



DISKUSIKAN MATERI

LAPORKAN MATERI

## Daftar Modul

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:

```
1. function makeCoffee() {
2.   const coffee = getCoffee(); // Async proses menggunakan promise
3.   console.log(coffee);
4. }
5.
6. makeCoffee();
```

Melainkan, harus menuliskannya seperti ini:

## Promise Approach    Callback Approach

```
1. function makeCoffee() {
2.   getCoffee()
3.   .then(coffee => {
4.     console.log(coffee)
5.   })
6. }
7.
8.
9. makeCoffee();
10.
11.
12. /* output
13. Coffee didapatkan!
14. */
```

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 perhatikan juga fungsi `getCoffee()` bagaimana ia dibuat menggunakan promise.

```
1. const getCoffee = () => {
2.   return new Promise((resolve, reject) => {
3.     const seeds = 100;
4.     setTimeout(() => {
5.       if(seeds >= 10) {
6.         resolve("Coffee didapatkan!");
7.       } else {
8.         reject("Biji kopi habis!");
9.       }
10.    }, 1000)
11.  })
12. }
```

Fungsi `getCoffee()` yang dibangun menggunakan promise.

```

1. function makeCoffee() {
2.   getCoffee()
3.   .then(coffee => {
4.     console.log(coffee)
5.   })
6. }
7.
8. makeCoffee();
9.
10. /* output
11. Coffee didapatkan!
12. */

```

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:

```

1. function makeCoffee() {
2.   const coffee = getCoffee();
3.   console.log(coffee);
4. }
5.
6. makeCoffee();

```

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

```

1. async function makeCoffee() {
2.   const coffee = await getCoffee();
3.   console.log(coffee);
4. }
5.
6. makeCoffee();
7.
8. /* output
9. Coffee didapatkan!
10. */

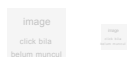
```

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 →



#### PERUSAHAAN

Tentang Kami

Blog

Berita Terbaru



#### PROGRAM

Academy

Challenge

Event

Job

Rewards

#### SUPPORT

Bantuan

FAQ

Hubungi Kami

