**JAVASCRIPT LANJUTAN | 6.2 Destructuring Function**

Sekarang kita akan liat bagaimana destructuring ini berpengaruh pada Function, langsung aja ke contoh misal nya saya punya sebuah function sederhana yg namanya penjumlahanPerkalian() dimana saya nanti punya 2 buah argument atau parameter yg isi function nya dia akan mengembalikan penjumlahan dari 2 buah nilai dan juga perkalian nya, nilai nya nanti kita akan kembalikan dalam bentuk array jadi supaya nilai nya lebih dari satu.

Jadi isi array yg pertama adalah a+b lalu array k-2 a \* b, gini :

function penjumlahanPerkalian(*a*, *b*) {

    return [*a* + *b*, *a* \* *b*];

}

console.log(penjumlahanPerkalian(2, 3));

Nah kalo kita liat hasil di console maka function itu akan menghasilkan angka 5 dan 6, lalu kalo tiap-tiap nilai nya mau saya simpan kedalam variable kan saya bisa lakukan gini ya :

function penjumlahanPerkalian(*a*, *b*) {

    return [*a* + *b*, *a* \* *b*];

}

const jumlah = penjumlahanPerkalian(2, 3);

Kalo saya console.log jumlah maka jumlah adalah array isi nya ada 2 buah array angka 5 itu dari penjumlahan nya dan angka 6 dari perkalian nya, kalo saya mau ngambil penjumlahan nya saaja maka saya bisa kasih index array disitu nya, gini :

const jumlah = penjumlahanPerkalian(2, 3)[0];

console.log(jumlah);

Maka begitu kita liat di console isi dari variable jumlah itu angka 5 karena kita kasih index disitu nya begitu pula dengan perkalian gini :

function penjumlahanPerkalian(*a*, *b*) {

    return [*a* + *b*, *a* \* *b*];

}

const jumlah = penjumlahanPerkalian(2, 3)[0];

const kali = penjumlahanPerkalian(2, 3)[1];

console.log(kali);

Maka yg akan tampil di console angka 6, agak ribet ya harus satu” begitu. Nah ini dengan menggunakan Destructuring kita bisa langsung aja membuat sebuah variable yg isi nya juga array nah array nya kita bongkar, tinggal begini :

function penjumlahanPerkalian(*a*, *b*) {

    return [*a* + *b*, *a* \* *b*];

}

const [jumlah, kali] = penjumlahanPerkalian(2, 3);

console.log(kali);

Jadi kita bisa juga menDestructurig value dari function, dan sekarang hasilnya ttp sama kaya tadi.

// Beda kasus

Kasus berikut nya gimana kalo return value nya bertambah ga cuman jumlah dan kali aja misalkan pengurangan sama pembagian gini :

function penjumlahanPerkalian(*a*, *b*) {

    return [*a* + *b*, *a* - *b*, *a* \* *b*, *a* / *b*];

}

Jadi nama function nya juga ikut berubah ya kita ganti jadi kalkulasi(), kalo gitu sama aja kita bisa bikin destructuring nya lebih lengkap lagi, gini :

function kalkulasi(*a*, *b*) {

    return [*a* + *b*, *a* - *b*, *a* \* *b*, *a* / *b*];

}

const [tambah, kurang, kali, bagi] = kalkulasi(2, 3);

console.log(kurang);

Nah sekarang gimana kalo misalnya yg Bagi gaada jadi begitu saya console.log(bagi) hasilnya akan undefined kan nah kalo kalian gamau lakukan itu kita bisa kasih nilai default ya, gini :

function kalkulasi(*a*, *b*) {

    return [*a* + *b*, *a* - *b*, *a* \* *b*];

}

const [tambah, kurang, kali, bagi = 'tidak ada'] = kalkulasi(2, 3);

console.log(bagi);

Jadi sekarang kalo di return value nya gaada dia muncul tulisan ‘tidak ada’, tapi kalo ada bagi di return value nya maka hasil nya ada, jadi kita bisa DESTRUCTURING ketika return value nya array.

Nah sekarang problem nya kalo misalkan kita mereturn nya array adalah ketika kita melakukan destructuring itu urutan indexnya harus pas ya gaboleh ketuker, jadi saya kalo tulis variable nya ga sesuai yg dimau misalnya di return nya itu tambah tapi nama variable nya jadi kurang gini :

function kalkulasi(*a*, *b*) {

    return [*a* + *b*, *a* - *b*, *a* \* *b*];

}

const [kurang, tambahs, kali, bagi = 'tidak ada'] = kalkulasi(2, 3);

console.log(kurang);

Jadi kalo saya console kurang karena nama variable nya ga sesuai urutan return nya yg harus nya 2-3 itu -1 ya malah jadi angka 5 karena kurang berasosiasi ke return tambah nya jadi urutan sangat berpengaruh.

Nah kalo misalkan kalian ingin urutan ga berpengaruh return value nya bisa kita ganti jadi object jadi jangan array return nya, gini :

function kalkulasi(*a*, *b*) {

    return {

        tambah: *a* + *b*,

        kurang: *a* - *b*,

        kali: *a* \* *b*,

        bagi: *a* / *b*

    };

}

Jadi sekarang return value nya jadi object bukan array lagi, nah maka kalo object urutan ga berpengaruh kalian mau urutan pertama bagi dulu lalu ke-2 tambah itu ga masalah karena dia sudah terasosiasi nya dengan nama property nya bukan lagi urutan nya, gini :

function kalkulasi(*a*, *b*) {

    return {

        tambah: *a* + *b*,

        kurang: *a* - *b*,

        kali: *a* \* *b*,

        bagi: *a* / *b*

    };

}

const { bagi, tambah, kali, kurang } = kalkulasi(2, 3);

console.log(kurang);

Jadi begitu kita console log kurang maka hasilnya norma yaitu -1 walaupun kurang itu kita taro di akhir urutan nya itu ga masalah padahal dia kalo di property nya urutan ke-2 jadi urutan kalo di object ga berpengaruh.

**Jadi kalo ingin urutan nya ga berpengaruh bisagunakan object destructuring pada return value nya.**

/**/ Destructuring function arguments** .

Sekarang kita akan liat gimana kita destructuring atau kita melakukan distruktur pada argument kalo tadi kan destructuring function return value ya. Misal nya gini saya punya mhs1 karena mungkin aja kalian punya data mahasiswa nya banyak, gini :

const mhs1 = {

    nama: 'Sandhika Galih',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id'

};

Ini kita belum masuk kedalam destructuringnya ya, ceritanya saya mau bikin sebuah function untuk mencetak sebuah string berdasarkan property yg ada pada object itu, simpel aja saya mau ngambil nama dan umur aja ya, gini :

function cetakMhs(*nama*, *umur*) {

    return `Halo, nama saya ${*nama*}, saya berumur ${*umur*} tahun`;

}

Nah function itu kalo kita jalankan itukan baru deklarasi nya aja ya, kalo saya jalankan saya bisa panggil cetakMhs() dengan parameternya property dari object mhs1 jadi saya nulis nya gini :

function cetakMhs(*nama*, *umur*) {

    return `Halo, nama saya ${*nama*}, saya berumur ${*umur*} tahun`;

}

console.log(cetakMhs(mhs1.nama, mhs1.umur));

Jadi nama dan umur ngambil dari property object mhs1, nah itu kalo kita pake function declaration biasa parameternya kita pecah sendiri saat ngecetak atau kalo mau cara yg lain kalian bisa kiriminnya object, jadi saya bisa kasih di parameter nya object, gini :

const mhs1 = {

    nama: 'Sandhika Galih',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id'

};

function cetakMhs(*mhs*) {

    return `Halo, nama saya ${*mhs*.nama}, saya berumur ${*mhs*.umur} tahun`;

}

console.log(cetakMhs(mhs1));

Jadinya kita pecahkan nya langsung di return jadi mhs1 mengirimkan object ke parameter mhs baru di bongkar dan yg diambil hanya nama dan umur.

Nah sekarang kalo kalian mau pake destructuring supaya disitu nya di return nya tetap nama dan umur supaya lebih simpel dan ga dipecah ya, jadi kalian kirimkan object lalu destructuring di argument nya.

Jadi pada saat dikirim ke argument kita langsung pecahkan, gini :

function cetakMhs({ *nama*, *umur* }) {

    return `Halo, nama saya ${*nama*}, saya berumur ${*umur*} tahun`;

}

console.log(cetakMhs(mhs1));

Nah jadi hasilnya sama aja gitu. Nah itu untuk mempermudah penulisan kalo misalnya object nya nanti kompleks ya jadi kita pecah langsung di argument nya supaya didalam function nya kalian bisa pake nama variable nya.

Nah itu akan kepake banget ketika object nya kompleks ya, contoh nya saya tambahin property nilai di object mhs1 nah nilai itu isi nya object juga, jadi ada object didalam object gitu dan ini sebetulnya bisa lebih kompleks lagi ya kalian bisa mulai dari mata kuliah baru didalam matakuliah ada komponen nilai dan seturusnya.

Nah untuk kasus kita nilai ini punya 3 komponen ada tugas, uts, dan uas. Gini :

const mhs1 = {

    nama: 'Sandhika Galih',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id',

    nilai: {

        tugas: 80,

        uts: 85,

        uas: 75

    }

};

Nah ini kalo misalkan saya mau nambahin return value nya gini :

function cetakMhs({ *nama*, *umur*, *nilai* }) {

    return `Halo, nama saya ${*nama*}, saya berumur ${*umur*} tahun, dan nilai uas saya adalah ${*nilai*.uas}`;

}

console.log(cetakMhs(mhs1));

Nah kan harusnya gitu ya jadi ngambil komponen uas kek gitu, tapi itu ribet lagi ada titik yg ngambil property disitu nya, nah gimana kalo didalam situ pun kita destructuring lagi jadi ini kaya destructuring bersarang gitu ya.

Kan itu object nya udah kita destructuring si nilai itu kan object nah property nya kita destructuring lagi jadi cara nya kalian bisa tambahkan titik 2 ( : ), lalu kurung kurawal di isi property yg ada didalam nilai nya, jadinya {tugas, uts, uas} gini :

const mhs1 = {

    nama: 'Sandhika Galih',

    umur: 33,

    email: 'sandhikagalih@unpas.ac.id',

    nilai: {

        tugas: 80,

        uts: 85,

        uas: 75

    }

};

function cetakMhs({ *nama*, *umur*, nilai: { *tugas*, *uts*, *uas* } }) {

    return `Halo, nama saya ${*nama*}, saya berumur ${*umur*} tahun, dan nilai uas saya adalah ${*uas*}`;

}

console.log(cetakMhs(mhs1));

Jadi nilai kita pecah lagi jadi begitu sehingga gaperlu nulis ( nilai.uas ) lagi cukup tulis uas aja atau nama variable nya. Jadi ini lebih simpel ya langsung to the point.

Ini cukup penting dan sering dipakai ketika kalian pake front end framwork seperti react, vue dll.