**JAVASCRIPT LANJUTAN 4.2 Filter, Map & Reduce**

Kita akan membahas mengenai 3 contoh dari Higher Order Function yg paling sering digunakan menurut pak dhika, jadi 3 function itu merupakan prototype dari tipe data array :

* Array.prototype.map()
* Array.prototype.filter()
* Array.prototype.reduce()

Ya walaupun didalam nya bisa macem” ya bisa array of angka, bsia array of string atau bahkan array of object dan sebetulnya kalo kalian ngikutin seri javascript dasar kita sudah membahas beberapa dari itu ya, terutama fiterl tapi waktu itu kita membahasnya masih function biasa, kalo sekarang kita membahasnya menggunakan contoh arrow function. Nah contoh Higher Order Function lain misalnya seperti find, foreach, splice dll itu gaakan kita bahas.

Dan sekarang kita akan liat ilustrasi dari penggunaan masing-masing function itu, misalnya saya punya beberapa bahan makanan seperti (cek ss 1) anggap aaj bahan-bahan makanan itu nantinya kita masukan kedalam sebuah array dan ketiak array ini nantinya saya masukan ke Higher Order Function filter(), dan saya akan memfilter mana diantara bahan” makanan itu yg merupakan sayur.

Maka nanti function itu akan menghasilkan sebuah array baru yg isi nya sayu-sayur dari bahan makanan itu, jadi di array baru bahan makanan yg bukan sayur itu tidak akan terpilih, itu ilustrasi contoh penggunaan filter. <https://www.datasciencecentral.com/forum/topics/what-is-map-reduce>, <https://www.youtube.com/watch?v=e-5obmIG_Fy>. – sumbernya

Nah sekarang gimana kalo misalkan dengan menggunakan kasus yg sama bahan makanan yg sama kita akan gunakan fitur lain atau function lain, misalkan kita gunakan map(), nah dengan menggunakan map() itu artinya kita akan petakkan setiap element array nya kedalam sebuah function sendiri.

Nah contoh nya untuk kasus ini setiap element pada bahan makanan itu akan saya lakukan “potong”, jadi dengan memberikan callback function didalam map nya nanti adalah “potong“ maka semua element nya ketika masuk kedalam Heghir Order Function itu akan menghasilkan array baru berupa bahan makanan yg telah terpotong seperti ini (cek ss2).

Nah itu kalok kita menggunakan map() akan ada array baru ya walaupun ilustrasinya didunia nyata nya itu pada saat kalian potong itu tidak menghasilkan bahan makanan baru melainkan bahan makanan yg sudah terpotong.

Sebetulnya akan lebih tepat kalo kita ilustrasikan menggunakan forEach() karena map() dan forEach() itu mirip bedanya kalo forEach() itu tidak menghasilkan array baru tapi kalo map() itu menghasilkan array baru., bahan makanan yg sebelumnya yg belum dipotong itu sebetulnya masih ada, jadi anggap aja seperti itu lah jadi kita kasih setiap element pada array nya function yg sama, semua dipotong.

Nah hasil dari function map() itu bisa kita kasih Heghir Order Function berikutnya yaitu reduce(), nah kalo reduce() ini bisa lakukan menggabungkan ke semua bahan makanna itu untuk menjadi satu sandwich atau satu hotdog yg utuh begitu lah (cek ss3).

JADI ITULAH ILUSTRASI MAP, FILTER DAN REDUCUE, dan reduce() itu sama menghasilkan array baru dan nanti keren nya ke-3 Heghir Order Function itu kita bisa gabungkan menggunakan yg namanya method chaining itu nanti ya yg penting kalian punya gambaran untuk ke-3 fungsi tersebut.

IMPLEMENTASIKAN DIDALAM CODING

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

Saya mempunyai variabel angka yg berupa array yg didalam nya terdapat beberapa angka yg random, kita akan coba function filter() cerita nya saya pengen mencari angka yang lebih besar >= 3 jadi dari array diatas saya mau cari mana saja angka yg lebih besar dari angka 3.

Nah sebelum kita menggunakan filter() gimana kalo kita menggunakan looping biasa dulu menggunakan for seperti ini ya :

const newAngka = [];

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

    if (angka[i] >= 3) {

        newAngka.push(angka[i]);

    }

}

console.log(newAngka);

Jadi gitu kalo pake yg for nah itu kita pake const ga masalah karena yg diganti itu bukan yg semisal dari yg tadinya array jadi integer nah itu gabisa harus menggunakan let, ga perlu dijelasin lagi karena itu dasar nya ya. Nah kalo kita gunakan Filter gini :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.filter(function (*a*) {

    return *a* >= 3;

});

console.log(newAngka);

Nah kalo ini kita langsung panggil array mana yg mau kita filter(), array angka yg mau kita gunakan lalu parameter nya filter() itu berisikan function nah disitu ada CALLBACK karena function yg parameter nya berisi function lain gitu.

Nah function itu juga ada paramter (a) yg a nya itu kita tulis aja sembarang, sebenernya kita bisa tulis dia misal b atau c dan lain nya, tapi variabel nya kita gunakan anggapan sebagai angka[i] di dalam for sebelum nya, jadi itu buat mewakilkan semua isi array yg ada didalam array angka apa aja gitu.

Atau a itu variabel yg merepesentasikan tiap-tiap element didalam array angka, Nah untuk tiap-tiap element didalam array angka kita bisa return ( a >= 3 ) artinya jika a nya lebih besar sama dengan 3, jadi otomatis dia memfilter tiap element nya.

Jadi anggap saja filter() itu didalam nya berjalan seperti yg For punya bedanya itu udah menjadi fungsi gitu. Nah kalo kita console.log yg filter() itu hasil nya sama persis sama yg For tapi filter() lebih simpel aja.

Dan kalo kalian inget ada arrow function nah callback nya itu kita bisa gunain arrow function yg simpel itu, :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.filter(*a* => *a* >= 3);

console.log(newAngka);

Jadi kita bisa hilangkan kurung kurawal nya dan kurung biasanya juga lalu tidak perlu lagi tulisan return karena hanya ada satu baris begitu saja sangat simpel sekali. Nah itulah contoh sederhana dari filter().

**// MAP()**

Nah kalo untuk map() itu kan artinya kita akan memetakan tiap-tiap element yg ada didalam array nya menggunakan fungsi yg baru, misalnya saya mau kali kan semua angka dengan 2, nah itu artinya saya pengen mempunyai array baru yg isinya kali 2 semua element yg didalam array angka yg tadi itu.

Misalnya kita punya variabel kaya tadi newAngka, ini kita langsung aja pettakan map() nya dan langsung gunakan arrow function nya :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.map(*a* => *a* \* 2);

console.log(newAngka);

Nah ini artinya telusuri semua element pada angka lalu tiap-tiap element nya kalikan 2,nah maka nanti akan ada array baru coba itukan udah di console.log, dan itu tidak mengubah isi dari array angka nya karena yg keluar itu array baru. :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.map(*a* => *a* \* 2);

console.log(newAngka);

console.log(angka);

Nah jadi isi angka nya masih ada ga berubah, jadi itu map() petakkan semua element pada array dengan sebuah fungsi yg baru, fungsi nya bisa macem-macem bukan hanya ( a \* 2 ), misalkan bisa ( a + 2) atau nambahin karakter mau ngubah warna nantinya atau apapun itu ya.

**// REDUCE** .

Reduce itu untuk melakukan sesuatu terhadap seluruh elemen pada array nya, misalkan yg ingin saya lakukan itu array angka tadi saya jumlahkan semua isi nya, caranya sama kita baut variabel baru karena itu 3 function itu menghasilkan array baru ya.

Nah untuk reduce itu agak beda ya, reduce itu punya 2 argument nah kalo punya 2 argumen dia harus dikasih kurung di function nya ya, ini parameter nya ada 2 ya pertama ada yg disebut dengan accumulator, nanti nama parameter nya boleh bebas a atau b atau c bebas Cuma itu biar nandain aja namanya gitu buat mempermudah.

Parameter kedua nya currentValue atau argument array yg sedang di looping nya misalkan sekarang lagi looping di element yg mana sedangkan accumulator adalah hasil dari proses nya misalkan dijumlahkan berarti awal nya 0 jadi 0 tambah mines 1 nah -1 adalah currentValue, Jadi nya -1 si accumulator nya nah nanti currentValue nya ngeser secara berurut sesuai dengan isi array angka nya, kan tadi nya -1 terus ngegeser jadi 8 currentValue, terus tadi accumulator nya -1 jadi kalo dijumlakan -1 + 8 jadi nya 7 nah currentValue nya berarti 7.

Nah jadi hasilnya newAngka kalo dijumlahin jadi 26 harusnya, nah jadi yg terjadi anggapan nya gini :

*// -1 + 8 + 9 + 1 + 4 + -5 + -4 + 3 + 2 + 9*

Nah coba kalo kalian itung pasti hasilnya 26 kan.

Nah ini ada yg harus diperhatikan, secara default itu sebetulnya ada satu nilai disitu dengan menggunakan koma :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.reduce((*accumulator*, *currentValue*) =>

*accumulator* + *currentValue*, );

console.log(newAngka);

Nah itu ada koma di setelah accumulator + currentValue nya, itu adalah nilai awal nah kalo kalian ga tulis kan nilai awal disitu nah itu nilai awal nya 0 :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.reduce((*accumulator*, *currentValue*) =>

*accumulator* + *currentValue*, 0);

console.log(newAngka);

Jadi kalo dijalankan hasilnya tetap 26 ga berubah jadi sebetulnya ada 0 disebelum -1 nya gitu, nah nanti nilai awal nya itu bisa kita ubah ga 0 terserah kalian mau isi nilai awal berapa misalkan saya tulis 5. :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const newAngka = angka.reduce((*accumulator*, *currentValue*) =>

*accumulator* + *currentValue*, 5);

console.log(newAngka);

Jadi kalo dijumlahkan semua harusnya hasilnya 31, jadi nanti itu akan bermanfaat ketika kalian mau bikin nilai nya udah ada isi nya ga 0 gitu tapi berdasarkan history nilai user gitu tapi nilai default nya kalo ga kalian tulis itu 0 ya.

**// METHOD CHAINING**

Method chaining atau method berantai yg artinya kita bisa menggabungkan fungsi-fungsi pada Heghir Order Function itu dalam satu kali eksekusi gaperlu disimpan kedalam variabel dulu, misalnya saya punya khasus gini dari array yg kita punya saya mau mencari angka yg lebih besar dari 5.

Setelah itu hasilnya saya mau kalikan 3, jadi setelah ketemu nih angka berapa aja yg di atas 5 maka kalikan 3 baru terakhir saya mau di jumlahkan, kalian bisa lakukan satu” kaya tadi atau kita mau gunakan method Chaining.

Yang pertama kita cari dulu angka > 5 lalu kita filter element array nya nah disitu ga kita akhiri pake titik koma tapi kita chaining, jadi kalian bisa tambahkan titik baru kita masuk kedalam perkalian menggunakan map(), bisa juga menuliskan titik nya dibawah ya biar lebih rapih.

Nah kita pake map() langsung jadinya map lalu element yg udah ketemu yg a itu kita kalikan 3, nah hasil dari filter() itu ada angka 8, 9, dan 9 nah jadi angka itu dikalikan dengan 3 jadi nya hasil di map() itu 24, 27, 27. Nah terakhir jumlahkan semuanya itu pake reduce() inget parameter nya itu ada 2 dan kita bisa kasih nama nya bebas aja :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const hasil = angka.filter(*a* => *a* > 5) *// 8,9,9*

    .map(*a* => *a* \* 3) *// 24, 27, 27*

    .reduce((*acc*, *cur*) => *acc* + *cur*); *// 78*

console.log(hasil);

Jadi hasil akhir nya itu 78 gitu. Jadi itu kalo kita lakukan menggunakan method chaining, itu kalo kalian mau tulis berulang-ulang juga masalah ya, misalkan sudah ketemu 24,27,27 lalu kalian mau map() lagi sekarang dibagi 2 gini :

const angka = [-1, 8, 9, 1, 4, -5, -4, 3, 2, 9];

const hasil = angka.filter(*a* => *a* > 5) *// 8,9,9*

    .map(*a* => *a* \* 3) *// 24, 27, 27*

    .map(*a* => *a* / 2) *// 12, 13.5, 13.5*

    .reduce((*acc*, *cur*) => *acc* + *cur*); *// 39*

console.log(hasil);

Nah gitu mau ditambah filter() lagi lebih dari satu silahkan bebas.