**JAVASCRIPT LANJUTAN 14. Async Await**

Topik ini sudah pernah kita implementasikan di video sebelum nya saat kita melakukan refactoring pada kode fetch kita, jadi kita kebalik ini implementasi dulu baru dokumentasi kita sekarang liat konsep dan definisinya.

**Asyncronous function**

“Sebuah function yang bekerja secara asyncronous (melalui event loop), yang dia menghasilkan (implisit) promise sebagai return value-nya, (implisit promise itu intinya akan mengembalikan promise meskipun didalam nya tidak ada kode yg **asynchronous),** tapi cara penulisan code-nya menggunakan penulisan yang syncronous atau (standard).”

(<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function> )

Dari definisi itu ada 3 hal yg bisa kita liat :

**Asyncronous function**

* Fungsi yang berjalan secara asynchronous
* Menghasilkan (implisit) promise
* Cara penulisannya sama seperti synchronous function atau (function biasa)

Mungkin dari implementasi sebelumnya kalian udah bisa liat ke 3 definisi itu. Selanjutnya penjelasan lebih lanjtnya “Sebuah async function dapat memiliki keyword await di dalamnya, (jadi keyword await hanya akan berjalan ketika function nya kita kasih async didepan nya) keyword await ini digunakan untuk memberhentikan sementara eksekusi dari fungsinya sambil menunggu promise-nya selesai atau resolve.”

Jadi walaupun penulisan nya secara synchronous ketemu keyword await nah dia nunggu dulu promise nya udah resolve belum kalo belum hentika eksekusi nya sampai resolve. Baru lanjutkan ke baris berikut nya.

Yang kita coba divideo sebelumnya menggunakan API fetch, API fetch sebuah API yg menggunakan promise didalam nya. Nah sekarang kita mau coba Async Await ini menggunakan promise biasa atau promise langsung ga menggunakan API Fetch.

Lets CODE!!

Pertama kita buat promise() nya dulu, buat nya kan didalam nya ada callback function bisa ada resolve dan reject ya, resolve itu kalo berhasil dan reject itu kalo gagal. Sebenernya kalo kita tulis resolve nya aja bisa jadi kita ga menangani promise nya gagal. Gini :

const coba = new Promise((*resolve*) => resolve("selesai"));

console.log(coba);

kalo kita liat di browser nya kita bisa langsung liat resolve nya selesai, gaada masalah ya karena memang disitu nya langsung selesai ya. Nah sekarang gimana kalo promise nya ga langsung resolve tapi nunggu dulu selama beberapa waktu supaya nanti keliatan Asynchronous nya.

Gini :

const coba = new Promise((*resolve*) => {

  setTimeout(() => {

    resolve("selesai");

  }, 2000);

});

console.log(coba);

harusnya kan nunggu dulu 2detik terus kalo udah resolve nya disimpan ke dalam coba ya, dan harusnya si coba isi nya resolve selesai ya. Dan ternyata hasil nya promise nya pending karena kita jalanin nya secara Synchronous langsung pake console.log gitu aja.

Nah supaya nunggu dulu selama 2detik kita bisa pake keyword then ketika terjadi resolve dan keyword cath ketika terjadi reject ya. Gini :

const coba = new Promise((*resolve*) => {

  setTimeout(() => {

    resolve("selesai");

  }, 2000);

});

coba.then(() => console.log(coba));

Jadi kalo coba terjadi resolve kita ngapain didalam then nya, misalnya saya kasih arrow function baru saya console.log coba nya. Jadi pas saya jalanin dia tunggu dulu 2 detik baru muncul resolve selesai.

/**/ Menerapkan async & await**

Untuk menerapkan asycn dan await kita butuh function, jadi promise nya akan saya simpan kedalam function sebagai return valu-nya, gini :

function cobaPromise() {

  return new Promise((*resolve*) => {

    setTimeout(() => {

      resolve("selesai");

    }, 2000);

  });

}

Jadi kita sekarang promise nya disimpan kedalam function dijadikan return, sekarang kita ngejalanin nya gimana, kita butuh function lagi untuk memanggil function cobaPromise() itu, gini :

function cobaPromise() {

  return new Promise((*resolve*) => {

    setTimeout(() => {

      resolve("selesai");

    }, 2000);

  });

}

async function jalankan() {

  const coba = await cobaPromise();

  console.log(coba);

}

jalankan();

Jadi kita tinggal buat aja tuh function didalam nya manggil function cobaPromise() terus baru console deh si coba nya, akan berhasil jika kita menggunakan keyword async pada awal function untuk memberi tahu bahwa javascript didalam fungsi itu bakalan ada sciprt yg Asynchronous dan harus ada Await untuk memberi tahu itu kode yg Asynchronous jadi sebelum dijalanin kita tunggu dlu sampai promise nya resolve begitu intinya.

**Error Handling**

Nah bagaimana melakukan error handling pada script async dan await kita, kan kalo kita menggunakan promise ada resolve dan reject nah resolve itu bisa ditangkap menggunakan then jadi apa yg terjadi didalam resolve kita tangkap menggunakan method then dan kalo gagal kita punya reject bisa ditangkap menggunakan method catch.

Misalnya gini saya sekarang mau punya parameter resolve dan reject terus saya mau cek kalo waktu tunggu nya di bawah 5 detik langsung aja resolve tapi kalo di atas 5 detik kita reject soalnya kelamaaan, gini :

function cobaPromise() {

  return new Promise((*resolve*, *reject*) => {

    const waktu = 3000;

    if (waktu < 5000) {

      setTimeout(() => {

        resolve("selesai");

      }, waktu);

    } else {

      reject("kelamaam!!");

    }

  });

}

const coba = cobaPromise();

coba

.then(() => console.log(coba))

.catch(() => console.log(coba));

Jadi kalo kita jalanin hasil nya resolve selesai karena sesuai lah ya baca kode nya sendiri. Kalo kita ganti waktu nya jadi 6000 itu kan udah false ya berarti yg muncul reject.

Nah sekarang gimana cara nya kalo kita lagi pake async dan await, masalah nya kalo kita pake asycn dan await kita gabisa pake then atau catch, nah kita bisa pake blok try dan catch, jadi kalo try itu untuk kondisi ketika resolve dan kalo catch itu ada parameter nya itu untuk yg reject nah pesan didalam reject nya yg ( ‘kelamaan!!’ ) itu masuk kedalam pesan parameter catch nya, gini :

function cobaPromise() {

  return new Promise((*resolve*, *reject*) => {

    const waktu = 6000;

    if (waktu < 5000) {

      setTimeout(() => {

        resolve("selesai");

      }, waktu);

    } else {

      reject("kelamaam!!");

    }

  });

}

async function jalankan() {

  try {

    const coba = await cobaPromise();

    console.log(coba);

  } catch (err) {

    console.log(err);

  }

}

jalankan();

Nah jadi begitu cara nya kalo pake async dan await bisa menggunakan blok try dan catch mudah bukan.