**JAVASCRIPT LANJUTAN 12. Fetch**

Fetch adalah sebuah API pada Javascript modern yg tugas nya untuk mengambil data secara Asynchronous atau Ajax sama seperti kita melalukan XML HTTP Request pada vanila Js sebelum nya atau menggunakan Method Ajax pada Jquery.

Di video sebelum nya kita sudah berhasil membuat terhubung ke public API atau open API yg namanya omdbapi kita sudah berhasil untuk mencari film dan menampilkan detail dari film tersebut, nah aplikasi tersebut kita buat dengan bantuan Jquery menggunakan method nya ajax.

Nah kita akan mencoba ganti Method ajax nya menggunakan fetch API sehingga nantinya kita tidak lagi mengandalkan liblary external seperti Jquery, jadi nanti kita performance penulisan menggunakan vanila js nya dapet tapi penulisan nya akan menjadi lebih sederhana karena kita menggunakan Fetch yg mirip dengan Jquery.

**PENJELASAN FETCH LEBIH DETAI DI VIDEO SELANJUTNYA**

Kita buka lagi codingan di video sebelum nya ya, sekarang kita mau ubah codingan nya jadi fetch dulu ya. Kita panggil dulu search button nya evenlistener nya gini :

const searchButton = document.querySelector(".search-button");

searchButton.addEventListener("click", function () {

  fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=avengers");

});

Jadi ketika tombol search-button di klik maka jalankan function ini, kita pake function biasa karena kalo pake function arrow kita gabisa pake this, disini kita mungkin butuh this. Jadi kalo tombol nya diklik kita mau jalankan method fetch untuk mengganti ajax punya nya Jquery.

Jadi cara menggunakan nya kalian cukup panggil fetch lalu di isi dengan parameter wajib nya adalah Resouses nya alamat kita mau ngambil Resouses nya dalam hal ini adalah URL kek di atas.

Nah avangers itu kan search query nya apa yg kita tulis didalam search nya muncul disitu, nah biar search query nya bisa sesuai yg dituliskan oleh client misalnya kita harus gabungkan gini :

const searchButton = document.querySelector(".search-button");

searchButton.addEventListener("click", function () {

  const inputKeyword = document.querySelector(".input-keyword");

  fetch(

    "http://www.omdbapi.com?apikey=78c0d1a4&s=" + inputKeyword.value

  );

});

Jadi nanti apapun yg dituliskan oleh client maka akan masuk ke search query URL nya gitu, kalo ajax ada opciton nya didalam nya, fetch juga bisa kita kasih opction didalam nya tapi karena dia GET gaperlu pake opction.

Nah jadi di atas itu harusnya mengembalikan hasilnya selesai, nah tapi fetch itu adalah fungsi di javascript yg mengembalikan PROMISE karena itu kita harus belajar promise dulu sebelum nya.

Jadi kalo kita jalanin fetch itu data nya ga langsung dapet karena itu bentuk nya promise. Yang nama nya promise kita harus kasih method buat menangani ketika data nya sudah fullfiled, ada dua ya kalo berhasil ngapain kalo gagal ngapain, pertama kalo berhasil kan pake then() ya ini tinggal lanjutin aja kaya Chaining.

Jadi kalo data nya sudah dikembalikan kita bisa ambil response nya lalu kita masukan kedalam arrow function, terus si respose nya mau kita apain, kalo kita liat response nya di console kita gabisa liat data nya, masi tersimpan didalam situ, nah kalo kita mua data nya berupa JSON kita bisa tambahkan method JSON gini ( response.json ). Method json itu merupakan bagian dari object response itu, nah yg namanya method pasti pake kurung buka dan kurung tutup giniiiiiiii ( response.json() ).

const searchButton = document.querySelector(".search-button");

searchButton.addEventListener("click", function () {

  const inputKeyword = document.querySelector(".input-keyword");

  fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + inputKeyword.value)

  .then((*response*) => console.log(*response*.json()));

});

Nah udah gitu kita coba lagi, dan ternyata data nya dikembalikan tapi lagi-lagi bentuk nya masih promise walaupun kita bisa liat udah ada isi data nya si film yg kita cari itu, tapi yang namanya promise kalo kita mau jalankan nya secara Asynchronous maka si ( response.json() ) harus kita jalankan lagi kedalam method then, makannya nanti keliatan nya then nya dua kali, gini :

const searchButton = document.querySelector(".search-button");

searchButton.addEventListener("click", function () {

  const inputKeyword = document.querySelector(".input-keyword");

  fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + inputKeyword.value)

    .then((*response*) => *response*.json())

    .then((*response*) => console.log(*response*));

});

Nah kalo sudah seperti itu maka bentuk nya sudah Object normal. Jadi sebetulnya menggunakan fetch hanya seperti itu aja. Noted ini hanya untuk mengambil si data nya aja tinggal si data nya udah dapet terus kita mau apain.

Nah si data nya kan mau kita simpan kedalam card, tinggal gini aja :

const searchButton = document.querySelector(".search-button");

searchButton.addEventListener("click", function () {

  const inputKeyword = document.querySelector(".input-keyword");

  fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + inputKeyword.value)

    .then((*response*) => *response*.json())

    .then((*response*) => {

      const movies = *response*.Search;

      let cards = "";

      movies.forEach((*e*) => {

        cards += showCard(*e*);

      });

      const movieContainer = document.querySelector(".movie-container");

      movieContainer.innerHTML = cards;

    });

});

Jadi sebenernya isinya hampir sama aja dengan menggunakan JQUery, karena liat dari coding const movies sampe forEach Nah itu kan tinggal copy aja dari codingan yg menggunakna Jquery, bedanya saat menggunakan DOM nya.

Jadi kalo yg then response itu bisa dianggap ketika aksinya success maka lakukan hal ini gitu jadi kalo dibaca sama dengan codingan menggunakan Jquery.

**Ketika tombol show Detail di KLIK**

const mdButton = document.querySelectorAll(".modal-detail-button");

      mdButton.forEach((*btn*) => {

*btn*.addEventListener("click", function () {

          console.log(**this**);

        });

      });

Jadi untuk menggunakan DOM memanggil tombol nya kita pake querySelectorAll karena tombol nya lebih dari satu jadi nya kaya array gitu, nah karena bentuk nya kaya array kita gabisa langsung addEventListener kita harus looping dulu baru deh addEventListener, dan kita disitu menggunakan this, jadi kalo tombol nya diklik di console nya muncul tag HTML tombol tersebut itu kegunaan nya this.

Nah selanjutnya yg kita butuhkna adalah imdbID nya itu, gini :

const mdButton = document.querySelectorAll(".modal-detail-button");

      mdButton.forEach((*btn*) => {

*btn*.addEventListener("click", function () {

          const dbS = **this**.dataset.imdbid;

          console.log(dbS);

        });

      });

Jadi const dbS itu kan isi nya this.dataset.imdbid, artinya this itu kan isinya tag HTML tombol nya ya, nah dataset itu mengambil attribute data yg ada didalam tag HTML tombol nya itu, dan imdbid ini nama data nya, misalnya gini di tag HTML tombol nya itu ada attribute data yg nama nya gini ( data-imdbid=” tt0046593”) jadikan kalo liat di console pas tombol nya diklik yg keluar isi dari data-imdbid itu sendiri yg ini “tt0046593”.

Nah kalo udah dapet ID nya baru deh kita Fetch, gini :

const mdButton = document.querySelectorAll(".modal-detail-button");

      mdButton.forEach((*btn*) => {

*btn*.addEventListener("click", function () {

          const dbS = **this**.dataset.imdbid;

          fetch("http://www.omdbapi.com?apikey=78c0d1a4&i=" + dbS)

            .then((*response*) => *response*.json())

            .then((*r*) => {

              const movieModal = showMovieDetail(*r*);

              const movieContainer = document.querySelector(".modal-body");

              movieContainer.innerHTML = movieModal;

            });

        });

      });

Jadi kan udah dapet tuh ID nya tinggal kita salin aja codingan yg sebelum nya yg menggunakan Jquery terus yg then() nya itu sama ajakan sama kaya yg card cara baca nya yg then() nya juga sama kaya yg card, disini bedanya cara manggil DOM nya kalo vanila js emang lebih panjang aja kali ya.

Dan jadi segitu aja perbedaan nya kalo menggunakan fetch dan menggunakna jquery bisa diliat sendiri, semua itu mudah kalo kita mau belajar secara perlahan. Dan itu baru fetch secara sederhana saja karena fetch itu bisa kita pakai untuk kasus yg sangat kompleks.

**Teori Fetch()**

“Sebuah method pada API javascript untuk mengambil resources dari jaringan, dan mengembalikan promise yang selesai (fullfilled) ketika ada response yang tersedia.”

<https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/fetch>

Jadi dia mengembalikan bentuknya promise, jadi ga langsung jadi json ga langsung text atau apapun tapi dia bentuk nya promise, nah promise ini akan fullfilled ketika ada response yg tersedia.

Nah lalu yg dikembalikan adalah response nya dan sebenernya isi nya macem-macem, inti nya fetch itu nulis nya gini ( fetch(resources, init); ), init itu bisa dibilang konfigurasi tambahan, tadi kita sama sekali ga pake init kalo mau pun bisa.

Nah resources nya itu sebetulnya bisa 2, gini :

* URL

Alamat dari sumber yang kita akan ambil seperti yg kita pake itu.

* Request Object

Representasi pemintaan sumber

Nah ini sama seperti promise ya, Object yg spesial yg dimilikin oleh javascript walaupun ini jarang kita gunakan dan ini sebetulnya representasi dari ketika kita minta ke resources nya. Jadi kita bisa simpan object kedalam resources nya.

Sedangkan init ini (optional) :

Kita bisa tambahkan konfigurasi tambahan saat kita ngirimin request nya bentuk nya object, jadi nanti ngasih nya sama kaya kita ngasih ID jquery, pake kurung kurawal isi nya bisa macem” property dan method nya, beberapa di antara nya ini :

* Method
* Headers
* Body
* Mode
* Cache
* Referrer
* referrerPolicy
* integrity
* keepalive
* signal

Yang tadi kita coba di codingan inti nya itu method nya GET dari default nya, kita bisa ngirimin POST juga misalnya ketika Ajax nya mau dijalanin ketika kita mau bikin form ada header, body dll. Jadi itu parameter dari FETCH.

**Response**

Sedangkan response nya sendiri hasil dari fetch nya itu sendiri yg berupa promise ya, nah didalam nya ada 2, ada property :

* headers
* ok
* redirected
* status
* statusText
* type
* url
* body

Jadi misalkan kalo kalian mau ngecek seperti di jquery kan di jquery ada misal bisa success atau ketika error gitu, menggunakan fetch juga bisa kalian pake response titik ok ( .ok ) gitu. Itu kalo hasilnya success, nah success itu kalo kita representasikan pake status kode itu dari 200-299 jadinya kalo itu terjadi maka yg dilakukan apa gitu.

Atau kalo status nya redirected jadi kalian bisa misalkan pake kondisi misalkan if response.redirected mau ngapain gitu, atau kalo kalian butuh angka status nya jadi if status = 200 ngapain gitu.

Atau butuh text nya jadi setiap status itu ada angka nya ada text nya, jadi kalo 200 itu text nya adalah ok, jadi kalian bisa tulis if statusText = ok ngapain gitu.

**Dan ada juga response (method) :**

* clone()
* error()
* redirect()
* blob()
* formData()
* json()
* text()

Jadi yang udah kita pake method nya itu ada yg json() itu. Jadi kalo pake json() kalo kita mau jadiin yg tadinya bentuk nya promise mau bentuknya json bisa pake method itu. Atau kalian mau jadiin promise nya bentuk nya jadi blob() atau binary large object misalnya kalian mau upload file terus kita bisa tahanin si file nya menjadi blob().

Dan rata-rata dari method itu mengembalikan promise juga, nah kenapa kita harus paham promise dulu sebelum kita belajar fetch nah itu.