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)

Async-Await Syntax

Pembahasan terakhir mengenai *asynchronous operator* kali ini adalah penggunaan sintak async/await. Apa itu?

Seperti yang sudah kita ketahui, pada JavaScript callback dan Promise merupakan teknik yang digunakan selama kita berurusan dengan proses asynchronous. Namun penulisan kode akan sedikit berbeda dari proses synchronous. Contohnya, untuk mendapatkan nilai coffee dari sebuah proses asynchronous, kita tidak dapat melakukannya dengan teknik seperti ini:

```
function makeCoffee() {
 const coffee = getCoffee(); // Async proses menggunakan promise
 console.log(coffee);
}

makeCoffee();
```

Melainkan, harus menuliskannya seperti ini:

Promise Approach **Callback Approach**

```
function makeCoffee() {
 getCoffee(function(coffee) {
   console.log(coffee)
 })
}


makeCoffee();


/* output
Coffee didapatkan!
*/
```

Namun semenjak **ES8 (ECMAScript 2017)** kita dapat menuliskan *asynchronous process* layaknya *synchronous process* dengan bantuan keyword ***async*** dan ***await***.

Fitur async/await sebenarnya hanya syntactic sugar. Itu berarti secara fungsionalitas bukan merupakan sebuah fitur baru dalam JavaScript. Namun hanya gaya penulisan baru yang dikembangkan dari kombinasi penggunaan Promise dan generator (pembahasan mengenai generator bisa Anda pelajari *di sini*). Sehingga async/await ini tidak dapat digunakan jika tidak ada Promise.

Using async/await

Lantas bagaimana cara menggunakan async/await ini? Oke kembali lagi pada contoh kode sebelumnya. Namun kita perlihatkan juga fungsi ***getCoffee()*** bagaimana ia dibuat menggunakan promise.

```
const getCoffee = () => {
 return new Promise((resolve, reject) => {
   const seeds = 100;
   setTimeout(() => {
     if(seeds >= 10) {
       resolve("Coffee didapatkan!");
     } else {
       reject("Biji kopi habis!")
     }
   }, 1000)
 })
}
```

Fungsi ***getCoffee()*** yang dibangun menggunakan promise.

```
function makeCoffee() {
 getCoffee()
 .then(coffee => {
   console.log(coffee)
```

```
  })
}

makeCoffee();

/* output
Coffee didapatkan!
*/
```

Kode di atas merupakan cara biasa kita menggunakan sebuah promise. Untuk mendapatkan nilai *coffee*, kita dapat memperolehnya di dalam method *.then()* dari hasil yang dikirimkan oleh *resolve*.

Dengan async/await, kita dapat menggunakan promise dengan cara seperti ini:

```
function makeCoffee() {
  const coffee = getCoffee();
  console.log(coffee);
}

makeCoffee();
```

Namun, tentu dengan menambahkan keyword *async* dan *await* seperti ini:

```
async function makeCoffee() {
  const coffee = await getCoffee();
  console.log(coffee);
}

makeCoffee();

/* output
Coffee didapatkan!
*/
```

Keyword *async* digunakan untuk memberitahu JavaScript untuk menjalankan fungsi *makeCoffee()* secara *asynchronous*. Lalu keyword await digunakan untuk menghentikan proses pembacaan kode selanjutnya sampai fungsi *getCoffee()* mengembalikan nilai *promise resolve*.

> "Walaupun *await* menghentikan proses pembacaan kode selanjutnya pada fungsi *makeCoffee*. Tapi ini tidak akan mengganggu proses runtime sesungguhnya pada JavaScript (global). Karena fungsi *makeCoffee* berjalan secara asynchronous. Kita tidak dapat menggunakan *await* tanpa membuat function dalam scope-nya berjalan secara asynchronous."

← KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA →