**JAVASCRIPT LANJUTAN | 7. for..of VS for..in**

Kita akan mempelajari 2 Looping baru yg ada pada modern Javascript, sintaks looping baru ini dinamakan :

* For..of
* For..in

Yg sebelum nya kita tau kita punya beberapa statement looping ya kita bisa gunakan for kita bisa gunakan while dan kita juga bisa gunakan forEach misalnya untuk looping pada array forEach nya.

**// for..of**

Nah sekarang kita liat dulu bagaimana cara kerja dari 2 looping baru itu, yg pertama for..of yg kalo kita liat definisi nya *“Creates a loop iterating over iterable object.”* Jadi for..of ini adalah sebuah looping yg bisa mengulang atau menelusuri object-object yg iterable.

Nah object-object yg iterable ini ada banyak :

* String
* Array
* Arguments / NodeList
* TypedArray
* Map
* Set
* User-defined Iterables

Ada String lalu Array, terus ada object yg mirip dengan array yg dinamakan arguments dan Nodelist, arguments itu adalah array untuk menampung argument-argument pada function sedangkan NodeList adalah ketika kita melakukan query pada DOM nah hasilnya di sebut dengan nodeList.

Nah yg lain nya ada typedArray, map, set, dan User-defined iterables atau iterable yg kita bikin sendiri, nah di video kali ini kita hanya membahas String, Array, dan array like object atau object yg mirip array yaitu Arguments dan NodeList.

LETS CODE!!

**// for..of**

Misalnya saya punya kasus sebuah array mahasiswa yg isi nya nama-nama mahasiswa berbentuk array, cerita nya saya mau menampilkan ke console hanya nama” nya saja, kan banyak ya cara nya kalian bisa pake for biasa, gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

for (let i = 0; i < mhs.length; i++) {

    console.log(mhs[i]);

}

Nah seperti itu kalo pake for biasa gaada masalah, lalu kita juga bisa pake forEach() yg kalian tau ini sebuah method yg ada pada prototype array, gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

mhs.forEach(*m* => console.log(*m*));

Hasilnya sama dengan menggunakan for biasa tapi forEach() jauh lebih simpel. Nah sekarang karena array itu termasuk salah satu iterable object maka pasti kita bisa gunakan for..of ya, jadi cara nulis nyakalian tulis for lalu kurung yg didalamnya kita representasikan tiap-tiap element pada array nya sebagai variable baru apa misalkan kita bisa gunakan const.

Nah di for..of itu kita bisa gunakan const karena variable m itu tidak akan berubah berbeda dengan I di for biasa, jadi bikin nya const nama variable bebas lalu of dan kasih nama array nya, gini :

for (const m of mhs) {

}

Jadi cukup nulis seperti itu aja lalu tiap-tiap element kita representasikan sebagai m dari array mahasiswa jadi hampir mirip seperti forEach() kalo menurut saya ini masih awala ya. Nah sehingga kita bisa langsung tulis console.log(m) disitu nya :

for (const m of mhs) {

    console.log(m);

}

Jadi hasilnya sama aja begitulah ada beberapa cara untuk melooping isi dari array, nah salah satunya karena array mhs itu iterables maka pasti bisa pake for..of itu, tapi kalo menurut kalian lebih simpel pake forEach() itu silahkan aja ya tapi tau bahwa sekarang kita bisa pake for..of.

/**/ Looping string for..of**

Nah string merupakan iterables nah ini merupakan perbedaan menggunakan forEach(), forEach itu gabisa digunakan untuk looping string karena dia bukan array forEach khusus untuk array.

Nah contoh nya misal saya punya variable namanya nama di isi ‘Sandhika’, lalu kalo saya pake for..of, nah tulis variable const lalu nama variable bebas misalnya n itu buat repsentasi dari satu buah elemen tiap-tiap pengulangan string nya jadi nanti variable n itu akan berisi huruf string nya.

Jadi kalo kita console.log(n) nya maka dia akan melooping tiap-tiap kharakter string nya secara otomatis gayss, gini :

const nama = 'Sandhika';

for (const n of nama) {

    console.log(n);

}

Jadi itu salah satu cara buat ngelooping string, lalu selanjutnya klo kita mau coba pake forEach() itu ga akan bisa ya.

Nah sekarang kalo kita kembali ke array lalu misalkan kalian pake forEach(), gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

mhs.forEach(*m* => console.log(*m*));

Kalo saya tulis m aja maka m itu akan merepsentasikan elemen” pada array nya ya, tapi kalo saya mau ngambil index nya misalkan ‘Sandhika’ itu index ke-0 ya. Nah forEach() punya parameter tambahan jadi ada parameter ke-2 untuk menjadi index nya, gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

mhs.forEach((*m*, *i*) => console.log(*i*));

Jadi kalo saya console.log(i) gitu yg tampil index nya, sehingga kalo saya mau tulis jadi sebuah string, gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

mhs.forEach((*m*, *i*) => {

    console.log(`${*m*} adalah mahasiswa ke-${*i* + 1}`);

});

Nah kalo pake forEach() bisa ya looping seperti itu. Nah tap I kalo pake for..of secara default dia ga punya index, gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

for (const m of mhs) {

    console.log(m);

}

Nah kalo jaya gitu kalian ga dapet index nya Cuma dapet string nya aja, nah ngakalin nya kalo kalian tetep mau pake for..of itu kalian bisa pake sebuah method yg namanya entries(), gini :

const mhs = ['Sandhika', 'Doddy', 'Erik'];

for (const m of mhs.entries()) {

    console.log(m);

}

Jadi sekarang kalo liat hasil console nya si m itu isi nya ada 2, isi nya index dan string nya itu jadi kalo kalian mau langsung dapetin 22 nya kan itu isi m nya di awalin dengan index ya, kalian langsung getstruktur aja langsung tangkep satu variable sebagai index dan satu variable sebagai nama.

Jadi kalo masih inget kita bisa pake kurung jadinya kita menggunakan destructuring dong, gini:

const mhs = ['Sandhika', 'Doddy', 'Erik'];

for (const [i, m] of mhs.entries()) {

console.log(`${m} adalah mahasiswa ke-${i + 1}`);

}

Nah jadinya hasilnya sama saja gitu karena kita ketolong sama kurung siku dengan menggunakan DESTRUCTURING.

**// NodeList** .

NodeList ini kan ketika kita melakukan query pada DOM ya, cerita nya kita mau tambahin list item yg nantinya kita mau tangkep pake query selector jadi biar gampang kita kasih class, gini:

    <ul>

        <li *class*="nama">Sandhika</li>

        <li *class*="nama">Doddy</li>

        <li *class*="nama">Erik</li>

    </ul>

Nah kita mau ngambil semua li jadinya pake querySelectorAll ya, gini :

const liNama = document.querySelectorAll('.nama');

console.log(liNama);

Jadi kalo kita console kalian bisa liat li nya berbentuk NodeList bukan array ya, walaupun keliatan nya seperti array ada index nya ada isi nya tapi itu nodeList, nah ini kalo kalian menggunakan browser terbaru saat ini untuk looping NodeList ini kita bisa pake forEach() walaupun sebetul nya khusus untuk array, nah tapi kalo misalkan kalian pake browser versi lama itu ada aja yg ga mendukung forEach().

Jadi kita harus paksa si variable liNama jadi array murni dulu, nah tapi sekarang kalian bisa pake forEach(), gini :

const liNama = document.querySelectorAll('.nama');

liNama.forEach(*n* => console.log(*n*.textContent));

Nah disitu saya pake textContent ya itu bisa juga pake innerHTML. Nah kalian juga bisa menggunakan for..of, nah kalo for..of langsung bisa karena nodeList merupakan iterables, gini:

const liNama = document.querySelectorAll('.nama');

for (n of liNama) {

    console.log(n.innerHTML);

}

Nah jadi hasil nya sama persis gitu ya jadi bisa digunakan untuk NodeList.

**// Arguments**

Di arguments ini adalah yg ada pada sebuah function, misalkan saya mau bikin sebuah function namanya jumlahkanAngka() jadi ceritanya function nya akan menjumlahkan angka yg ada pada parameter ketika dipanggil.

Jadi kita ga akan kasih argument di situ nya, saya mau nya waktu saya panggil gini :

function jumlahkanAngka() {

}

jumlahkanAngka(1, 2, 3, 4, 5);

Nah saya mau hasilnya itu penjumlahan 1+2+3+4+5, kalo saya kirim parameter gitu harusnya kan disitu nya ada argument yg menangkap nya kan ( argument beda dengan arguments ) nah kalo misal kita ga tangkap pake argument disitu nya secara default argument nya ke tampung ke variable namanya arguments itu otomatis.

Nah kalo kita coba console arguments nya :

function jumlahkanAngka() {

    console.log(**arguments**);

}

jumlahkanAngka(1, 2, 3, 4, 5, 6);

Hasil di console nya terlihat seperti array padahal dia arguments bukan array, nah gimana kalo kita mau jumlahin itu, sebetulnya kalo pake array itu gampang banget tinggal reduce() aja masih inget kan higher Order Function, gini :

function jumlahkanAngka() {

    return **arguments**.reduce((*a*, *i*) => *a* + *i*);

}

jumlahkanAngka(1, 2, 3, 4, 5, 6);

Nah kalo kaya gitu kan kita pake arguments ya bukan array jadi hasilnya eror tapi kalo kita ganti jadi array itu udah bisa gitu deh.

Begitu pula kalo kita pake forEach(), misalnya kita gunakan variable bantuan yg awal nya 0 gini :

function jumlahkanAngka() {

    let jumlah = 0;

**arguments**.forEach(*n* => jumlah += *n*);

    return jumlah;

}

jumlahkanAngka(1, 2, 3, 4, 5, 6);

Nah hasilnya ternyata masih eror katanya forEach() bukan bagian dari arguments karena dia bukan array tapi kalo saya ganti arguments nya jadi array itu baru bisa. Jadi nya kalo arguments gabisa pake forEach().

Jadi kita bisa menggunakan for..of kita bisa tulis gini :

function jumlahkanAngka() {

    let jumlah = 0;

    for (a of **arguments**) {

        jumlah += a;

    }

    return jumlah;

}

console.log(jumlahkanAngka(1, 2, 3, 4, 5));

Jadi kita bisa looping arguments menggunakan for..of itulah beberapa cara menggunakan looping pada iterables.

**// for..in**

*“Creates a loop only iterating over* ***enumerable****.”* Link:

( <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in> )

Jadi ini hanya untuk enumerable, enumerable disini maksudnya adalah property pada object nah kalo tadikan didalam iterables itu tidak ada object ya karena object itu bisa di looping isi nya atau property nya menggunakan for..in.

Misalkan saya punya variable nmanya mhs sekarang dia bentuknya object literal, gini :

const mhs = {

    nama: 'Sandhika',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id'

}

Kalo misalkan saya mau melooping isi dari porperty nya itu kita gabisa pake for..of ya, kalo kalian pake for..of dia eror katanya si mhs itu bukan bagian dari for..of atau iterable object.

Nah kalo kita pake for..in baru mau :

const mhs = {

    nama: 'Sandhika',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id'

}

for (m in mhs) {

    console.log(m);

}

Dan hasil di console nya itu hasil looping nya bukan isi dari property si object itu, isi nya itu nama property itu sendiri atau enumerable yaitu nama, umur, dan email gitu.

Jadi kalo kalian mau looping nya property dari object pake nya for..in kalo mau mengulah isi array dan iterables object lain pake nya for..of, dan kalo kita mau ngambil value nya atau isi dari property object itu bukan property nya itu kita bisa panggil m nya sebagai index dari object jadi saya tulis nya gini :

const mhs = {

    nama: 'Sandhika',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id'

}

for (m in mhs) {

    console.log(mhs[m]);

}

Nah jadi yg ke ambil isi dari property array nya seperti ‘Sandhika’, 33, dan ‘sandhikagalih@unpas.ac.id’ gitu kalo m nya sja dia index nya.