**JAVASCRIPT LANJUTAN | 5.3 Tagged Template Literals**

Definisi Tagged Template dari MDN Web Docs : *“Bentuk yang lebih kompleks dari Template Literals, memungkinkan kita untuk membaca template literals melalui sebuah function.”* Jadi kalo kita sudah membuat Template literal string tersebut bisa kita masukan kedalam function untuk mengubah bentuk dari string tadi. Lebih detail nya lagi di link berikut :

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

Oke mulai misal kita punya kasus seperti video sebelum nya dimana saya punya variabel namanya nama yg isi nya string biasa lalu ada variabel umur, gini :

const nama = 'Sandhika Galih';

const umur = 33;

Nah sekarang misalnya saya mau membuat sebuah template literals biasa ya, ini misal untuk merangkai string menggunakan expression dari variabel nama dan umur di atas, gini :

const str = `Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah sekarang bagaimana menjadikan template literals biasa itu menjadi Tagged template, nah caranya itu kita tinggal menuliskan nama function di depan string nya dan nanti string template literals itu akan masuk seolah-olah sebagai parameter dari sebuah fungsi tadi.

Contoh nya misalkan saya punya sebuah function nama nya coba() dan itu belum kita buat ya, jadi cerita nya string yg di variable str itu akan masuk kedalam fungsi coba(), gini :

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Jadi kalo saya buat function coba() nya, lalu kita return gini :

function coba() {

    return 'coba';

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah maka nanti string yg Halo itu akan masuk ke function nya dan function nya itu mengembalikan string ‘coba’, jadi hasil nya tulisan coba. Agak aneh emang ya string Halo nya yg udah kita tulis sama sekali tidak dipakai.

Tapi lihat ya secara default tagged template itu si coba() nya bisa menerima paameter atau argument, kalo misalkan saya tulis parameter pertama misal saya kasih nama parameter nya strings, nah kalo saya return strings nya, gini :

function coba(*strings*) {

    return *strings*;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Ohh ternyata hasilnya itu ARRAY tapi pas saya coba kita pakein kurung pas manggil function nya () nah itu hasilnya string yg keluar di console, tapi karena contoh di atas itu ga pake kurung buka dan kurung tutup nya hasil nya jadi ARRAY tapi itu bukan alesan utama nya keknya.

Nah ada sebuah array yg berisi 3 elemen lohh, elemen pertama berupa string tulisan nya ‘Halo, nama saya’, element ke-2 string ‘, saya’ element ke-3 string ‘tahun.’ Nah ini di ambil dari mana? Jadi ketika kita jalankan template literals itu maka Tagged templae nya akan mengecek tiap-tiap string yg ada didalamnya dan dipisahkan oleh expression nya yg ini ${} nah expression nya ga masuk ke string nya gitu.

Jadi itu yg pertama tiba-tiba ada parameter strings yg bisa kita pake, nah kalo gaada expression maka dia akan menjadi string aja satu index atau elemen, nah sekarnag expression nya masuk kemana? Nah dia masuk sebagai parameter berikutnya, gini :

function coba(*strings*, *nama*, *umur*) {

    return *nama*;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah maka kalo return nya saya tulis nama maka tampil nama nya dan kalo umur juga sama yg tampil umur nya. Nah itu karena kita punya 2 expression, nah kalo misalkan saya tamahkan lagi parameter email, gini :

function coba(*strings*, *nama*, *umur*, *email*) {

    return *email*;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah karena kita gapunya didalam templae literals nya email maka hasilnya udefined. Nah ada kemungkinan nanti ketika kita membangun sebuah string itu kita ngga tau ada berpa expression didalam template literals nya, jadi ktia ga mungkin punya parameter arg1, arg2, arg3 nah kita gatau kan nantinya ada berapa argument gitu.

Nah javascript versi baru atau modern javasript itu punya sebuah argument yg namanya rest parameter jadi dia bisa menampung seluruh expression yg ada didalam template literals nya, cara nulis nya kalian tulis titik 3 kali lalu nama variabel nya nama variabel nya bebas ya, gin:

function coba(*strings*, ...*values*) {

    return *values*;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah sebelum nama variabel nya ada titik 3 kali gitu disitu saya kasih nama variabel nya values nah nantinya values itu akan menjadi sebuah array yang isi nya semua expression yg ada didalam template literals jadi kalian gapelu ambil satu” lagi sebagai parameter jadi kalo saya jalanin values nya kita return gitu.

Nah maka yg tampil di console nya hanya expression dari template literal, Jadi gitu ya parameter strings atau parameter yg pertama itu untuk ambil string biasa nya seperti yg Halo dan kalo parameter ke-2 atau yg values itu untuk ngambil expression nya.

Nah kedepan nya kalo misal kita mau menampilkan string yg persis sama dengan kalo ktia langsung console.log seperti diatas artinya kita harus rangkai kita harus gabungkan parameter strings dengan parameter values nya untuk menampilkan strings yg persis sama.

**MENCOBA**

Jadi cara gampang nya gini deh kita bikin let namanya str untuk nampung strings nya awal nya isi string kosong gini :

function coba(*strings*, ...*values*) {

    let str = '';

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah lalu kita akan looping isi dari parameter strings nya nanti kita akan gabungkan dengan isi dari parameter values nya, kalian bisa pake forEach() atau map() tapi karena kita ga butuh array baru kita bisa gunakan forEach() aja, kalo pake map() kan bikin array baru ya.

Jadi saya bisa bikin strings.forEach() parameter forEach() itu bisa 2 ya, yg pertama adalah tiap-tiap elemen pada array strings nya mau kita representasikan sebagai apa misalnya str dan variabel sebelum nya yg let str ganti jadi result aja.

Lalu parameter kedua nya bisa ada index nya, index nya nanti buat ngelooping isi dari parameter values, terus kita gabungkan result dengan parameter str dan values yg values nya kita kasih index ke-i gini :

function coba(*strings*, ...*values*) {

    let result = '';

*strings*.forEach((*str*, *i*) => {

        result += `${*str*}${*values*[*i*]}`;

    });

    return result;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Hasil nya ternyata di akhir nya da undefined karena values terakhir itu gaada isi nya, dan parameter strings itu index nya sampe 3kali jadi nya dibuat 3kali dan yg ke 3kali nya gaada isi si values nya atau penjelasan singkat nya parameter strings itu selalu lebih besar satu dari pada parameter values.

Nah jadi pada saat strings terakhir terus dia harus tetep cetak value nya sedangkan disitu udah gaada expression lagi makannya keluar undefined, nah kita bisa akalin pake operator OR kasih string kosong gini :

function coba(*strings*, ...*values*) {

    let result = '';

*strings*.forEach((*str*, *i*) => {

        result += `${*str*}${*values*[*i*] || ''}`;

    });

    return result;

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Jadi itu artinya kalo values isi nya tampilin tapi begitu values nya gaada isi nya tampilin strings kosong supaya undefined nya ga keluar, nah hasil nya aman. Jadi string literal yg Halo dijalanin ke function coba menghasilkan string yg sama seperti yg dituliskan yg string Halo itu.

Mungkin samape sini agak aneh ngapain kita membuat Tagged tamplate untuk menghasilkan string yg sama dengan kita ngga pake Tagged tamplate, jadi gini :

HASIL NO Tagged tamplate : Halo, nama saya Sandhika Galih, saya 33 tahun.

HASIL YANG Tagged template : Halo, nama saya Sandhika Galih, saya 33 tahun.

Sama aja kan nah nanti kita liat manfaat nya. Sebelum kita ke situ kita lia dulu function coba itu kita lakukan cara yg gampang untuk membuat Tagged template, Nah sebetulnya itu ada cara gampang nya dengan menggunakan Higher Order Function yaitu reduce(), reduce itu bisa menggabungkan isi dari array.

**//MENGGUNAKAN reduce()**

Jadi kita bisa langsung return aja gini :

function coba(*strings*, ...*values*) {

    return *strings*.reduce((*result*, *str*, *i*) =>

`${*result*}${*str*}${*values*[*i*] || ''}`, ''

);

}

const str = coba`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Jadi kan reduce() punya 2 parameter ya nah disitu ada 3 parameter yg terakhir nya parameter ( I ) kan nah itu gunanya untuk index aja sisa nya 2 parameter itu sama seperti yg udah di jelasin Nah dan disitu juga kita kasih nilai awal karena itu reduce() biasanya kita kasih nilai awal nya ( 0 )tapi itu kan untuk angka ya kalo mau ngejumlahin, nah kalo sekarang kita awal nya string kosong ( ‘’ ) kaya variabel result yg di komentarin itu.

Nah sekarang kenapa kita cape-cape lakuin itu kalo hasil nya sama persis? Nah coba bayangkan contoh implementasi sederhana nya kalo kalian pengen menampilkan hasil sebuah string hasil pencarian berdasarkan keyword tertentu.

Misalkan kalian cari berdasarkan nama nah kalian pengen tampil hasil pencarian nya dengan nama yg kalian masukan itu sudah ada highlight, misalkan nanti tampil ‘Halo nama saya Sandhika’ tapi si sandhikan ya itu ada background color nya sehingga orang kalo liat di mudahkan string nya tampil berikut juga highlight nya.

**//MENCOBA Highlight**

Kita ubah jadi gini codingan nya :

const nama = 'Sandhika Galih';

const umur = 33;

function highlight(*strings*, ...*values*) {

    return *strings*.reduce((*result*, *str*, *i*) =>

        `${*result*}${*str*}${*values*[*i*] || ''}`, '');

}

const str = highlight`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Nah sekarang gini setelah ditampilkan saya pengen nama dan umur nya itu ada highlight nya nah nama dan umur itu dia kan values ya, jadi saya pengen nantinya disitu sebelum saya tampilkan saya bungkus dulu dengan sebuah tag span misal nya yg nama class nya hl untul highlight, gini :

const nama = 'Sandhika Galih';

const umur = 33;

function highlight(*strings*, ...*values*) {

    return *strings*.reduce((*result*, *str*, *i*) =>

        `${*result*}${*str*}<span class="hl">${*values*[*i*] || ''}</span>`, '');

}

const str = highlight`Halo, nama saya ${nama}, saya ${umur} tahun.`;

console.log(str);

Jadi nanti setiap tampil expression nya ada span nya walaupun diakhir ttp ada span yg kosong itu gapapa ya karena itu gaakan keliatan. HASIL NYA :

Halo, nama saya <span>Sandhika Galih</span>, saya <span>33</span> tahun.<span></span>

Jadi kalo saya sekarang punya didalam HTML nya itu sebauh style yg class hl tadi kita pake ya kasih background color nya terus saya mau nya tampil langsung di halaman bukan di console nya, gini :

CSS NYA :

  <style>

        .hl {

            background-color: salmon;

        }

    </style>

const nama = 'Sandhika Galih';

const umur = 33;

function highlight(*strings*, ...*values*) {

    return *strings*.reduce((*result*, *str*, *i*) =>

        `${*result*}${*str*}<span class="hl">${*values*[*i*] || ''}</span>`, '');

}

const str = highlight`Halo, nama saya ${nama}, saya ${umur} tahun.`;

document.body.innerHTML = str;

Nah lihat hasil nya ada tampil string nya yg expression nya sudah ada highlight nya yg background color salmon itu. Nah baru disini terlihat string nya tetap sama tapi perilaku expression nya berubah.

Dan ini keren nya kalo misalkan saya punya variabel baru misalnya email lalu saya tampilkan disitu nya, gini :

const nama = 'Sandhika Galih';

const umur = 33;

const email = 'sandhikagalih@unpas.ac.id';

function highlight(*strings*, ...*values*) {

    return *strings*.reduce((*result*, *str*, *i*) =>

        `${*result*}${*str*}<span class="hl">${*values*[*i*] || ''}</span>`, '');

}

const str = highlight`Halo, nama saya ${nama}, saya ${umur} tahun, dan email saya adalah : ${email} `;

document.body.innerHTML = str;

Nah otomatis email nya juga ikut ada highlight nya jadi itu adalah penggunaan sederhana tagged template.

**//Penggunaan lain?**

* Escaping HTML Tags
* Translation & Internationalization
* Styled Components

Kalo kalian tanya penggunaan lain dari Tagged template ada beberapa ya, pertama itu untuk Escaping HTML Tags atau ada yang bilang Sanitize HTML Tags untuk menghindari ketika ada script atau kharakter yang tidak di ingin kan pada halaman web kalian atau digunakan juga untuk Translation atau Internationalization atau isitilah nya ini ahli bahasa dari websaite kalian.

Yang terakhir yg saat ini sering digunakan untuk front end framwork yaitu untuk Styled Components misalnya kalian pake vue js atau react js, contoh nya adalah ketika kalian mau Escaping atau Sanitize HTML Tags misalnya kalian punya script seperti ini :

function sanitize(*strings*, ...*values*) {

    return DOMPurify.sanitize(aboutMe);

}

const name = 'petyr baelish';

const aboutMe = `I love to do evil <img src="http://unsplash.it/100/100?random" onload="alert('I hacked you. Haha');" />`;

const html = sanitize`

        <h3>${name}</h3>

        <p>${aboutMe}</p>

    `;

Jadi disitu kita punya variabel nama dan aboutMe yg nanti nya mungkin aja isi dari variabel 2 itu bukan kalian tulis sendiri tpai misalkan inputan dari user, Nah kalian gamau dong user nya memasukan script yg jhat contoh nya yg alert itu, onload itu script ketika halaman nya di load itu kali alert gamasalah sih tapi gimana kalo script yg lain yg jahat.

Nah untuk ngebersihin nya kita bisa lewatkan string yg sudah di rangkai itu yg isi dari variable const html itu, kedalam sebuah tagged template misalnya namanya sanitize nanti dijalankan lah si fungsi itu. Nah itu menggunakan liblary lain misalnya untuk melakukan sanitize nya.

Tapi tetep intinya adalah tagged tamplate yg string nya itu dilewatkan dulu masuk ke function sanitize disitu anggap lah dibersihin dulu isi nya dihilangkan alert yg jahat nya itu. Lebih detail nya kunjungi ini :

<https://codeburst.io/javascript-es6-tagged-template-literals-a45c26e54761>

**// Translation & Internationalization**

Selanjutnya untuk Translation atau internationalization mengubah bahasa dari websaite kalian contoh nya kalian punya template literal seperti ini :

const html = i18n`Hello ${name}, you have ${amount}:c in your bank account.`;

*// Hallo Steffen, Sie haben US$ 1,250,33 auf Ihrem Bankkonto.*

Nah ( i18n ) itu library nah itu internationalization ya sesuai dengan bahasa yg digunakan nanti otomatis di translate dan otomatis menjadi seperti yg Hallo Steffen itu. Nah untuk menggunakan itu ada sebuah library kalian bisa cek disini :

<https://github.com/skolmer/es2015-i18n-tag>

tapi intinya itu adalah tagged tamplate yg melewatkan semua string itu kedalam function nya itu yg (i18n).

**//Styled Components**

Misalkan kalo mau menjalankan styled components :

const Title = styled.h1`

    font-size: 1.5em;

    text-align: center;

    color: palevioletred;

`;

Misalnya kalo kalian bikin tag h1 nah nanti dilewatkan dulu kedalam tagged tamplate itu sehingga otomatis perilaku nya berubah dan untuk lebih lanjut cek :

<https://www.styled-components.com/docs/basics#getting-started>

dan contoh di atas itu adalah ketika kalian menggunakan react js.