**JAVASCRIPT LANJUTAN | 8.1 Spread Operator**

Di video kali ini kita akan membahas Spread Operator vs. Rest Parameter, kedua hal itu di reprentasikan dengan kharakter 3 titik seperti ini ( **…** ) jadi meskipun notasinya sama tapi kedua hal tadi sebetulnya berbeda, dan sebetulnya kita udah pernah mencoba menggunakan notasi 3 titik itu ya di beberapa video sebelum nya.

**// Spread Operator**

Kita mulai di video kali ini dengan yg pertama yaitu Spread Operator, definisi :

*“Memecah atau isitilah nya (expand / unpack), iterables menjadi single element.”* . link :

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax> .

Yang tadinya iterables yg didalam nya banyak isi nya dipecah menjadi single element, kalo kalian tau iterables itu ada banyak ya kita sudah bahas beberapa di video sebelum nya.

Nah cara memecahnya kita langsung liat aja contoh kasus berikut ini :

**//Spread Operator // memecah iterables menjadi single element**

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

Inget array itu termasuk iterables dan array itu bisa kita destructuring nah kalo kita destructuring tiap” elment nya itu langsung masuk kedalam variable tapi kalo menggunakan spread Operator.

Kalo saya console.log(mhs); itu yg muncul pasti array nya tapi kalo saya tambahkan notasi 3 titik didepan nya gini :

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

    console.log(...mhs);

Maka yang tadi nya arrya itu dipecah di tiap-tiap element nya jadi hasil nya seperti string gitu coba sendiri aja karena array iterables dia di pecah.

Nah sekarang liat bedanya ketika saya masuk kedalam array misalkan saya ambil array index ke-0, 0 itu string ‘Sandhika’ inget string juga merupakan salah satu dari iterables, sehingga kalo string kita pecah pake spread Operator, gini :

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

    console.log(...mhs[0]);

hasil string nya begini ( S a n d h i k a ) jadi dipecah ya masingg-masing menjadi satu huruf terpisah.

Nah kapan kita menggunakan Spread Operator ini ada beberapa contoh ya, yang pertama kita bisa menggabungkan 2 array jadi kalo misalkan saya punya array namanya mahasiswa lalu saya juga punya array yg namanya dosen misalnya. Gini :

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

    const dosen = ['Ade', 'Hendra', 'Wanda'];

contoh nya seperti itu aja nah saya pengen bikin satu buah array lagi yang menggabungkan 2 array itu ya, Nah di isi dengan sebuah array yg didalam nya kita spread array mhs digabungkan dengan kita spread array dosen, gini :

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

    const dosen = ['Ade', 'Hendra', 'Wanda'];

    const orang = [...mhs, ...dosen];

    console.log(orang);

Jadi nanti array orang itu isi nya array mhs dan donsen keren banget. Tapi kalo kalian tulis nya gini :

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

    const dosen = ['Ade', 'Hendra', 'Wanda'];

    const orang = [mhs, dosen];

    console.log(orang);

Nah kalo begitu salah ya karena nanti jadinya array orang berisi array didalam array itu bukan di gabungin. Nah sebetulnya kita juga bisa pake cara yg lain dengan menggunakan sebuah method array concat(), kalo kalian liat didalam method array itu ada yg namanya concat() ya.

Nah kalo pake concat() kalian bisa gini :

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

    const dosen = ['Ade', 'Hendra', 'Wanda'];

    const orang = mhs.concat(dosen);

    console.log(orang);

Nah menggunakan concat() hasilnya sama dengan menggunakan Spread Operator, keliatan nya lebih simpel concat() ya. Tapi dengan menggunakan Spread Operator itu kita bisa lebih fleksible jika ingin menambahkan element array di tengah” antara mhs dan deosen.

Misalnya saya mau gabungin antara mhs dan dosen tapi di tengahnya saya mau gabungin elemen array lain, nah kalo pake spread operator bisa kalo pake concat() ga semudah ini :

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

    const dosen = ['Ade', 'Hendra', 'Wanda'];

    const orang = [...mhs, 'Aji', ...dosen];

    console.log(orang);

Nah si ‘aji’ masuk kedalam array orang jadi lebih fleksible ketika kalian menggunakan spread. Jadi itu penggabungan nya kalian bisa menggabungkan 2 atau lebih array, kalo kalian mau gabungkan 3 array juga bisa silahkan.

**// Meng-copy nilai pada array**

Jadi saya punya array mahasiswa yg tadi lalu saya mau bikin array baru yg isi nya sama dengan array mahasiswa itu, misalkan kalian punya lagi variable baru misalnya namanya mhs1 yg isi nya mhs, gini :

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

const mhs1 = mhs;

Nah harapan nya variable mhs1 isi nya sama dengan variable mhs nah kalo kita console.log(mhs1) hasilnya sama ternyata isi nya sama variable mhs. Kaya nya udah oke nih dengan begitu aja kita bisa meng-copy in isi array nya jadi saya punya variable yg isi nya sama.

Nah tapi problem nya kalo melakukan seperti itu dia tidak benar-benar melakukan copy, coba kalo kita bikin mhs1 element ke-0 itu kita timpa dengan elemen lain misalnya ‘Fajar’ gitu, nah harapan nya yg berubah hanya array mhs1 saja, jadi kalo saya console :

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

    const mhs1 = mhs;

    mhs1[0] = 'Fajar';

    console.log(mhs1);

nah betul tuh hasil nya index ke-0 nya keganti jadi “Fajar”, nah tapi ternyata kalo kita liat array mhs juga console.log(mhs); gitu berubah juga dia, kan seharusnya nilai nya tetap seperti yg awal ya karena itu yg kita isi [‘Sandhika’, ‘Doddy’, ‘Erik’].

Nah jadi gabisa meng-copy array seperti itu karena kalo kalian melakukan itu artinya kalian membuat referensi nya jadi tetep aja variable mhs1 nunjuk nya ke array utamanya si mhs jadi tidak benar-benar meng-copy.

Nah jadi cara nya buat variable baru yg isi nya array yg kita spread gini :

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

    const mhs1 = [...mhs];

    console.log(mhs1);

Jadi sekarang isi dari mhs1 sama persisi seperti variable mhs tapi itu copyan nya sehingga kalo kita ubah elemen ke-0 dari mhs1 jadi “Fajar” kita timpa, gini :

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

    const mhs1 = [...mhs];

    mhs1[0] = 'Fajar';

    console.log(mhs1);

kalo kita jalanin mhs1 berubah elemen pertama nya yg tadinya “Sandhika” menjadi “Fajar” dan kalo kita liat variable utama nya si mhs kita console.log(mhs); nah dia ga berubah isi nya masih tetap sama, gini :

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

    const mhs1 = [...mhs];

    mhs1[0] = 'Fajar';

    console.log(mhs);

Jadi isi array awal nya tetap tidak berubah sama sekali gitu. Jadi spread operator bisa juga digunakan untuk mengcopy array.

Kita coba contoh menggunakan HTML cerita nya didalam body kita mau bikin liast seperti baisa list nama-nama orang, gini :

    <ul>

        <li>Sandhika</li>

        <li>Doddy</li>

        <li>Erik</li>

    </ul>

Cerita nya sekarang saya ingin mengambil nama-nama nya pake a lalu saya simpan kedalam array, caranya kita harus ambil dulu element li nya itu, gini :

    const liMhs = document.querySelectorAll('li');

    console.log(liMhs);

Kalo kita console dia kan bentuk nya NodeList ya yg didalam nya ada 3 buah element, nah karena bentuk nya nodeList berartikan kita bisa looping nanti ya, terus kalo mau ambil isi dari teks nya itu didalam tiap-tiap element nya itu ada sebuah property textContent, jadi kalo saya tulis liMhs ke-0 textContent itu pasti “Sandhika” gini :

        const liMhs = document.querySelectorAll('li');

        console.log(liMhs[0].textContent);

Nah nanti yg di console.log nya itu akan saya masukan kedalam array, contohnya kalo mau pake looping biasa kan, kalian bisa gini :

    const liMhs = document.querySelectorAll('li');

    const mhs = [];

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

        mhs.push(liMhs[i].textContent);

    }

    console.log(mhs);

Nah sekarang hasilnya menghasilkan array jadi udah betul yg kita mau menjadi array. Sebetul nya kan itu bisa kita permudah ya kalian bisa pake higher order function map(), jadi kita bisa jalankan liMhs itu kedalam function map() gini:

const mhs = liMhs.map(*a* => *a*.textContent);

Nah kalo kit gitu aja sebetulnya bisa ya, tapi hasilnya eror karena map() itu khusus array nah yg kita ituin itu nodeList bukan array. Nah makannya karena dia bukan array kita ubah dia jadi array dengan cara kita spread disitu nya gini :

        const liMhs = document.querySelectorAll('li');

        const mhs = [...liMhs].map(*a* => *a*.textContent);

        console.log(mhs);

jadi yang tadinya isi nya NodeList kita spread yg ada titik 3 nya itu namanya spread, lalu kita simpan kedalam array, jadi sekarang bentuknya array begitu. Jadi hasilnya sama dengan menggunakan for biasa tapi dengan adanya map() dan spread jadi lebih praktis.

**//BEDA KASUS**

Misalnya didalam HTML nya saya punya h1 yg tulisan nya “Sandhika” h1 nya saya kasih class nama, gini :

    <h1 *class*="nama">Sandhika</h1>

Nah kita coba kasih style dulu untuk HTML nya ya :

    <style>

        body {

            background-color: salmon;

            text-align: center;

        }

        .nama {

            font-family: arial;

            font-size: 80px;

            color: white;

        }

    </style>

Nah nanti cerita nya saya pengen bikin supaya kalo saya hover pake kursor nanti akan ada animasi perhuruf, jadi kalo saya sorot atau hover kursor nya ke huruf s berubah perilaku nya gitu.

Saya bisa aja begini :

        .nama {

            font-family: arial;

            font-size: 80px;

            color: white;

            transition: .3s;

        }

        .nama:hover {

            transform: scale(2);

        }

Nah dia memperbesar ketika di hover tapi itu kata bukan satu huruf, saya pengen nya perhuruf kalo saya deketin huruf nya, kalo saya deketin hurufnya huruf nya aja yg jadi besar. Nah ini bisa kita lakukan dengan membuat string itu menjadi single element.

Jadi nanti kita spread string nya supaya jadi single elemen lalu kita sisipkan tag <span> misalnya kedalam situ, sehingga nanti tiap-tiap huruf nya akan di kelilingi oleh tag <span>, jadi nanti S nya ada <span> nya A ada <span> nya, tapikan kalo bikin manual ribet ya.

Nah jadi kita bisa melakukan itu dengan javascript, kita ambil dulu si namanya dulu ya, gini :

    const nama = document.querySelector('.nama');

Nah kao sudah di ambil kita pecah menjadi huruf, langsung aja gini :

const nama = document.querySelector('.nama');

        const huruf = [...nama.textContent];

        console.log(huruf);

Nah jadi ada array yg isi nya huruf dari string “Sandhika”, kalo udah gitu tinggal kita looping pake map() supaya tiap” element nya kita bungkus pake <span> cara nya kita map() lalu tiap-tiap huruf nya saya repsentasikan variable h yg mereturn span gini :

    const nama = document.querySelector('.nama');

    const huruf = [...nama.textContent].map(*h* => `<span>${*h*}</span>`);

    console.log(huruf);

Jadi hasilnya liat aja diconsole gitu, nah itu kan hasilnya array kalo kita mau jadi string aja kita pake join() pake string kosong, gini :

    const nama = document.querySelector('.nama');

    const huruf = [...nama.textContent].map(*h* => `<span>${*h*}</span>`).join('');

    console.log(huruf);

Nah tinggal si variable huruf itu kita tempelkan kita timpa ke variable nama itu, jadi caranya nama.innerHTML nya isi dengan huruf, gini :

    const nama = document.querySelector('.nama');

    const huruf = [...nama.textContent].map(*h* => `<span>${*h*}</span>`).join('');

    nama.innerHTML = huruf;

Jadi sekarang kalo ktia liat tampilan nya aja keliatan nya ga terjadi apa-apa, tapi kalo kita liat di insprect element disitu kita bisa liat tag-tag HTML kita apa aja ya, nah jadinya didalam h1 isi nya span semua.

Nah sekarang kalo udah gini gimana? Gampang tinggal CSS nya aja yg kita ubah jadi yg di hover itu span bukan nama nya, nah kalo udah diubah tapi ternyata gajalan itu karena span display nya inline sedangkan inline itu gabisa kita kasih style transform jadi biar jalan kita ubah display nya jadi inline-block, lal biar enak diliat kita kasih kursor pointer, gini :

        .nama {

            font-family: arial;

            font-size: 80px;

            color: white;

        }

        .nama span {

            display: inline-block;

            transition: .3s;

            cursor: pointer;

        }

        .nama span:hover {

            transform: scale(2);

        }