**JAVASCRIPT LANJUTAN | 3.2 this pada Arrow Function**

Di video kali ini kita akan mencari tau konsep dari this di arrow function bagaimana, kalian tahu kan kalo kita menggunakan function entah itu function expression atau declaration itu didalam function nya memiliki konsep this.

Nah kalo kita mengubah function expression menjadi arrow function maka konteks this nya itu menjadi berbeda, untuk lebih jelas nya kita langsung liat contoh kode nya.

Sekarang saya mau kasih contoh jika kita membuat Constructor function, ini adalha salah satu cara kita ketika ingin membuat object pada javascript ya menggunakan Constructor function, yg sebenarnya Constructor function ini cara bikin nya sama dengan kita membuat function Expression.

Misalnya gini saya ingin membuat object mahasiswa seperti biasa ya huruf pertama nya huruf besar ya, lalu kita bikin function expression nya yg didalam nya kita cukup tulis kan misalnya saya mau bikin property nama dan umur pada mahasiswa gitu, gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

}

Nah kalo kita membuat sebuah constractor function nah maka jika kita menuliskan this didalam nya itu akan mengacu pada object nya si Mahasiswa itu, jadi misalnya saya console.log this nya dan sekarang kita coba instansiasi object nya menggunakan variabel baru yg namanya sandhika, gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

    console.log(**this**);

}

const sandhika = new Mahasiswa();

Nah maka kalo kita jalanin tampil object mahasiswa itu adalah kontesk this nya jadi begitu this dia mengacu ke object Mahasiswa nya.

Sekarang kalo kita punya sebuah method, misalnya sayHello yg isi nya function expression, gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = function () {

        console.log(`Hallo, nama saya ${nama}, dan saya ${umur} tahun.`);

    };

}

const sandhika = new Mahasiswa();

Nah terus kita panggil di console langsung sandhika.sayHello() nah disitu muncul tulisan eror nya, nah itu karena kalo kita mau manggil nama dan umur di method sayHello nya itu kita membutuhkan this, gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = function () {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

}

const sandhika = new Mahasiswa();

Nah baru ga eror karena nama dan umur nya udah ngambil ke this diluar method sayHello makannya itu perlu pake this kalo atas nya udah this. Kalo gitu gaada masalah kita buat sebuah function yg didalamnya punya this yg otomatis mengacu ke object Mahasiswa nya ketika di instansiasi.

Nah sekarang gimana kalo kita ubah itu semua menjadi arrow function, Nah untuk menggunakan arrow function kalian tidak bisa begitu saja mengubah semua function menjadi arrow, karena kalo kita ubah itu jadi arrow function, gini :

const Mahasiswa = () => {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = function () {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

}

const sandhika = new Mahasiswa();

Nah terjadi eror, jadi gabisa kalo mau bikin constractor function menggunakan arrow function, jadi gabisa semua function tiba” menjadi arrow function, nah tapi kalo method bisa method itu function yg ada di dalam object ya, gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

}

const sandhika = new Mahasiswa();

Tidak terjadi eror dan keliatan nya sama aja kalo pake arrow function dan function expression fungsi nya sama, Nah tapi sebetul nya pada saat kita menggunakan arrow function, arrow function sebenernya ga menyimpan konteks this. Jadi gaada konsep this didalam arrow function nya disitu dia akan cari keluar, keliatan nya gaada masalah dan sama aja mau menggunakan function expression atau arrow function ya.

Tapi coba lihat ketika kita bikin object nya itu tidak menggunakan constractor function melainkan object Literal originalnya cara bikin object, gini :

const mhs1 = {

    nama: 'Sandhika',

    umur: '33',

    sayHello: function () {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    }

}

Kalo saya mau manggil sayHello gausah instansiasi pake new seperti constractor function, saya hanya perlu tulis di console nya mhs1.sayHello() gitu, Jadi kalo menggunakan function this nya akan mencari ke lexical scope nya dia akan cari keluar ketemu nama maka di ambil yg property nama isi sandhika itu karena itu yg pertama kali dia temuin.

Nah tapi ini akan beda ketika saya ganti function nya menjadi arrow function, gini :

const mhs1 = {

    nama: 'Sandhika',

    umur: '33',

    sayHello: () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    }

}

Nah kalo saya panggil mhs1.sayHello() sekarang dia nilai nya undefined, karena sekarang this yg di function nya itu tidak mencari ke lexical scope nya karena sekali lagi harusnya si arrow function itu tidak memiliki this makannya dia nyari ke luar dari function nya ada this ga, dan ngga ada ya maka dia akan cari keluar dari object mhs1 itu ada this ga sampai ke window nah kalo di window gaada this.nama maka tampilkan undefined.

Nah kalo yg sebelum nya pake constractor function begitu dijalankan yg method sayHello() nya kan pake arrow function ya itu kan hasil nya bisa ga undefined, padahal sama pengecekan nya liat didalam method sayHello() ada this.nama ga nah gaada kan terus dia nyari ke atas ketemu yg this.nama di isi ‘Sandhika’ makannya hasilnya ada bukan undefined.

Makannya kalo pake arrow function buat method di object sedangkan this nya itu bersifat berbeda kalo di object literal punya method yg method nya arrow function didalam nya pake this itu akan terjadi undefined hasilnya, nah kalo untuk constractor function karena ini bisa pake this maka ketika buat method didalam object nya terus pake this dia masih berfungsi selama nama variabel nya sama dengan variabel atau property di dalam object itu, dan property itu adalah sebutan variabel di dalam object dan method itu adalah sebutan function di dalam object inget ini udah di ajarin ya.

Nah sekarang lanjut lagi, kalo saya console.log this kan harusnya this itu dia object nya si mhs1 ya nih :

const mhs1 = {

    nama: 'Sandhika',

    umur: '33',

    sayHello: () => {

        console.log(**this**);

    }

}

Nah sekarang kalo saya jalanin di console kita panggil mhs1.sayHello() nah itu hasil nya window bukan isi dari object mhs1, Jadi itulah kenapa arrow function tidak memiliki konsep this.

CONTOH LAINNYA

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

}

const sandhika = new Mahasiswa();

Itu kan menggunakan function expression ya karena function expression punya konsep this dia langsung nyari ke atasnya, sekarang saya mau kasih kasuss gini didalam constractor function itu saya mau bikin sebuah fucntion yg otomatis dijalanin tapi bentuk method.

Misalnya saya mau menggunakan sebuah fungsi namnya setInterval yg akan otomatis dijalankan pada interval tertentu, kan bikin nya itu didalam nya ada function anonymous yg kita kasih durasi dari interval nya yg durasi nya 500ms (mili second).

Jadi artinya saya mau dijalankan isi function nya setiap 500ms sekali, contohnya saya pengen setiap setengah detik umur saya nambah satu kan gampang nya gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

    setInterval(function () {

        console.log(**this**.umur++);

    }, 500);

}

const sandhika = new Mahasiswa();

Jadi setiap setengah detik umur nya nambah satu, kalo kita menggunakan function declaration nah setInterval pake itu function declaration ya, kalo function expression itu disimpan dulu kedalam variabel maka function itu ga akan kena hoisting.

Nah tapi begitu kita gunakan function declaration itu akan kena hoisting sehingga pada saat this nya di panggil dia ga ngecek ke this di atas nya tapi dia cek di global nya, jadi this yg di setInterval itu ga akan mengacu ke this.umur yg isi nya 33 itu karena terkena hoisting.

Nah tenyata hasil nya itu Nan (not a number) bukan angka itu karena yg di jumlahin bukan 33 tapi window nya yg dijumlahin, kita liat gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

    setInterval(function () {

        console.log(**this**);

    }, 500);

}

const sandhika = new Mahasiswa();

Jadi hasilnya diluar dari itu ngambil nya si window yg dia ambil, jadi kalo saya tulis this.umur ya gaada karena kita tulis this aja dia hasil nya aja window. :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

    setInterval(function () {

        console.log(**this**.umur);

    }, 500);

}

const sandhika = new Mahasiswa();

Nah this.umur itu hasilnya undefined, nah kalo kita tulis this.umur++ hasilnya Nan, nah itu scope this kalo kita bikin function declaration karena di hoisting nyari this didalam function setInterval nya gaada maka dia cari diluar itu karena berada diluar dan nyampelah di global.

Nah gimana cara mengatasi itu, nah cara mengatasinya kita bisa pake arrow function gitu :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

    setInterval(() => {

        console.log(**this**.umur++);

    }, 500);

}

const sandhika = new Mahasiswa();

Nah sekarang coba pahami logika nya, sekarang kalo ini dijalanin arrow function gapunya konsep this karena dia gapunya konsep this maka this nya akan mencari ke lexical scope nya keluar ke object nya, maka kalo kita console this di setInterval gini :

const Mahasiswa = function () {

**this**.nama = 'Sandhika';

**this**.umur = 33;

**this**.sayHello = () => {

        console.log(`Hallo, nama saya ${**this**.nama}, dan saya ${**this**.umur} tahun.`);

    };

    setInterval(() => {

        console.log(**this**);

    }, 500);

}

const sandhika = new Mahasiswa();

Jadi sekarang hasil nya di console yaitu isi object nya si Mahasiswa gitu, jadi this nya ngambil keluar situ bukan di global sehingga kalo saya tulis this.umur++ pasti dapet this.umur yg isi nya 33 jadinya setiap setengah detik angka 33 itu jadi nambah 1 gitu.

Jadi itu perbedaan konteks THIS pada ARROW FUNCTION.

**CONTOH REAL DARI ARROW FUNCTION**

Kita coba menggunakan element HML ya, kita buat sebuah kotak yg transisi :

*<!-- html nya -->*

    <div *class*="box"></div>

*/\* CSS NYA \*/*

\* {

    margin: 0;

    padding: 0;

}

.box {

    width: 100px;

    height: 100px;

    border: 1px solid black;

    margin: 50px auto;

    transition: .3s width,

        .3s height .3s,

        .3s margin-top .3s,

        1s background-color;

}

.size {

    width: 150px;

    height: 150px;

    margin-top: 25px;

}

.caption {

    background-color: black;

}

Kode nya sederhana aja, kita fokus ke .box dulu kalo kita lihat hasilnya itu ada persegi yg ukuran nya sesuai yg ditulis di .box, terus saya punya 2 buah class ada class size dan caption.

Ceritanya saya mau memberikan animasi ketika tombol di klik, cara ngasih animasinya kita awal nya punya class box kan di div nya nah nanti dengan menggunakan javascript saya akan kasih class yg namanya size.

Nah size itu gunanya untuk mengubah lebar dan tinggi persegi tersebut dan nanti di situ nya saya kasih animasi, jadi nanti lebar dan tinggi nya nambah tapi dikasih delay begitu supaya nunggu dulu animasi lebarnya selesai.

Jadi ketika diklik saya mau lebar dan tinggi nya berubah, nah nanti setelah nunggu 0.6 detik saya mau tambah kan lagi class baru namanya caption untuk memberikan animasi gradasi pada background color.

Nah sekarang yg akan kita lakukan adalah melakukan javascript DOM agar terhubung dengan kotak nya sehingga bisa diklik :

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

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

    console.log(**this**);

});

Nah kita cek this nya, kalo box di klik yg di console muncul tag html nya si box gini (<div class="box"></div>), nah kalo function nya itu gabisa di ubah arrow function karena sekali lagi arrow function itu gapunya konsep this, kalo kita pake arrow function nanti this nya itu yg muncul di console adalah window global nya.

Jadi saya pengen ketika diklik nambah class nya di div itu, karena kita udah tau this disitu adalah konteks nya .box maka kita bisa tulis untuk mengelola daftar class pada halaman nya menggunakan toggle :

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

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

**this**.classList.toggle('size');

});

Jadi kalo diklik tambahkan class size kalo diklik lagi hilangin class nya jika ada. Nah sekarang yg saya akan lakukan adalah saya akan menunggu selama 0.6 detik karena animasi nya berjalan 0.3s dan 0.3s jadi nya 0.6s baru saya akan kasih class caption disitu.

Disini saya bisa gunakan sebuah fungsi setTimeout() nah kalo setTimeout() ini kita nunggu dulu beberapa detik baru jalankan sebuah fungsi, gini :

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

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

**this**.classList.toggle('size');

    setTimeout(function () {

**this**.classList.toggle('caption');

    }, 600);

});

Nah kita klik tapi setelah diklik yg jalan hanya class size yg caption nya ga jalan, nah ini karena ketika kita menggunakan function maka this di setTimeout() itu dia akan berisi apapun yg ada diluar nya, nah karena setTimeout() itu dijalankan ketika hoisting maka dia posisinya ada di global.

Nah jadi kalo kita console.log(this) di dalam setTimeout nya kita bisa liat hasil nya itu Window nah jadi gaada tuh classList, harusnya this itu isi nya box nah cara ngatasinnya kalo dulu sebelum ada arrow function kita akalin, akalinnya dengan membuat sebuah variabel yg namanya apalah yg berisi this, gini :

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

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

    let that = **this**;

**this**.classList.toggle('size');

    setTimeout(function () {

*// this.classList.toggle('caption');*

        console.log(that);

    }, 600);

});

Jadi yg kita tulis di setTimeout() nya itu bukan this melain kan nama variabel yg berisi this diluar itu kek di atas. Nah tapi sekarang kita udah tau ada cara lain yaitu menggunakan arrow function, gini :

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

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

**this**.classList.toggle('size');

    setTimeout(() => {

**this**.classList.toggle('caption');

    }, 600);

Nah sekarang baru bisa ketika diklik baru ngaruh semua nya. Jadi sebenernya udah selesai sampe situ tapi saya mau ketika udah diklik terus diklik lagi saya mau yg jalan duluan itu class caption jadi warna hilang dulu terus baru size nya baru balik semula ke awal.

Kalo mau dibenerin kita bisa bikin gini aja :

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

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

    let satu = 'size';

    let dua = 'caption';

    let tempt;

    if (**this**.classList.contains(satu)) {

        satu = tempt;

        satu = dua;

        dua = tempt;

        dua = 'size';

    }

**this**.classList.toggle(satu);

    setTimeout(() => {

**this**.classList.toggle(dua);

    }, 1000);

});

Jadi saya buat 2 variabel yg isi nya si class itu dan 1 variabel tidak ada isi nya liat aja ya, nah lalu kita cek pake IF jadi jika this.classList.contains(satu) jadi maksud contains itu kalo didalam this nya ada variabel satu itu maka jalankan perogram nya, satu itu isi nya class ‘size’ ya.

Nah jika udah ada jalankan program nya ya kita balikin pake mengisi variabel nya kek gitu biar ketuker gitu loh makannya ada variabel sementara gitu. Nah tapi dengan javascript yg baru ada syntak yg keren, gini :

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

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

    let satu = 'size';

    let dua = 'caption';

    if (**this**.classList.contains(satu)) {

        [satu, dua] = [dua, satu];

    }

**this**.classList.toggle(satu);

    setTimeout(() => {

**this**.classList.toggle(dua);

    }, 700);

});

Nah kaya gitu jadi nuker isi nya dengan notasi seperti itu. Jadi kalo kita klik sekali terus klik satu kali lagi dia buat animasi yg pas gitu.