# Pertemuan 23 - Recap Materi Javascript Manipulasi DOM

## Pendahuluan

Di Javascript memiliki `DOM` atau `Document Object Model` ini merupakan suatu representasi elemen struktur HTML kita, dengan ini kita dapat mengubah bentuk struktur HTML dan CSS kita ketika halaman web dibuka, ini sangat bermanfaat ketika kita memerlukan interaksi yang sangat tinggi, untuk memakainya kita hanya memanggil `document` object lalu kita akan mendapatkan banyak sekali method yang bisa kita gunakan, kita hanya membahas yang sering dipakai saja.

## HTML Selector

Karena kita akan mengubah struktur HTML dan CSS kita, kita harus menggunakan `Selector` untuk men-select elemen yang kita inginkan, perlu di-ingat cara penyebutannya akan berbeda elemen yang di-select di dalam Javascript disebut `Node`.

Untuk contoh kita akan menggunakan struktur HTML seperti ini

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <div id='main-content' class='container'>
            <p data-my-data='Data ku di attribute'>Hello, World!</p>
        </div>
    </body>
    <script>
        // Kode Javascript kita ada disini
        // Karena Javascript hanya dapat men-select kalau strukturnya sudah ada
        // Dan hanya mengambil sekali
    </script>
</html>
```

### getElementById

Yang paling mudah di-ingat ialah `getElementById('id')` ini akan men-select `element` pertama yang memiliki `id` yang di spesifikan di parameter tersebut, perlu di-ingat ini hanya akan men-select 1 saja jadi kalau mau select banyak `element` harus ganti `method`.

```
const myContentNode = document.getElementById('main-content');
```

### querySelector

`method` yang ini hampir mirip dengan `getElementById('id')` tetapi kita dapat men-select tidak hanya `id` saja, misal `tag`, `class`, `data-{nama}`, yang `data-{nama}` itu merupakan data attribute, ini sering digunakan untuk memberikan data tambahan kepada Javascript tetapi bisa juga menjadi bahan untuk di-select.

```
const bodyNode = document.querySelector('body'); // Select memakai nama tag
const mainContentNode = document.querySelector('#main-content'); // Select memakai
id
const containerClassNode = document.querySelector('.container'); // Select memakai
CSS class
const myDataNode = document.querySelector('[data-my-data]'); // Select memakai
data attribute
```

`querySelector` ini juga dapat men-select lebih dari satu dengan menggunakan `All` di bagian akhir `querySelectorAll`.

## Node

Kan tadi sudah tau cara mengambil elemen di HTML lalu kita jadikan `Node` di script Javascript kita, kita dapat menambahkan elemen didalam `Node` itu tadi agar ke render ke browser kita, kita juga dapat melakukan hal-hal yang menarik juga, tetapi kita akan fokus didasarnya saja dan yang akan sering kita gunakan karena `Node` memiliki banyak fitur.

### innerHTML

`property` ini digunakan untuk menambah teks ke elemen HTML tetapi akan juga me-render kalau di berikan elemen HTML, ini sangat bermanfaat untuk membuat elemen baru dengan peforma tinggi tetapi perlu di-ingat jangan menggunakan user input untuk membuat elemen yang baru dinamis karena bisa saja menjadi bahan `hack`.

kalau `innerHTML` di-isi sesuatu, nanti-nya yang ada di browser akan ke `override` dengan kata lain di-injak, tetapi kita juga bisa memakainya untuk mengambil struktur HTML yang ada.

```
const bodyNode = document.querySelector('body');

// Dengan contoh html di-atas kita akan mendapatkan tulisan "Zagar" dengan tulisan
besar bukan "Hello, World!" lagi
bodyNode.innerHTML = "<h1>Zagar!</h1>";
console.log(bodyNode.innerHTML) // Ini akan menprint struktur HTML
```

Kalau misal pengen append atau menambahkan kita menggunakan simbol `+=` bukan `=` lagi, ini akan menambahkan apa apa yang ditambahkan setelah konten yang sudah ada di browser.

### textContent

`property` ini sama dengan `innerHTML` tetapi tidak akan me-render elemen yang diberikan kedalam `property` tersebut

```
const bodyNode = document.querySelector('body');

// Dengan contoh html di-atas kita akan mendapatkan tulisan "<h1>Zagar</h1>" bukan
"Hello, World!" lagi
bodyNode.innerHTML = "<h1>Zagar!</h1>";
```

## append

`method` ini digunakan untuk membuat `Node` yang lain untuk menjadi anak dari `Node` yang memanggil `method` tersebut.

Masih dengan contoh di-atas

```
const bodyNode = document.querySelector('body');
const pNode = document.querySelector('p');

bodyNode.append(pNode);
```

Di contoh di-atas akan membuat elemen HTML `<p>` keluar dari `<div>` dan langsung menjadi anak dari `<body>`, perlu di-ingat `append` ini juga membawa elemen anak yang akan di-append juga.

Ilustrasi :

```
<body>
    <div id='main-content' class='container'>
        <p data-my-data='Data ku di attribute'>Hello, World!</p>
    </div>
    <div id="your-content">
        <!--  -->
    </div>
</body>
<script>

    const mainContent = document.querySelector('#main-content');
    const yourContent = document.querySelector('#your-content');

    yourContent.append(mainContent);
</script>
```

Hasilnya akan menjadi

```
<body>
    <div id="your-content">
        <!--  -->
        <div id='main-content' class='container'>
            <p data-my-data='Data ku di attribute'>Hello, World!</p>
```

```
        </div>
    </div>
</body>
```

## style

`property` ini digunakan untuk menambah `style attribute` didalam `Node` kita juga bisa mengambil style tersebut dengan `property` yang sama, cara pemakaiannya mirip dengan `innerHTML` dan `textContent`.

```
const pNode = document.querySelector('p');

pNode.style = "color: red;"; // Membuat teks didalam Elemen yang diselect menjadi merah
```

### id

`property` ini dapat menambahkan `attribute` id atau mengambil `attribute` tersebut, cara penggunaanya sama dengan `style`, `textContent`, `innerHTML`.

### value

`property` ini hanya bekerja kepada element `<input>` karena `property` ini lebih mengarah apa apa yang di-input di elemen tersebut, ini dapat digunakan untuk menambah data ke input atau mungkin di-ambil.

### dataset

`property` ini sama seperti dengan `style` dan yang lainnya tetapi ini mengarah di attribute `data-{nama}`, `property` ini memiliki tipe data `Javascript Object` jadi kita bisa mengakses `property` tersebut seperti halnya kita mengubah variabel `Javascript Object` kita, kita juga dapat menambah data ke dalam dataset agar `ter-render` sebagai data tambahan di elemen.

```
<div data-container="1">

</div>
<script>
    const containerNode = document.querySelector('[data-container]');

    console.log(containerNode.dataset); // akan men-print "DOMStringMap { container → "1" }"
    console.log(containerNode.dataset.container); // akan men-print "1"

    // Untuk menambah dataset kita hanya memberi key yang kita inginkan misalnya target
    containerNode.dataset.target = "Target";
</script>
```

Perlu di-ingat kalau `attribute data`-nya berbentuk `data-my-target` atau dengan kata lain ada `-` lagi setelah `data-` di dataset akan menjadi `myTarget` bukan `my-target` karena tidak diperbolehkan di Javascript

## addEventListener, removeEventListener & Event Attribute

Sebelum kita ke `EventListener` kita belajar dulu apa itu `Event Attribute`, `Event Attribute` merupakan suatu cara kita men-trigger kode Javascript kita ketika suatu elemen HTML mendapat event tertentu seperti diklik, kena tunjuk cursor dll, ada banyak attribute yang dapat kita pakai, kita akan membahas sedikit saja, semua ini di tangani lewat `attribute` yang akan diberikan nama function yang nanti akan dijalankan atau kode Javascript langsung, semua `attribute` tersebut dimulai dengan nama `on...`.

### Event Attribute

- `onclick`
  Ini akan jalan ketika diklik sesuai dengan namanya.
- `ondblclick`
  Ini akan jalan ketika di double click.
- `oninput`
  Ini akan dijalankan ketika elemen-nya diketik sesuatu, misalnya `form input`.

```
<!-- Konteks this disini bermaksud elemen HTML-nya dengan kata lain Node -->
<button onclick='myFunction(this)'></button>

<script>
    function myFunction(node) {
        console.log(node);
    }
</script>
```

### addEventListener

`Event Listener` merupakan suatu cara kita memberikan interaksi di HTML kita yang lebih mengarah ke `event based` karena `Event attribute` hanya support satu saja kalau mau support banyak jenis interaksi kita akan mendapatkan kode yang tidak berbentuk, nah itu `Event Listener` bisa membantu, dengan memiliki fungsi dan tugas yang sama dengan `Event Attribute`, `addEventListener` ini memiliki semua event yang ada di `Event Attribute` tetapi tidak ada `on`-nya.

`addEventListener` ini menerima 2 parameter, yang pertama tipe eventnya, yang kedua closure yang menerima 1 parameter yaitu `event`, `event` ini merupakan `object` yang berisi `property` yang sangat banyak sekali, tetapi kita paling sering di-ambil ialah `target` karena kita bisa mendapatkan `Node` yang bisa kita pakai.

```
<button id="myBtn"></button>

<script>
    const myBtn = document.getElementById('myBtn');
    myBtn.addEventListener('click', event => {
        console.log(event);
```

```
    })
</script>
```

**removeEventListener**

Ini method ini sama dengan yang `addEventListener` tetapi digunakan untuk menghapus function yang digunakan untuk menjalankan `event`, karena `Event Listener` itu bisa banyak kita harus memberikan function yang sama persis dan `Event` yang digunakan untuk men-trigger.

```
<button id="myBtn"></button>

<script>
    const myBtn = document.getElementById('myBtn');

    // Buat function yang nanti kita pakai
    const myFunc = event => console.log(event);

    // tambahkan ke event listener
    myBtn.addEventListener('click', myFunc);

    // hapus
    myBtn.removeEventListener('click', myFunc);
</script>
```

## Membuat Node

Terkadang kita juga pengen membuat `Node` ketika web kita sedang dijalankan jadi kita tidak mengandalkan elemen HTML yang sudah ada, ini dapat dicapai dengan memanggil `document.createElement("nama-tag")` dan mengisi nama tag-nya

```
<body>
    <div id='main-content' class='container'>
        <p data-my-data='Data ku di attribute'>Hello, World!</p>
    </div>
    <div id="your-content">
        <!--  -->
    </div>
</body>
<script>
    // Kita select terlebih dahulu
    const mainContent = document.querySelector('#main-content');
    const yourContent = document.querySelector('#your-content');

    // Kita buat element nodenya lalu kita beri isi
    const newH1Node = document.createElement('h1');
    newH1Node.textContent = "My new Node!";

    // Lalu kita masukan sebagai anak elemen dari yourContent
```

```
    yourContent.append(newH1Node);
</script>
```

dari Contoh di-atas kita menggunakan method `createElement` karena ini lebih disarankan ketika kita menggunakan input user untuk memberikan konten tambahan atau styling dll, karena demi keamanan, tetapi lebih lambat kalau kebanyakan operasi seperti ini, tetapi kalau kita tidak menggunakan input user kita bisa langsung skip, memakai langsung method `innerHTML` saja.

```
<body>
    <div id='main-content' class='container'>
        <p data-my-data='Data ku di attribute'>Hello, World!</p>
    </div>
    <div id="your-content">
        <!--  -->
    </div>
</body>
<script>
    const yourContent = document.querySelector('#your-content');
    yourContent.innerHTML += "<h1>My new Node!</h1>"
</script>
```