**JAVASCRIPT LANJUTAN 4.1 Higher Order Function**

Materi ini cukup penting ketika kalian mau belajar mengenai modern javascript di video kali ini kita akan bahas dulu teori nya apa itu definisinya dan contoh nya seperti apa baru nanti di video setelah ini kita akan bahas implementasi dari Higher Order Function ini.

DEFINISI

“Function yang beroperasi pada function yang lain. Baik itu digunakan dalam argument, maupun sebagai value.” **-** [**https://eloquentjavascript.net**](https://eloquentjavascript.net) .

Kita simpan dulu ini definisinya karena didalam javascirpt itu sebuah function itu disebtu dengan First Class Function itu artinya inti dari javascript itu sebetulnya function, nah dimana function itu diperlakukan sebagai :

“Javascript memperlakukan function sebagai object.” **– sitepoint.com**

Dan kalian tau kalo object itu merupakan value sama seperti integer atau string misalnya ya, nah maka function juga bisa kita simpan sebagai argument maupun return value dari sebuah function yg lain, nah sebetulnya ini sudah kita coba ketika kita belajar mengenai closure.

CONTOH misalkan saya punysa sebuah function :

function kerjakanTugas(*matakuliah*, *selesai*) {

    console.log(`Mulsai mengerjakan tugas ${*matakuliah*} ...`);

    selesai();

}

Jadi ada function yg nerima 2 argument matakuliah dan selesai nah matakuliah nanti kita simpan berupa string, lalu argument ke-2 itu selesai yg merupakan function isi function nya gampang aja cetak ‘Mulai mengerjakan tugas…..’ lalu nama matakuliah nya apa.

Dan setelah console.log dijalankan kita panggil function selesai() nya nah jadi diluar function mengerjakan tugas itu kita harusnya punyafunction yg nama nya selesai(), gini :

function kerjakanTugas(*matakuliah*, *selesai*) {

    console.log(`Mulsai mengerjakan tugas ${*matakuliah*} ...`);

    selesai();

}

function selesai() {

    alert('Selesai mengerjakan tgas!');

}

Jadi kalian bisa liat nih kita menggunakan function sebagai argument, dan ketika kita panggil caranya bisa begini :

function kerjakanTugas(*matakuliah*, *selesai*) {

    console.log(`Mulai mengerjakan tugas ${*matakuliah*} ...`);

    selesai();

}

function selesai() {

    alert('Selesai mengerjakan tugas!');

}

kerjakanTugas('Pemrograman Web', selesai);

Jadi kirim parameter pertama nya yaitu tugas parameter kedua nya itu function, jadi ketika kita jalankan diconsole akan ada tulisan ‘Mulai mengerjakan tugas Pemrograman Web …’ dan di browsernya tampil alert , nah itu hanya contoh aja ketika kita menggunakan function kita bisa menyimpan function lagi sebagai argument.

Nah kalo sudah seperti itu function kerjakanTugas itu sudah bisa kita sebut Higher Order Function, dan jika kalian punya function sebagai argument artinya itu kita sebut dengan **CALLBACK** seperti argument selesai itu.

Jadi function yg kita simpan sebagai argument itu disebut callback, Sedangkan function yang memiliki callback sebagai argument nya itu kita sebut dengan Higher Order Function, simpel nya jika function itu terdapat callback berarti dia Higher Order Function.

CONTOH LAINNYA :

setTimeout(function () {

    console.log('Hello World!');

}, 1000);

Kita menggunakan setInterval atau setTimeout yg mempunyai 2 parameter, parameter pertama itu function parameter ke2 itu integer untuk menyimpan milisecond waktu tunggu nya kalo misalkan setTimeout() kalo setInterval berapa milisecond interval nya.

Nah tapi kalian bisa liat parameter pertama nya sebuah function atau isitilah nya itu callback sehingga setTimeout() dan setInterval() bisa kita sebut dengan Higher Order Function.

CONTOH LAINNYA 3 :

const tombol = document.querySelector('.sumbit');

tombol.addEventListener('click', function () {

    console.log('tombol ditekan!');

});

Kalo kalian mau buat sebuah tombol ketika tombol itu diklik mengeluar kan aksi tertentu gitu. Nah kalo menggunakan DOM itukan pake addEventListener nah itu juga bisa kita sebut Higher Order Function.

CONTOH LAINNYA 4 ::

function ucapkanSalam(*waktu*) {

    return function (*nama*) {

        console.log(`Halo ${*nama*}, Selamat ${*waktu*}, semoga harimu

menyenangkan`);

    }

}

let selamatMalam = ucapkanSalam('Malam');

console.dir(selamatMalam('Sandhika'));

Ini adalah function yg return value nya adalah function lagi nah ini udah pernah kita coba di video closure ya. Saya punya function ucapkanSalam() yg memiliki argument waktu lalu didalam function nya kita return function lagi yg menampilkan console.log.

Nah sekarang function ucapkanSalam juga bisa kita sebut Higher Order Function karena didalam nya memliki return value yg juga function.

**KENAPA?**

Kenapa kita harus menggunakan Higher Order Function kenapa kita gabikin function biasa aja, ada beberapa alasannya. :

* Abstraksi, untuk membuat agar kode yang kita buat itu bisa lebih sederhana atau bisa lebih simpel gitu, karena dengan menggunakan function itu artinya kita menyembunyikan ke rumitan. Karena katanya *“Semakin besar sebuah program, semakin tinggi kompleksitasnya, semakin membingungkan programmernya.”* . Jadi semakin banyak juga kode nya apalagi kode nya disimpan dalam satu blok persedural yg sama misalnya nah itu akan semakin membingungkan, nah ada baik nya kita pisahkan menjadi function-function sendiri.

Dan ketika kita punya function didalam function nah itu harusnya akan membuat program nya semakin simpel, nah terus ada anekdot seperti ini : *“Ada dua cara untuk merancang sebuah software: Cara pertama adalah untuk membuat programnya se-sederhana mungkin sehingga jelas-jelas tidak ada kekurangannya, dan cara lainnya adalah untuk membuat programnya se-kompleks mungkin sehingga tidak ada kekurangan yang jelas. Mungkin banyak kekurangan nya tapi kekurangan nya gajelas.”* **-C.A.R Hoare, 1980 ACM Turing Award Lecture** , nah dari situ kita bisa lihat bahwa semakin simpel programnya semakin kecil kemungkinan nya ada kesalahan didalam nya.

CONTOH misalnya saya punya program sederhana yg buat menghitung sebuah angka :

let total = 0, count = 1;

while (count <= 10) {

    total += count;

    count += 1;

}

console.log(total);

Nah dengan menggunakan while kita bisa tulis program nya seperti itu , Nah coba kita bedain kalo kita tulis nya begini :

console.log(sum(range(1, 10)));

Nah seperti itu, kalo kita jalankan itu akan eror ya, tapi coba bayangkan aja ada definisi function yg namanya sum dan ada definisi function namanya range, nah ini ceritanya didalam program kalian punya function sum dan function range.

Nah tapi bayangkan ketika kalian memanggil dua function itu bersamaan nah jadi ada function yg jadi parameter dari function yg lain nah itu lebih simpel cukup satu baris aja karena kerumitan nya kita sembunyikan didalam function.

CONTOH LAINNYA :

for (let i = 0; i < 10; i++) {

    console.log(i);

}

Misalnya kita punya looping seperti itu sederhana aja, keliatan nya simpel tapi gimana kalo saya ingin looping nya ga 10kali misal 100kali atau 1000kali atau puluhan ribu berarti kita harus merubah angka 10 itu ya, jadi tidak dinamis nah gimana misalkan kita gunakan function gini :

function repeatLog(*n*) {

    for (let i = 0; i < *n*; i++) {

        console.log(i);

    }

}

repeatLog(10);

Jadi nulis for nya Cuma sekali terus kita ganti angka 10 nya jadi variabel n yg menjadi argument dari function repeatLog() sehingga kalo kita mau mengulang sebanyak apapun tinggal panggil repeatLog kasih parameter berapa mau 100kali juga bisa.

Nah jadi dengan menggunakan function itu abtraksi Nah tapi kita bisa bikin itu lebih efektif lagi, nah gimana kalo aksi yg dilakukan itu bukan console.log tapi saya punya aski yg lain misalnya tampil dalam alert , nah kita bisa bikin menjadi seperti ini :

function repeat(*n*, *action*) {

    for (let i = 1; i <= *n*; i++) {

        action(i);

    }

}

repeat(10, console.log);

repeat(3, alert);

Jadi sekarang function nya saya ganti namanya jadi repeat yg mempunyai 2 argument, n untuk menentukan aksi nya mau diulang berapa kali lalu yg kedua action yg berisi aksi apa yg mau dilakukan gitu, jadi parameter keduanya function jadi musibah itu bisa lebih efektif dengan menggunakan Hgher Order Function.

Dan alasan yang lain adalah dengan terbiasa kita membuat segala sesuatu nya kedalam function itu kita masuk ke sebuah pendekatan yg namanya FUNCTIONAL PROGRAMMING, tapi dengan melakukan pendekatan ke functional programming karena javascript bukan bahasa fungctional pemrograman murni tapi bisa kita buat menjadi sebuah bahasa Functional.

Dan ketika kita menggunakan paradigma itu dengan benar itu membuat program kita lebih efektif lebih aman dan lebih efisien lagi, terakhir ini ada contoh dari beberapa Higher Order Function terutama dari penggunaan array :

* Array.prototype.map(), untuk maping pada array
* Array.prototype.filter(), untuk menyeleksi isi array
* Array.prototype.reduce()