DISKUSIKAN MATERI
LAPORKAN MATERI

Daftar Modul

Masukkan kata kunci

Persetujuan Hak Cipta
Modul 1: Introduction Course (Gratis)
Prasyarat Kelas (Gratis)
Apa yang Akan Kita Pelajari? (Gratis)
Tools Requirement
Modul 2: ECMAScript 6 (Gratis)
Running Snippet Code (Gratis)
Persiapan Project Latihan (Gratis)

Constructuring a Promise Object

Situs MDN mengatakan Promise merupakan sebuah objek yang digunakan untuk membuat sebuah perhitungan (kode) ditangguhkan dan berjalan secara asynchronous. Untuk membuat objek promise, kita gunakan keyword new diikuti dengan constructor dari Promise:

```
const coffee = new Promise();
```

Namun jika kita jalankan kode tersebut, akan mengakibatkan eror seperti ini:

```
/* ERROR: Promise resolver undefined is not a function */
```

Di dalam constructor Promise kita perlu menetapkan *resolver function* atau bisa disebut *executor function* di mana fungsi tersebut akan dijalankan secara otomatis ketika constructor Promise dipanggil.

```
const executorFunction = (resolve, reject) => {
 const isCoffeeMakerReady = true;
 if(isCoffeeMakerReady) {
   resolve("Kopi berhasil dibuat");
 } else {
   reject("Mesin Kopi tidak bisa digunakan!")
 }
}

const makeCoffee = new Promise(executorFunction);
console.log(makeCoffee);

/* output:
Promise { 'Kopi berhasil dibuat' }
*/
```

Executor function dapat memiliki dua parameter, yang berfungsi sebagai `resolve()` dan `reject()` function. Berikut penjelasan detailnya:

- `resolve()` merupakan parameter pertama pada executor function. Parameter ini merupakan fungsi yang dapat menerima satu parameter, biasanya kita gunakan untuk mengirimkan data ketika promise berhasil dilakukan. Ketika fungsi ini terpanggil, kondisi Promise akan berubah dari ***pending*** menjadi ***fulfilled***.
- `reject()` merupakan parameter kedua pada executor function. Parameter ini merupakan fungsi yang dapat menerima satu parameter yang digunakan untuk memberikan alasan mengapa Promise tidak dapat terpenuhi. Ketika fungsi ini terpanggil, kondisi Promise akan berubah dari ***pending*** menjadi ***rejected***.

Executor function akan berjalan secara asynchronous hingga akhirnya kondisi Promise berubah dari pending menjadi ***fulfilled*** / ***rejected***.

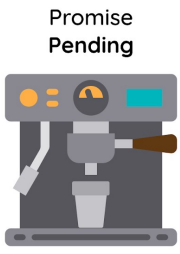
Pada contoh kode di atas, berikut ini outputnya:

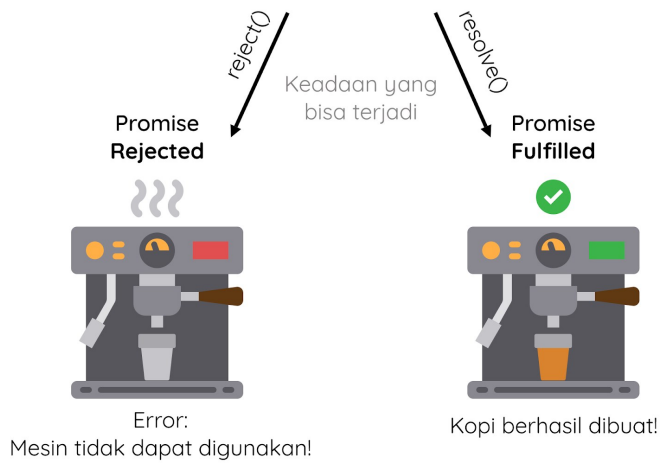
```
/* output:
Promise { 'Kopi berhasil dibuat' }
*/
```

Kenapa demikian? Executor function mengeksekusi `resolve()` dengan membawa data string "Kopi berhasil dibuat". Coba kita ubah nilai dari variabel `isCoffeeMakerReady` menjadi `false`, maka *executor function* akan mengeksekusi `reject()` dengan membawa pesan *rejection* "Mesin Kopi tidak bisa digunakan!".

```
/* output:
Promise { <rejected> 'Mesin Kopi tidak bisa digunakan!' }
*/
```

Untuk mudah menggambarkan alurnya, silakan lihat ilustrasi berikut:





Dari output yang dihasilkan baik ketika ***fulfilled*** ataupun ***rejected*** masih berupa ***Promise***, bukan nilai yang dibawa oleh fungsi `resolve` atau `reject` itu sendiri. Lantas bagaimana cara untuk mengakses nilai yang dibawa oleh fungsi-fungsi tersebut? Caranya adalah dengan menggunakan method `.then()` yang tersedia pada objek Promise.

← KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA →

Copyright © 2020 - Dicoding Indonesia. All rights reserved.