WEB PROGRAMMING

MODUL PRAKTIKUM 5 Document Object Model (DOM)

DAFTAR ISI

[1 Pengenalan Document Object Model 3](#_Toc51540745)

[1.1 HTML DOM 3](#_Toc51540746)

[1.2 Akses Node Pada DOM 4](#_Toc51540747)

[1.2.1 Node Tree 4](#_Toc51540748)

[1.3 Informasi pada Node 5](#_Toc51540749)

[1.3.1 Tipe Node 5](#_Toc51540750)

[1.3.2 nodeName 6](#_Toc51540751)

[1.3.3 nodeValue 6](#_Toc51540752)

[1.4 Hubungan antar Node 6](#_Toc51540753)

[2 Akses DOM 8](#_Toc51540754)

[2.1 getElementById( ) 8](#_Toc51540755)

[2.1.1 Contoh Penggunaan getElementById() 8](#_Toc51540756)

[2.2 getElementsByName() 9](#_Toc51540757)

[2.2.1 Contoh getElementsByName() 10](#_Toc51540758)

[2.3 getElementsByTagName( ) 12](#_Toc51540759)

[2.3.1 Contoh Penggunaan getElementsByTagName() 12](#_Toc51540760)

[2.4 getElementsByClassName( ) 14](#_Toc51540761)

[2.4.1 Contoh getElementsByClassame( ) 14](#_Toc51540762)

[2.5 querySelector( ) 16](#_Toc51540763)

[2.5.1 Contoh Dasar querySelector( ) 16](#_Toc51540764)

[2.5.2 Contoh Penggunaan querySelector( ) 18](#_Toc51540765)

[3 Akses Lintas Node 20](#_Toc51540766)

[3.1 parrentNode 20](#_Toc51540767)

[3.2 childNodes[nodenumber] 22](#_Toc51540768)

[3.2.1 Mendapatkan elemen pertama pada child 22](#_Toc51540769)

[3.2.2 Mendapatkan elemen terakhir (lastchild) 24](#_Toc51540770)

[3.2.3 Mendapatkan seluruh elemen child. 26](#_Toc51540771)

[4 Manipulasi Elemen 28](#_Toc51540772)

[4.1 createElement( ) 28](#_Toc51540773)

[4.1.1 Membuat elemen div 28](#_Toc51540774)

[4.1.2 Menambahkan id pada elemen 29](#_Toc51540775)

[4.1.3 Menambahkan class pada elemen 30](#_Toc51540776)

[4.1.4 Menambahkan teks pada elemen node 31](#_Toc51540777)

[5 Event Handling 32](#_Toc51540778)

# Pengenalan Document Object Model

Document Object Model (DOM) adalah *Aplication Programming Interface* (API) untuk memanipulasi dokumen HTML dan XML.

DOM merepresentasikan dokumen sebagai struktur pohon simpul (*tree node*). Ini menyediakan API yang memungkinkan Anda untuk menambah, menghapus, dan memodifikasi bagian-bagian dokumen secara efektif.

Perhatikan bahwa DOM adalah cara lintas platform dan bahasa-independen untuk memanipulasi dokumen HTML dan XML.

DOM dibagi menjadi beberapa bagian yaitu :

1. Core DOM : untuk mendefinisikan sebuah standar objek pada struktur dokumen.
2. XML DOM : untuk mendefinisikan sebuah standar objek pada dokumen XML.
3. HTML DOM : untuk mendefinisikan sebuah standar objek pada dokumen HTML.

## HTML DOM

HTML DOM dapat digunakan dengan Javascript untuk membaca dan mengubah dokumen HTML, XHTML dan XML. Dasar skrip yang harus dikuasai adalah HTML/XHTML dan Javascript.

DOM membuat sebuah dokumen HTML sebagai *struktur tree* dengan elemen, atribut dan text, sehingga dengan DOM dapat dikatakan :

1. Semua yang ada di dalam dokumen HTML adalah sebuah node.
2. Setiap tag HTML adalah sebuah elemen node.
3. Teks dalam HTML adalah teks node.
4. Setiap atribut HTML adalah atribut node.
5. Comment adalah comment node.

Sehingga, HTML DOM adalah sebuah standar bagaimana mengakses dan memanipulasi dokumen HTML.

contoh dokumen HTML :

|  |
| --- |
| <html>  <head>  <title> My Title</title>  </head>  <body>  <a href: http://iinilink.com>My llink</a>  <h1>My Header</h1>  </body>  </html> |

Jika diilustrasikan dalam bentuk gambar maka dapat dilihat pada gambar berikut :



Gambar 6 Tree Structure DOM HTML.

Semua node diatas mempunyai hubungan antara satu dengan yang lain. Setiap node, kecuali node dokumen, mempunyai sebuah *parent* node, misalkan *parent* node dari <head> dan <body> adalah node <html> dan parent node dari teks “**My header**” adalah node <h1>.

## Akses Node Pada DOM

Dengan DOM, dokumen baik dokumen HTML dan XML dapat diakses disetiap node-nya.

### Node Tree

Sebuah *node tree* menunjukan sebuah dokumen sebagai set koneksi di antara mereka. Ada beberapa hal yang perlu diperhatikan :

1. Dalam *node tree*, *node* paling atas disebut *root*.
2. Setiap node, kecuali root mempunyai satu *parent node*.
3. Sebuah node dapat mempunyai beberapa *children* (anak).
4. Sebuah *leaf* (daun) adalah node yang tidak mempunyai *children* (anak).
5. Sibling adalah node yang mempunyai parent yang sama.
6. Child yang pertama kali dapat di list dinamakan *firstchild*.
7. Child yang terakhir di list dinamakan *lastchild*.



Gambar 9 Ilustrasi Node Tree file.

## Informasi pada Node

### Tipe Node

Setiap node di pohon DOM diidentifikasi oleh tipe node. JavaScript menggunakan bilangan bulat untuk menentukan jenis node.

Tabel 1 Element Type Node.

|  |  |  |
| --- | --- | --- |
| Element Type | NodeType | Constant |
| Element | 1 | Node.ELEMENT\_NODE |
| Attribute | 2 | Node.Attribute\_NODE |
| Text | 3 | Node.TEXT\_NODE |
| CData | 4 | Node.CDATA\_SECTION\_NODE |
| Intruction | 7 | Node.PROCESSING\_INSTRUCTION\_NODE |
| Comment | 8 | Node.COMMENT\_NODE |
| Document | 9 | Node.DOCUMENT\_NODE |
| Document Type | 10 | Node.DOCUMENT\_TYPE\_NODE |
| Document Fragment | 11 | Node.DOCUMENT\_FRAGMENT\_NODE |

Untuk mendapatkan tipe node, Anda menggunakan properti nodeType:

**node.nodeType**

### nodeName

Properti nodeName berisi nama node.

* nodeName dari sebuah elemen adalah nama tag.
* nodeName dari sebuah atribut adalah nama atribut.
* nama dari sebuah teks node selalu #text
* nama dari node dokumen selalu #document

### nodeValue

Dalam teks node, properti nodeValue berisi teks. Dalam atribut node, properti nodeValue berisi nilai atribut. properti nodeValue tidak tersedia dalam dokumen dan elemen node.

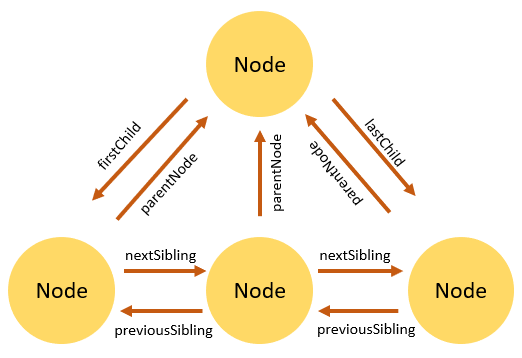
## Hubungan antar Node

Setiap node memiliki hubungan dengan node lain di pohon DOM. Hubungannya sama seperti yang dijelaskan dalam silsilah keluarga tradisional.

Misalnya, <body> adalah simpul anak (*child node*) dari node <html>, dan <html> adalah simpul induk (*parrent node*) dari node <body>.

Simpul <body> adalah *sibling* dari simpul <head> karena mereka berbagi induk (*parrent*) langsung yang sama, yaitu elemen <html>.

Gambar berikut mengilustrasikan hubungan antar node:



# Akses DOM

Dengan DOM maka semua node dalam dokumen dapat diakses.

Untuk dapat mencari dan mengakses elemen pada dokumen HTML untuk dilakukan manipulasi, dapat digunakan beberapa cara : Menggunakan *method*

1. ***getElementById()***,
2. ***getElementsByTagName(),***
3. ***getElementsByClassName(),***
4. ***getEementsByName(),***
5. ***querySelector().***

## getElementById( )

getElementById() memungkinkan Kita untuk memilih elemen berdasarkan Id-nya.

Sintaks yang digunakan :

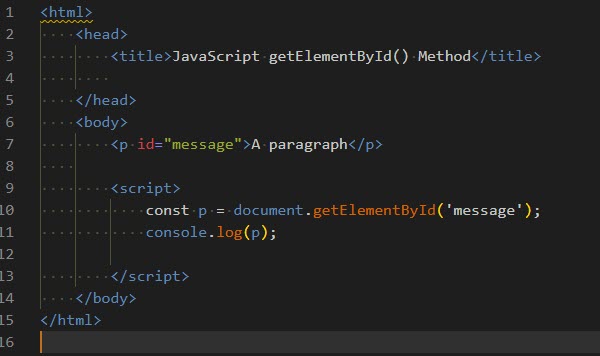
var element = document.getElementById(id);

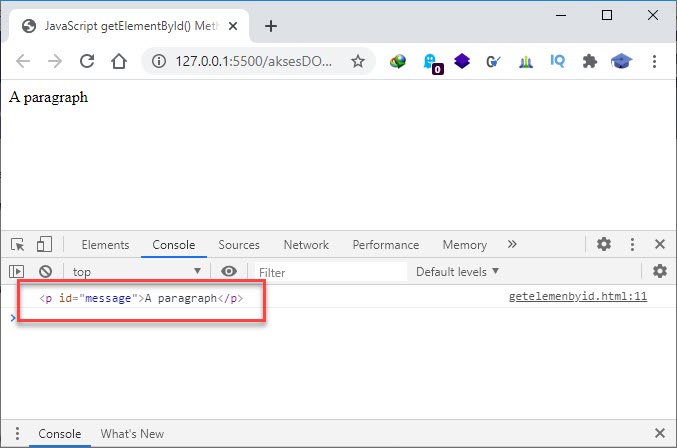
Dalam sintaks ini, Id mewakili “id” dari element yang Kita pilih. Id adalah *case sensitive*, misalnya “pesan” dan “Pesan” adalah Id yang berbeda.

getElementById() mengembalikan objek elemen yang mendeskripsikan objek elemen DOM dengan id atau **null** yang ditentukan jika tidak ditemukan elemen yang dimaksud.

Id pada dokumen HTML bersifat unik. Jika dokumen memiliki lebih dari satu elemen dengan id yang sama, maka metode getElementById() hanya akan mengembalikan yang pertama ditemuinya.

### Contoh Penggunaan getElementById()





## getElementsByName()

getElementsByName() merupakan metode untuk mendapatkan elemen dengan nama tertentu dalam dokumen.

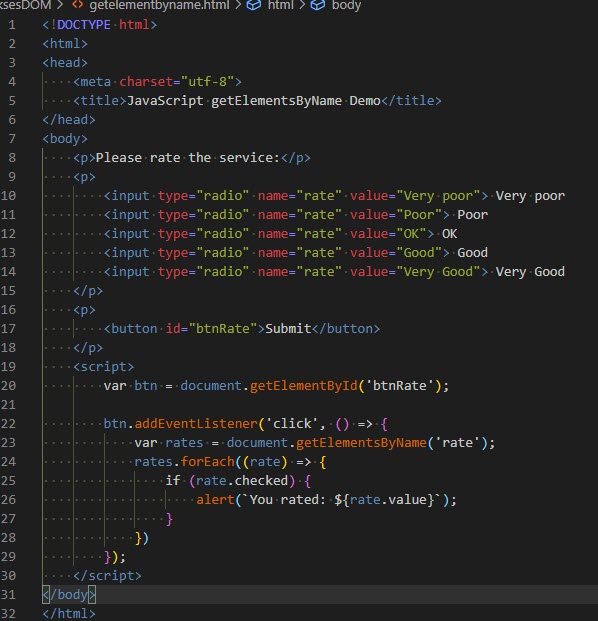
Untuk mendapatkan semua elemen dengan nama yang ditentukan, Kita menggunakan metode getElementsByName () dari objek dokumen:

var elements = document.getElementsByName(name);

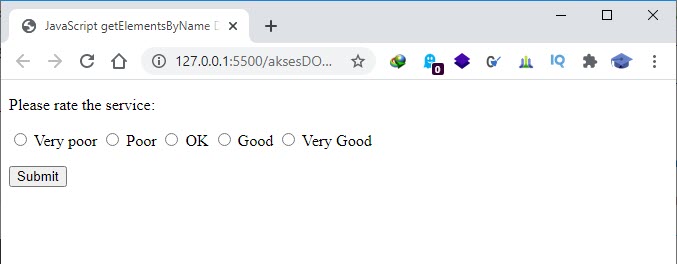
getElementsByName () menerima nama yang merupakan nilai atribut nama elemen dan mengembalikan NodeList elemen secara langsung.

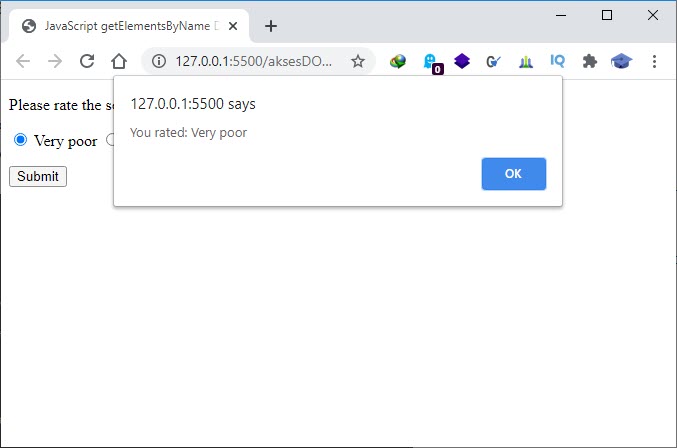
Koleksi elemen yang dikembalikan secara langsung. Ini berarti bahwa elemen secara otomatis diperbarui ketika elemen dengan nama yang sama ditambahkan dan / atau dihapus dari dokumen.

### Contoh getElementsByName()



Hasil dapat di lihat pada gambar berikut :





Keterangan :

* Pertama, pilih tombol Rate dengan id btnRate-nya menggunakan metode getElementById ().
* Kedua, kaitkan event klik ke tombol Rate sehingga ketika tombol diklik, fungsi anonim dijalankan.
* Ketiga, di click event handler, panggil getElementsByName () untuk memilih semua radio button yang memiliki name rate.
* Terakhir, lakukan iterasi melalui tombol radio. Jika tombol radio dicentang, maka akan ditampilkan dialog peringatan yang menunjukkan nilai tombol radio yang dipilih.

## getElementsByTagName( )

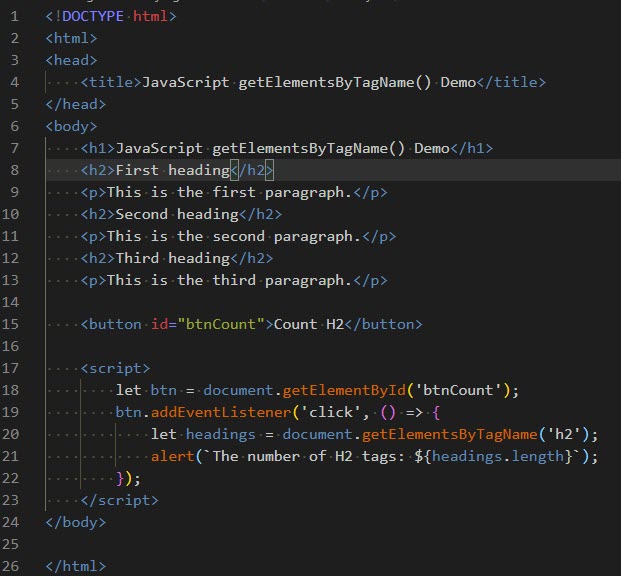
getElementsByTagName () adalah metode objek dokumen atau elemen DOM tertentu.

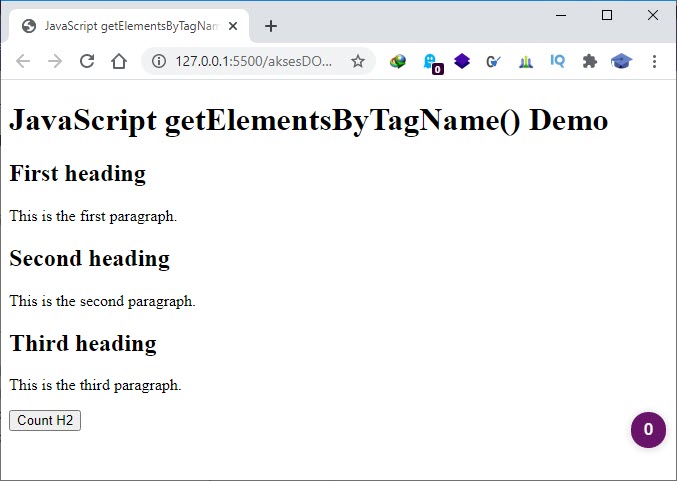
Metode getElementsByTagName () menerima nama tag - tag HTML dan mengembalikan langsung dari elemen dengan nama tag yang cocok sesuai urutan kemunculannya di dokumen.

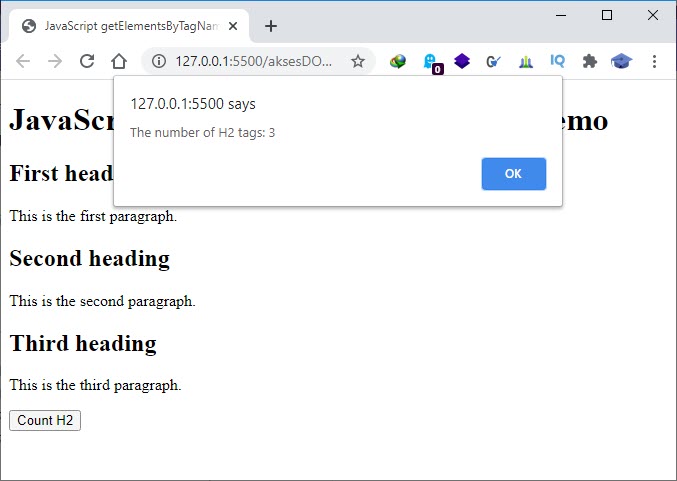
Sintaks sebagai berikut :

var elements = document.getElementsByTagName(tagName);

### Contoh Penggunaan getElementsByTagName()







Keterangan :

* Pertama, pilih tombol Count H2 dengan menggunakan metode getElementsById().
* Kedua, kaitkan tombol click event ke fungsi anonim.
* Ketiga, dalam fungsi anonim, gunakan document.getElementsByTagName () untuk mendapatkan daftar tag H2.
* Terakhir, tunjukkan jumlah tag H2 menggunakan fungsi alert ().

## getElementsByClassName( )

Metode getElementsByClassName () tersedia di objek dokumen dan elemen HTML apa pun.

Metode getElementsByClassName () menerima satu argumen, yaitu string yang berisi satu atau lebih nama kelas:

var elements = document.getElementsByClassName(classNames)

var elements = parentElement.getElementsByClassName(classNames)

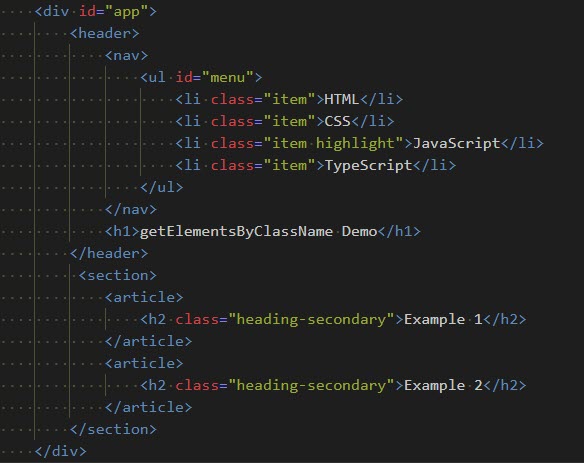
Dalam sintaks ini, parameter classNames adalah string yang mewakili nama kelas atau daftar nama kelas yang dipisahkan koma untuk dicocokkan.

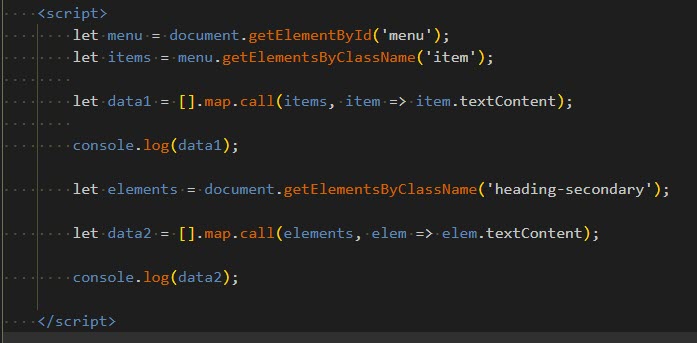
Jika Kita memanggil metode getElementsByClassName () pada objek dokumen, metode tersebut mencari elemen dengan nama kelas yang ditentukan di seluruh dokumen.

Namun, saat Kita memanggil metode getElementsByClassName () pada elemen tertentu, metode ini hanya mengembalikan elemen yang cocok di subtree elemen tersebut.

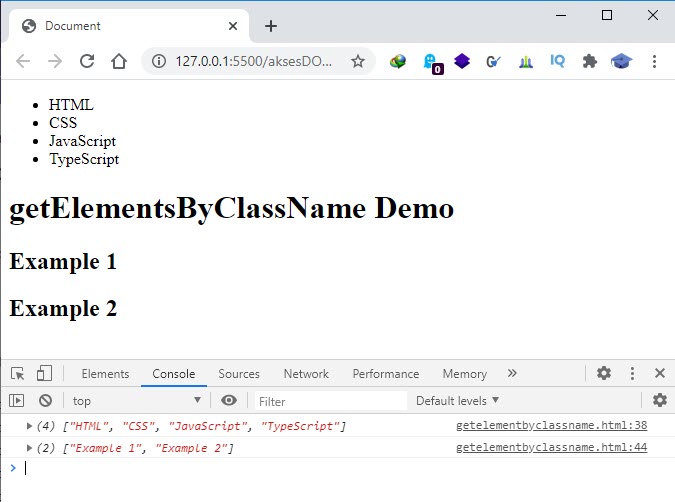
### Contoh getElementsByClassame( )

Pada dokumen HTML masukin script ini diantara tag body :





Maka akan terlihat pada browser sebagai berikut :



Keterangan :

Memanggil pada objek dokumen

* Pertama, pilih elemen <ul> dengan id = menu menggunakan metode getElementById ().
* Kemudian, pilih <li> elemen, yang merupakan turunan dari elemen <ul>, menggunakan metode getElementsByClassName ().
* Terakhir, buat array konten teks dari elemen <li> dengan meminjam metode map () dari objek Array.
* data1 log menggunakan metode getElementsByClassName () pada objek dokumen,
* data2 log metode ini mencari menggunakan elemen pada dokumen, elemen dengan kelas heading-secondary di seluruh dokumen.

## querySelector( )

Dengan querySelector( ), Kita dapat menemukan elemen - elemen yang akan dipanggil untuk manipulasi DOM, cocok digunakan untuk selector CSS atau selector group CSS.

Sintaks :

let element = parentNode.querySelector(selector);

Dalam sintaksis ini, selektor adalah pemilih CSS atau sekelompok pemilih CSS untuk mencocokkan elemen turunan dari parentNode.

Jika selector bukan sintaks CSS yang valid, metode ini akan memunculkan pengecualian SyntaxError.

Jika tidak ada elemen yang cocok dengan pemilih CSS, querySelector () mengembalikan null.

Selain querySelector (), Kita dapat menggunakan metode querySelectorAll () untuk menemukan semua elemen yang cocok dengan selector CSS atau sekelompok selector CSS:

let elementList = parentNode.querySelectorAll(selector);

Metode querySelectorAll () mengembalikan NodeList statis elemen yang cocok dengan selector CSSnya. Jika tidak ada elemen yang ditemukan, ia mengembalikan NodeList kosong.

### Contoh Dasar querySelector( )

#### Universal Selector

Universal selector ditandai dengan simbol asterik ( \* ) yang mencocokan dengan semua elemen jenis apapun.

sintaks :

let element = document.querySelector('\*');

let elements = document.querySelectorAll('\*');

#### Type Selector

untuk memilih elemen - elemen dengan nama node, Kita dapat menggunakan type selector contoh **a** untuk memilih seluruh elemen <a>, h1 untuk memilih seluruh elemen <h1>.

sintaks :

let firstHeading = document.querySelector('h1');

let heading2 = document.querySelectorAll('h2');

#### Class Selector

Untuk Mencari elemen yang sudah diberikan atribut class, Kita dapat menggunakan class selector.

Sintaks :

mencari elemen dengan class menu-item yang akan ditampilkan adalah yang pertama kali ditemukan.

let note = document.querySelector('.menu-item');

Mencari seluruh elemen dengan class menu-item.

let notes = document.querySelectorAll('.menu-item');

#### ID Selector

Untuk memilih elemen berdasarkan nilai dari suatu Id, Kita dapat menggunakan Id Selector.

Sintaks :

Memilih elemen yang pertama kali ditemukan dengan id logo

let logo = document.querySelector('#logo');

karena Id harus bersifat unik maka Kita tidak dapat menggunakan querySelectorAll( ) secara relevan.

#### Attribute selector

Untuk memilih semua elemen yang memiliki atribut tertentu, Kita menggunakan salah satu dari Attribute Selector berikut :

* [attribute]
* [attribute=value]
* [attribute~=value]
* [attribute|=value]
* [attribute^=value]
* [attribute$=value]
* [attribute\*$\*=value]

Contoh Kita mencari elemen pertama pada elemen dengan atribut “[autoplay]” dengan nilai apapun :

let autoplay = document.querySelector('[autoplay]');

dan contoh berikut menemukan semua elemen yang memiliki atribut “[autoplay]” dengan nilai apapun :

let autoplays = document.querySelectorAll('[autoplay]');

#### Grouping Selector

Untuk mengelompokan berbagai selector, Kita dapat menggunakan sintaks :

selector, selector, ...

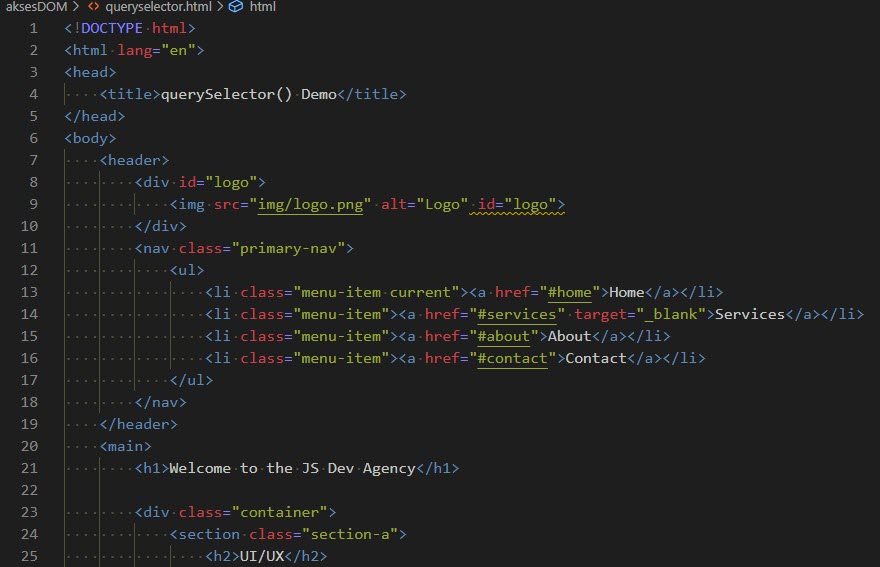
Contoh Kita akan mencari seluruh elemen <div> dan <p> :

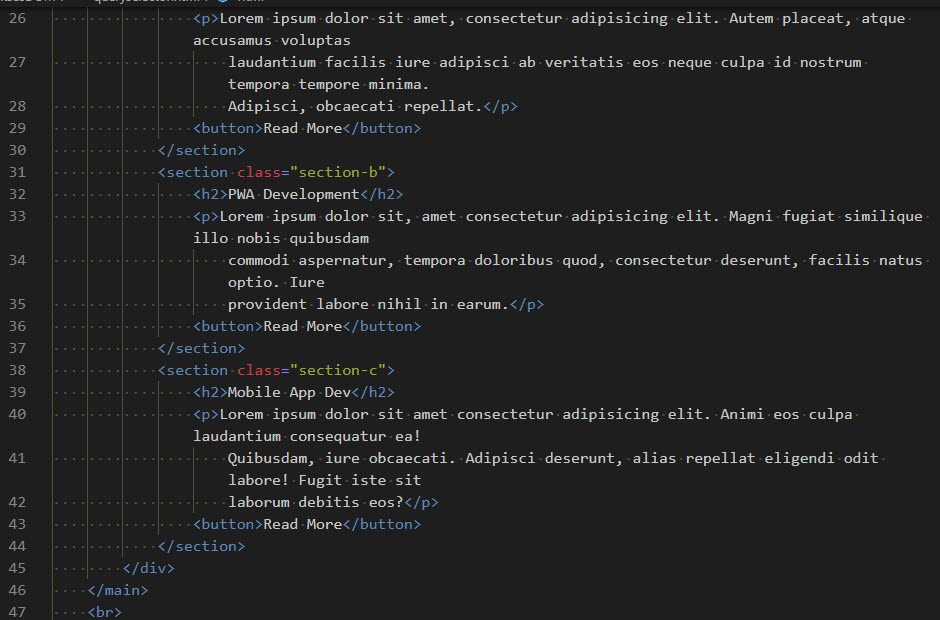
let elements = document.querySelectorAll('<div>, <p>');

maka yang akan dipilih adalah elemen pertama yang ditemukan terlebih dahulu antara elemen div atau p yang akan dimanipulasi.

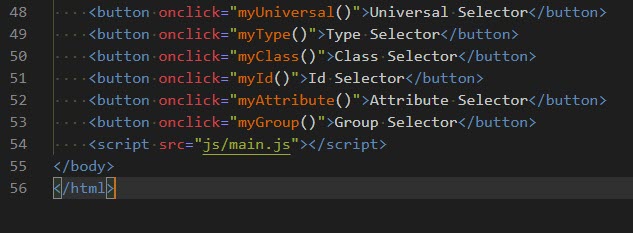
### Contoh Penggunaan querySelector( )

file html





button untuk memanggil fungsi selector



main.js



# Akses Lintas Node

Kita dapat menggunakan properti node berikut untuk menavigasi antar node dengan JavaScript:

* **parentNode**
* **childNodes[nodenumber]**
  + **firstChild**
  + **lastChild**
* **nextSibling**
* **previousSibling**

## parrentNode

Untuk mendapatkan node induk dari node yang ditentukan di struktur tree DOM, Kita dapat menggunakan properti parentNode:

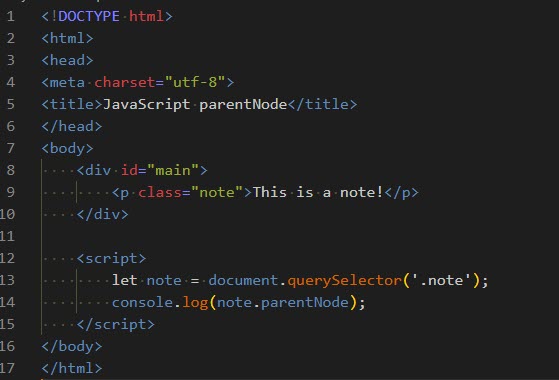
let parent = node.parentNode;

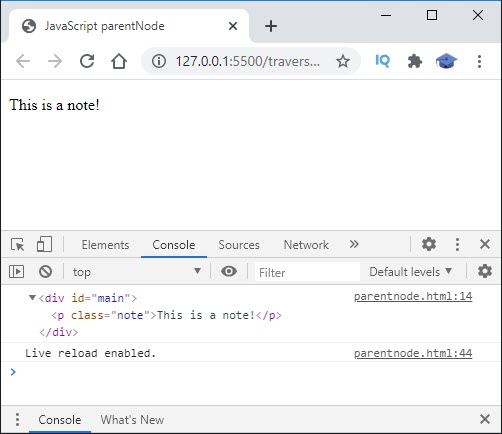
parentNode bersifat read-only.

Node Document dan DocumentFragment tidak memiliki parent, oleh karena itu parentNode akan selalu null.

Jika Kita membuat node baru tetapi belum memasangnya ke tree DOM, parentNode dari node tersebut juga akan menjadi null.

Contoh Penggunakan parentNode



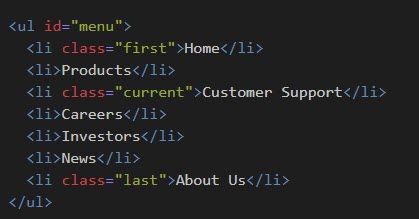


Pada contoh diatas maka Kita telah memilih parentNode dari class note yaitu id main.

## childNodes[nodenumber]

Dengan menggunakan childNodes, firstChild, dan lastChild Kita dapat mengakses elemen dibawah parentNode (child) pada elemen tertentu.

Kita akan menggunakan skrip HTML dibawah ini untuk struktur DOM tree yang akan dimanipulasi.



### Mendapatkan elemen pertama pada child

Untuk mendapatkan elemen anak pertama dari elemen tertentu, Kita dapat menggunakan properti **firstChild** :

sintaks :

let firstChild = parentElement.firstChild;

Jika parentElement tidak memiliki anak maka firstChild akan mengembalikan nilai null. Properti firstChild akan mengembalikan child node yang dapat berupa jenis apapun seperti node element, node text, atau node comment.

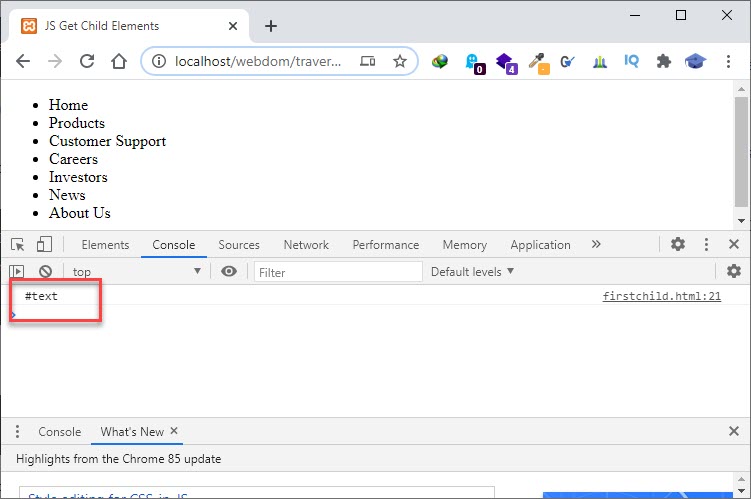
Berikut ini adalah contoh menunjukan firstChild dari elemen dengan id menu :

let content = document.getElementById('menu');

let firstChild = content.firstChild.nodeName;

console.log(firstChild);

maka output yang akan keluar pada browser adalah sebagai berikut :

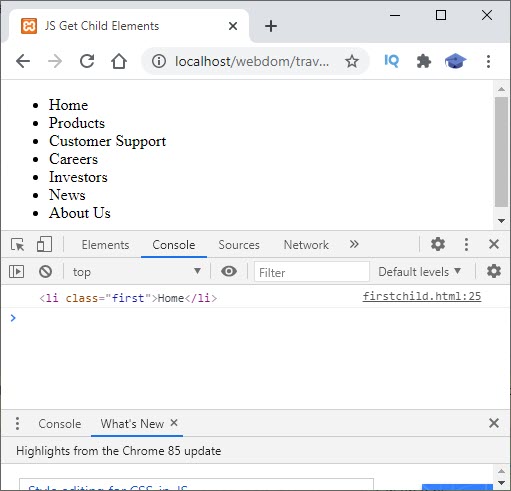


Pada console log menunjukan karakter **#text** karena node teks yang disisipkan untuk menjaga whitespace antara tag pembuka <ul> dan <li>. whitespace ini membuat sebuah #text node.

Perhatikan bahwa whitespace apapun seperti single space, multiple space, return dan tab akan membuat sebuah #text node.

Untuk mendapatkan first child dengan element only kita dapat menggunakan properti **firstElementChild.**

Maka pada browser console akan terlihat element pertama pada node yang dimaksud.



Pada contoh diatas :

Pertama memilih elemen #menu menggunakan metode getElementById( ).

Kemudian mendapatkan elemen first child menggunakan properti **firstElementChild**.

### Mendapatkan elemen terakhir (lastchild)

Untuk mendapatkan elemen terakhir dari sebuah node, Kita dapat menggunakan properti lastChild.

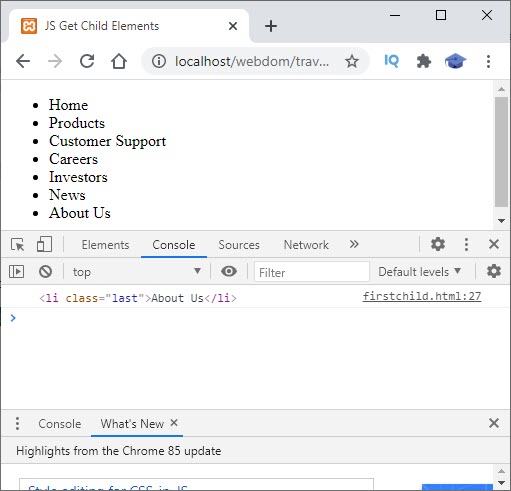
let lastChild = parentElement.lastChild;

Jika parentElement tidak memiliki turunan, lastChild mengembalikan nilai null. Mirip dengan properti firstChild, properti lastChild mengembalikan node elemen pertama yang ditemukan, node teks, atau node komentar. Jika Kita ingin memilih hanya elemen anak terakhir dengan jenis node elemen, Kita menggunakan properti lastElementChild :

let lastChild = parentElement.lastElementChild;

berikut adalah contoh penggunaan properti lastChild :

 console.log(content.lastElementChild);

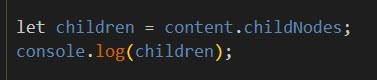


Maka output yang akan tampil pada console window adalah elemen terakhir dari child node.

### Mendapatkan seluruh elemen child.

Untuk mendapatkan NodeList langsung child element dari elemen tertentu Kita dapat menggunakan properti **childNodes** :

**let children = parentElement.childNodes;**



A screenshot of a social media post

Description automatically generated

Properti childNodes mengembalikan semua elemen child dengan tipe node apa pun. Untuk mendapatkan elemen child dengan hanya jenis node elemen, Kita dapat menggunakan properti **children**:

**let children = parentElement.children;**

**A picture containing drawing

Description automatically generated**

A screenshot of a social media post

Description automatically generated

# Manipulasi Elemen

## createElement( )

Untuk membuat elemen HTML, Kita dapat menggunakan script method document.createElement( )

**let element = document.createElement(htmlTag);**

**document.createElement( )** menerima nama tag HTML dan mengembalikan Node baru dengan tipe elemen.

### Membuat elemen div

Sebelumnya Kita akan mempersiapkan dokumen HTML dengan struktur sebagai berikut :

A screenshot of a cell phone

Description automatically generated

Kemudian buat sebuah tag script createElement( ) untuk membuat elemen div yang baru dan tambahkan element <p> untuk menampilkan paragraf pada browser.

gunakan properti **appendChild( )** untuk menambahkan elemen HTML.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

### Menambahkan id pada elemen

Untuk menambahkan sebuah id pada element tertentu Kita dapat menambahkan script berikut :



Maka pada dokumen HTML akan terlihat sebagai berikut :

A screenshot of a social media post

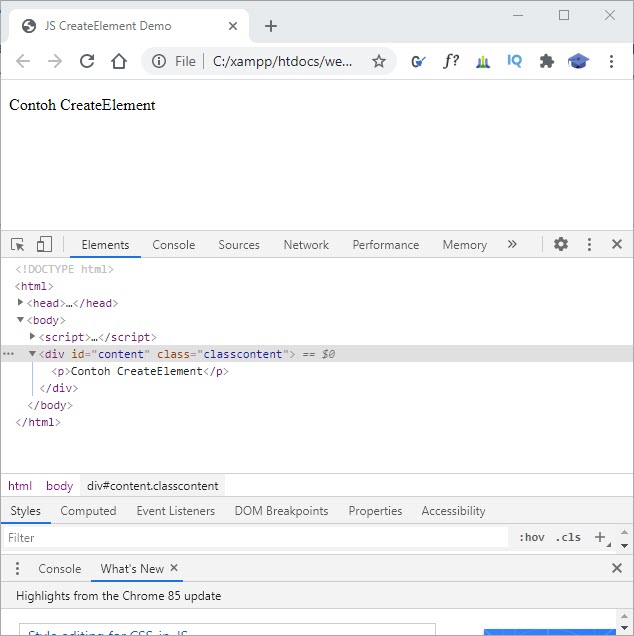
Description automatically generated

### Menambahkan class pada elemen

Untuk menambahkan class pada elemen yang akan dibuat maka tambahkan script berikut :

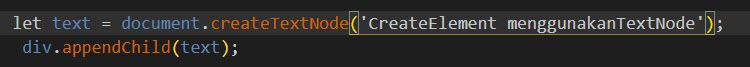


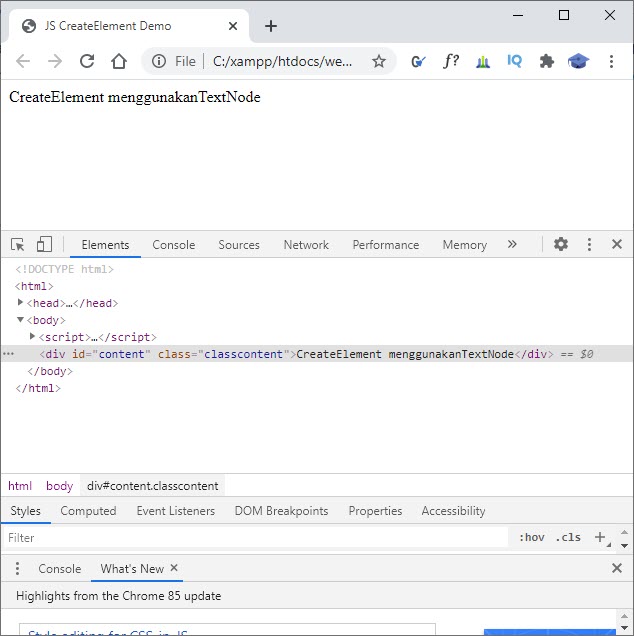
Maka pada dokumen HTML akan terlihat sebagai berikut :



### Menambahkan teks pada elemen node

Untuk menambahkan sebuah teks pada elemen yang akan Kita buat kita dapat menggunakan properti **innerHTML** atau membuat Text Node baru untuk ditambahkan pada elemen yang sudah dibuat.





# Event Handling

Event adalah sebuah tindakan yang terjadi di browser web, yang diberikan browser kepada Kita sehingga Kita dapat menanggapinya.

Misalnya, saat pengguna mengklik tombol di sebuah dokumen HTML, Kita mungkin ingin menanggapi event klik ini dengan menampilkan kotak dialog.

Setiap event mungkin memiliki penanganan event (event handling) yang merupakan blok kode yang akan dijalankan saat event terjadi.

Event handling dikenal juga sebagai event listener. Ini akan menanggapi sebuah event dan mengeksekusi ketika event terjadi.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated