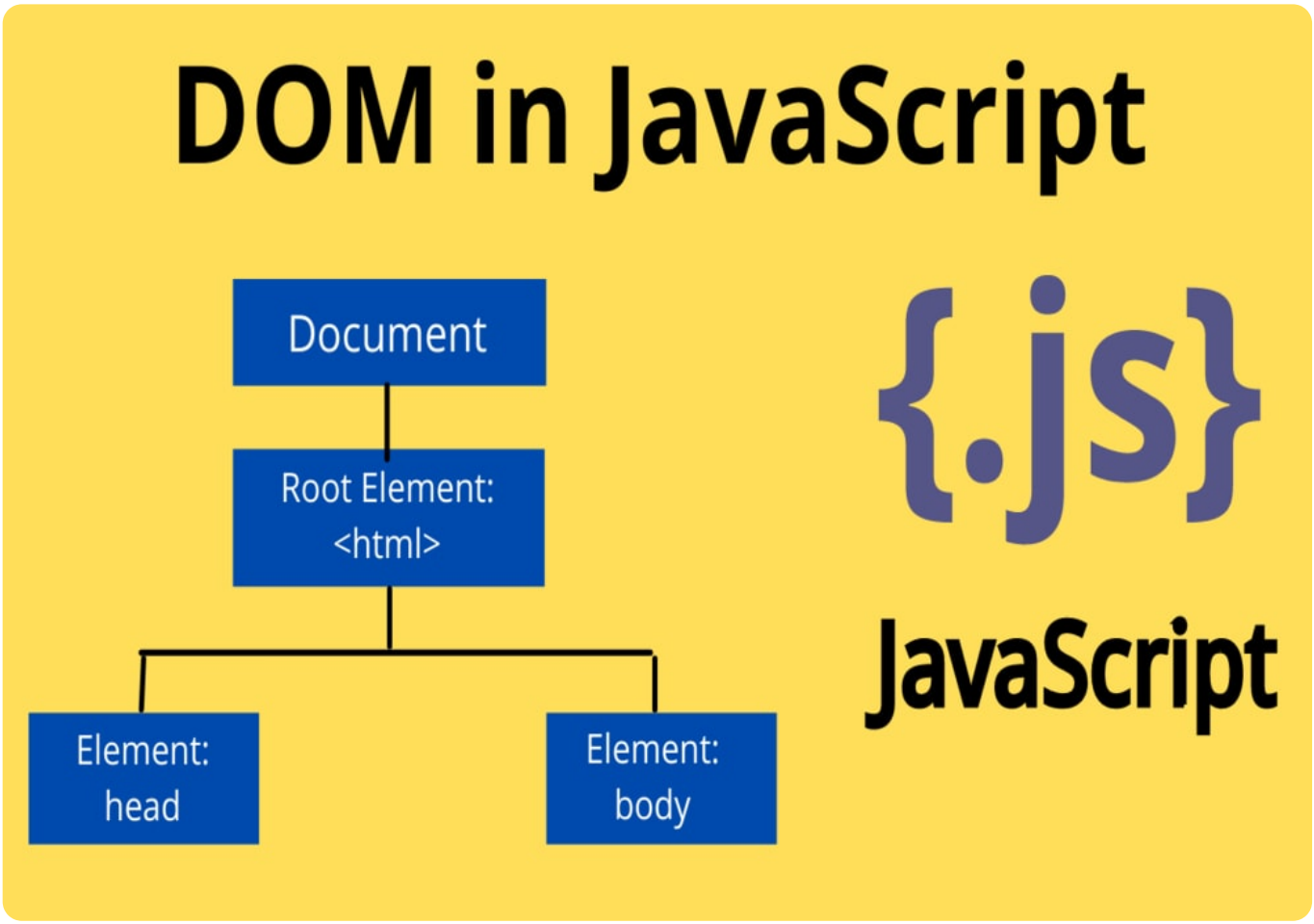

Selamat Datang! 👋

Halo teman-teman! Selamat datang di dunia JavaScript DOM. Jangan khawatir jika kamu belum pernah mendengar istilah "DOM" sebelumnya. Dalam e-book ini, kita akan belajar bersama-sama dengan bahasa yang mudah dipahami dan contoh-contoh yang menarik.

Bayangkan jika kamu bisa membuat website yang bisa berubah-ubah ketika diklik, atau membuat tombol yang bisa mengubah warna halaman. Semua itu bisa kamu lakukan dengan DOM JavaScript!

> *Oleh: Abdul Yamin, S.Pd., M.Kom*
> *Berdasarkan referensi W3Schools*

📖 Daftar Isi

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Bab 1: Pengantar DOM JavaScript

Apa itu DOM?

DOM adalah singkatan dari **Document Object Model**. Kedengarannya rumit ya? Tapi sebenarnya sederhana kok!

Bayangkan DOM seperti **pohon keluarga** dari halaman web kamu. Setiap bagian dari halaman web (seperti judul, paragraf, gambar, tombol) adalah anggota keluarga dalam pohon ini. JavaScript bisa "berbicara" dengan anggota keluarga ini untuk mengubah mereka.

Analogi Sederhana: Rumah dan Furniture

Coba bayangkan halaman web seperti sebuah rumah:

- **Dokumen HTML** = Rumah
- **Elemen HTML** (seperti `<h1>`, `<p>`, `<button>`) = Furniture di rumah
- **JavaScript DOM** = Remote control yang bisa mengubah furniture

Dengan remote control (JavaScript DOM), kamu bisa:

- Mengubah warna sofa (mengubah warna teks)
- Memindahkan meja (mengubah posisi elemen)
- Menyalakan/mematikan lampu (menampilkan/menyembunyikan elemen)
- Dan masih banyak lagi!

Mengapa DOM Penting?

Tanpa DOM, halaman web hanya bisa menampilkan informasi yang sama terus-menerus. Dengan DOM, kamu bisa membuat halaman web yang:

1. **Interaktif** - Merespons ketika pengguna mengklik atau mengetik
2. **Dinamis** - Berubah-ubah sesuai dengan situasi
3. **Menarik** - Memberikan pengalaman yang lebih seru untuk pengguna

⭐ github.com/ocikyamin
✉ ocikyamin@gmail.com

Struktur Pohon DOM

Mari kita lihat contoh sederhana struktur HTML dan bagaimana DOM melihatnya:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Website Saya</title>
</head>
<body>
    <h1>Selamat Datang</h1>
    <p>Ini adalah paragraf pertama.</p>
    <button>Klik Saya</button>
</body>
</html>
```

DOM melihat struktur di atas seperti pohon keluarga:

```
└── html (Kakek)
    ├── head (Ayah)
    │   └── title (Anak)
    │       └── "Website Saya" (Cucu)
    └── body (Ibu)
        ├── h1 (Anak pertama)
        │   └── "Selamat Datang" (Cucu)
        ├── p (Anak kedua)
        │   └── "Ini adalah paragraf pertama." (Cucu)
        └── button (Anak ketiga)
            └── "Klik Saya" (Cucu)
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

Contoh Pertama: Mengubah Teks

Mari kita coba contoh sederhana. Kita akan membuat halaman yang bisa mengubah teks ketika tombol diklik.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Contoh DOM Pertama</title>
</head>
<body>
    <h1 id="judul">Halo Dunia!</h1>
    <button onclick="ubahJudul()">Ubah Judul</button>

    <script>
        function ubahJudul() {
            // Cari elemen dengan id "judul"
            var judulElement = document.getElementById("judul");

            // Ubah teksnya
            judulElement.innerHTML = "Judul Sudah Berubah!";
        }
    </script>
</body>
</html>
```

Penjelasan Kode:

1. `<h1 id="judul">` - Kita memberi nama "judul" pada elemen h1 agar mudah dicari
2. `document.getElementById("judul")` - JavaScript mencari elemen dengan id "judul"
3. `innerHTML = "..."` - Mengubah isi teks dari elemen tersebut
4. `onclick="ubahJudul()"` - Ketika tombol diklik, jalankan fungsi ubahJudul()

Coba Sendiri!
Salin kode di atas ke file HTML dan buka di browser. Klik tombolnya dan lihat apa yang terjadi!

github.com/ocikyamin
ocikyamin@gmail.com

Apa yang Bisa Dilakukan DOM?

Dengan DOM JavaScript, kamu bisa:

1. Mengubah Konten

- Mengubah teks dalam paragraf
- Mengganti gambar
- Mengubah judul halaman

2. Mengubah Gaya (Style)

- Mengubah warna teks
- Mengubah ukuran font
- Mengubah warna background

3. Menambah atau Menghapus Elemen

- Menambah paragraf baru
- Menghapus gambar
- Membuat tombol baru

4. Merespons Interaksi Pengguna

- Ketika mouse diklik
- Ketika keyboard ditekan
- Ketika mouse bergerak di atas elemen

Latihan Sederhana

Sebelum lanjut ke bab berikutnya, coba buat halaman HTML sederhana dengan:

1. Satu judul (h1)
2. Satu paragraf (p)
3. Satu tombol (button)

Kemudian coba ubah warna teks judul menjadi merah ketika tombol diklik. (Petunjuk: gunakan `style.color = "red"`)

Ringkasan Bab 1

Di bab ini, kita sudah belajar bahwa DOM itu seperti pohon keluarga dari halaman web kita. Setiap bagian dari halaman web adalah anggota keluarga yang bisa kita ubah-ubah menggunakan JavaScript. Kita juga sudah mencoba contoh sederhana mengubah teks dengan DOM. Keren, kan?

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Di bab selanjutnya, kita akan belajar lebih dalam tentang bagaimana cara menemukan dan mengubah elemen-elemen di halaman web kita. Siap? Ayo lanjut!

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Bab 2: Mengakses dan Mengubah Elemen HTML

Di bab sebelumnya, kita sudah tahu kalau DOM itu seperti remote control untuk halaman web kita. Nah, di bab ini, kita akan belajar bagaimana cara menggunakan remote control itu untuk menemukan "furniture" (elemen HTML) yang kita inginkan, lalu mengubahnya.

Mencari Elemen HTML: Cara Menemukan "Furniture"

Ada beberapa cara untuk menemukan elemen HTML di halaman web kita. Ini seperti mencari furniture di rumah: bisa pakai nama, jenis, atau ciri-ciri lainnya.

1. Mencari Berdasarkan ID (`getElementById()`)

Ini adalah cara paling gampang dan paling sering dipakai. Setiap elemen HTML bisa punya ID unik, seperti nomor rumah. Kalau kamu tahu ID-nya, kamu bisa langsung menemukannya.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<p id="pesanSelamatDatang">Halo, selamat datang!</p>

<script>
    // Cari elemen dengan ID "pesanSelamatDatang"
    const pesan = document.getElementById("pesanSelamatDatang");

    // Tampilkan teksnya di konsol browser (untuk developer)
    console.log(pesan.innerHTML);
</script>

</body>
</html>
```

Penjelasan:

- `id="pesanSelamatDatang"`: Kita memberi ID unik pada paragraf ini.

github.com/ocikyamin
ocikyamin@gmail.com

- `document.getElementById("pesanSelamatDatang")`: Ini adalah perintah JavaScript untuk "cari elemen di dokumen yang ID-nya 'pesanSelamatDatang'".
- Hasilnya akan disimpan di variabel `pesan`.

2. Mencari Berdasarkan Nama Kelas (`getElementsByClassName()`)

Kadang, ada beberapa "furniture" yang punya jenis yang sama, misalnya semua kursi makan. Di HTML, kita bisa pakai `class` untuk menandai elemen-elemen yang punya ciri-ciri yang sama.

Penting: Karena bisa ada banyak elemen dengan kelas yang sama, JavaScript akan mengembalikan **daftar** elemen. Jadi, kita harus menentukan elemen ke berapa yang mau kita pakai (dimulai dari 0).

Contoh:

```html
<!DOCTYPE html>
<html>
<body>

<p class="info">Ini adalah informasi penting.</p>
<p class="info">Ini juga informasi penting.</p>
<p>Ini paragraf biasa.</p>

<script>
    // Cari semua elemen dengan kelas "info"
    const daftarInfo = document.getElementsByClassName("info");

    // Ubah teks elemen pertama (indeks 0) yang punya kelas "info"
    daftarInfo[0].innerHTML = "Informasi sudah diubah!";

    // Tampilkan teks elemen kedua (indeks 1) yang punya kelas "info"
    console.log(daftarInfo[1].innerHTML);
</script>

</body>
</html>
```

Penjelasan:

- `class="info"`: Kita memberi kelas "info" pada dua paragraf.

github.com/ocikyamin
ocikyamin@gmail.com

- `document.getElementsByClassName("info")`: Ini akan mengembalikan daftar elemen yang punya kelas "info".
- `daftarInfo[0]`: Mengakses elemen pertama dari daftar.

3. Mencari Berdasarkan Nama Tag (`getElementsByTagName()`)

Kalau kamu mau mencari semua elemen dengan jenis tertentu, misalnya semua paragraf (`<p>`) atau semua judul (`<h1>`), kamu bisa pakai cara ini.

Sama seperti mencari berdasarkan kelas, ini juga akan mengembalikan **daftar** elemen.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<h1>Judul Utama</h1>
<p>Paragraf pertama.</p>
<p>Paragraf kedua.</p>

<script>
    // Cari semua elemen <p>
    const semuaParagraf = document.getElementsByTagName("p");

    // Ubah teks paragraf kedua (indeks 1)
    semuaParagraf[1].innerHTML = "Paragraf ini sudah diganti!";
</script>

</body>
</html>
```

Penjelasan:

- `document.getElementsByTagName("p")`: Mencari semua elemen `<p>` di dokumen.

4. Mencari dengan Lebih Canggih (`querySelector()` dan `querySelectorAll()`)

Ini adalah cara yang lebih modern dan fleksibel, seperti mencari furniture dengan kombinasi kriteria (misalnya, "kursi warna merah di ruang tamu"). Kamu bisa pakai aturan CSS untuk mencari elemen.

⋆ github.com/ocikyamin
ocikyamin@gmail.com

- `querySelector()` : Mengembalikan **elemen pertama** yang cocok dengan kriteria.
- `querySelectorAll()` : Mengembalikan **semua elemen** yang cocok dengan kriteria (dalam bentuk daftar).

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<div id="kontainer">
  <p class="teks-penting">Teks ini sangat penting.</p>
  <p class="teks-biasa">Teks biasa.</p>
</div>

<script>
    // Cari elemen paragraf pertama di dalam div dengan ID "kontainer" yang punya ke
    const elemenPenting = document.querySelector("#kontainer .teks-penting");
    elemenPenting.style.color = "blue"; // Ubah warnanya jadi biru

    // Cari semua elemen paragraf yang punya kelas "teks-biasa"
    const semuaTeksBiasa = document.querySelectorAll("p.teks-biasa");
    semuaTeksBiasa[0].style.fontWeight = "bold"; // Tebalkan teks biasa yang pertama
</script>

</body>
</html>
```

Penjelasan:

- `"#kontainer .teks-penting"` : Ini adalah "selector" CSS. Artinya, cari elemen dengan kelas `teks-penting` yang ada di dalam elemen dengan ID `kontainer`.
- `"p.teks-biasa"` : Cari semua elemen `<p>` yang punya kelas `teks-biasa`.

Mengubah Elemen HTML: Mengganti "Furniture"

Setelah kamu menemukan elemen yang diinginkan, saatnya mengubahnya!

1. Mengubah Konten Teks atau HTML (`innerHTML`)

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Properti `innerHTML` adalah cara paling umum untuk mengubah isi dari sebuah elemen. Kamu bisa mengubah teks biasa, bahkan menambahkan tag HTML baru di dalamnya.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Teks awal.</p>

<button onclick="gantiTeks()">Ganti Teks</button>
<button onclick="gantiHTML()">Ganti HTML</button>

<script>
    function gantiTeks() {
        document.getElementById("demo").innerHTML = "Teks sudah diganti!";
    }

    function gantiHTML() {
        document.getElementById("demo").innerHTML = "<b>Teks ini tebal sekarang!</b>
    }
</script>

</body>
</html>
```

2. Mengubah Atribut HTML (`setAttribute()` , `getAttribute()` , `removeAttribute()`)

Atribut adalah informasi tambahan pada elemen HTML, seperti `src` pada gambar atau `href` pada link. Kamu bisa mengubah, mengambil, atau menghapus atribut ini.

- `setAttribute(namaAtribut, nilaiBaru)` : Mengatur atau mengubah nilai atribut.
- `getAttribute(namaAtribut)` : Mengambil nilai atribut.
- `removeAttribute(namaAtribut)` : Menghapus atribut.

Contoh:

⋆ github.com/ocikyamin
ocikyamin@gmail.com

```html
<!DOCTYPE html>
<html>
<body>

<img id="gambarSaya" src="https://www.w3schools.com/images/smiley.gif" alt="Smiley F

<button onclick="gantiGambar()">Ganti Gambar</button>
<button onclick="cekAlt()">Cek Teks Alternatif</button>
<button onclick="hapusLebar()">Hapus Lebar</button>

<script>
    function gantiGambar() {
        const gambar = document.getElementById("gambarSaya");
        gambar.setAttribute("src", "https://www.w3schools.com/images/img_girl.jpg");
        gambar.setAttribute("alt", "Gadis Cantik");
    }

    function cekAlt() {
        const gambar = document.getElementById("gambarSaya");
        alert("Teks alternatif gambar: " + gambar.getAttribute("alt"));
    }

    function hapusLebar() {
        const gambar = document.getElementById("gambarSaya");
        gambar.removeAttribute("width");
        alert("Atribut 'width' sudah dihapus!");
    }
</script>

</body>
</html>
```

3. Mengubah Gaya (Style) CSS (`style`)

Kamu bisa mengubah tampilan elemen langsung dari JavaScript dengan properti `style`. Ingat, nama properti CSS yang ada tanda hubungnya (misalnya `background-color`) di JavaScript ditulis dengan `camelCase` (misalnya `backgroundColor`).

Contoh:

⋆ github.com/ocikyamin
ocikyamin@gmail.com

```html
<!DOCTYPE html>
<html>
<body>

<p id="teksGaya" style="color: red; font-size: 20px;">Teks ini akan berubah gayanya.

<button onclick="ubahGaya()">Ubah Gaya</button>

<script>
    function ubahGaya() {
        const teks = document.getElementById("teksGaya");
        teks.style.color = "green";
        teks.style.fontSize = "30px";
        teks.style.backgroundColor = "yellow";
        teks.style.border = "2px solid blue";
    }
</script>

</body>
</html>
```

Latihan Mandiri

Coba buat halaman HTML dengan:

1. Sebuah `div` kosong dengan ID `kotakWarna`.
2. Tiga tombol: "Merah", "Biru", "Hijau".

Ketika tombol "Merah" diklik, ubah `background-color` dari `div` menjadi merah. Lakukan hal yang sama untuk tombol "Biru" dan "Hijau".

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Bab 3: Berinteraksi dengan Pengguna (Events)

Di bab sebelumnya, kita sudah belajar cara menemukan dan mengubah elemen HTML. Tapi, bagaimana caranya agar perubahan itu terjadi *ketika* pengguna melakukan sesuatu? Nah, di sinilah peran **Events** atau **Peristiwa**.

Bayangkan Events seperti "sinyal" yang dikirim oleh browser ketika sesuatu terjadi. Misalnya, ketika kamu mengklik tombol, browser mengirim sinyal "klik". Ketika kamu menggerakkan mouse, browser mengirim sinyal "mouse bergerak". JavaScript bisa "mendengarkan" sinyal-sinyal ini dan melakukan sesuatu sebagai respons.

Apa itu Event?

Event adalah hal-hal yang terjadi pada elemen HTML. Contohnya:

- **Klik mouse:** Pengguna mengklik tombol atau teks.
- **Halaman dimuat:** Seluruh halaman web sudah siap ditampilkan.
- **Mouse bergerak:** Kursor mouse digerakkan di atas suatu area.
- **Input diubah:** Pengguna mengetik sesuatu di kotak teks.
- **Tombol keyboard ditekan:** Pengguna menekan tombol di keyboard.

Bagaimana JavaScript Merespons Event?

Untuk merespons event, kita perlu "memberi tahu" JavaScript apa yang harus dilakukan ketika event tertentu terjadi. Ini disebut **Event Handler** atau **Penanganan Peristiwa**.

Ada beberapa cara untuk melakukannya, tapi kita akan fokus pada cara yang paling modern dan direkomendasikan.

Menggunakan `addEventListener()` (Cara Terbaik!)

`addEventListener()` adalah cara paling keren dan fleksibel untuk "mendengarkan" event. Kenapa keren? Karena:

1. **Bisa banyak aksi:** Satu elemen bisa punya banyak respons untuk satu event. Misalnya, satu tombol klik bisa menampilkan pesan DAN mengubah warna.
2. **Kode lebih rapi:** Kode JavaScript terpisah dari HTML, jadi lebih mudah dibaca dan diatur.

Sintaks (Cara Penulisan):

- `elemen`: Elemen HTML yang ingin kamu dengarkan event-nya (misalnya, tombol, paragraf).

⋆ github.com/ocikyamin
ocikyamin@gmail.com

- `"namaEvent"`: Nama event yang ingin kamu dengarkan (misalnya, "click", "mouseover", "keydown"). **Penting: Jangan pakai "on" di depannya!** Cukup "click", bukan "onclick".
- `fungsiYangAkanDijalankan`: Kode JavaScript yang akan dijalankan ketika event terjadi.

Contoh 1: Tombol Klik Sederhana

Mari kita buat tombol yang menampilkan pesan ketika diklik.

```
<!DOCTYPE html>
<html>
<body>

<button id="tombolPesan">Klik Saya!</button>

<script>
    // 1. Cari tombolnya
    const tombol = document.getElementById("tombolPesan");

    // 2. Tambahkan event listener untuk event "click"
    tombol.addEventListener("click", function() {
        alert("Tombol diklik!");
    });
</script>

</body>
</html>
```

Penjelasan:

- Kita mencari tombol dengan ID `tombolPesan`.
- Lalu, kita "menempelkan" pendengar event (`addEventListener`) ke tombol itu.
- Kita bilang, "kalau ada event `click` di tombol ini, jalankan fungsi ini: `alert("Tombol diklik!");`".

Contoh 2: Mengubah Teks Saat Mouse Lewat

Kita bisa membuat teks berubah ketika mouse kita lewat di atasnya.

```
<!DOCTYPE html>
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

```
<html>
<body>

<p id="teksHover">Arahkan mouse ke sini!</p>

<script>
    const teks = document.getElementById("teksHover");

    // Ketika mouse masuk (mouseover)
    teks.addEventListener("mouseover", function() {
        teks.innerHTML = "Mouse masuk!";
    });

    // Ketika mouse keluar (mouseout)
    teks.addEventListener("mouseout", function() {
        teks.innerHTML = "Arahkan mouse ke sini lagi!";
    });
</script>

</body>
</html>
```

Penjelasan:

- Kita menggunakan dua event: `mouseover` (saat mouse masuk area elemen) dan `mouseout` (saat mouse keluar area elemen).
- Setiap event punya fungsi sendiri yang akan mengubah teks paragraf.

Objek Event: Informasi Tambahan

Ketika sebuah event terjadi, JavaScript secara otomatis membuat "objek event" yang berisi banyak informasi tentang event tersebut. Kamu bisa mengakses objek ini di dalam fungsi event handler.

Contoh: Mengetahui Posisi Klik Mouse

```
<!DOCTYPE html>
<html>
<body>

<p id="infoKlik">Klik di mana saja di sini!</p>
```


github.com/ocikyamin
ocikyamin@gmail.com

```
<script>
    const info = document.getElementById("infoKlik");

    info.addEventListener("click", function(event) {
        // event.clientX dan event.clientY memberikan koordinat X dan Y mouse
        const posisiX = event.clientX;
        const posisiY = event.clientY;
        info.innerHTML = `Kamu mengklik di posisi X: ${posisiX}, Y: ${posisiY}`;
    });
</script>

</body>
</html>
```

Penjelasan:

- Fungsi `function(event)` menerima objek `event` sebagai parameter.
- `event.clientX` dan `event.clientY` adalah properti dari objek `event` yang memberitahu kita koordinat horizontal dan vertikal tempat mouse diklik.

Latihan Seru!

Coba buat halaman HTML dengan:

1. Sebuah kotak (`div` dengan `background-color` dan `height` / `width` di CSS).
2. Ketika kotak diklik, ubah warna background-nya menjadi warna acak (cari di internet cara membuat warna acak dengan JavaScript).
3. Ketika mouse masuk ke kotak, tampilkan pesan "Siap diklik!" di konsol browser.
4. Ketika mouse keluar dari kotak, tampilkan pesan "Sampai jumpa!" di konsol browser.

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Bab 4: Navigasi Antar Elemen

Di bab sebelumnya, kita sudah belajar cara mencari elemen tertentu dan merespons interaksi pengguna. Sekarang, bayangkan kalau kamu punya sebuah elemen, dan kamu ingin tahu siapa "orang tuanya", "saudaranya", atau "anak-anaknya" di pohon DOM. Nah, inilah yang akan kita pelajari di bab ini: **Navigasi Antar Elemen**.

Ini seperti kamu sedang berada di sebuah ruangan di rumah, dan kamu ingin tahu di mana letak pintu ke ruangan lain, atau siapa saja yang ada di ruangan ini, atau ruangan apa di sebelahnya.

Hubungan Keluarga dalam DOM

Setiap elemen di DOM punya hubungan dengan elemen lain. Hubungan ini mirip dengan hubungan keluarga:

- **Orang Tua (Parent):** Elemen yang membungkus elemen lain. Contoh: `<body>` adalah parent dari `<h1>` dan `<p>`.
- **Anak (Child):** Elemen yang berada di dalam elemen lain. Contoh: `<h1>` dan `<p>` adalah child dari `<body>`.
- **Saudara (Sibling):** Elemen yang berada di tingkat yang sama dan punya parent yang sama. Contoh: `<h1>` dan `<p>` adalah sibling.

Properti untuk Navigasi Elemen

Ada beberapa properti khusus di JavaScript yang bisa kita gunakan untuk menelusuri hubungan ini. Kita akan fokus pada properti yang hanya melihat elemen HTML saja, mengabaikan teks kosong atau komentar.

1. `parentElement`: Mencari "Orang Tua"

Properti `parentElement` akan mengembalikan elemen induk (parent) dari elemen yang sedang kamu lihat.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<div id="kotakUtama">
  <p id="paragrafAnak">Ini adalah paragraf di dalam kotak.</p>
</div>
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

```
<script>
    const paragraf = document.getElementById("paragrafAnak");
    const orangTua = paragraf.parentElement;

    console.log("Nama tag orang tua paragraf adalah: " + orangTua.tagName); // Outpu
    orangTua.style.border = "2px solid blue"; // Beri border pada orang tuanya
</script>

</body>
</html>
```

Penjelasan:

- Kita punya paragraf dengan ID `paragrafAnak` yang dibungkus oleh `div` dengan ID `kotakUtama`.
- `paragraf.parentElement` akan mengembalikan elemen `div` tersebut.

2. `children`: Mencari "Anak-anak"

Properti `children` akan mengembalikan daftar semua elemen anak dari sebuah elemen. Ini seperti melihat siapa saja yang ada di dalam sebuah ruangan.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<ul id="daftarBelanja">
  <li>Apel</li>
  <li>Jeruk</li>
  <li>Mangga</li>
</ul>

<script>
    const daftar = document.getElementById("daftarBelanja");
    const anakAnakDaftar = daftar.children;

    console.log("Jumlah item di daftar: " + anakAnakDaftar.length); // Output: 3

    // Ubah warna item pertama
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

```
    anakAnakDaftar[0].style.color = "red";

    // Ubah warna item terakhir
    anakAnakDaftar[anakAnakDaftar.length - 1].style.color = "green";
</script>

</body>
</html>
```

Penjelasan:

- `daftar.children` akan mengembalikan daftar elemen `<li>` yang ada di dalam `<ul>`.
- Kita bisa mengakses setiap anak menggunakan indeks, seperti `anakAnakDaftar[0]` untuk anak pertama.

3. `firstElementChild` dan `lastElementChild`: Anak Pertama dan Terakhir

Kalau kamu cuma butuh anak yang paling pertama atau paling terakhir, properti ini lebih cepat daripada harus mengakses `children[0]` atau `children[length - 1]`.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<div id="konten">
  <h1>Judul Konten</h1>
  <p>Paragraf pembuka.</p>
  <img src="gambar.jpg" alt="Gambar">
  <p>Paragraf penutup.</p>
</div>

<script>
    const konten = document.getElementById("konten");

    const anakPertama = konten.firstElementChild;
    const anakTerakhir = konten.lastElementChild;

    console.log("Anak pertama: " + anakPertama.tagName); // Output: H1
```


github.com/ocikyamin
ocikyamin@gmail.com

```
    console.log("Anak terakhir: " + anakTerakhir.tagName); // Output: P

    anakPertama.style.textDecoration = "underline";
    anakTerakhir.style.fontStyle = "italic";
</script>

</body>
</html>
```

4. `nextElementSibling` dan `previousElementSibling`: Saudara Sebelah

Properti ini memungkinkan kamu untuk bergerak maju atau mundur di antara elemen-elemen yang berada di tingkat yang sama (saudara).

- `nextElementSibling`: Mengembalikan elemen saudara berikutnya.
- `previousElementSibling`: Mengembalikan elemen saudara sebelumnya.

Contoh:

```
<!DOCTYPE html>
<html>
<body>

<ul>
  <li>Item A</li>
  <li id="itemB">Item B</li>
  <li>Item C</li>
</ul>

<script>
    const itemB = document.getElementById("itemB");

    const saudaraBerikutnya = itemB.nextElementSibling;
    const saudaraSebelumnya = itemB.previousElementSibling;

    if (saudaraBerikutnya) {
        saudaraBerikutnya.style.backgroundColor = "lightblue";
        console.log("Saudara berikutnya dari Item B: " + saudaraBerikutnya.innerHTML
    }
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

```
    if (saudaraSebelumnya) {
        saudaraSebelumnya.style.backgroundColor = "lightgreen";
        console.log("Saudara sebelumnya dari Item B: " + saudaraSebelumnya.innerHTML
    }
</script>

</body>
</html>
```

Penjelasan:

- Kita mulai dari `itemB`.
- `itemB.nextElementSibling` akan mengembalikan `<li>Item C</li>`.
- `itemB.previousElementSibling` akan mengembalikan `<li>Item A</li>`.

Latihan Menjelajah Pohon DOM

Buat halaman HTML dengan struktur seperti ini:

```
<!DOCTYPE html>
<html>
<body>

<div id="container">
  <header>
    <h1>Judul Halaman</h1>
    <nav>
      <a href="#">Home</a>
      <a href="#">About</a>
    </nav>
  </header>
  <main>
    <section id="artikel">
      <h2>Artikel Utama</h2>
      <p>Ini adalah paragraf pertama artikel.</p>
      <p>Ini adalah paragraf kedua artikel.</p>
    </section>
    <aside>
      <h3>Sidebar</h3>
      <p>Informasi tambahan.</p>
```


github.com/ocikyamin
ocikyamin@gmail.com

```
    </aside>
  </main>
  <footer>
    <p>&copy; 2025 Belajar DOM</p>
  </footer>
</div>

</body>
</html>
```

Kemudian, gunakan JavaScript untuk:

1. Ubah warna teks `<h1>` menjadi ungu.
2. Ubah warna background dari `header` (orang tua dari `<h1>`).
3. Tebalkan teks dari paragraf kedua di dalam `<section id="artikel">`.
4. Tambahkan border merah pada elemen `aside` (saudara dari `section#artikel`).

⋆ github.com/ocikyamin
ocikyamin@gmail.com

Bab 5: Bekerja dengan Formulir dan CSS

Di bab ini, kita akan belajar dua hal penting yang sering kita temui di website: **Formulir** (seperti kotak isian nama, email, atau tombol kirim) dan **CSS** (untuk mengatur tampilan dan gaya halaman web). JavaScript DOM bisa membantu kita berinteraksi dengan keduanya!

Bagian 1: Berinteraksi dengan Formulir

Formulir adalah cara pengguna memasukkan data ke website. Dengan DOM, kita bisa membaca apa yang diketik pengguna, mengubah nilai di formulir, atau bahkan memeriksa apakah inputnya sudah benar.

Mengakses Elemen Formulir

Sama seperti elemen HTML lainnya, elemen formulir juga bisa diakses dengan `getElementById()`, `getElementsByClassName()`, atau `querySelector()`.

Contoh: Mengambil Nilai dari Kotak Input

```html
<!DOCTYPE html>
<html>
<body>

Nama: <input type="text" id="namaPengguna" value="Budi">
<button onclick="tampilkanNama()">Tampilkan Nama</button>

<p id="hasil"></p>

<script>
    function tampilkanNama() {
        // Cari kotak input dengan ID "namaPengguna"
        const inputNama = document.getElementById("namaPengguna");

        // Ambil nilai yang ada di kotak input (apa yang diketik pengguna)
        const nilaiNama = inputNama.value;

        // Tampilkan nilai tersebut di paragraf hasil
        document.getElementById("hasil").innerHTML = "Halo, " + nilaiNama + "!";
    }
</script>
```


github.com/ocikyamin
ocikyamin@gmail.com

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

Penjelasan:

- `inputNama.value`: Ini adalah properti yang berisi teks yang ada di dalam kotak input.

Mengubah Nilai Formulir

Kita juga bisa mengubah nilai di kotak input secara otomatis.

Contoh: Mengubah Nilai Input

```
<!DOCTYPE html>
<html>
<body>

<input type="text" id="kota" value="Jakarta">
<button onclick="gantiKota()">Ganti Kota</button>

<script>
    function gantiKota() {
        const inputKota = document.getElementById("kota");
        inputKota.value = "Bandung"; // Ubah nilai input menjadi "Bandung"
    }
</script>

</body>
</html>
```

Bekerja dengan Checkbox dan Radio Button

Untuk checkbox dan radio button, kita sering perlu tahu apakah mereka `checked` (tercentang) atau tidak.

Contoh: Memeriksa Checkbox

```
```


⋆ github.com/ocikyamin
ocikyamin@gmail.com

```
<!DOCTYPE html>
<html>
<body>

<input type="checkbox" id="setuju" checked>
<label for="setuju">Saya Setuju dengan Syarat dan Ketentuan</label><br><br>

<button onclick="cekStatus()">Cek Status</button>

<p id="statusCek"></p>

<script>
    function cekStatus() {
        const checkbox = document.getElementById("setuju");
        if (checkbox.checked) {
            document.getElementById("statusCek").innerHTML = "Kamu setuju!";
        } else {
            document.getElementById("statusCek").innerHTML = "Kamu belum setuju.";
        }
    }
</script>

</body>
</html>
```

Penjelasan:

- `checkbox.checked`: Properti ini akan bernilai `true` jika checkbox tercentang, dan `false` jika tidak.

Bagian 2: Mengubah Gaya (Style) dengan JavaScript

Di Bab 2, kita sudah sedikit menyentuh tentang mengubah gaya dengan `element.style.propertiCSS`. Sekarang kita akan melihat lebih dalam dan juga cara yang lebih rapi menggunakan kelas CSS.

Mengubah Properti CSS Langsung

Ini adalah cara paling dasar untuk mengubah gaya. Ingat, nama properti CSS yang ada tanda hubungnya (misalnya `background-color`) di JavaScript ditulis dengan `camelCase` (misalnya `backgroundColor`).

Contoh: Mengubah Warna dan Ukuran Teks

github.com/ocikyamin
ocikyamin@gmail.com

```html
<!DOCTYPE html>
<html>
<body>

<p id="teksUbah">Teks ini akan berubah gayanya.</p>

<button onclick="ubahGayaLangsung()">Ubah Gaya</button>

<script>
    function ubahGayaLangsung() {
        const teks = document.getElementById("teksUbah");
        teks.style.color = "purple"; // Ubah warna teks jadi ungu
        teks.style.fontSize = "28px"; // Ubah ukuran font jadi 28px
        teks.style.textDecoration = "underline"; // Tambah garis bawah
    }
</script>

</body>
</html>
```

Menggunakan Kelas CSS (`classList`)

Cara yang lebih baik dan rapi untuk mengubah gaya adalah dengan memanipulasi kelas CSS. Kamu bisa membuat aturan gaya di file CSS atau di `<style>` di HTML, lalu JavaScript hanya perlu menambahkan atau menghapus kelas tersebut dari elemen.

Ini seperti kamu punya beberapa setelan baju (kelas CSS). Kamu tinggal ganti setelan bajunya, daripada harus mengubah satu per satu kancing, warna, dan ukuran baju.

Properti `classList` memiliki beberapa metode penting:

- `add('nama-kelas')`: Menambahkan kelas ke elemen.
- `remove('nama-kelas')`: Menghapus kelas dari elemen.
- `toggle('nama-kelas')`: Jika kelas ada, hapus. Jika tidak ada, tambahkan.
- `contains('nama-kelas')`: Memeriksa apakah elemen punya kelas tertentu (mengembalikan `true` atau `false`).

Contoh: Mengubah Gaya dengan `classList.toggle()`

⋆ github.com/ocikyamin
ocikyamin@gmail.com

```html
<!DOCTYPE html>
<html>
<head>
<style>
  .highlight {
    background-color: yellow;
    font-weight: bold;
    border: 2px solid orange;
  }
  .tebal-merah {
    color: red;
    font-weight: bold;
  }
</style>
</head>
<body>

<p id="paragrafKelas">Ini adalah paragraf biasa.</p>

<button onclick="toggleHighlight()">Toggle Highlight</button>
<button onclick="tambahTebalMerah()">Tambah Tebal Merah</button>

<script>
    function toggleHighlight() {
        const paragraf = document.getElementById("paragrafKelas");
        paragraf.classList.toggle("highlight"); // Tambah/hapus kelas 'highlight'
    }

    function tambahTebalMerah() {
        const paragraf = document.getElementById("paragrafKelas");
        paragraf.classList.add("tebal-merah"); // Tambah kelas 'tebal-merah'
    }
</script>

</body>
</html>
```

Penjelasan:

- Kita punya dua kelas CSS: `highlight` dan `tebal-merah`.

⋆ github.com/ocikyamin
ocikyamin@gmail.com

- Ketika tombol "Toggle Highlight" diklik, kelas `highlight` akan ditambahkan atau dihapus dari paragraf. Ini akan mengubah background, ketebalan font, dan border secara bersamaan.
- Ketika tombol "Tambah Tebal Merah" diklik, kelas `tebal-merah` akan ditambahkan, membuat teks menjadi merah dan tebal.

Latihan Akhir Bab

Buat halaman HTML dengan:

1. Sebuah kotak input teks dengan ID `inputKata`.
2. Sebuah tombol dengan teks "Kirim".
3. Sebuah paragraf kosong dengan ID `pesanOutput`.
4. Sebuah tombol dengan teks "Ubah Warna Latar".

Tugas JavaScript:

- Ketika tombol "Kirim" diklik, ambil teks dari `inputKata` dan tampilkan di `pesanOutput`.
- Ketika tombol "Ubah Warna Latar" diklik, ubah warna latar belakang (background-color) dari `<body>` menjadi warna yang berbeda setiap kali diklik (misalnya, bergantian antara biru muda, hijau muda, dan kuning muda). Gunakan `classList.toggle()` atau `style.backgroundColor`.

Kesimpulan dan Tips

Selamat! Kamu sudah berhasil menjelajahi dasar-dasar JavaScript HTML DOM. Kamu sudah belajar bagaimana DOM bekerja seperti pohon keluarga di halaman web, cara menemukan dan mengubah elemen, merespons interaksi pengguna, menavigasi antar elemen, hingga berinteraksi dengan formulir dan mengubah gaya CSS.

DOM adalah alat yang sangat ampuh untuk membuat halaman webmu menjadi lebih hidup dan interaktif. Dengan DOM, kamu bisa membuat website yang tidak hanya menampilkan informasi, tetapi juga bisa "berbicara" dan "bereaksi" terhadap penggunanya.

Ingat Poin-Poin Penting Ini:

- **DOM itu Pohon Keluarga:** Setiap elemen HTML adalah "anggota keluarga" yang bisa diakses dan diubah.
- **Cari Dulu, Baru Ubah:** Gunakan `getElementById()`, `getElementsByClassName()`, `getElementsByTagName()`, atau `querySelector()`/`querySelectorAll()` untuk menemukan elemen yang tepat.
- `innerHTML` **untuk Konten:** Ubah teks atau HTML di dalam elemen dengan `innerHTML`.
- `style` **untuk Gaya Langsung:** Ubah properti CSS langsung dengan `element.style.propertiCSS` (ingat `camelCase`!).

⋆ github.com/ocikyamin
ocikyamin@gmail.com

- `classList` **untuk Gaya Rapi:** Cara terbaik untuk mengubah gaya adalah dengan menambah/menghapus kelas CSS menggunakan `classList.add()`, `classList.remove()`, atau `classList.toggle()`.
- `addEventListener()` **untuk Interaksi:** Gunakan ini untuk membuat JavaScript merespons klik, ketikan, atau gerakan mouse pengguna.
- **Navigasi untuk Hubungan:** Gunakan `parentElement`, `children`, `nextElementSibling`, `previousElementSibling` untuk menjelajahi hubungan antar elemen.

Tips untuk Belajar Lebih Lanjut:

1. **Praktik, Praktik, Praktik!** Ini adalah kunci utama. Coba semua contoh kode, modifikasi, dan buat proyek kecilmu sendiri.
2. **Eksplorasi W3Schools:** Situs W3Schools adalah sumber yang sangat bagus. Jangan ragu untuk menjelajahi bagian lain tentang DOM atau JavaScript secara umum.
3. **Lihat Kode Orang Lain:** Buka website favoritmu, lalu klik kanan dan pilih "Inspect Element" atau "Lihat Sumber Halaman". Kamu bisa melihat bagaimana developer lain menggunakan HTML, CSS, dan JavaScript.
4. **Jangan Takut Error:** Error adalah bagian dari proses belajar. Baca pesan errornya, cari tahu artinya, dan coba perbaiki. Itu akan membuatmu semakin pintar!
5. **Buat Proyek Kecil:** Coba buat proyek sederhana seperti:
    - Daftar tugas (To-Do List) yang bisa ditambah dan dihapus.
    - Game tebak angka sederhana.
    - Kalkulator sederhana.

Semoga e-book ini membantumu memahami JavaScript HTML DOM dan membuatmu semakin semangat untuk belajar pemrograman web. Dunia web sangat luas dan menarik, dan kamu sudah punya dasar yang kuat untuk menjelajahinya!

Selamat Belajar dan Berkarya!

⋆ github.com/ocikyamin
ocikyamin@gmail.com