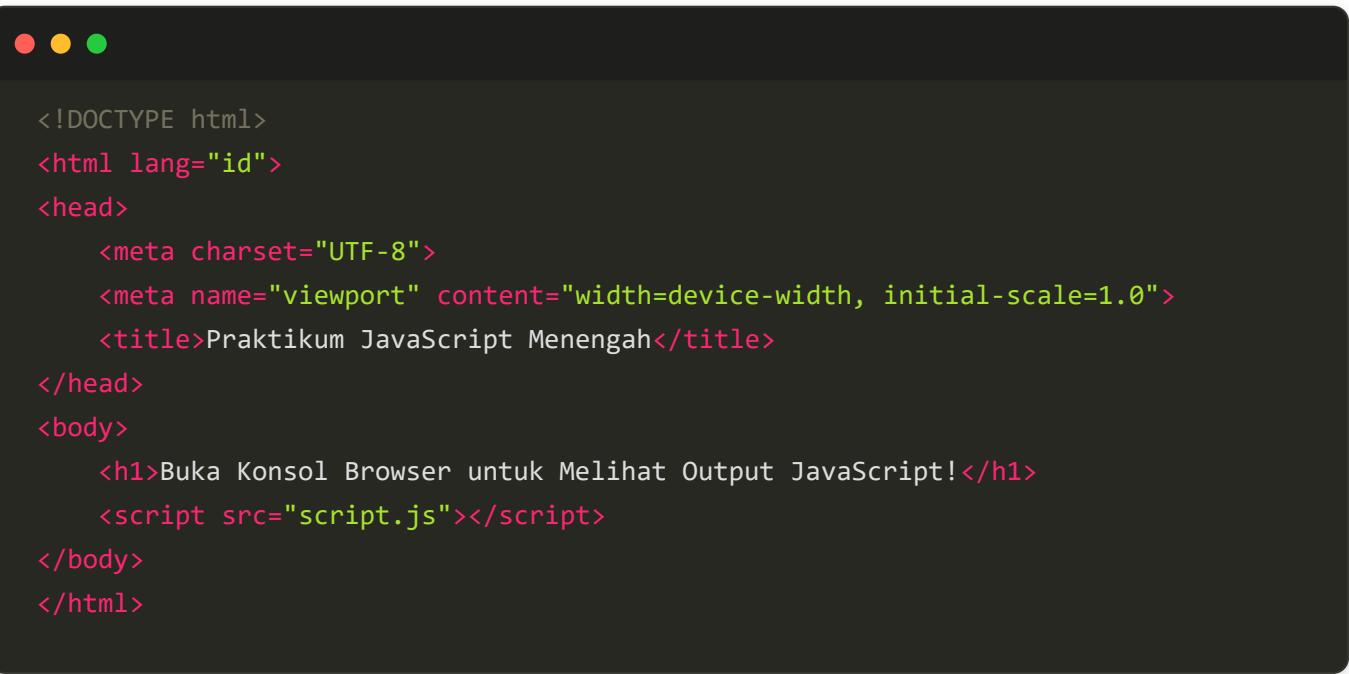
```

        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)



```

<!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:

```
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(indexTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
4. Buat fungsi bernama `cariTugas(keyword)` yang mengembalikan array tugas yang deskripsinya mengandung `keyword`.
5. Simulasikan "penyimpanan" dan "pengambilan" data dari backend dengan menggunakan `JSON.stringify()` saat "menyimpan" `daftarTugas` dan `JSON.parse()` saat "mengambil"nya. Tampilkan string JSON yang dihasilkan ke konsol.
6. Tampilkan hasil pencarian tugas ke konsol.

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");
semauLink.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 yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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 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");
semauaLink.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()`

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?

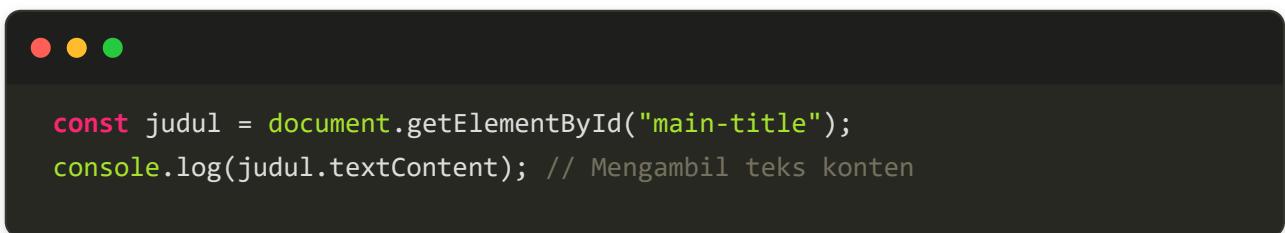
Bayangkan halaman HTML Anda sebagai sebuah pohon keluarga. Setiap tag HTML (seperti `<html>`, `<body>`, `<p>`, `<div>`) adalah anggota keluarga (node). Hubungan antara tag-tag ini (induk, anak, saudara) membentuk struktur pohon. DOM adalah cara JavaScript untuk "berbicara" dengan pohon ini, memungkinkan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

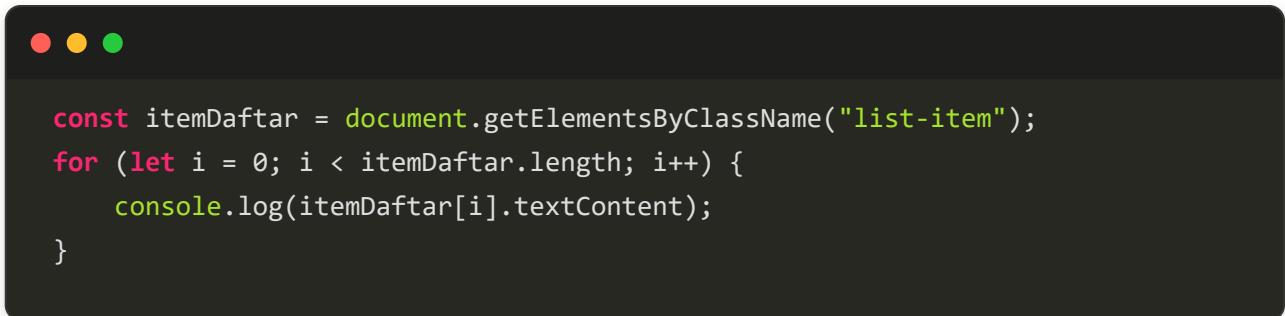
Ada beberapa metode untuk memilih elemen dari DOM:

- `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 yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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:** `click`, `dblclick`, `mouseover`, `mouseout`, `mousedown`, `mouseup`, `keydown`, `keyup`, `keypress`, `submit`, `change`, `focus`, `blur`, `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.

- **Pentingnya untuk Backend:** Mengurangi jumlah permintaan yang tidak valid yang mencapai server, sehingga menghemat sumber daya server dan mengurangi beban database.
- **Teknik Validasi Lanjutan:**
 - **Pola Regex:** Menggunakan ekspresi reguler untuk memvalidasi format input (misalnya, email, nomor telepon).
 - **Pengecekan Panjang:** Memastikan input memiliki panjang minimum atau maksimum.
 - **Pengecekan Angka:** Memastikan input adalah angka dan berada dalam rentang tertentu.

- **Pengecekan Kecocokan:** Memastikan dua input cocok (misalnya, password dan konfirmasi password).
- **Umpam Balik Pengguna:** Selain mengubah border input, tampilkan pesan error yang spesifik di dekat setiap input yang bermasalah.

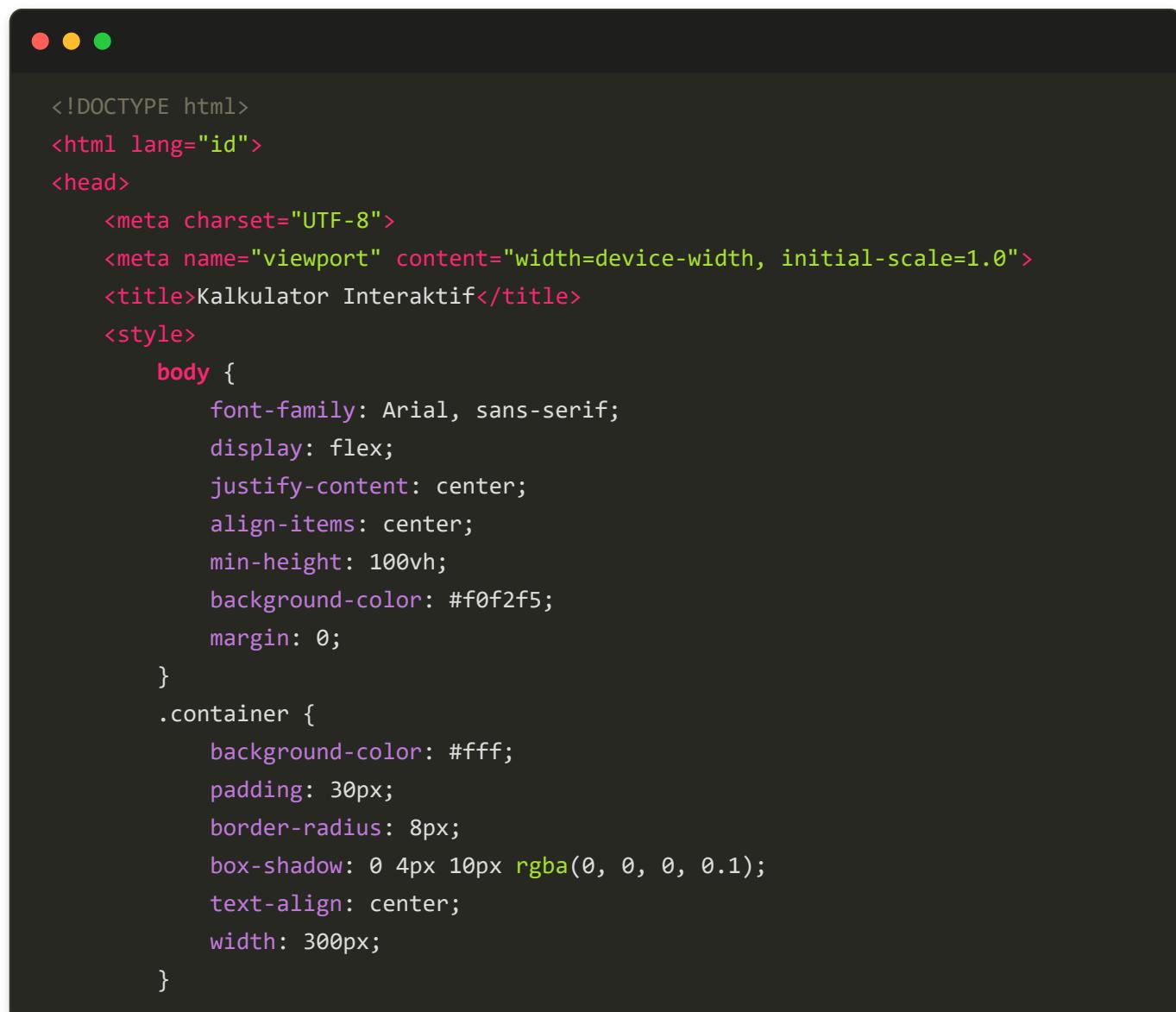
Pembahasan

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;
        }
    </style>
</head>
<body>
    <div class="container">
        <input type="text" id="input1" placeholder="Masukkan Angka 1" required>
        <input type="text" id="input2" placeholder="Masukkan Angka 2" required>
        <button id="button">Hitung</button>
        <div id="hasil"></div>
    </div>
</body>
</html>
```

```
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.
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 aplikasi To-Do List sederhana dengan fitur-fitur berikut:

1. **Form Input:** Sebuah input field untuk memasukkan tugas baru dan tombol "Tambah".
2. **Daftar Tugas:** Area yang menampilkan semua tugas yang telah ditambahkan.
3. **Validasi:** Pastikan input tidak kosong sebelum menambahkan tugas.
4. **Fitur Hapus:** Setiap tugas memiliki tombol "Hapus" untuk menghapusnya dari daftar.
5. **Fitur Selesai:** Setiap tugas memiliki checkbox untuk menandai tugas sebagai selesai (dengan gaya visual yang berbeda).
6. **Counter:** Tampilkan jumlah total tugas dan jumlah tugas yang sudah selesai.
7. **LocalStorage:** Simpan daftar tugas di browser menggunakan `localStorage` sehingga data tidak hilang saat halaman di-refresh.

Persyaratan teknis:

- Gunakan event handling untuk semua interaksi pengguna.
- Implementasikan validasi form yang tepat.
- Gunakan manipulasi DOM untuk menambah/menghapus elemen.
- Berikan styling CSS yang menarik dan responsif.

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

Kesimpulan

Modul ajar ini telah memperkenalkan mahasiswa pada fondasi pengembangan web modern melalui HTML, CSS, dan JavaScript. Setiap pertemuan dirancang untuk membangun pemahaman yang progresif, mulai dari struktur dasar HTML hingga manipulasi DOM yang kompleks.

Ringkasan Pembelajaran

Pertemuan 1-2 membangun fondasi dengan HTML untuk struktur konten dan CSS untuk presentasi visual. Mahasiswa belajar pentingnya pemisahan konten dari presentasi, yang merupakan prinsip fundamental dalam pengembangan web.

Pertemuan 3-4 memperkenalkan JavaScript sebagai bahasa pemrograman untuk menambahkan interaktivitas. Dari konsep dasar seperti variabel dan tipe data, hingga struktur data kompleks seperti array dan objek yang esensial untuk komunikasi dengan backend.

Pertemuan 5-6 mengintegrasikan semua pengetahuan melalui manipulasi DOM dan event handling. Mahasiswa belajar bagaimana JavaScript dapat mengubah halaman web secara dinamis dan merespons tindakan pengguna.

Persiapan untuk PHP dan MySQL

Modul ini secara khusus dirancang sebagai persiapan untuk mempelajari PHP dan MySQL. Mahasiswa telah dibekali dengan:

- **Pemahaman struktur data** melalui HTML forms dan JavaScript objects
- **Konsep validasi sisi klien** yang akan melengkapi validasi sisi server

- **Manipulasi DOM** untuk menampilkan data dari database secara dinamis
- **Event handling** untuk interaksi pengguna dengan aplikasi web

Langkah Selanjutnya

Dengan menguasai materi dalam modul ini, mahasiswa siap untuk:

1. **Mempelajari PHP** untuk pemrograman sisi server
2. **Menggunakan MySQL** untuk penyimpanan dan pengelolaan data
3. **Membangun aplikasi web full-stack** yang mengintegrasikan frontend dan backend
4. **Mengimplementasikan pola MVC** dalam pengembangan aplikasi web

Modul ini memberikan dasar yang kuat untuk perjalanan mahasiswa dalam pengembangan web modern, memastikan mereka memiliki pemahaman yang solid tentang teknologi frontend sebelum melangkah ke pengembangan backend dengan PHP dan MySQL.

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).

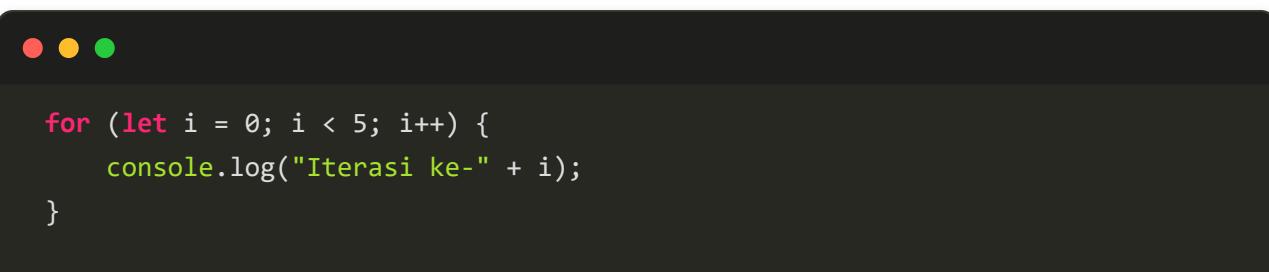
Materi

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);  
}
```

The screenshot shows a dark-themed code editor window. At the top left are three small colored circles (red, yellow, green). The main area contains a single line of JavaScript code: "for (let i = 0; i < 5; i++) { console.log("Iterasi ke-" + i); }". The code is written in a light-colored font against the dark background.

- **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: "Budi", 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("Budi", "Santoso"));
```

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: "Budi", 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)



```
<!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 propertiannya.
 - Gunakan `for...in` loop untuk mengulang properti objek `kalkulator`.

Penugasan

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

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

Contoh:

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

- Buat fungsi bernama `tambahTugas(deskripsiTugas)` yang menambahkan tugas baru ke `daftarTugas` dengan `selesai: false` secara default.
- Buat fungsi bernama `tandaiSelesai(indexTugas)` yang mengubah properti `selesai` dari tugas pada indeks tertentu menjadi `true`.
- Buat fungsi bernama `cariTugas(keyword)` yang mengembalikan array tugas yang deskripsinya mengandung `keyword`.
- Simulasikan "menyimpan" dan "mengambil" data dari backend dengan menggunakan `JSON.stringify()` saat "menyimpan" `daftarTugas` dan `JSON.parse()` saat "mengambil"nya. Tampilkan string JSON yang dihasilkan ke konsol.
- Tampilkan hasil pencarian tugas ke konsol.

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:

- Memahami apa itu DOM dan perannya dalam pengembangan web interaktif.
- Mengakses elemen HTML menggunakan berbagai metode seleksi DOM.
- Memanipulasi konten teks dan atribut elemen HTML.
- Mengubah gaya (CSS) elemen secara dinamis menggunakan JavaScript.
- 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");
semuanya.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



The screenshot shows a browser window with a dark theme. At the top, there are three circular icons: red, yellow, and green. The main content area displays an HTML page with the following structure:

```

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);
    });
})

```

The page contains an

element with the text "Selamat Datang di DOM!", a element with the text "Ini adalah paragraf informasi.", and a element with the text "Kotak Saya". There are several buttons: "Ubah Judul", "Tambah Paragraf", "Hapus Paragraf Terakhir", "Ubah Warna Kotak", "Toggle Highlight", and "ubahWarna". The "ubahWarna" button is highlighted with a blue background and white text.

```
// 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();
    }
});

// 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 yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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 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");
semaulink.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();  
    }  
});  
  
// 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 yang menampilkan daftar item yang dapat ditambahkan dan dihapus (seperti daftar belanja atau to-do list):

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.