1. **Cara JavaScript Mengontrol Website dan Browser**

Setelah membaca materi sebelumnya, maka akan muncul pertanyaan “Bagaimana JavaScript mengontrol sebuah website atau browser?” Jawabannya ada dua, dan keduanya akan kita bahas di modul ini, yakni melalui Browser Object Model (BOM) dan Document Object Model (DOM).

Yang pertama adalah BOM. Dengan BOM, kita dapat memberikan perintah-perintah khusus ke browser, misalnya melalui sebuah ‘atribut’ khusus milik browser yakni ‘window’ (akan kita membahasnya pada materi selanjutnya), sehingga kita bisa membuat browser menampilkan pesan pop-up. Caranya yakni menjalankan method alert() pada console milik browser.

Selain method alert(), objek window juga memiliki method-method lainnya, seperti prompt, console, dsb. Tenang, kita akan berjelajah lebih jauh pada modul-modul selanjutnya.

Cara kedua adalah DOM. DOM sama seperti BOM. Perbedaannya adalah kita menggunakan global objek bernama document. Melalui global objek ini, kita bisa menangkap seluruh elemen dalam dokumen HTML guna memanipulasi konten HTML melalui method getElementById(). Method ini akan menangkap elemen berdasarkan value dari atribut id.

1. Browser Object Model (BOM)

Anggota BOM

* Method Alert()

Method alert() merupakan salah satu method BOM yang menerima satu parameter opsional berupa data string yang akan ditampilkan melalui sebuah pop-up browser. Contohnya jika kita ingin menampilkan sebuah pesan berisi string "Halo user!" menggunakan console browser, maka caranya adalah sebagai berikut:

alert('Halo user!');

* Prompt()

Method ini akan menampilkan sebuah dialog browser yang meminta user untuk mengisi kolom inputnya. Karena bisa menangkap input dari user, method ini memiliki lebih banyak fungsionalitas ketimbang alert().

* Nilai Return dari prompt()

Jika kita lihat, sebenarnya method alert() mengembalikan sebuah value berupa undefined. Namun, prompt() akan mengembalikan sebuah nilai sesuai dengan inputkan oleh user. Sebagai contoh, mari jalankan kode berikut pada console browser.

let pesanInput = prompt('Masukkan pesan sesukamu...');

Ada satu hal yang perlu kita ingat selama menggunakan prompt(), yakni nilai apa pun yang dimasukkan oleh user akan diproses dan dikembalikan menjadi data string.

* Console

Berikut beberapa method dari objek console yang umum digunakan:

|  |  |
| --- | --- |
| Method | Penjelasan |
| log() | Menampilkan pesan secara umum ke console browser. |
| info() | Menampilkan pesan yang berisi informasi penting ke console browser. |
| warn() | Menampilkan pesan yang berisi informasi dalam bentuk peringatan ke console browser. |
| error() | Menampilkan pesan yang berisi informasi dalam bentuk error ke console browser. |

1. Document Object Model (DOM)

Apa sih DOM itu? Pada lingkungan browser, DOM memberikan izin bagi kode JavaScript untuk mengakses dan memanipulasi konten pada dokumen melalui sebuah Application Programming Interface (API), yakni sesuatu yang memungkinkan JavaScript dan dokumen HTML untuk "berkomunikasi". Masih ingat Browser Object Model yang diwakilkan oleh objek window yang bisa kita masukkan ke kode JavaScript? Nah, DOM diwakilkan oleh global objek bernama document, yang mana dapat dimanfaatkan oleh JavaScript.

* DOM Tree

Tree yang dimaksud di sini adalah nama dari sebuah struktur data pada komputer yang secara visual mirip seperti sebuah pohon. Struktur data ini disebut tree, karena layaknya pohon terdapat satu batang induk tunggal yang kemudian bercabang menjadi batang-batang lainnya dan bisa saja bercabang kembali. Jika batang tersebut buntu, maka ujungnya terdapat daun (pada struktur data tree, daun disebut sebagai node). Pada berkas HTML, batang induk tunggal adalah elemen <html>, sedangkan cabang-cabangnya adalah elemen-elemen yang terdapat di dalamnya. Misalkan kita mempunyai berkas HTML dengan struktur HTML berikut.

<!DOCTYPE html>

<html>

<head>

<title>DOM Tree</title>

</head>

<body>

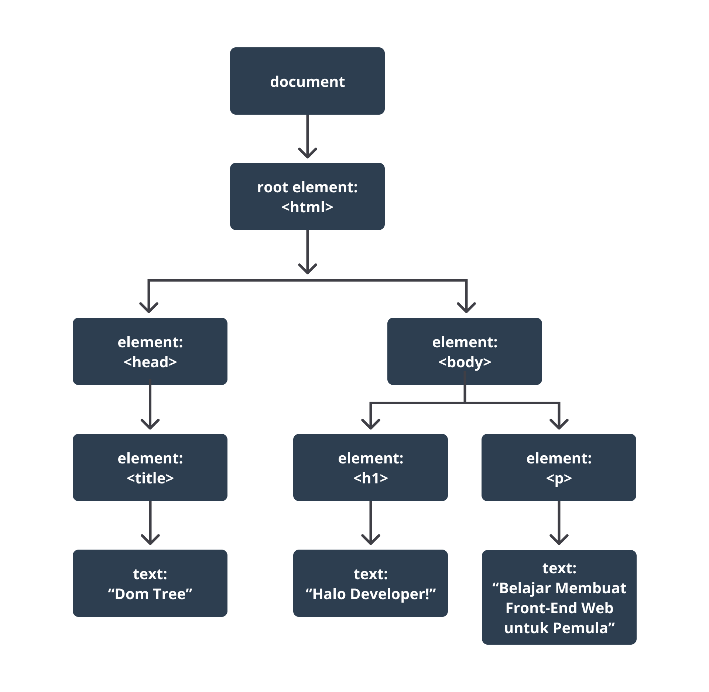
<h1>Hello Developer Front-End Web!</h1>

<p>Belajar Membuat Front-End Web untuk Pemula</p>

</body>

</html>

Jika kita buat berkas HTML di atas menjadi dalam bentuk DOM, strukturnya akan menjadi seperti berikut.



Pada contoh yang diberikan di atas, DOM memiliki bentuk struktur data tree yang dibuat berdasarkan berkas HTML di atas. Struktur data tree di atas inilah yang akan direpresentasikan dalam bentuk global obyek bernama document nanti.

Walaupun struktur dari DOM terbentuk berdasarkan isi dari berkas HTML, tetapi ada beberapa skenario tertentu yang menyebabkan struktur DOM berbeda dengan struktur elemen-elemen dalam berkas HTML. Salah satu skenarionya adalah jika terdapat kesalahan penulisan dalam berkas HTML. Mari kita lihat contohnya seperti berikut:

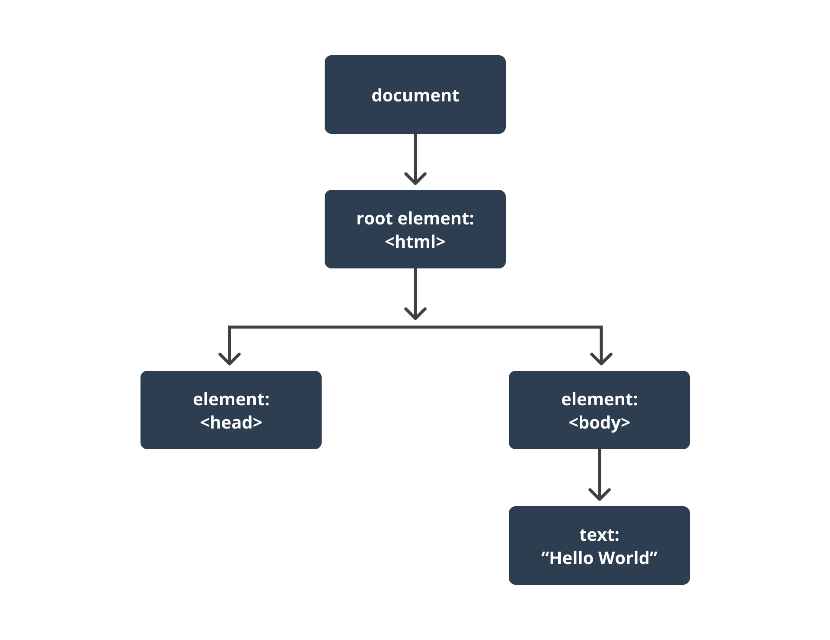
<!DOCTYPE html>

<html>

Hello World!

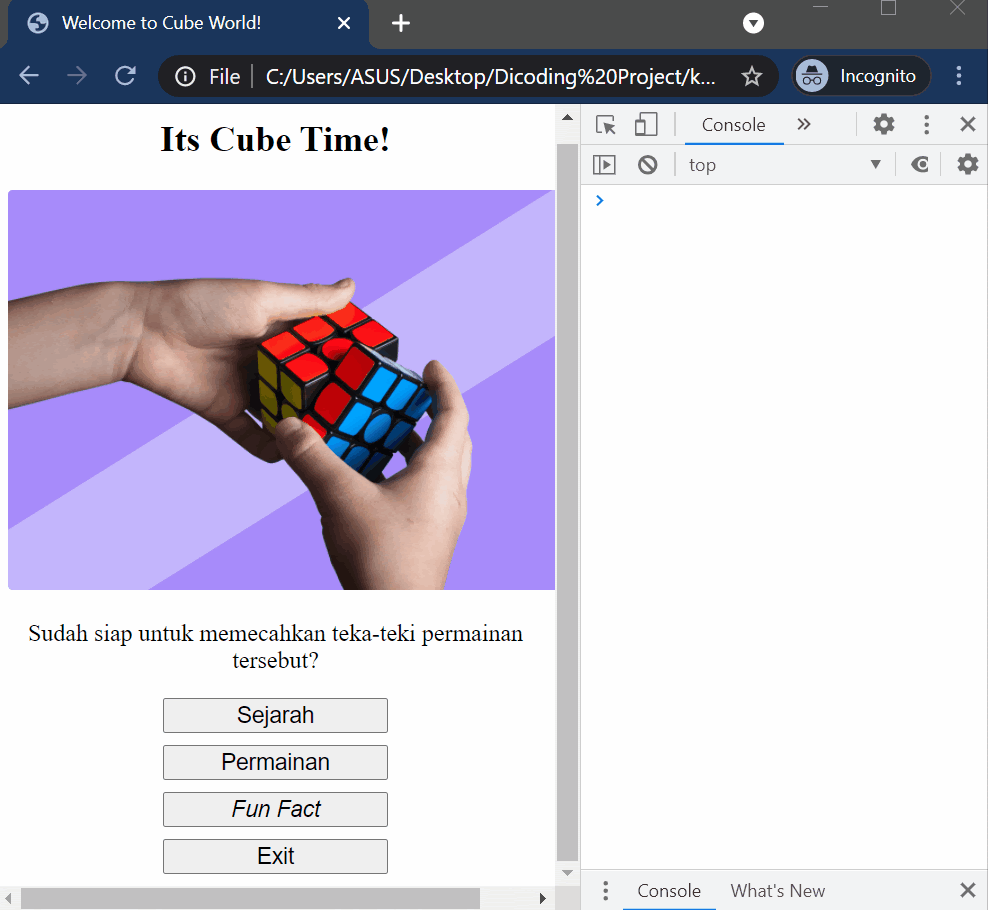
</html>

Pada berkas HTML di atas, jelas terlihat bahwa tidak ada elemen dengan tag <head> maupun <body> yang mengakibatkan berkas HTML tersebut tidak valid. Walaupun berkas HTML tersebut tidak valid pada bentuk DOM-nya, object structure-nya akan diperbaiki. Bagaimana caranya? Caranya yakni elemen dengan tag <head> maupun <body> akan ditambahkan secara otomatis kemudian teks “Hello World!” ditempatkan di elemen <body>.



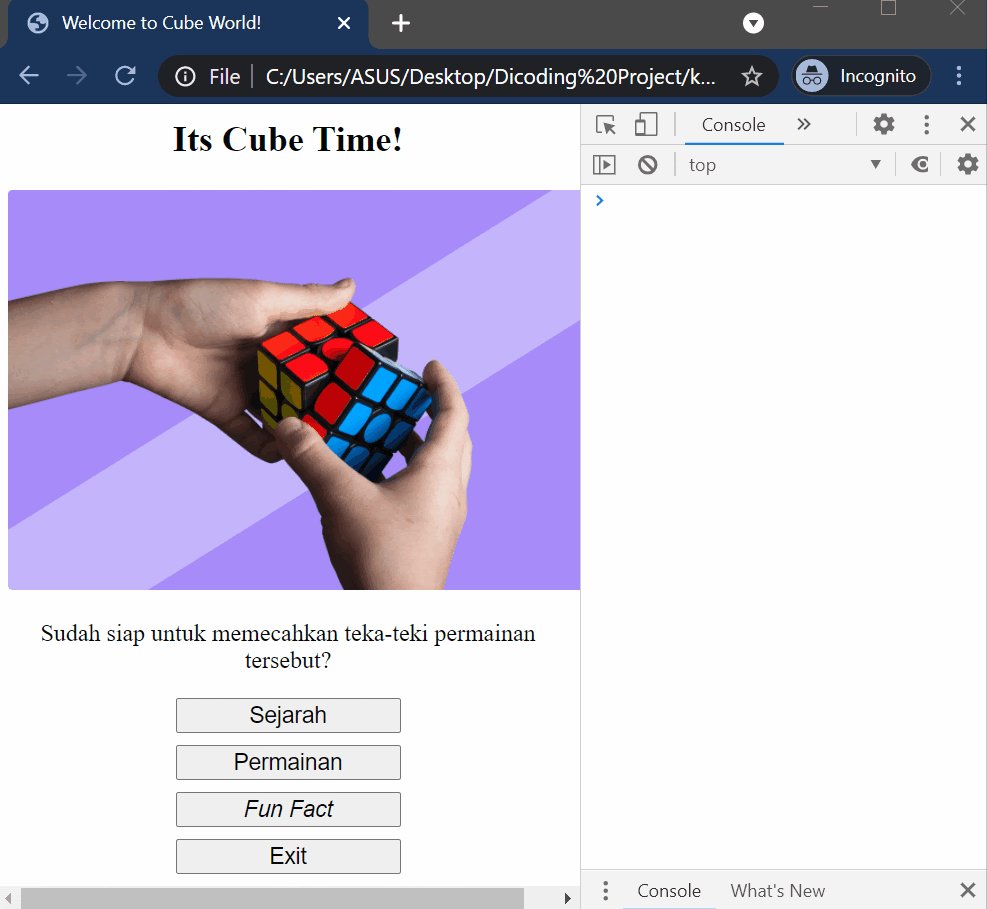
* Mencari DOM (Mendapatkan DOM)

Untuk mengakses sebuah elemen HTML melalui DOM, kita gunakan objek yang bernama document. Objek ini berisi semua elemen yang berada pada HTML atau keseluruhan halaman yang terlihat pada jendela browser saat itu. Jika kita coba mengakses nilai dari global obyek document pada console browser maka tampilannya akan sebagai berikut:

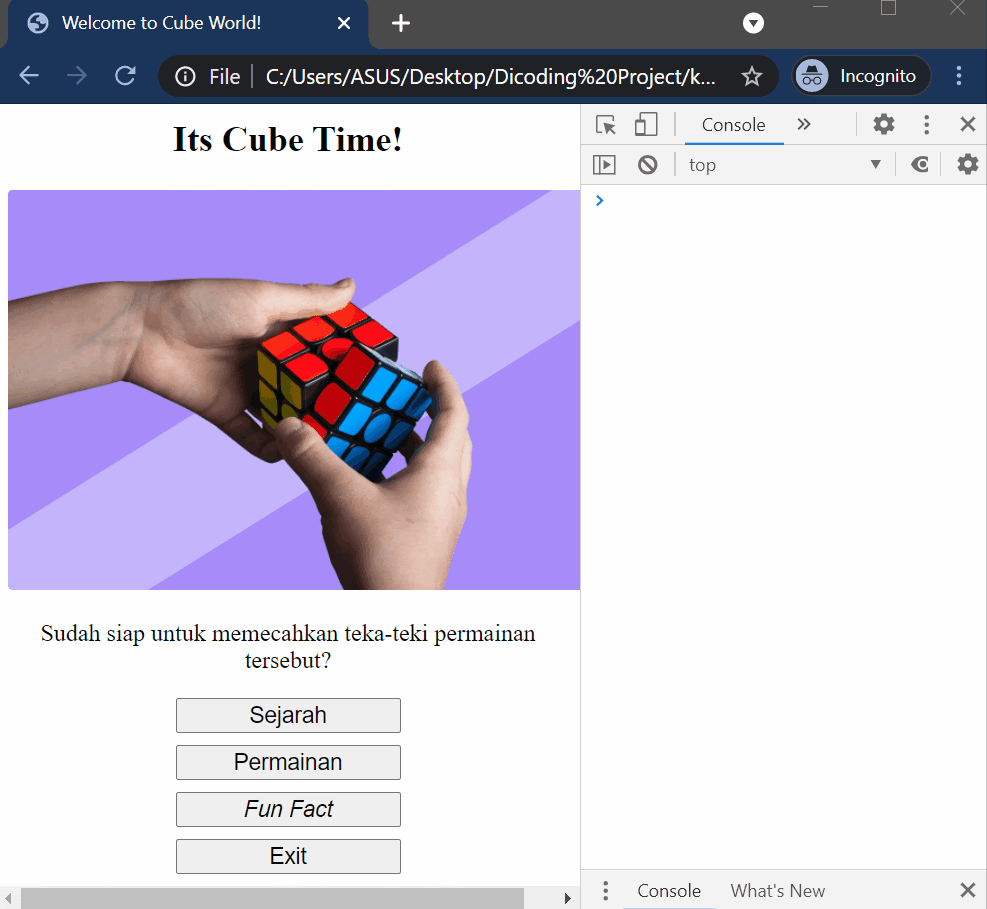


Cukup dengan objek document, kita dapat mengakses keseluruhan konten pada HTML yang terdapat root element <html>. Sehingga, jika dicermati kembali di dalamnya terdapat elemen lainnya seperti <head> dan <body>.

Secara sekilas isi dari objek document hanyalah terlihat seperti isi dari sebuah berkas HTML saja. Namun, layaknya sebuah objek JavaScript, objek document memiliki beberapa properti dan method yang bisa kita gunakan untuk mengakses koleksi elemen maupun elemen tertentu. Misalnya, mendapatkan konten-konten pada elemen dengan tag <head> melalui objek document dengan kode berikut :

const head = document.head;

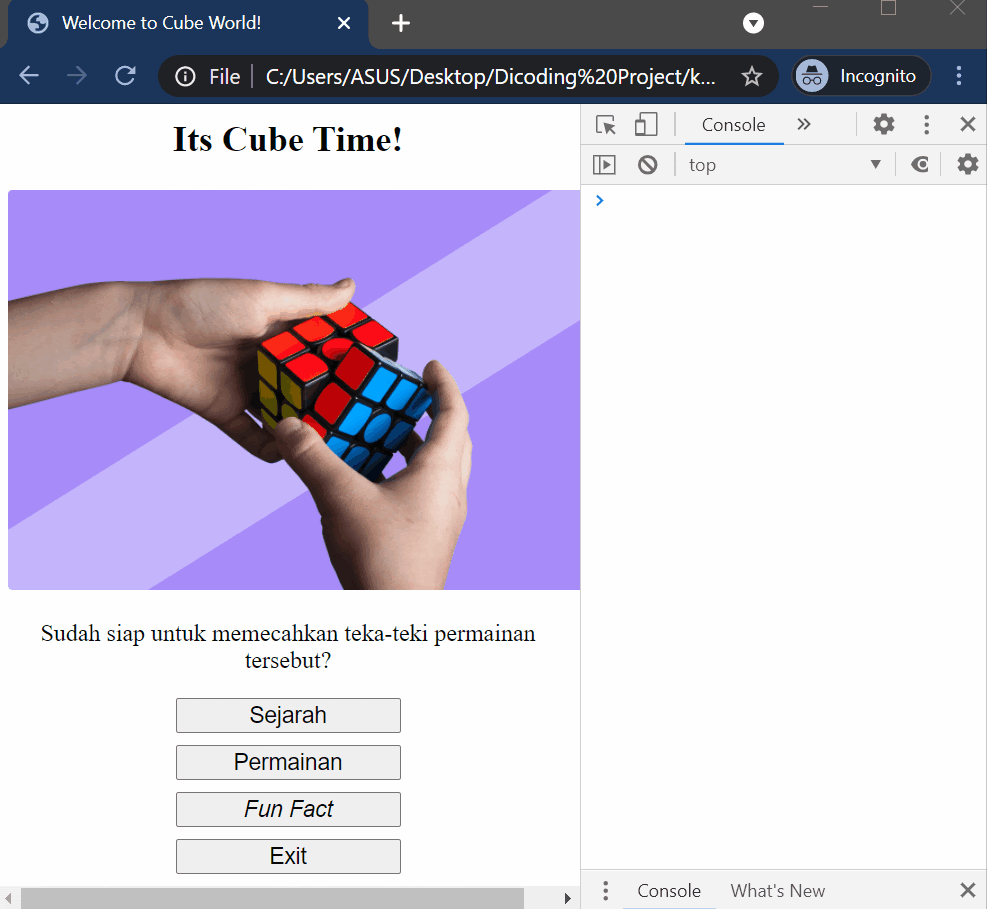
Bagaimana jika kita ingin mendapatkan elemen <body> saja? Caranya hampir sama, yakni dengan kode berikut:

const body = document.body;

Kedua properti yakni head dan body akan mengembalikan elemen yang sama seperti document yakni sebuah objek.

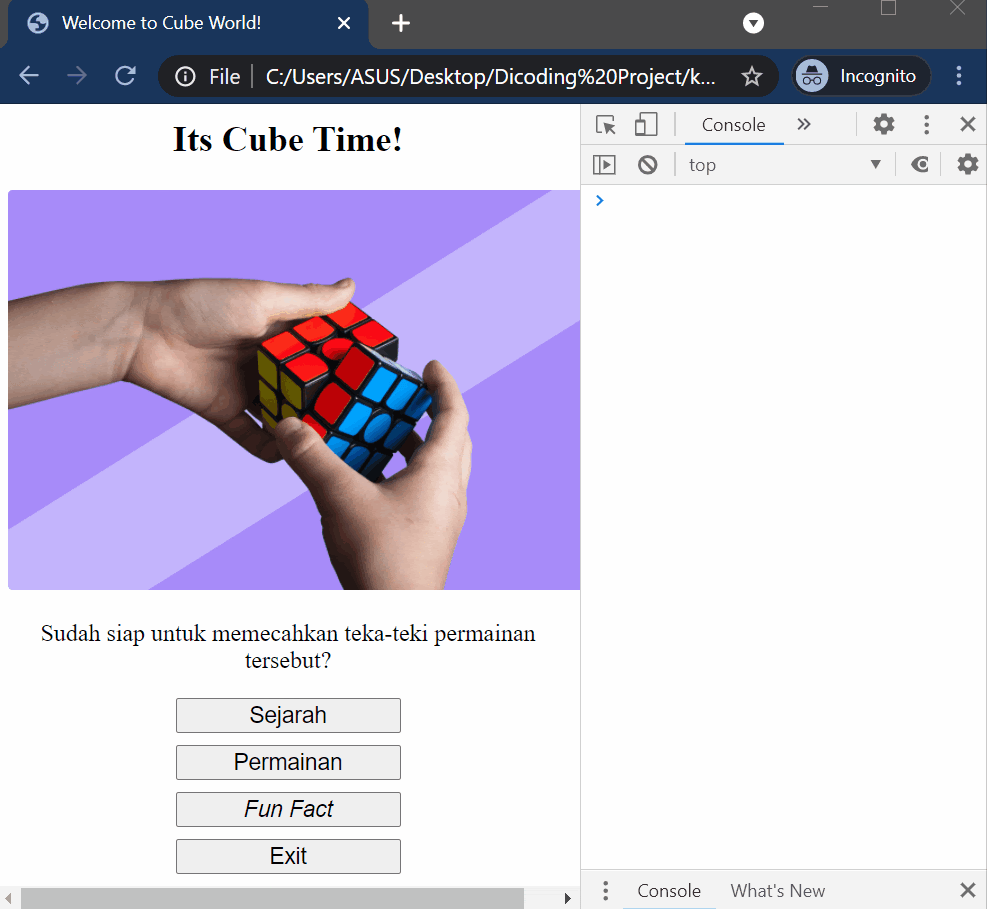
Bagaimana jika kita ingin lebih spesifik lagi? Misalnya sebuah elemen tunggal? Kita bisa menggunakan method-method yang tersedia oleh objek document. Salah satu method yang bisa kita gunakan adalah getElementById(). Method tersebut berfungsi untuk mendapatkan elemen berdasarkan nilai id-nya. Pada contoh di atas, elemen gambar memiliki id "gambarUtama". Jika kita ingin mengakses elemen tersebut, caranya sebagai berikut:

const gambar = document.getElementById('gambarUtama');



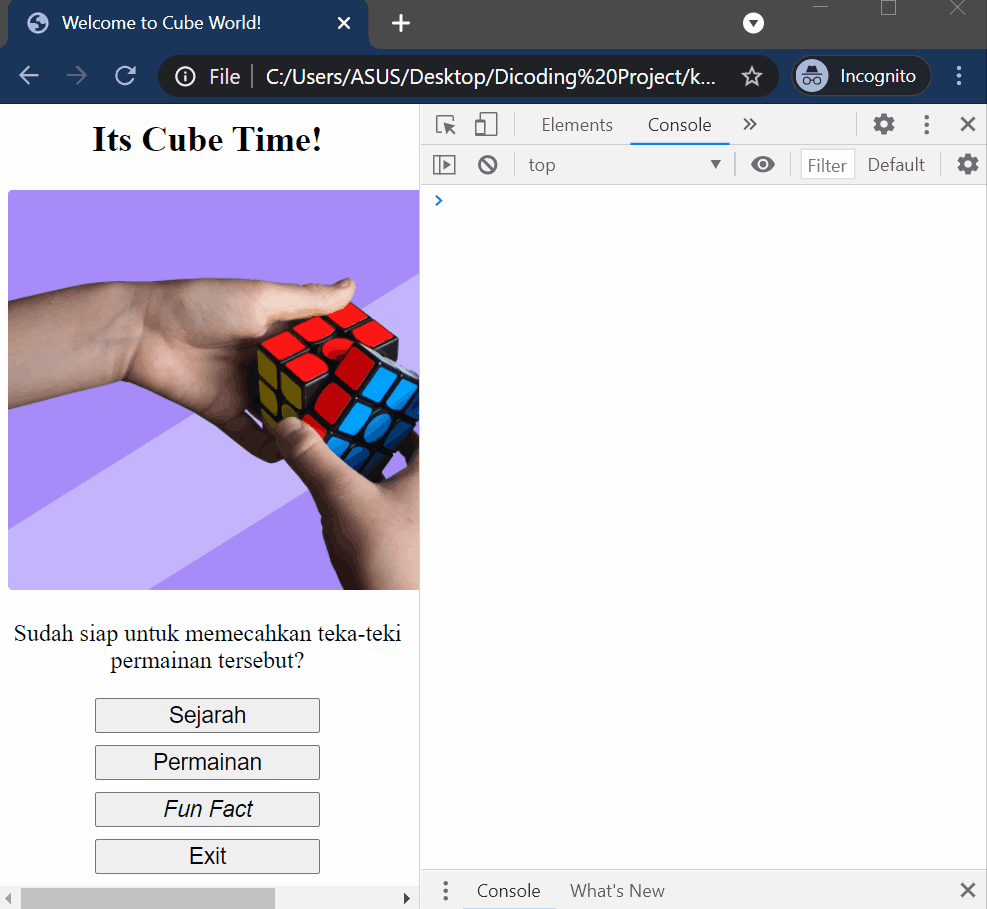
| **Method** | **Contoh** | **Fungsi** |
| --- | --- | --- |
| getElementById() | document.getElementById('display'); | Mengembalikan satu elemen yang memiliki nilai id "display". |
| getElementsByName() | document.getElementsByName('button') | Mengembalikan banyak elemen (HTMLCollection) yang memiliki attribute name dengan nilai "button". |
| getElementsByClassName() | document.getElementsByClassName('button') | Mengembalikan banyak elemen (HTMLCollection) yang memiliki attribute class dengan nilai "button". |
| getElementsByTagName() | document.getElementsByTagName('div') | Mengembalikan banyak elemen (HTMLCollection) yang merupakan <div> element. |
| querySelector() | document.querySelector('.button') | Mengembalikan elemen pertama (node) yang menerapkan class "button". |
| querySelectorAll() | document.querySelectorAll('.button') | Mengembalikan banyak Node dalam bentuk NodeList yang menerapkan class "button". |

Objek document juga memiliki banyak sekali method yang tersedia, berikut beberapa method yang umum digunakan serta fungsinya:

Pada method di atas ada yang mengembalikan nilai HTML elemen secara tunggal. Selain itu, ada juga yang mengembalikan banyak elemen dari sebuah berkas HTML yang biasa disebut sebagai HTMLCollection. Dikarenakan semua method di atas dimiliki oleh objek document, maka jangan lupa mengawali semua pemanggilan method-method di atas dengan sintaks document.<nama\_method> ya. Misalnya kita bisa mendapatkan elemen paling pertama yang menerapkan nilai "button" pada atribut class dengan method querySelector('.button').

Sedangkan untuk mendapatkan semua elemen yang nilai atribut class-nya "button", gunakan method querySelectorAll('.button'). Method tersebut akan mengembalikan semua elemen yang sesuai dalam bentuk HTMLCollection.

Ada fakta menarik tentang HTMLCollection, yakni memiliki karakteristik yang mirip dengan array. Contohnya, kita bisa menggunakan properti length untuk mendapatkan jumlah elemen yang terdapat di dalamnya. Selain itu, kita bisa mengakses nilai individual elemennya menggunakan indexing.

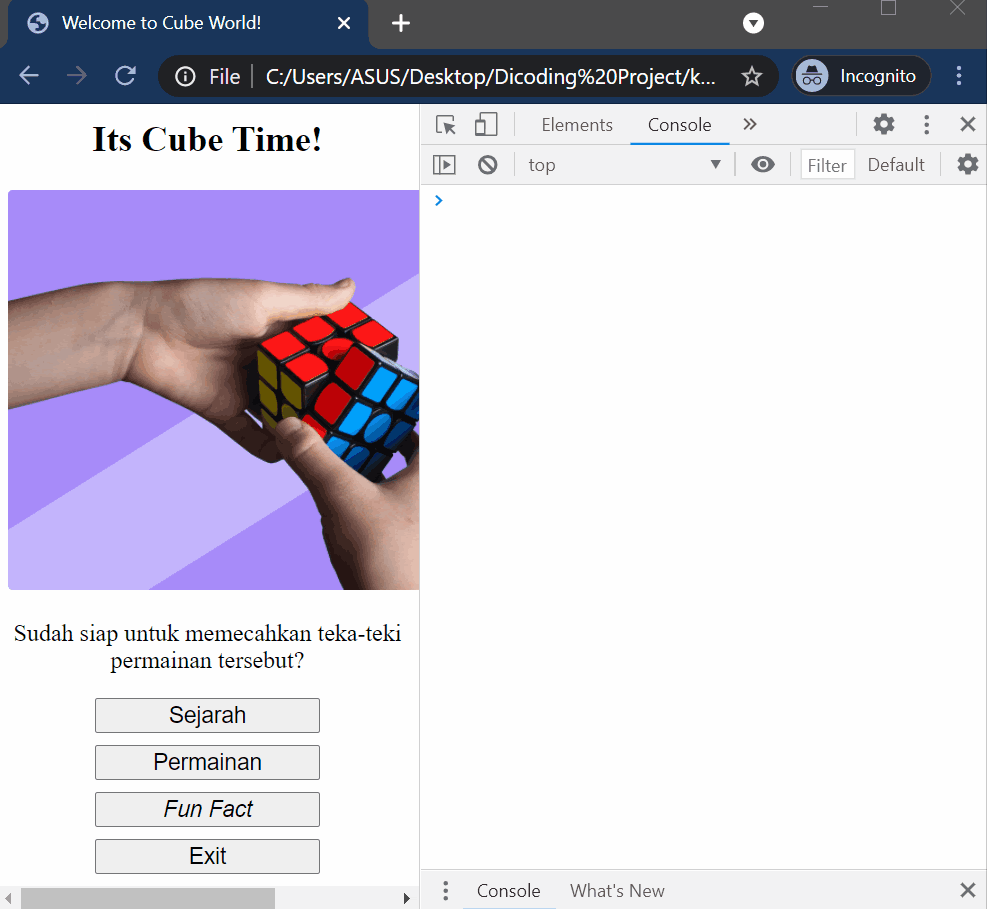


Karena HTMLCollection memiliki karakteristik yang mirip dengan array, maka kita juga bisa melakukan looping terhadap elemen-elemennya, yakni melalui sintaks looping for of. Berdasarkan contoh di atas, maka implementasinya sebagai berikut:

for (let item of buttons) {

console.log(item);

}



Sesuai dengan nama method tersebut, method querySelector() dan querySelectorAll() membutuhkan query khusus sebagai parameternya. Jika tujuan kita adalah mendapatkan elemen berdasarkan atribut class, maka parameternya harus diawali dengan tanda titik ("."), sedangkan jika berdasarkan atribut id, kita harus mengawali nilai parameternya dengan tanda pagar ("#"). Hal ini sama seperti ketika kita bekerja dengan selector di CSS.

Cobalah untuk bereksperimen dengan method-method yang sudah dijelaskan sebelumnya untuk mendapatkan pemahaman yang baik dan elemen yang sesuai. Kemudian carilah perbedaan dari fungsi method-method tersebut.

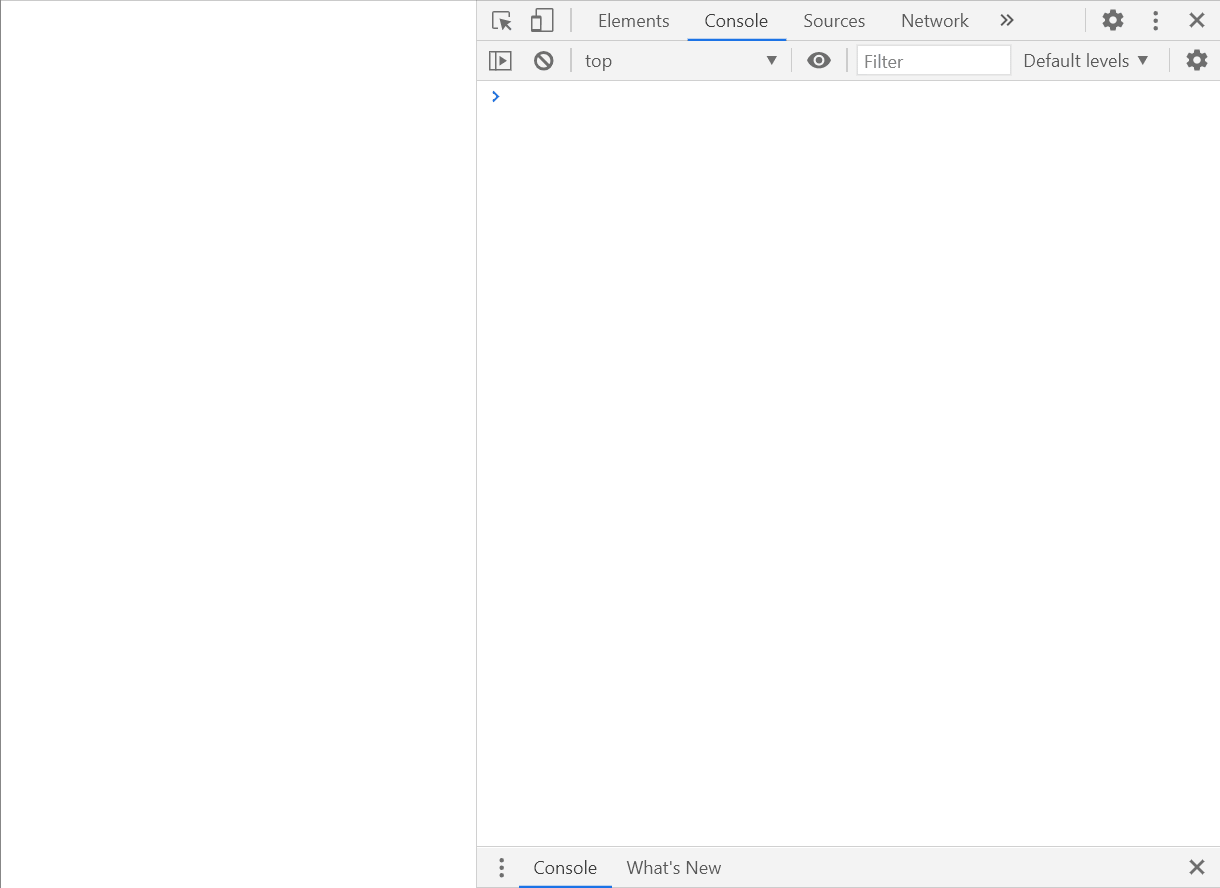
* Membuat Elemen HTML

Manipulasi DOM memberikan kemampuan bagi kita untuk membuat elemen-elemen HTML melalui kode JavaScript. Selain itu, melalui DOM kita mampu membuat konten HTML.

Sekarang mari kita mulai dengan membahas method createElement(). Dengan method tersebut, kita bisa membuat sebuah elemen HTML yang benar-benar baru tanpa memanipulasi isi konten berkas HTML.

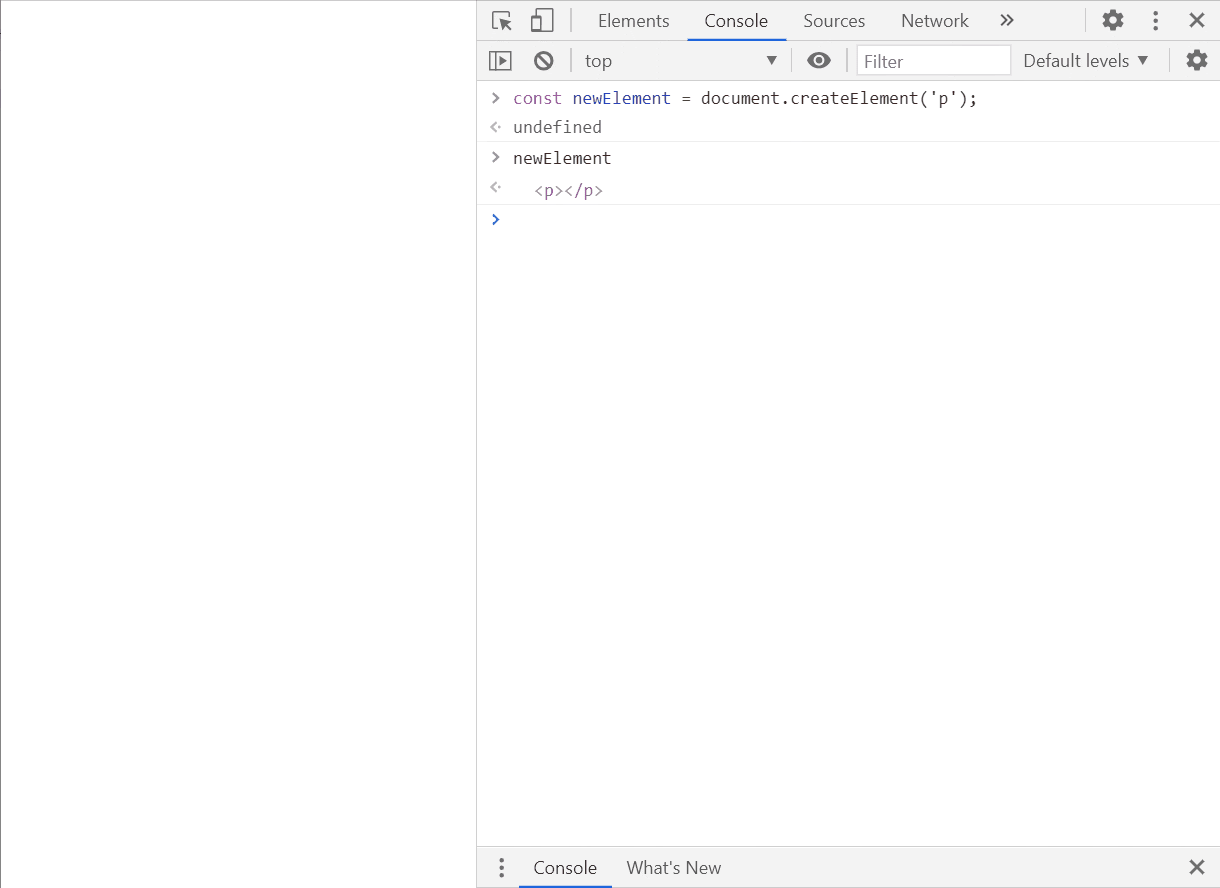
Contohnya, jika kita ingin membuat sebuah elemen HTML dengan tag <p> maka sintaksnya adalah sebagai berikut:

const newElement = document.createElement('p');



Mudah sekali bukan? Tetapi tidak cukup jika hanya membuat elemen baru, bagaimana jika kita menambahkan teks? Caranya adalah kita berikan nilai string yang baru dan berikan pada properti innerText.

newElement.innerText = 'Selamat datang ke HTML kosong ini :)';



Misalnya kita ingin menambahkan tag <b> untuk membungkus kata "Selamat datang", maka kita tulis ulang nilai string dan melakukan assignment menggunakan properti innerHTML.

newElement.innerHTML = '<b>Selamat datang</b> ke HTML kosong ini :)';



Contoh di atas menerapkan createElement() untuk membuat teks melalui tag <p>. Apakah method tersebut juga memungkinkan untuk membuat elemen gambar? Jawabannya adalah sangat mungkin! Berikut ini contohnya.

const newImg = document.createElement('img');