



HTML DOM

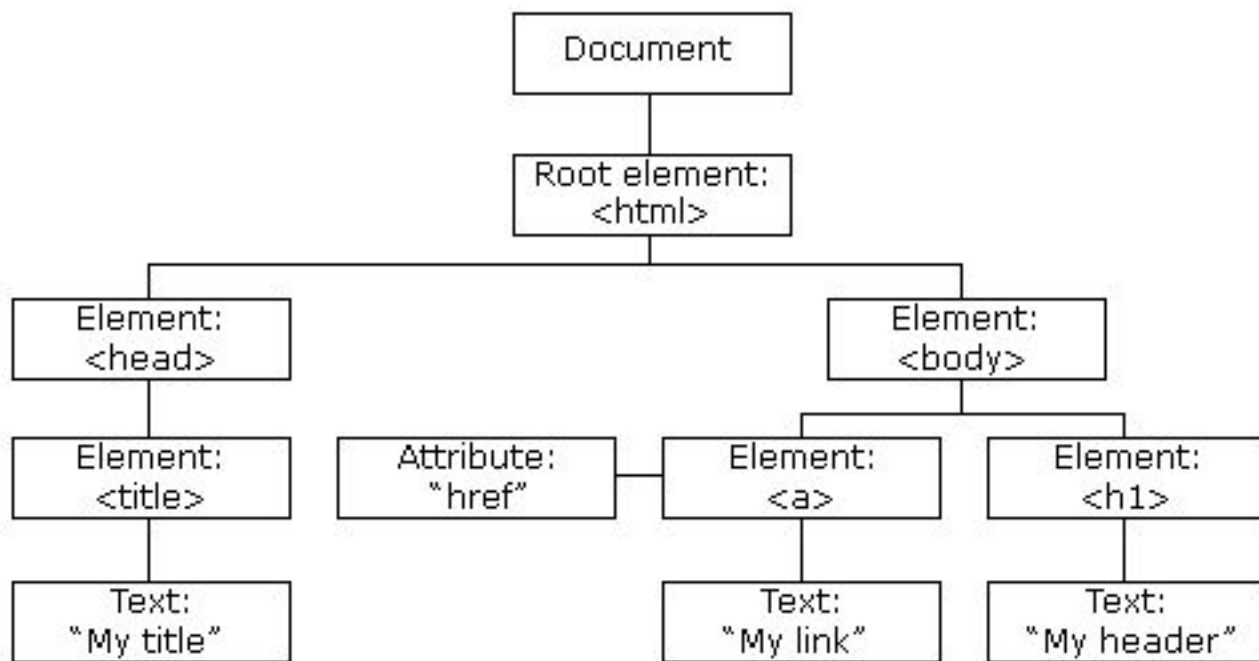


Apa itu DOM?

Merupakan representasi data dari objek yang menyusun struktur dan dokumen di web. DOM ini merupakan *programming interface* untuk *web documents*, yang memperbolehkan kita untuk mengubah struktur dokumen, seperti menambahkan *element*, mengubah isi *element*, mengganti *style* pada *element*, dan lainnya. Sederhananya, DOM ini lah yang akan membuat sebuah bahasa pemrograman dapat berinteraksi dengan halaman HTML kita.

Ketika sebuah halaman dimuat, *browser* akan membuat *Document Object Model* dari halaman tersebut.

Ketika sebuah halaman dimuat, browser akan membuat Document Object Model dari halaman tersebut.



Ilustrasi pohon *Document Object Model*
Sumber: [W3Schools](https://www.w3schools.com)



DOM dan JavaScript

Dalam materi ini, digunakan bahasa *JavaScript* sebagai bahasa pemrograman yang akan berinteraksi dengan DOM. Perlu diingat, bahwa DOM bukanlah bahasa pemrograman, dan juga bukan merupakan bagian dari *JavaScript*. DOM merupakan *Web API* yang digunakan untuk membangun *website*.



Mengakses DOM

Kita tidak perlu cara khusus untuk mengakses *DOM*. Ketika kita menuliskan *code* didalam tag `<script>`, langsung menuliskan *code* pada *element* HTML, maupun pada *script external* yang kita buat secara terpisah.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>HTML DOM</title>
  </head>
  <body onload="console.log('Hello world')"></body>
</html>
```

Code pada index.html

Attribute `onload` aktif ketika suatu object telah dimuat.



Internal Script

Kita dapat mengakses *DOM* menggunakan *internal script* yang kita kurung dalam *element* `<script>`, didalam *element* `<body>`.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>HTML DOM</title>
  </head>
  <body onload="console.log('hello world')">
    <script>
      console.log('hello world from script');
    </script>
  </body>
</html>
```



Code pada index.html

External Script

Sama seperti CSS, kita dapat menuliskan *script* kita diluar *file HTML*, dengan menggunakan *element* `<script>`, dan mendefinisikan *source code* script kita.

Code pada index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>HTML DOM</title>
  </head>
  <body onload="console.log('hello world')">
    <script src="./index.js"></script>
    <script>
      console.log('hello world from script');
    </script>
  </body>
</html>
```

```
alert('Hello from external script');
```

Code dalam index.js



Core Interfaces dalam DOM

Object document dan *window* merupakan *interfaces* yang paling sering digunakan dalam DOM. Sederhananya, *window object* merupakan representasi dari *browser*, dan *document object* merupakan *root* dari *document* itu sendiri. Berikut beberapa *list API* yang ada pada *web*;

- [Document.querySelector\(\)](#)
- [EventTarget.addEventListener\(\)](#)
- [Node.textContent](#)

**list hyperlink diatas, wajib dibuka, dan dibaca jika ingin tahu lebih dalam*



Document: `querySelector()` method

Method `querySelector()` akan me-returns *element* pertama di dalam dokumen yang cocok dengan CSS *selector* yang ada pada *element HTML*.

Syntax: → `querySelector("selectors")`

- *class selector* :
 - `.className`
- *id selector*:
 - `#idName`

```
const printText = document.body.querySelector('#print-text');
```

→ Code dalam index.js

* kita harus memiliki sebuah tag html yang memiliki id "print-text"



EventTarget: addEventListener() method

Method `addEventListener()` dari interface `EventTarget` menyiapkan *function* yang akan dipanggil setiap kali *event* yang ditentukan dikirimkan ke *target*.

Syntax → `addEventListener(type, listener)`

Dokumentasi lengkap ada di;

- [EventTarget: addEventListener\(\)](#)

```
printText.addEventListener('click', () => {  
  alert('Clicked');  
});
```

→ Code dalam index.js

* maksud dari *code* diatas adalah, menambahkan *eventListener* ke *node* `printText` yang sudah ada pada slide 9



Node: textContent property

Properti *textContent* dari *interface Node* me-representasikan konten teks dari *node* dan turunannya.

```
<p id="printed">Printed element</p>
```

→ Code dalam index.html

```
const printed = document.querySelector('#printed');  
console.log(printed.textContent);
```

→ Code dalam index.js



Task


1. Buat sebuah *element* yang ketika di *click* akan menambahkan teks “Printed text” ke dalam *element* lain.



Form Validation

Merupakan sebuah istilah yang kita gunakan untuk memvalidasi sebuah *form*(elemen *input*). Contohnya, ketika kita tidak ingin sebuah elemen *input* kosong, disinilah kita membuat sebuah *form validation*.

Untuk melanjutkan pembelajaran ini, pembaca wajib membaca dan menulis *code* pada [link berikut](#).




Sederhananya, *form validation* ini “hanya” melakukan pengecekan terhadap *value* dari sebuah *input* sebelum melakukan *submit*(mengirimkan data ke-server) misalnya.

```
<form action="" name="sign-up-form">
  <label for="first-name">First Name</label>
  <input
    type="text"
    value="John Doe"
    placeholder="Masukkan Nama"
    id="first-name"
    name="first-name"
  />
  <p id="error-first-name" class="error"></p>
  <p id="success-first-name" class="success"></p>
</form>
```

Logikanya adalah, kita membuat “penampung” untuk menampilkan *error* maupun *success message* yang ingin ditampilkan. Nantinya, si penampung inilah yang akan kita masukkan pesan *error* maupun *success* nya.

Code pada index.html



```
const signUpForm = document.forms['sign-up-form'];
const firstNameInput = signUpForm['first-name'];
const errorFirstName = document.querySelector('#error-first-name');
const successFirstName = document.querySelector('#success-first-name');
```

Code pada index.js

- *signUpForm*
Menyimpan object form dengan name "sign-up-form"
- *firstNameInput*
Menyimpan object didalam sign-up-form dengan name "first-name"
- *errorFirstName*
Menyimpan object dengan id "error-first-name"
- *successFirstName*
Menyimpan object dengan id "success-first-name"

```
signUpForm.onsubmit = (event) => {  
  event.preventDefault();  
  firstNameInput.value.trim() === ''  
    ? (errorFirstName.innerText = 'First name tidak boleh kosong')  
    : (successFirstName.innerText = 'Oke, input diterima');  
};
```

Code pada index.js

- *onSubmit*
Argument event ini merupakan sebuah object yang berisikan informasi tentang "event".
- *preventDefault()*
Ini merupakan *method* dari *event* untuk mencegah tindakan default, yaitu adalah mengirim data ke server dan melakukan refresh halaman.