**JAVASCRIPT LANJUTAN 15. Error Handling**

Sebenarnya divideo sebelum nya udah kita coba ketika kita buat sebuah promise itu kita punya method yg kita gunakan sebagai error handling.

**Error Handling : Promise**

* Then()
* Catch()

Kalo misalnya kita punya code seperti ini :

function cobaPromise() {

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

    const waktu = 3000;

    if (waktu < 5000) {

      setTimeout(() => {

        resolve("selesai");

      }, 2000);

    } else {

      reject("kelamaan!!");

    }

  });

}

Jadi disitu kita punya sebuah function yg didalam nya mengembalikan sebuah Promise dan didalam promise nya terdapat method resolve dan method reject, kita tau resolve itu ketiak promise nya berhasil dan reject itu ketika promise nya tidak terpenuhi.

Nah ketika kita memanggil promise nya kita bisa gunakan then() untuk jalanka apa yg ada didalam resolve, dan catch() untuk menjalankan apa yg ada didalam reject.

const coba = cobaPromise();

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

.catch((*err*) => console.error(*err*));

Jadi isitilah nya ketika ada eror kita bis tangkap erornya kedalam method catch(), ya gaada masalah kalo pake PROMISE ini udah kita coba di video sebelum nya.

**Error Handling : Async Await**

Terus yang kedua kita udah pernah juga menangani handling eror pada kode async dan await, kalo menggunakan nya itu ita ga pake then() dan catch() tapi kita pake :

* Try
* Catch()

Karena kode nya kita buat seolah-olah kode yg Synchronous, jadi kalo kita punya promise yg sama atau function yg mengembalikan promise untuk menangkap resolve kita bisa pake TRY dan reject nya pake catch seperti ini :

function cobaPromise() {

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

    const waktu = 3000;

    if (waktu < 5000) {

      setTimeout(() => {

        resolve("selesai");

      }, 2000);

    } else {

      reject("kelamaan!!");

    }

  });

}

async function cobaAsync() {

  try {

    const coba = await cobaPromise();

    console.log(coba);

  } catch (err) {

    console.error(err);

  }

}

**Error Handling : fetch()?**

Nah tapi ada satu Error Handling yg belum kita coba, kalian tau kan fetch() ini didalam nya merupakan promise ya, tapi masalahnya kita ga bikin sendiri fetch ini karena ini fungsi punya javascript, kita gatau apa yg ada didalam nya isi method reject nya seperti apa kan.

Yang udah kita coba ketika berhasil aja atau istilah nya kita menjalankan apa yg ada didalam method resolve aja. Jadi kita akan mencoba kembali aplikasi pencarian film kita ya.

Kita mau coba error nya ketika kita mau carai film tapi kosong atau gimana kalo kita cari nama film nya gaada di API nya itu, di console nya mah ada error tapi keterangan nya itu karena si forEach nya itu mengembalikan nilai yg tidak sesuai atau kosong isitlahnya.

Jadi kita mau cari tau gimana si cara ngatasin error handling ini, nah kita tau async await kita untuk error handling kita bisa gunakan blok try and catch(), gini :

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

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

  try {

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

    const movies = await getMovies(inputKeyword.value);

    updateUI(movies);

  } catch (err) {

    console.log(err);

  }

});

Jadi kalo ada error tampilin gitu, seskarang ketika saya cari film yg tidak ada error nya sudah ditangkap catch() itu, jadi muncul nya udah gak lagi pake bawaan browser nya. Sekarang kita mau lihat apa aja error yg bisa ditangkap.

Nah sekarang kalo misalnya URL nya salah tulis typo gitu, misalnya kurang ( p ) yg harusnya (http), nah yg harus kalian tau kalo menggunakan fetch() error yg ditangkap itu hanya error yg ada pada network nya atau yg ada pada URL nya aja.

Jadi sebetul nya kalo error yg tadi ketika searching film nya tidak sesuai itu sebetul nya bukan error dari fetch, itu error dari browser karena begitu mau menjalankan function updateUI parameter nya kosong gitu.

Jadi kalo kita matikan dulu function updateUI() nya, gini :

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

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

  try {

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

    const movies = await getMovies(inputKeyword.value);

*// updateUI(movies);*

  } catch (err) {

    console.log(err);

  }

});

Jadi kalo kita cari film yg gaada ternyata gaada error nya gitu, jadi sebetul nya gaada yg masuk kedalam catch(), karena fetch() hanya akan error kalo URL nya yg bermasalah, nah untuk kita mau nanganin error nya ketika misal film nya ga gaada. Nah ternyata cara nya kita harus tanganin secara manual didalam, gini :

Nah method response.json() itu akan jalan ketika data nya sudah benar, ketika film nya ada, ketika ada yg di isi kan pas searching, jadi kita bisa cek dulu ya kita bisa kasih kondisi disitu, jadi kalo response nya gagal gimana.

Nah sebelum itu coba deh kita console.log(response) gitu biar tau isi nya apa aja. Isi nya it ada ok ada status, jadi kalo searching selagi kosong itu sebener nya berhasil Cuma data nya gaada, kalo misalkan tulis film ngasal berhasil juga Cuma data nya gaada.

Nah jadi dia mengembalikan error kalo pas key nya salah, yg tampil itu ( status: 401 dan ok: false) kan 401 satu artinya error ya biasanya, nah itu yg mau kita ambil jadi kalian bisa cek status nya atau ok nya.

Pengkondisan nya gini jika (response.ok === false) atau bisa juga gini ( !response.ok ) jadi kalo response nya tidak ok, maka kita akan kirim kan error untuk dilempar ke catch() tadi itu, menggunakan keyword nya itu throw new Error(). Jadi kita lemparkan error isi error nya saya pengen unauthorized, kalo pengen tampil unauthorized berarti kalian ambil response.statusText nya gitu.

Jadi nanti statusText nya akan dilempar dan ditangkap oleh catch(), gini :

function getMovies(*keyword*) {

  return fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + *keyword*)

    .then((*response*) => {

      if (!*response*.ok) {

        throw new Error(*response*.statusText);

      }

    })

    .then((*response*) => *response*.Search);

}

Itu API key nya typo in dikit biar Error gitu ya, nah kita balikin ke semula lagi key nya, sekarang kalo udah berhasil jangan lupa tulis return response.json() nya gini :

function getMovies(*keyword*) {

  return fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + *keyword*)

    .then((*response*) => {

      if (!*response*.ok) {

        throw new Error(*response*.statusText);

      }

      return *response*.json();

    })

    .then((*response*) => *response*.Search);

}

Jadi sekarang kalo ok yg dikerjakan yg json() kalo ga ok yg dikerjakan yg if, nah tapi sekarang kalo kita klik Search ga ada yg terjadi karena sekarang udah masuk ke then() yg response.Search.

Nah sekarang kita cek lagi misal API key nya udah aman kita cek ada isi nya ga, nah coba kita cek lagi isi dari error nya apa console.log(response), dan ternyata ketika kita klik tombol search dan kita ga nulis apa” di input nya, fetch() nya itu sebenernya ada pesan error nya “Movie not found!” ini bisa kita manfaatin, gini:

function getMovies(*keyword*) {

  return fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + *keyword*)

    .then((*response*) => {

*// console.log(response);*

      if (!*response*.ok) {

        throw new Error(*response*.statusText);

      }

      return *response*.json();

    })

    .then((*response*) => {

*// console.log(response);*

      if (*response*.Response === "False") {

      }

    });

}

Yg response.Response ( itu yg R nya kapital itu nama variable atau obj dari si fetch itu sendiri yg kita cek di console atau kalo yg R kapital itu response dari key json nya), jadi kalo Response nya “False” kita kirim error lagi gini :

function getMovies(*keyword*) {

  return fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + *keyword*)

    .then((*response*) => {

*// console.log(response);*

      if (!*response*.ok) {

        throw new Error(*response*.statusText);

      }

      return *response*.json();

    })

    .then((*response*) => {

*// console.log(response);*

      if (*response*.Response === "False") {

        throw new Error(*response*.Error);

      }

      return *response*.Search;

    });

}

Jadi kalo error ditampilin pesan yg “Movie not found!” dan kalo ga error yaudah jalanin deh kita return aja gitu. Atau kalo kalian mau liat apa si isi dari response ketika hasil nya True gini

function getMovies(*keyword*) {

  return fetch("http://www.omdbapi.com?apikey=78c0d1a4&s=" + *keyword*)

    .then((*response*) => {

*// console.log(response);*

      if (!*response*.ok) {

        throw new Error(*response*.statusText);

      }

      return *response*.json();

    })

    .then((*response*) => {

*// console.log(response);*

      if (*response*.Response === "False") {

        throw new Error(*response*.Error);

      }

      console.log(*response*);

    });

}

Nah coba kalo kalian ketik nama film disney misal keluar tuh isi response nya, nah didalam response nya itu ada Search yg isi nya Array seperti nya. Kalo mau lebih keren kalian bisa tampilin dihalaman ketika error kan keren ya.

Atau kalo bisa dibagian pas ketika tombol di klik didalam nya kan ada try sama catch() nah pas catch() nangkep error itu jangan console.log(err) bisa di buat apa gitu tampil alert di layar misal nya, gini :

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

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

  try {

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

    const movies = await getMovies(inputKeyword.value);

    updateUI(movies);

  } catch (err) {

    alert(err);

  }

});

Jadi ketika terjadi error nanti muncul alert lumayan lah bisa keliatan ya, nah untuk yg tombol detail kalian bisa kreasikan sendiri ya sama aja lah metode nya gitu juga.