Proses belajar

- Pengenalan DOM (Teori) *document tree*
- Penggunaan 3 Elemen dasar yang umum digunakan
    Targeting
- getElementById()
- querySelector()
- querySelectorAll()
    Manipulating
- createElement()
- textContent
- innerText
- innetHTML
    Applying CSS using DOM
- Event Dasar menggunakan addEventListener()
- Real World Projects :
    - hex-color
    - carousel-image

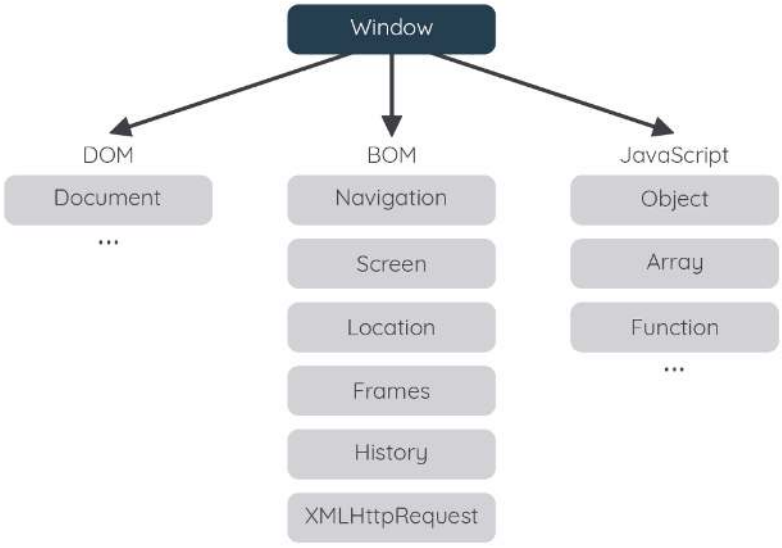
DOM

Definition :

*DOM scripting, which allows us to manipulate the elements, attributes, and text on a page. (*Jenifer Robbins*, Learning Web Design 5th Edition*. Oreilly, 2018.)

DOM: providing JavaScript with a map of all the elements on our page, and providing Us With A Set Of Methods For Accessing Those elements,Their attributes, and Their contents. (Mat Marquis, *Javascript for Web Designers*. A book Apart, 2016)

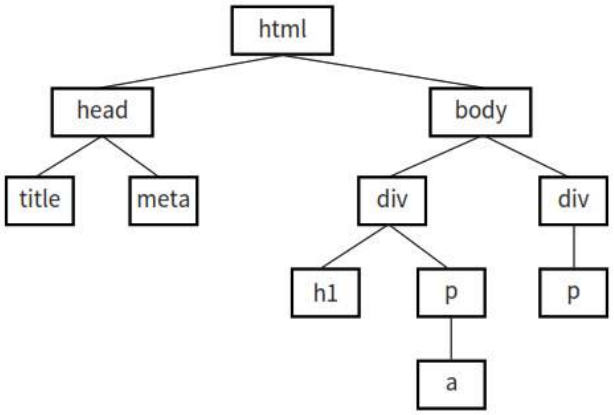
DOM Tree

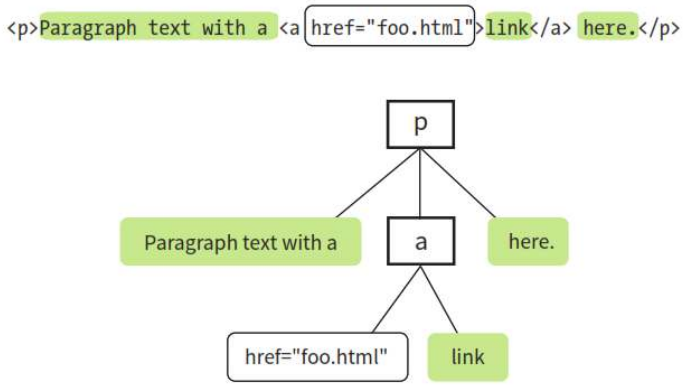


Struktur code :

Ketika kita membuat struktur html maka browser akan mem *converted* ke dalam *object javascript* . Setiap titik disebut node. Banyak node, disebut nodelist.

```
<!DOCTYPE html>
<html>
<head>
  <title>Document title</title>
  <meta charset="utf-8">
</head>
<body>
  <div>
    <h1>Heading</h1>
    <p>Paragraph text with a <a href="foo.html">link</a> here.</p>
  </div>
  <div>
    <p>More text here.</p>
  </div>
</body>
</html>
```





Selection Method

1. **document.getElementById()**

Method tersebut berfungsi untuk mendapatkan elemen berdasarkan nilai *id*-nya. Pada contoh di atas, elemen gambar memiliki *id* "gambar". Jika kita ingin mengakses elemen tersebut, caranya sebagai berikut:

```
const imgId = document.getElementById('gambar');
```

Hasil :

```
> imgId
<- <img id="gambar" src="cube.png" alt="Its a rubiks cube!" width="300" height="21
   5">
```

2. **document.querySelector()**

kita bisa mendapatkan elemen paling pertama yang menerapkan nilai "button" pada atribut *class* dengan *method* ***querySelector(".button")***

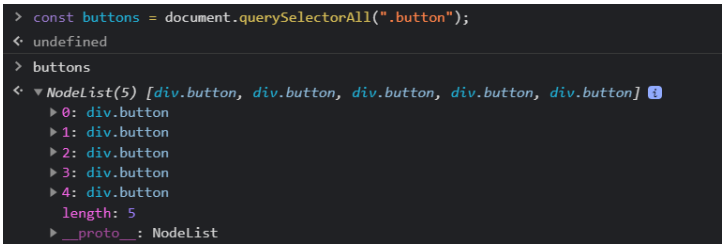
```
document.querySelector(".button");
```

```
> document.querySelector(".button");
<- ▶<div class="button">…</div>
```

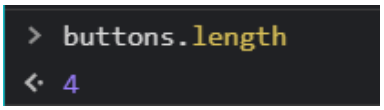
3. **document.querySelectorAll()**

   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.

```
const buttons = document.querySelectorAll(".button");
```



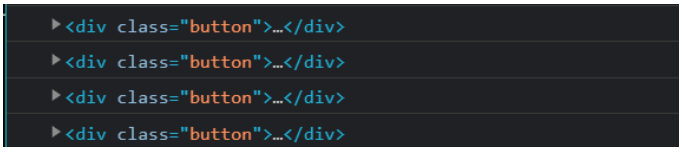
Ada fakta menarik tentang HTMLCollection, yakni memiliki karakteristik yang mirip dengan *array*. Contohnya, kita bisa menggunakan atribut *length* untuk mendapatkan jumlah elemen yang terdapat di dalamnya.



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*.

```
for (let item of buttons) {
   console.log(item);
}
```



Manipulating Element
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 kontek berkas HTML.

1. **createElement()**

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

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

```
> newElement
<-   <p></p>
```

bagaimana jika kita ingin menambahkan teks? Jawabannya terletak pada *assignment* melalui a*tribut innerText.*

```
newElement.innerText = "Hallo, saya senang belajar disini";
```

```
> newElement
<-   <p>Hallo, saya senang belajar disini</p>
```

Misalnya kita ingin menambahkan *tag* tambahan seperti *tag <b>* antara kata "Selamat datang", maka bisa kita tulis ulang pesannya dengan melakukan *assignment* menggunakan *innerHTML*.

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

```
> newElement
<- ▼<p>
     <b>Selamat datang</b>
     " ke HTML kosong ini :)"
   </p>
```

method createElement memungkinkan kita untuk membuat elemen dengan tag HTML lain.contohnya kita akan membuat element *tag <img>*.

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

Kita pun dapat menambahkan attribut element tag yang penting, yaitu src. Melalui method setAttribute yang menerima 2 parameter. Parameter pertama isikan nama attribute, parameter kedua isi dengan nilainya.

```
newElement.setAttribute("src", "https://picsum.photos/200/300");
```

```
> newImg
<-   <img src="https://picsum.photos/200/300">
```

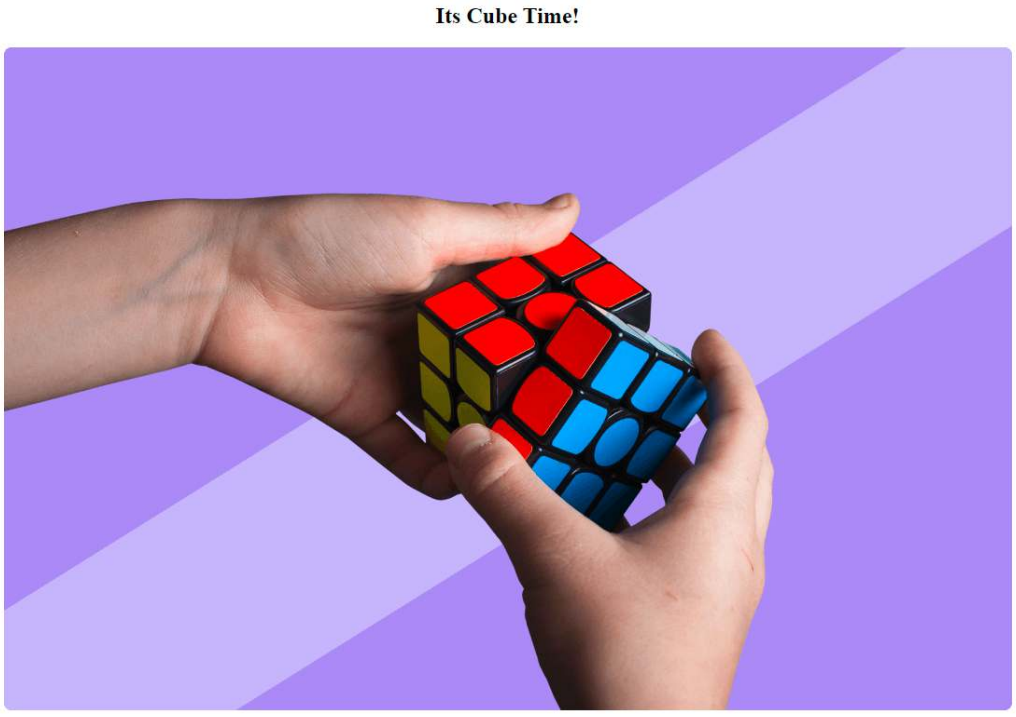
Namun , kita belum melihat hasil apapun pada browser. Itu karena kita belum menambahkan itemnya. Kita akan mempelajari bagaimana cara menambahkan item nanti. Namun sebelumnya, mari kita terapkan pemahaman kita kedalam studi kasus cube. Buka kembali proyek cube nya.

Lalu, apa yang akan kita lakukan selanjutnya. Berikut list yang harus kita kerjakan :

- Mengubah ukuran gambar
- *Disabled* button "play button"

Mari kita praktekan bersama-sama, pertama kita selesaikan dulu list pertama :

Sebelumnya, kita telah melihat pada page html kita, bahwa ukuran gambar terlalu besar.



Pertama kita harus dapatkan element dengan ID 'gambar' :

```
const getImg = document.getElementById('gambar');
```

Sehingga, jika kita tampilkan di console, maka akan muncul seperti berikut :

```
> getImg
<-   <img id="gambar" src="cube.png" alt="Its a rubiks cube!">
```

Kedua, kita atur lebar dan tinggi nya menggunakan method setAttribute();

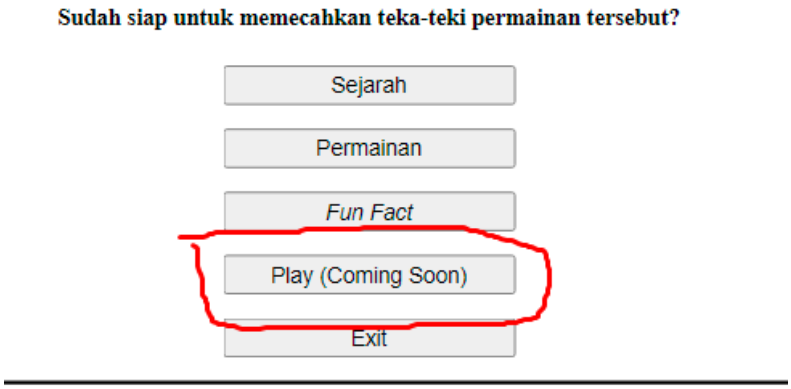
```
getImg.setAttribute("width", 300);
getImg.setAttribute("height", 215);
```

```
> const getImg = document.getElementById('gambar');
<- undefined
> getImg
<-   <img id="gambar" src="cube.png" alt="Its a rubiks cube!">
> getImg.setAttribute("width", 300);
<- undefined
> getImg.setAttribute("height", 215);
<- undefined
>
```

Maka hasilnya seperti berikut. Gambar sudah berhasil kita atur ulang ukurannya :

Its Cube Time!

Oke, sudah cukup sepertinya. Sekarang mari kita kerjakan list kedua, yaitu *disabled* play button.



Pertama kita ambil semua button yang memiliki class (.button) button menggunakan method querySelectorAll();

```
const buttons = document.querySelectorAll(".button");
```

Maka ketika variable dipanggil akan mengembalikan HTMLCollection berupa array.

```
> buttons
< ▶ NodeList(5) [div.button, div.button, div.button, div.button, div.button]
```

Karena yang dikembalikan beruba array. Berarti sifat nya bisa kita terapkan. Sifat array yaitu jika kita ingin mendapatkan nilainya, maka kita harus akses dengan indeks nya. Play button berada di indeks ke 3.

```
const playButton = buttons[3];
```

```
> buttons[3]
< ▼<div class="button">
    <button>Play (Coming Soon)</button>
  </div>
```

Elemen sudah berhasil diakses. Namun kita justru mendapatkan div. elemen yang sebenenarnya ingin kita dapatkan adalah button nya. Tenang, kita ambil gunakan childNodes. Kita buat variabel yang menampung childNodes nya :

```
const playButtonElement = playButton.childNodes[0];
```

Sehingga kita dapat element button nya :

```
> playButtonElement
<-   <button>Play (Coming Soon)</button>
```

Setelah kita mendapatkan element nya. Kita kembali gunakan setAttribute("disabled", "disabled") untuk *disable* Play button.

```
playButtonElement.setAttribute("disabled", "disabled");
```

Selesai, dua list sudah kita kerjakan. Mari lanjut belajar ke topik berikutnya.

Manipulating using *innerText* , *innerHTML, and style property*

Perbedaan mendasar dari keduanya adalah :
innerText : hanya menambahkan *plain text* atau teks biasa dan tidak bisa menambahkan tag html.

innerHTML : menambahkan teks dengan tag nya. Mendukung tag html.

Mari kita manipulasi link dibawah ini :

Situs lainnya yang tidak kalah menarik:

- Wikipedia
- Google

Kita buat lagi pekerjaan kita menjadi dua list berikut ini :

- Mengambil id dari link wiki lalu melakukan perubahan teks menggunakan innerText.
- Hal yang sama diatas. Namun menggunakan innerHTML.

- Styling button dengan style property

Kita kerjakan list pertama. Pertama kita ambil id dari link wikipedia anchor.

```
const wiki = document.getElementById('wikiLink');
```

Lalu kita tambahkan innerText sebagai berikut :

```
wiki.innerText = "Belajar menambah wawasan dunia dengan wiki";
```

Maka hasilnya sebagai berikut :

Situs lainnya yang tidak kalah menarik:

- Belajar menambah wawasan dunia dengan wiki
- Google

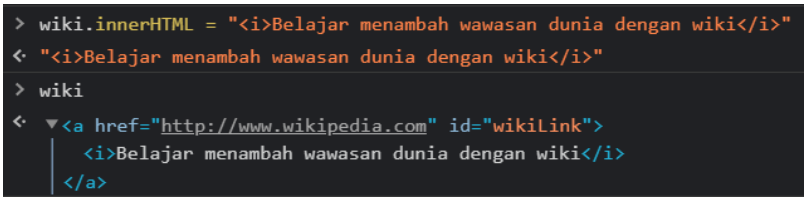
Selanjutnya , kita akan mengerjakan list kedua. Yaitu mengubah text menggunakan innerHTML.

```
wiki.innerHTML = "<i>Belajar menambah wawasan dunia dengan wiki</i>"
```

Maka hasilnya seperti berikut :

Situs lainnya yang tidak kalah menarik:

- *Belajar menambah wawasan dunia dengan wiki*
- Google



Selanjutnya kita akan styling menggunakan style.property

Pertama , kita ambil seluruh element button yang mempunyai class button.

```
const buttons = document.getElementsByClassName("button");
```

Kedua, karena getElementByClassName return HTMLCollection, kita bisa menggunakan perulangan for of karena bersifat array

```
for(let item of buttons) {
    console.log(item)
}
```

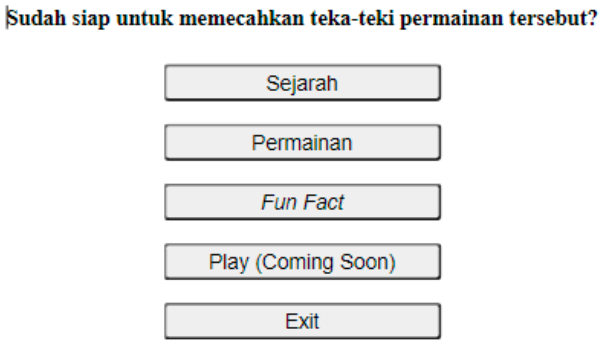
Jika sudah maka hasilnya seperti berikut :

```
▶ <div class="button">…</div>  VM5837:1
▶ <div class="button">…</div>  VM5837:1
▶ <div class="button">…</div>  VM5837:1
▶ <div class="button">…</div>  VM5837:1
▶ <div class="button">…</div>  VM5837:1
```

Lalu kita gunakan lagi perulangan for of untuk apply styles ke tiap button.

```
for (let button of buttons) {
    button.childNodes[0].style.borderRadius = "4px";
}
```

Maka hasilnya sebagai berikut :



Adding HTML to Document

Melalui DOM, kita sudah belajar bagaimana cara membuat konten HTML dan memanipulasi konten HTML sehingga dapat "berubah bentuk". Namun, bagaimana jika kita ingin menambahkan elemen HTML yang benar-benar baru? Pada materi ini kita akan mempelajarinya melalui 2 *method* yakni *appendChild()* dan *insertBefore()*.

Sebelumnya , mari kita tambahkan script berikut :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Memasak Air</title>
    </head>
    <body>
        <p id="name">Langkah-langkah memasak air</p>
        <ol id="daftar">
            <li id="awal">Masukkan air ke dalam wadah.</li>
```

```
            <li id="akhir">Matikan kompor.</li>
        </ol>
        <hr size="2" width="95%" color="black">
    </body>
</html>
```

Maka awal konten akan seperti ini

Langkah-langkah memasak air

1. Masukkan air ke dalam wadah.
2. Matikan kompor.

---

appendChild

Apa fungsi dari *method appendChild*? Fungsinya adalah menambahkan sebuah elemen HTML pada bagian paling akhir dari sebuah elemen yang kita gunakan untuk memanggil *method* ini. Jika Anda belum begitu paham maksudnya, *yuk* kita praktikkan.

Pertama kita buat element list baru.

```
const newElement = document.createElement('li');
```

Maka jika variabel nya kita panggil. Akan muncul tag list nya :

```
> newElement
<-   <li></li>
```

Lalu kita isi list dengan teks menggunakan innerText :

```
newElement.innerText = "Selamat menikmati";
```

Kita panggil variabel nya :

```
> newElement
<-   <li>Selamat menikmati</li>
```

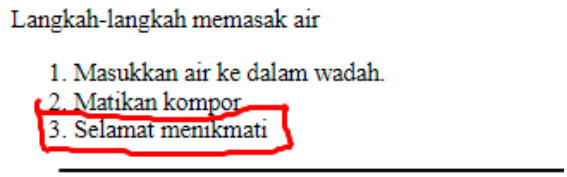
Item sudah dibuat, lalu kita panggil id yang akan menampung konten kita. Yaitu ol dengan id "daftar".

```
const daftar = document.getElementById('daftar');
```

Setelah itu lalu gabungkan item yang tadi dibuat dengan element ol yang akan menampungnya menggunakan appendChild.

```
daftar.appendChild(newElement);
```

Maka hasilnya akan seperti berikut :



Sebentar, pada langkah ke-2 tulisannya "Matikan kompor". Sejak kapan kita menghidupkan kompornya? Tampaknya kita harus menambahkan langkah yang berisi pesan "Hidupkan kompor". Langkah ini kita atasi dengan menggunakan method insertBefore().

insertBefore

method insertBefore() memberikan kemampuan untuk memasukkan elemen sebelum elemen HTML tertentu selama elemen *parent*-nya masih bisa diakses, sama seperti *method appendChild()*.

Pertama kita buat terlebih dahulu element nya

```
const elementAwal = document.createElement('li');
elementAwal.innerText = "Hidupkan Kompor";
```

Kedua, kita panggil id yang ingin kita sisipkan terhadap item yang sudah dibuat :

```
const itemAwal = document.getElementById('awal');
```

Ketiga, panggil id yang akan menampung konten kita.

```
const daftar = document.getElementById('daftar')
```

Terakhir kita gunakan method insertBefore. Parameter pertama pada *method* tersebut diisi dengan elemen baru yang ingin ditambah berdasarkan elemen yang sudah ditentukan di parameter kedua.

```
daftar.insertBefore(elementAwal, itemAwal)
```

Maka hasilnya seperti berikut :

Langkah-langkah memasak air

1. Hidupkan kompor
2. Masukkan air ke dalam wadah.
3. Matikan kompor.
4. Selamat menikmati

Event

AddEventListener() mengambil *jenis event* dan fungsi. Jenis acara akan menjadi acara klik dan fungsi akan memicu pesan peringatan.

Ini adalah kode untuk menambahkan event listener ke variabel button.

```
const btn = document.querySelector('button');

    btn.addEventListener('click', () => {
        alert('Ouch, you click me!')
    })
```

Referensi :

DOM : https://www.geeksforgeeks.org/dom-document-object-model/
DOM 2 : https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/