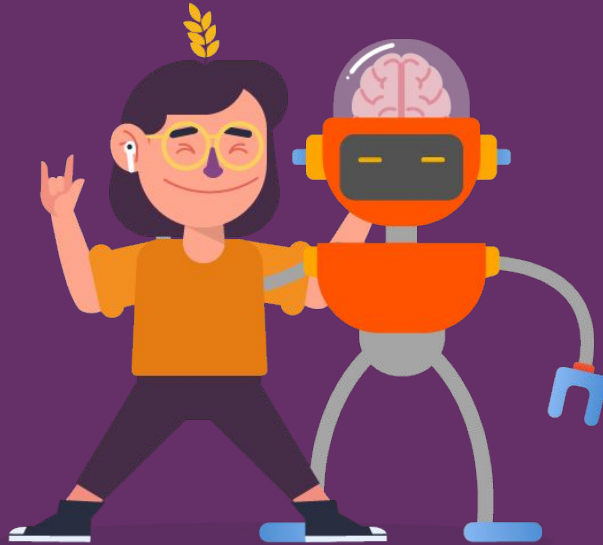




DOM

Gold - Chapter 4 - Topic 3

Halo, kakak-kakak~
Ini adalah sesi terakhir di **Chapter 4**
online course **Full-Stack Web**.

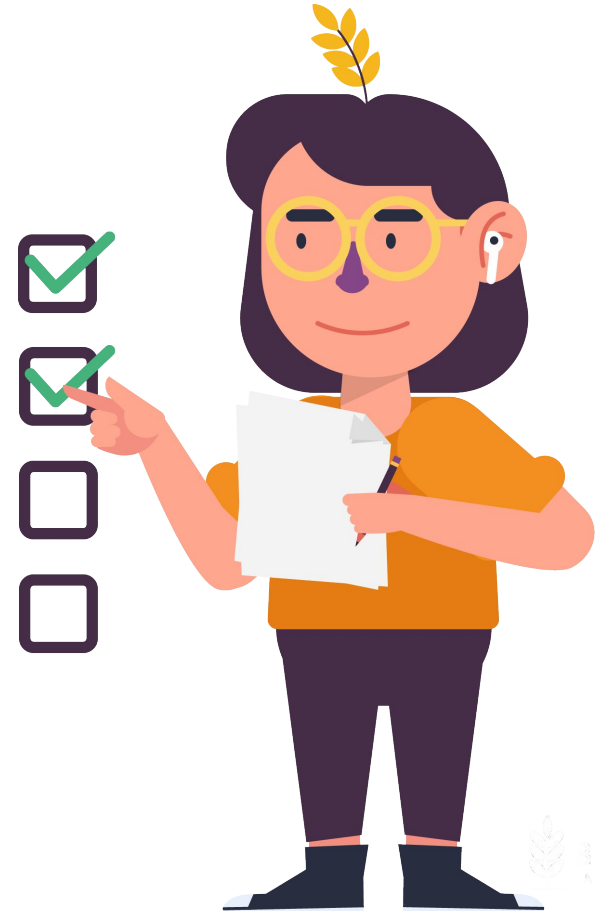




Di sesi ini, kita akan belajar tentang apa itu **DOM (Document Object Modeling) API** dan gimana cara menerapkannya di JavaScript.

Setelah sesi ini selesai, harapannya kamu bisa mendapatkan beberapa hal, antara lain:

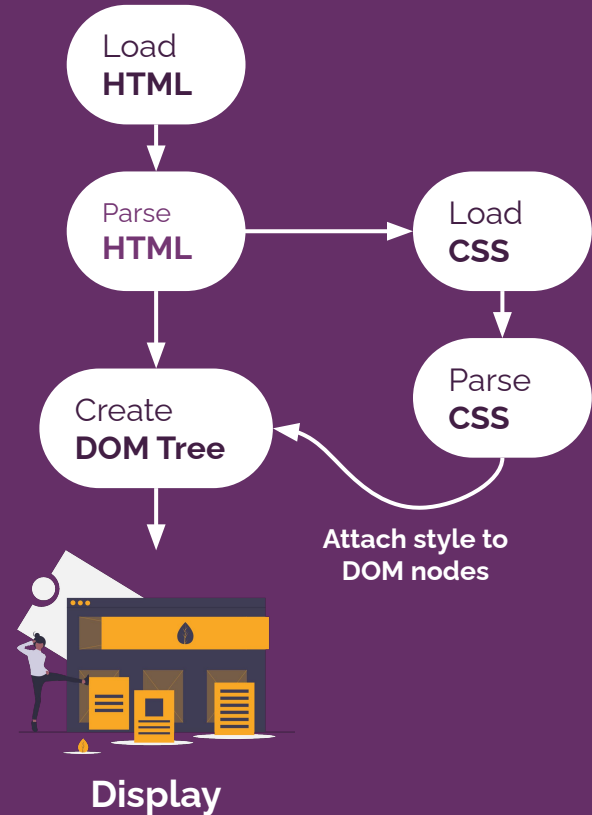
1. Memahami DOM API
2. Memahami cara menggunakan DOM di JavaScript, yaitu:
 - mengakses element tertentu dengan DOM
 - membuat element dengan DOM API
 - menghapus element dengan DOM API
 - contoh program yang menggunakan DOM





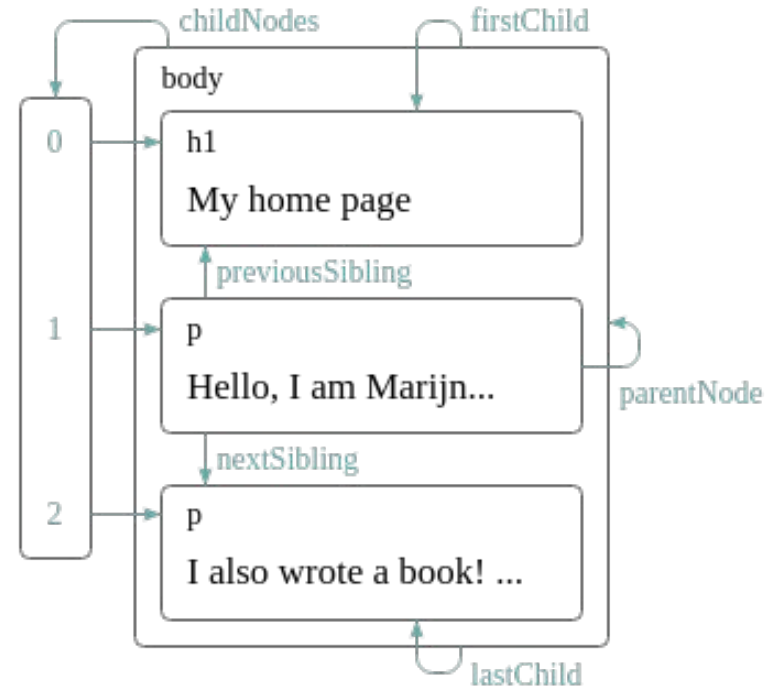
Kalian masih ingat materi di **Chapter 1**, kan?

Kalau lupa, coba perhatikan gambar di samping dulu deh~



Flashback sedikit yaa

1. Web browser menampilkan/me-render dokumen dengan mengubah HTML dan CSS menjadi **DOM (Document Object Modeling)**.
2. DOM memiliki struktur seperti pohon. Setiap **element**, **attribute**, dan **potongan teks** dalam bahasa *markup* menjadi **DOM node** dalam **struktur pohon**.
3. **Node** ditentukan oleh **hubungannya** dengan **DOM node lainnya**. Beberapa *element* merupakan **parent** dari **node anak**, dan **node anak** memiliki **saudara kandung**.





Nah, DOM menyediakan sekumpulan fungsi dan *attribute/data* yang bisa kita manfaatkan dalam membuat program JavaScript. Fungsi itu dikenal dengan sebutan **API** (***Application Programming Interface***).

Oh, iya! DOM nggak cuma untuk dokumen HTML saja, tetapi juga bisa digunakan untuk dokumen XML dan SVG. Dan, tentunya bisa digunakan di bahasa pemrograman lain.

Tapi, khusus sesi ini, kita hanya akan belajar **penggunaan DOM pada JavaScript** saja ya~



Pak haji siap?
Zidan siap?

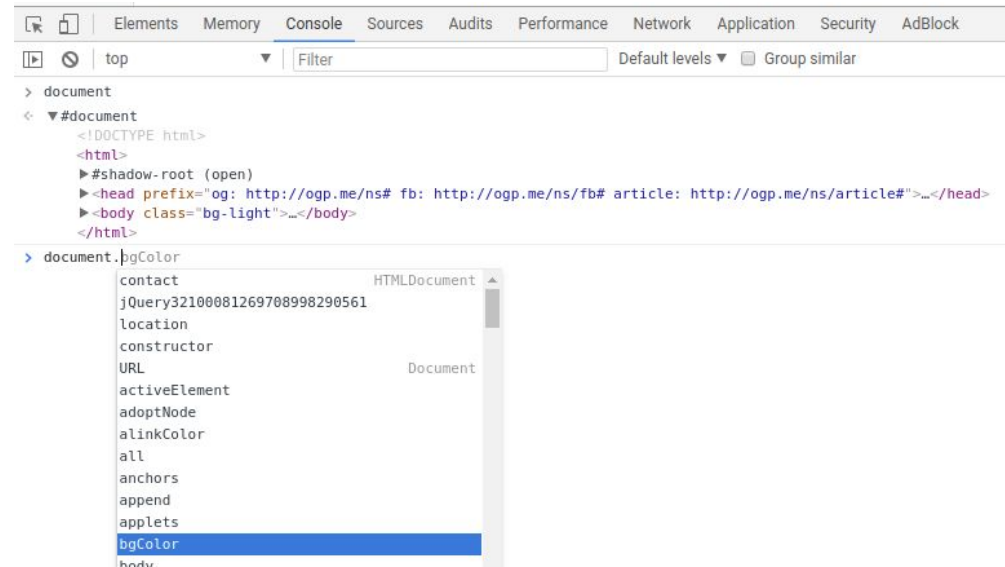
Mari kita masuk ke
bahasan tentang
DOM di JavaScript~



Seperti yang sudah kita ketahui, DOM adalah sebuah *object* buat memodelkan dokumen HTML.

Object DOM di JavaScript bernama ***document***. *Object* ini berisi segala hal yang kita butuhkan untuk memanipulasi HTML.

Jadi, jika kita coba ketik *document* pada *console* JavaScript seperti contoh di gambar, maka yang akan tampil adalah **kode HTML**.





Nah, di dalam *object document*, terdapat fungsi-fungsi dan atribut yang bisa kita gunakan untuk memanipulasi dokumen HTML.

Sebagai contoh fungsi **`document.write()`**. Fungsi ini digunakan untuk menulis sesuatu ke dokumen HTML.

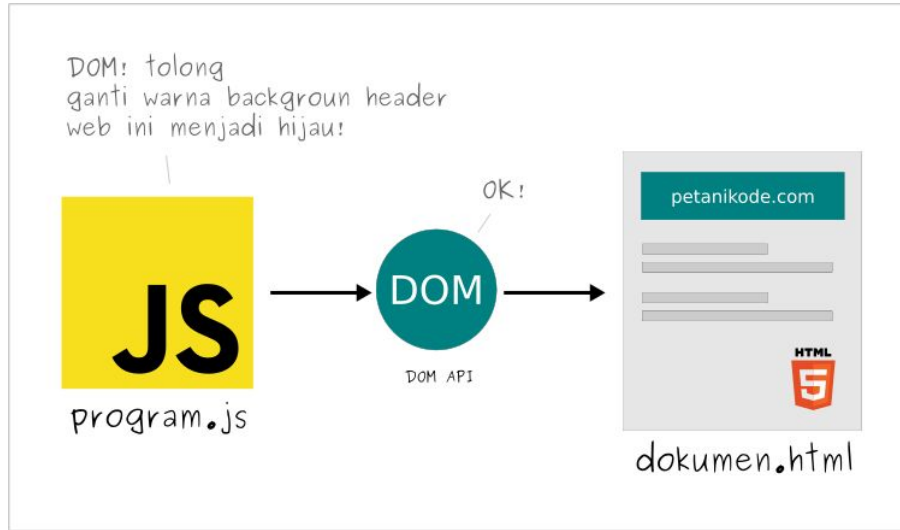
Mari, bapak-bapak, ibu-ibu. Kita ketik kode berikut ini di dalam *console* JavaScript Chrome:

```
document.write("Hello World");  
document.write("<h2>Saya Sedang Belajar Javascript</h2>");
```

Nanti hasilnya akan langsung berdampak pada dokumen HTML.
Bagaimana? Bagus kan? Besok bisa langsung dicoba di rumah yaa~

Hello World

Saya Sedang Belajar Javascript



Lalu, kapan kita harus menggunakan DOM?

DOM bisa kamu gunakan dalam beberapa keperluan, seperti:

- membuat pengalaman pengguna (*user experience*) yang lebih baik.
- memanipulasi elemen HTML secara *real time*.
- menghindari *refresh* halaman setiap kali *user* ingin melakukan interaksi

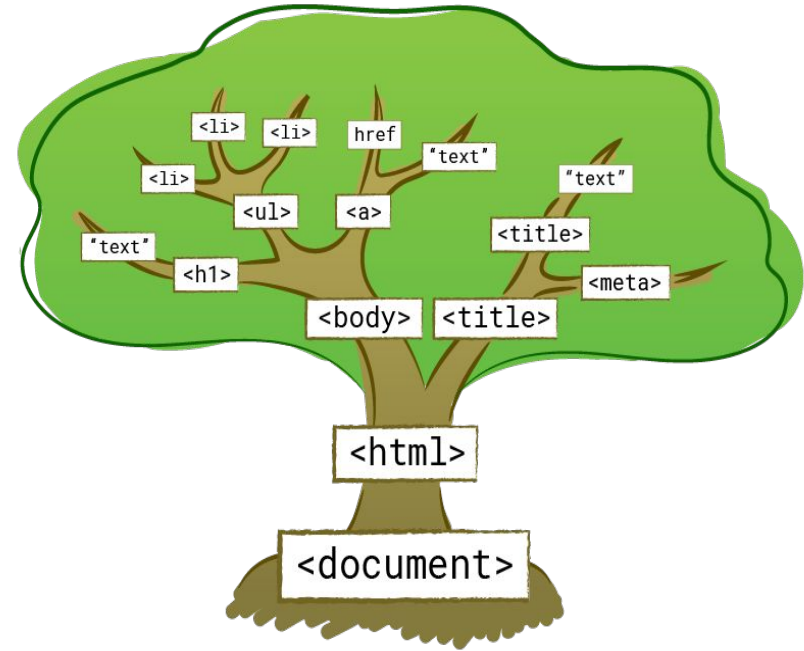


Terus, kalau kita mau
mengakses *element*
tertentu dengan DOM,
caranya gimana?

Jangan lupa, struktur DOM itu bentuknya seperti pohon. Maksudnya, *object document* yang ada di DOM itu sebagai representasi model dari dokumen HTML.

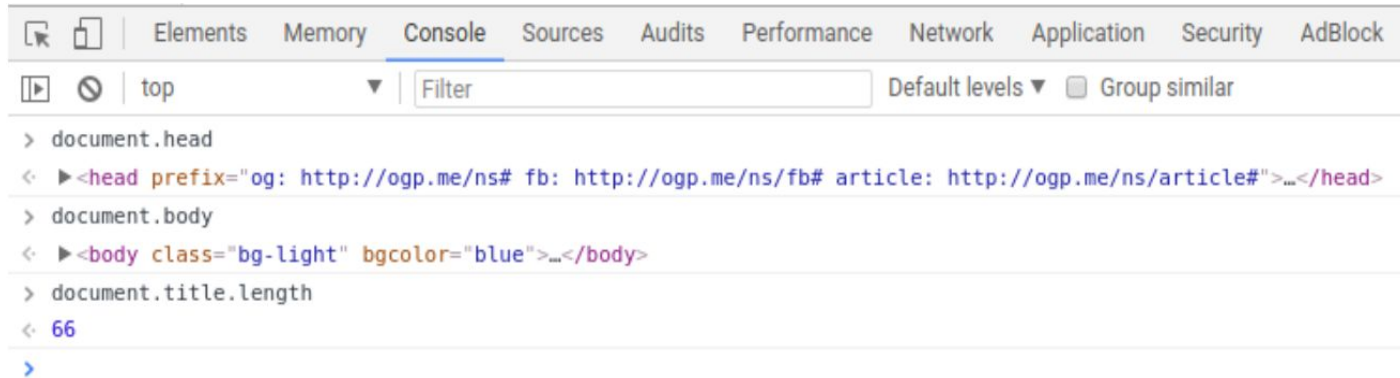
Object ini berisi kumpulan fungsi dan atribut berupa *object* dari elemen HTML yang bisa digambarkan dalam bentuk pohon. Ilustrasinya seperti pada gambar.

Nah, ini akan memudahkan kita dalam menggunakan *element* tertentu dengan DOM.



Jika kita mengetik kode seperti contoh pada gambar *console*, maka kita bisa mengakses *element* tertentu yang kita inginkan. Misalnya **<head>**, **<body>**, dll.

Bahkan, kita juga bisa mengetahui panjang dari suatu *element* **<title>**.



```
> document.head
< ▶<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# article: http://ogp.me/ns/article#">...</head>
> document.body
< ▶<body class="bg-light" bgcolor="blue">...</body>
> document.title.length
< 66
>
```

Jika kita ingin mengakses *element* yang spesifik, terdapat beberapa fungsi yang sering digunakan:

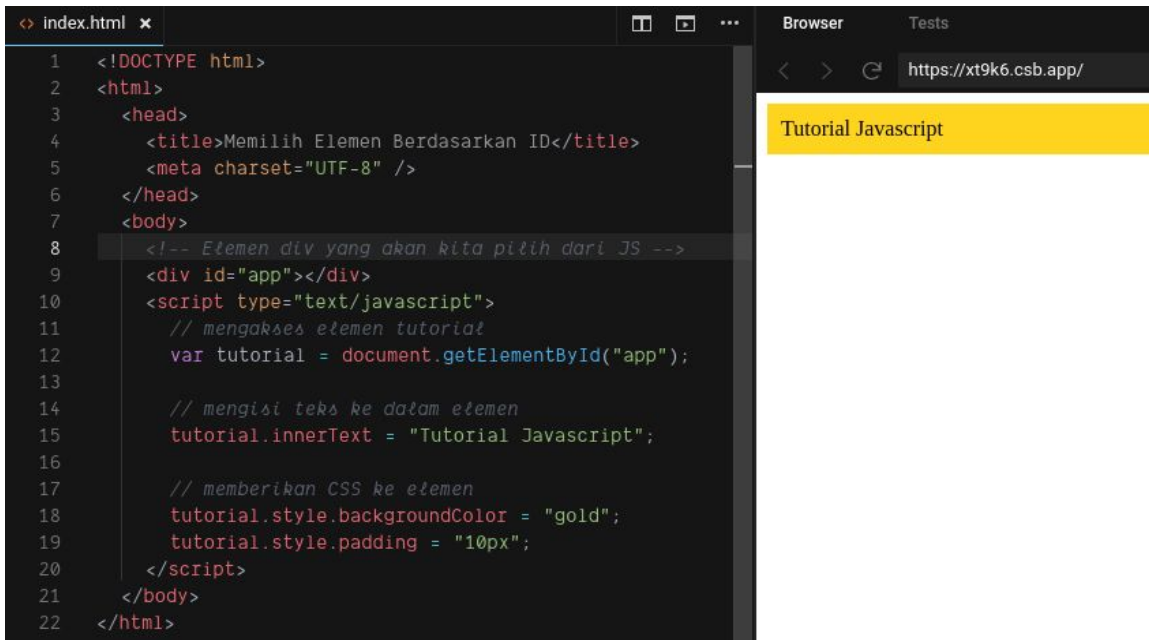
Fungsi	Kegunaan
getElementById()	memilih <i>element</i> berdasarkan atribut <i>id</i>
getElementByName()	memilih <i>element</i> berdasarkan atribut <i>name</i>
getElementsByClassName()	memilih <i>element</i> berdasarkan atribut <i>class</i>
getElementsByTagName()	memilih <i>element</i> berdasarkan nama tag dan bisa mengembalikan nilai berupa <i>array</i> , mengingat <i>element</i> html dengan tag tertentu bisa jadi lebih dari satu
querySelector()	mencari <i>element</i> DOM pertama yang sesuai dengan aturan <i>selector</i> CSS yang diberikan ke fungsi
querySelectorAll()	sama seperti <i>querySelector</i> , tapi mengembalikan semua <i>element</i> yang memenuhi aturan (bukan hanya <i>element</i> pertama)

Sekarang, kita coba praktikkan ya!

Misalnya kita punya kode HTML seperti di gambar.

Dalam kode tersebut, kita coba memilih *element* `<div>` yang memiliki *id* bernama **"app"**. Lalu, kita ingin mengaksesnya dengan fungsi **`getElementById()`** dan menyimpannya ke dalam suatu variabel yaitu `tutorial`.

Variabel `tutorial` akan menjadi sebuah *object* DOM dari *element* yang kita pilih. Setelah itu, kita bisa lakukan apapun yang diinginkan, seperti mengubah teks dan *style* CSS.



```
index.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Memilih Elemen Berdasarkan ID</title>
5     <meta charset="UTF-8" />
6   </head>
7   <body>
8     <!-- Elemen div yang akan kita pilih dari JS -->
9     <div id="app"></div>
10    <script type="text/javascript">
11      // mengakses elemen tutorial
12      var tutorial = document.getElementById("app");
13
14      // mengisi teks ke dalam elemen
15      tutorial.innerText = "Tutorial Javascript";
16
17      // memberikan CSS ke elemen
18      tutorial.style.backgroundColor = "gold";
19      tutorial.style.padding = "10px";
20    </script>
21  </body>
22 </html>
```

Browser Tests
https://xt9k6.csb.app/
Tutorial Javascript

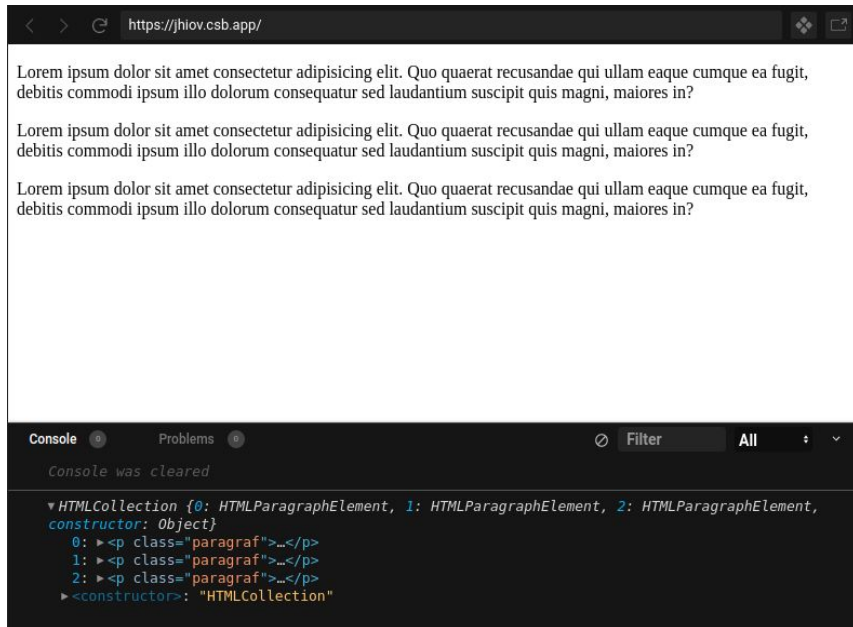


Tapi, gimana kalau ternyata ada **lebih dari satu elemen yang dipilih?**

Biasanya, kalau kita pilih *element* berdasarkan nama *tag* atau *attribute class*, akan ada lebih dari satu *element* yang dipilih. Nah, *element* yang terpilih itu akan menjadi sebuah *array*. Kenapa? Karena kita memilih sekumpulan *element*.

Array itu akan berisi objek DOM dari *element-element* yang terpilih. Untuk lebih jelasnya bisa lihat contoh kode berikut ini.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Mengakses lebih dari satu elemen di DOM</title>
5      <meta charset="UTF-8" />
6    </head>
7    <body>
8      <p class="paragraf">
9        Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaeat
10       recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
11       dolorum consequatur sed laudantium suscipit quis magni, maiores in?
12      </p>
13      <p class="paragraf">
14        Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaeat
15       recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
16       dolorum consequatur sed laudantium suscipit quis magni, maiores in?
17      </p>
18      <p class="paragraf">
19        Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaeat
20       recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
21       dolorum consequatur sed laudantium suscipit quis magni, maiores in?
22      </p>
23      <script>
24        var paragraf = document.getElementsByClassName("paragraf");
25        console.log(paragraf);
26      </script>
27    </body>
28  </html>
```



The screenshot shows a web browser at the URL `https://jhiov.csb.app/`. The page contains three paragraphs of Lorem Ipsum text. Below the text, the browser's developer console is open, showing the result of a JavaScript command. The console output is:

```
▼ HTMLCollection (0: HTMLParagraphElement, 1: HTMLParagraphElement, 2: HTMLParagraphElement, constructor: Object)
  0: ▶ <p class="paragraf">_</p>
  1: ▶ <p class="paragraf">_</p>
  2: ▶ <p class="paragraf">_</p>
  ▶ <constructor>: "HTMLCollection"
```

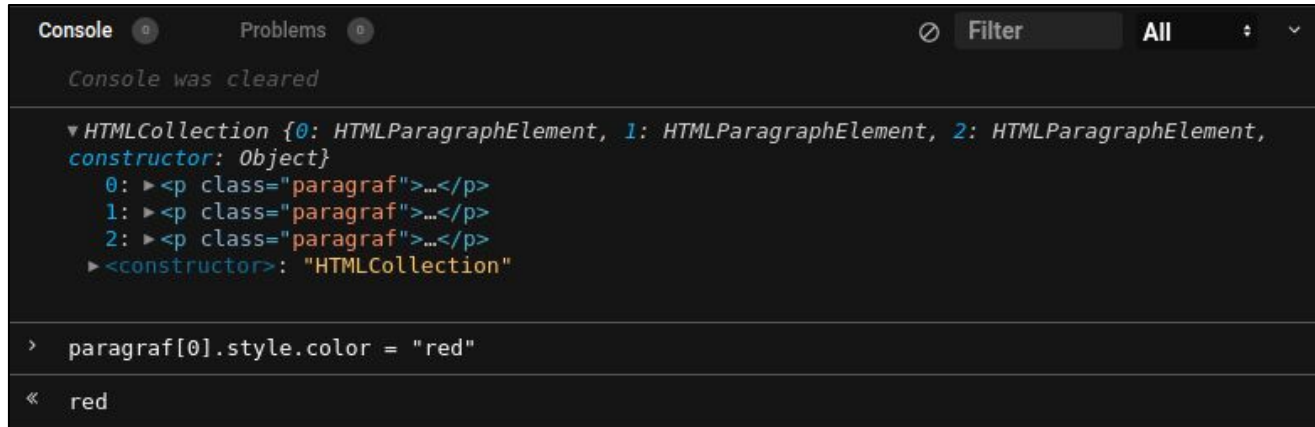
Hasil dari kode yang kita buat di *slide* sebelumnya akan tampak seperti pada gambar jika dibuka di *web browser*.

Pada contoh ini, kita memiliki tiga buah paragraf dengan nama *class* “paragraf”. Lalu, kita coba memilih ketiga paragraf tersebut melalui JavaScript dengan *method* atau fungsi ***getElementsByClassName()***.

Voila! Variabel paragraf akan berisi sebuah *array*, yang ternyata di dalamnya terdapat tiga buah objek DOM dari paragraf.

Oke, sekarang kita coba bereksperimen dengan mengubah warna teks!

Paragraf pertama akan berada pada posisi indeks ke-0 di dalam *array*. Coba kamu ketik perintah berikut ini di dalam *console* Javascript:



```
Console Problems Filter All
Console was cleared

▼ HTMLCollection {0: HTMLParagraphElement, 1: HTMLParagraphElement, 2: HTMLParagraphElement,
constructor: Object}
  0: ▶ <p class="paragraf">...</p>
  1: ▶ <p class="paragraf">...</p>
  2: ▶ <p class="paragraf">...</p>
  ▶ <constructor>: "HTMLCollection"

> paragraf[0].style.color = "red"

« red
```

Voila!

Hasilnya, paragraf pertama akan **berwarna merah**.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?

```
<> index.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Mengakses lebih dari satu elemen di DOM</title>
5     <meta charset="UTF-8" />
6   </head>
7   <body>
8     <p class="paragraf">
9       Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat
10      recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
11      dolorum consequatur sed laudantium suscipit quis magni, maiores in?
12    </p>
13    <p class="paragraf">
14      Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat
15      recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
16      dolorum consequatur sed laudantium suscipit quis magni, maiores in?
17    </p>
18    <p class="paragraf">
19      Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat
20      recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo
21      dolorum consequatur sed laudantium suscipit quis magni, maiores in?
22    </p>
23    <script>
24      var paragraf = document.getElementsByClassName("paragraf");
25      setInterval(function() {
26        paragraf[0].style.color = "red";
27        paragraf[1].style.color = "green";
28        paragraf[2].style.color = "blue";
29
30        setTimeout(function() {
31          paragraf[0].style.color = "black";
32          paragraf[1].style.color = "black";
33          paragraf[2].style.color = "black";
34        }, 500);
35      }, 1000);
36    </script>
37  </body>
38 </html>
```

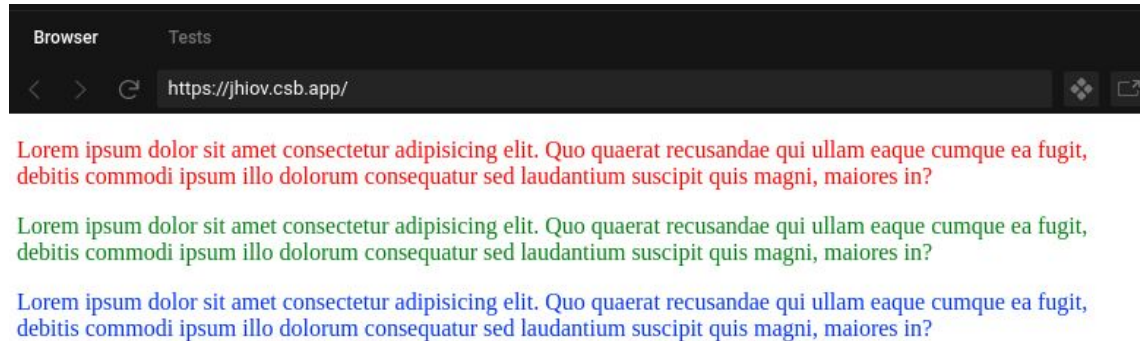
Lalu, kita coba buat sebuah animasi warna~

Biar nggak repot, coba praktikkan pada kode sebelumnya ya! Kita coba memanfaatkan fungsi ***setInterval()*** dan fungsi ***setTimeout()*** untuk menentukan waktu animasinya.

Pada kode ini, rentang waktu (interval) kita berikan 1000 milidetik, yang berarti 1 detik. Sedangkan untuk mengubah warnanya jadi hitam, kita berikan waktu 500 milidetik atau 0.5 detik. Maka akan terjadi animasi warna tersebut.

Boom!

Hasilnya jadi seperti ini:





Nah, DOM API ini punya fungsi untuk **membuat element HTML**.

Kita coba sekarang yaw :3



Salah satu fungsi untuk membuat *element* HTML adalah fungsi **`createElement()`**. Dengan fungsi ini maka akan tercipta *element* baru, yaitu paragraf.

```
document.createElement('p');
```

Tapi, hasilnya gak akan ditampilkan ke dalam halaman web. Karena kita belum menambahkannya ke dalam *body document*.

Cara menambahkannya ke *body document*, kita bisa gunakan fungsi **`append()`**.





Nah, setelah kita menggunakan fungsi ***append()***, maka hasilnya akan seperti ini:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Membuat element di DOM</title>
5     <meta charset="UTF-8" />
6   </head>
7   <body>
8     <script>
9       // membuat element h1
10      var judul = document.createElement("h1");
11
12      // mengisi content element
13      judul.textContent = "Belajar JavaScript";
14
15      // menambahkan element ke dalam tag body
16      document.body.append(judul);
17    </script>
18  </body>
19 </html>
```



Kalau kita bisa membuat *element* dengan DOM API, tentu kita juga bisa ***menghapus element***



Kalau fungsi **append()** untuk menambahkan *element*, untuk menghapus kita pakai **remove()**.

The screenshot shows a code editor on the left and a web browser on the right. The code editor displays an `index.html` file with the following content:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Menghapus element di DOM</title>
5     <meta charset="UTF-8" />
6   </head>
7   <body>
8     <h2 id="judul2">Delete Saya!</h2>
9     <script>
10      // memilih element berdasarkan ID
11      var h2 = document.getElementById("judul2");
12
13      // menghapus element yang sudah dipilih
14      h2.remove();
15
16      console.log("Element sudah dihapus");
17      console.log(h2);
18    </script>
19  </body>
20 </html>
```

The web browser on the right shows the URL `https://jhiov.csb.app/`. The console at the bottom of the browser displays the message "Console was cleared" and "Elemen sudah dihapus". Below this, a log entry shows the removed element: `<h2 id="judul2">Delete Saya!</h2>`.

Fungsi **remove()** ini nggak jauh beda seperti melupakan mantan.

Element memang terhapus dari halaman web. Tapi, sebenarnya ia masih tersimpan di dalam memori.

Biar makin mantap nih,
mari kita coba membuat
program menggunakan
DOM di JavaScript





Program ini berfungsi untuk mengubah warna latar belakang dari suatu *element* dan mengubah warna teksnya. Kita akan menggunakan event "*change*" pada *element* **bgColor** dan **txtColor**.

Maksudnya, ketika setiap nilai dari *element* ini berubah, kode di dalamnya akan dieksekusi.

The screenshot shows a web browser window on the right and a code editor on the left. The browser displays a page titled "Aplikasi Ubah Warna" with a light blue background. It has two input fields: "Warna latar:" and "Warna teks:". The code editor shows the following HTML and JavaScript code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Contoh program dengan DOM</title>
5     <meta charset="UTF-8" />
6   </head>
7   <body>
8     <h1>Aplikasi Ubah Warna</h1>
9     <label>Warna latar: </label>
10    <input type="color" id="bg-color" />
11    <br />
12    <label>Warna teks: </label>
13    <input type="color" id="text-color" />
14
15    <script>
16      var bgColor = document.getElementById("bg-color");
17      var txtColor = document.getElementById("text-color");
18
19      bgColor.addEventListener("change", event => {
20        document.body.style.backgroundColor = bgColor.value;
21      });
22
23      txtColor.addEventListener("change", event => {
24        document.body.style.color = txtColor.value;
25      });
26    </script>
27  </body>
28 </html>
```



DOM (*Document Object Model*)

Dokumen (HTML) yang dimodelkan dalam sebuah *object*.

API (*Application Programming Interface*)

Sekumpulan fungsi dan *attribute*/data yang disediakan oleh JavaScript yang bisa kita manfaatkan dalam membuat program.

DOM API untuk mengakses *element*

- `getElementById()` fungsi untuk memilih *element* berdasarkan atribut id.
- `getElementByName()` fungsi untuk memilih *element* berdasarkan atribut name.
- `getElementsByClassName()` fungsi untuk memilih *element* berdasarkan atribut class.
- `getElementsByTagName()` fungsi untuk memilih *element* berdasarkan nama tag.
- `getElementsByTagNameNS()` fungsi untuk memilih *element* berdasarkan nama tag.
- `querySelector()` fungsi untuk memilih *element* berdasarkan query.
- `querySelectorAll()` fungsi untuk memilih *element* berdasarkan query.

DOM API untuk membuat *element*

- `createElement()` fungsi untuk membuat *element* HTML.
- `append()` fungsi untuk menampilkan *element* yang sudah dibuat ke *body document*.

DOM API untuk menghapus *element*

- `remove()` fungsi untuk menghapus *element* HTML.





Saatnya QUIZ



1

DOM menyediakan sekumpulan fungsi dan *attribute*/data yang bisa kita manfaatkan dalam membuat program JavaScript yang dikenal dengan istilah ...

- A. Document Object Model
- B. Application Programming Interface
- C. Node



2

DOM API **getElementByClassName()** digunakan untuk ...

- A. Memilih *element* berdasarkan atribut id
- B. Memilih *element* berdasarkan atribut name
- C. Memilih *element* berdasarkan atribut class



3

Kode di bawah ini digunakan untuk ...

```
document.body.append(judul);
```

- A. Menampilkan *element* yang sudah dibuat ke *body document*
- B. Memilih *element* berdasarkan query
- C. Membuat *element* HTML baru



Pembahasan Quiz



1

DOM menyediakan sekumpulan fungsi dan *attribute*/data yang bisa kita manfaatkan dalam membuat program JavaScript yang dikenal dengan istilah ...

B. Application Programming Interface

Sekumpulan fungsi dan *attribute*/data yang disediakan oleh DOM untuk membuat program JavaScript, dikenal dengan istilah API (*Application Programming Interface*)



2

DOM API `getElementsByClassName()` digunakan untuk ...

- C.** Memilih *element* berdasarkan atribut class

DOM API `getElementsByClassName()` digunakan untuk memilih *element* berdasarkan atribut class



3

Kode di bawah ini digunakan untuk ...

```
document.body.append(judul);
```

A. Menampilkan *element* yang sudah dibuat ke *body document*

DOM API ***append()*** digunakan untuk menampilkan *element* yang sudah dibuat ke dalam *body document*



Referensi

- https://www.w3schools.com/js/js_htmlDOM.asp
- https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API
- <https://www.petanikode.com/javascript-dom/>
- https://eloquentjavascript.net/14_dom.html
- <https://blog.10pines.com/2018/08/27/reactjs-virtual-dom/>
- <http://dev.bertzzie.com/knowledge/javascript/Document-Object-Model.html>