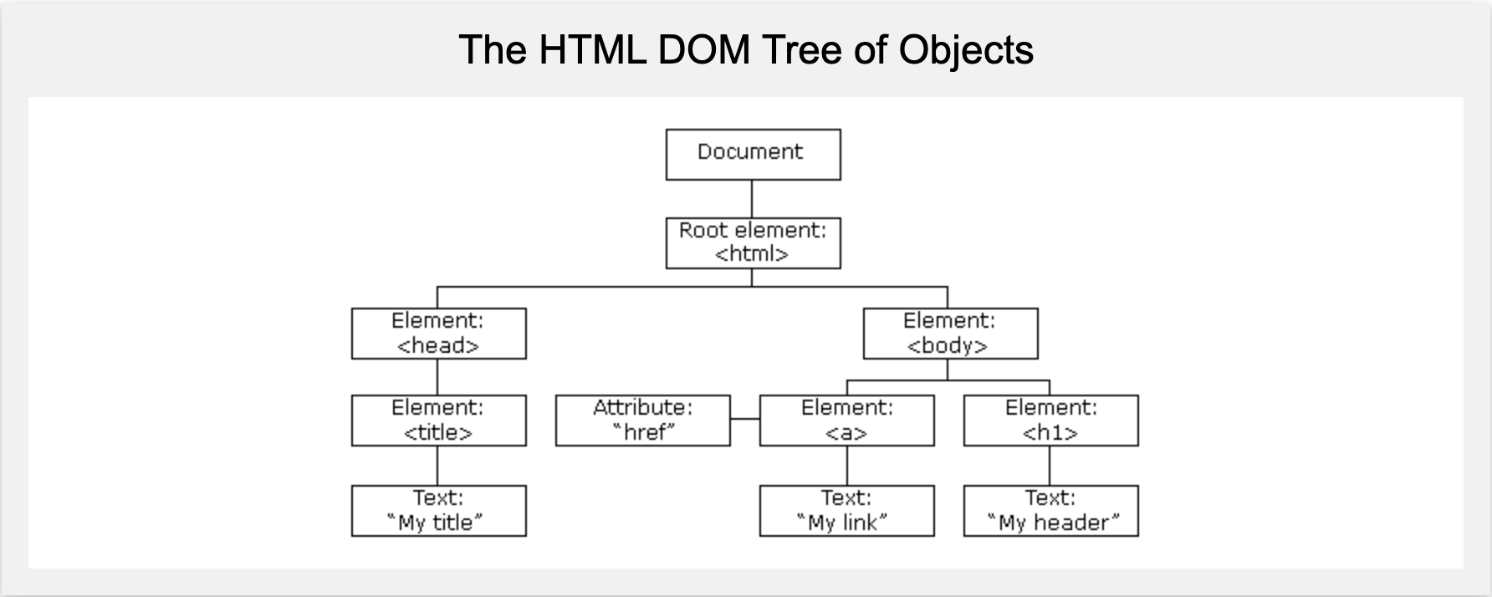
Kali ini kita akan mempelajari JavaScript HTML DOM. DOM adalah singkatan dari *Document Object Model*.

Jadi, ketika halaman website kita diload, browser kita akan membuat Document Object Model dari halaman website kita. Strukturnya kira-kira seperti ini:

  
Sumber gambar: [W3Schools](https://www.w3schools.com/js/js_htmldom.asp)

Dengan adanya DOM ini, JavaScript diberi akses untuk membuat HTML menjadi dinamis, seperti:

* Mengubah element HTML pada halaman website.
* Mengubah attribute HTML pada halaman website.
* Mengubah CSS style pada halaman website.
* Menambah dan/atau menghapus element maupun attribute HTML.
* Menambah HTML event (contoh: efek klik pada mouse, hover pada mouse, dan lain-lain) pada halaman website.
* Berinteraksi dengan semua HTML event di website.

Di HTML DOM, semua element HTML dari sebuah website dianggap sebagai objek. Dan sama seperti objek JavaScript pada umumnya, objek element HTML di HTML DOM juga mempunyai properti dan method atau yang lebih dikenal dengan istilah **DOM Property** dan **DOM Method**.

Jadi untuk mengubah nilai properti dari element HTML, kita bisa menggunakan **DOM Property** dan untuk memanggil fungsi dari suatu element HTML, kita bisa menggunakan **DOM Method**.

**Contoh Penggunaan**

Misalnya kita mempunyai element HTML sebagai berikut:

<input id="umur" type="text" value="20" />

Untuk mengakses value dari <input> di atas, maka bisa dilakukan dengan cara sebagai berikut:

let umur = document.getElementById("umur").value;

console.log(umur); // Output: 20

Penjelasan kode di atas:

* document adalah akar dari semua objek di sebuah website. Jadi untuk mengakses element HTML apapun di satu website, selalu dimulai dengan objek document ini.
* getElementById merupakan **method** dari objek document.
* value merupakan **properti** dari objek element HTML yang dikembalikan dari method getElementById yaitu <input>.

Karena ada banyak sekali jenis properti dan method di JavaScript HTML DOM, di topik selanjutnya kita akan membahas beberapa contoh penggunaan properti dan method yang seringkali digunakan.

Di topik sebelumnya, kita sudah melihat contoh cara menggunakan JavaScript HTML DOM untuk mengambil nilai dari sebuah element HTML <input>.

Mari kita perhatikan sekali lagi contoh tersebut:

<!-- html -->

<input id="umur" type="text" value="20" />

// js

let umur = document.getElementById("umur").value;

console.log(umur); // Output: 20

Sebelum kita bisa mengambil nilai attribute value dari element HTML <input> di JavaScript, kita terlebih dahulu harus mengakses ke element <input>. Di contoh di atas, kita menggunakan salah satu **DOM Method** untuk mengakses element HTML yaitu getElementById.

Di topik ini, kita akan bahas lebih lanjut DOM method lainnya untuk mengakses element HTML.

**getElementById(id)**

Method yang ini sudah kita lihat beberapa kali dari contoh-contoh sebelumnya. Kita bisa menggunakan getElementById untuk mengakses element HTML berdasarkan nilai id-nya.

**getElementsByTagName(tag)**

Untuk mengakses element-element HTML berdasarkan jenis tag-nya, kita bisa menggunakan getElementsByTagName.

Contoh:

<!-- html -->

<h1 id="title">Hello, World!</h1>

<p>Selamat Datang di Skilvul</p>

<h1 class="subtitle">Mari Belajar JavaScript</h1>

let semuaTagH1 = document.getElementsByTagName("h1");

console.log(semuaTagH1); // Output: HTMLCollection(2) [h1#title, h1.subtitle]

console.log(semuaTagH1[0]); // Output: <h1 id="title">Hello, World!</h1>

console.log(semuaTagH1[1]); // Output: <h1 class="subtitle">Mari Belajar JavaScript</h1>

Penjelasan kode di atas:

* Ketika kita memanggil document.getElementsByTagName("h1"), document sebagai akar objek dari halaman html kita akan mencari semua element <h1> di bawahnya.
* Karena di contoh di atas terdapat 2 <h1>, maka document.getElementsByTagName("h1") akan mengembalikan 2 element <h1>. Untuk mengakses masing-masing element, bisa menggunakan [noIndex], sama seperti cara Array mengakses elementnya.

**getElementsByClassName(className)**

Untuk mengakses element-element HTML berdasarkan nilai attribute class-nya, kita bisa menggunakan getElementsByClassName.

Contoh:

<!-- html -->

<h1 class="header">Hello, World!</h1>

<p>Selamat Datang di Skilvul</p>

<span class="header">Mari Belajar JavaScript</span>

let semuaClassHeader = document.getElementsByClassName("header");

console.log(semuaClassHeader); // Output: HTMLCollection(2) [h1.header, span.header]

console.log(semuaClassHeader[0]); // Output: <h1 class="header">Hello, World!</h1>

console.log(semuaClassHeader[1]); // Output: <span class="header">Mari Belajar JavaScript</span>

Penjelasan kode di atas:

* Ketika kita memanggil document.getElementsByClassName("header"), document sebagai akar objek dari halaman html kita akan mencari semua element dengan attribute class bernilai header di bawahnya.
* Karena di contoh di atas terdapat 2 element yang mempunyai class dengan nilai header, maka document.getElementsByClassName("header") akan mengembalikan 2 element, yaitu element <h1 class="header> dan <span class="header">. Untuk mengakses masing-masing element, bisa menggunakan [noIndex], sama seperti cara Array mengakses elementnya.

**querySelectorAll(cssSelector)**

Untuk mengakses element-element HTML berdasarkan CSS Selector-nya HTML, kita bisa menggunakan querySelectorAll.

Contoh:

<!-- html -->

<h1 class="header">Hello, World!</h1>

<p id="header2">Selamat Datang di Skilvul</p>

<span class="header">Mari Belajar JavaScript</span>

let h1ClassHeader = document.querySelectorAll('h1.header');

console.log(h1ClassHeader); // Output: NodeList [h1.header]

console.log(h1ClassHeader[0]); // Output: <h1 class="header">Hello, World!</h1>

let idHeader2 = document.querySelectorAll('#header2');

console.log(idHeader2); // Output: NodeList [p#header2]

console.log(idHeader2[0]); // Output: <p id="header2">Selamat Datang di Skilvul</p>

Penjelasan kode di atas:

* Ketika kita memanggil document.querySelectorAll("h1.header"), document sebagai akar objek dari halaman html kita akan mencari semua element <h1> dengan attribute class bernilai header di bawahnya. Begitu juga ketika memanggil document.querySelectorAll('#header2'), document akan mencari semua element dengan id bernilai header2 di bawahnya.

Di topik Pengenalan JavaScript HTML DOM, kita tahu bahwa salah satu kegunaan dari DOM adalah untuk berinteraksi dengan HTML Event di suatu website.

Pernahkan ketika kita klik 1 tombol di satu website, terkadang akan muncul popup window yang menyampaikan suatu pesan. Salah satu cara untuk melakukan itu adalah dengan menggunakan **DOM Event**.

Ada banyak sekali tipe HTML Event di JavaScript. Kali ini kita hanya akan membahas onclick yaitu HTML Event yang akan terjadi ketika user klik suatu element HTML.

Syntax-nya adalah sebagai berikut:

onclick = kodeJavaScript

**Contoh Penggunaan**

Seandainya kita mempunyai satu <button> yang akan menampilkan popup window dengan pesan "Hello, World!", maka caranya kira-kira seperti ini:

<!-- html -->

<button id="demo">Click Me!<button>

// js

let demo = document.getElementById("demo");

demo.onclick = showMessage;

function showMessage() {

alert("Hello, World!");

}



Penjelasan kode di atas:

* Di baris demo.onclick = showMessage;, kita menambahkan event onclick ke element <button id="demo"> dan memberi nilai showMessage yang merupakan fungsi untuk menampilkan alert dengan tulisan "Hello, World!".
* Jadi ketika <button id="demo"> di-klik, maka alert window dengan tulisan "Hello, World!" akan muncul, sesuai dengan gambar di atas.

📝**Catatan**:

Contoh di atas adalah salah satu dari banyak syntax untuk menambahkan onclick event di JavaScript.

Masih ada banyak sekali DOM Property, Method, dan Event lainnya di JavaScript. Kita akan bahas fungsi DOM lainnya di kelas JavaScript berikutnya.