

Events

Aplikasi Node.js biasanya dikenal memiliki pola *event-driven* atau memiliki alur berdasarkan suatu kejadian. Apa maksudnya itu? Mari kita jelajahi lebih dalam lagi.

Dunia nyata penuh dengan kejadian. Alarm berbunyi, ponsel berdering, turun hujan, ataupun kejadian lainnya. Sebagai manusia, kita membuat keputusan lantas bertindak berdasarkan kejadian yang ada. Contohnya ketika berjalan dan tiba-tiba turun hujan, kita bergegas menggunakan payung; ketika ponsel berdering, kita bereaksi dengan mengangkat panggilan; ketika merasa lapar, kita makan. Seperti inilah pola yang terjadi di kehidupan nyata, sudah sejak lama kita bertahan hidup dengan pola seperti ini. Inilah yang dimaksud dengan pola *event-driven*.

Kita kembali ke dunia komputer. Tradisionalnya, programming dilakukan dengan cara yang imperatif. Agar komputer dapat melakukan sesuatu hal, kita perlu banyak menuliskan instruksi secara runtut beserta langkah-langkahnya. Komputer akan membaca kode dari atas ke bawah sesuai dengan urutan yang kita definisikan.

Dengan pola yang kaku seperti itu, kita akan sulit membangun program yang dapat menangani suatu kejadian. Karena kita saja tidak tahu kapan suatu kejadian akan terjadi, lantas bagaimana cara memberikan instruksi pada komputer? Lalu bagaimana solusinya? Berkaca dari dunia nyata, program komputer juga harus bekerja dengan pola *event-driven*. Syukurlah dengan Node.js kita dapat menerapkan pola tersebut dengan mudah.

Node.js menyediakan **EventEmitter** class yang merupakan member **events** core module:

```
1. const { EventEmitter } = require('events');
2.
3. const myEventEmitter = new EventEmitter();
```

Setiap instance dari **EventEmitter** akan memiliki fungsi **on**. Pada fungsi tersebut, kita dapat menentukan aksi berdasarkan sebuah kejadian.

Contohnya seperti ini:

```
1. const { EventEmitter } = require('events');
2.
3. const myEventEmitter = new EventEmitter();
4.
5. // fungsi yang akan dijalankan ketika event coffee-order terjadi
6. const makeCoffee = ({ name }) => {
7.   console.log(`Kopi ${name} telah dibuat!`);
8. };
9.
10. // mendaftarkan fungsi makeCoffee sebagai listener event coffee-order
11. myEventEmitter.on('coffee-order', makeCoffee);
```

Fungsi **on** menerima dua buah argumen, yang pertama adalah nama event dan yang kedua adalah listener atau fungsi yang akan dieksekusi ketika event terjadi. Dari kode di atas, jika terjadi event **'coffee-order'**, maka

fungsi `makeCoffee` akan dijalankan.

Anda bebas menentukan nama event yang diberikan pada argumen fungsi `on`. Jika nama event lebih dari dua kata, latihan terbaiknya adalah memisahkannya dengan tanda garis (-) bukan menggunakan spasi.

Lantas bagaimana cara membangkitkan suatu event? Setiap instance dari `EventEmitter` juga memiliki fungsi `emit()` yang berguna untuk membangkitkan event.

```
3. const myEventEmitter = new EventEmitter();
4.
5. const makeCoffee = ({ name }) => {
6.   console.log(`Kopi ${name} telah dibuat!`);
7. };
8.
9. myEventEmitter.on('coffee-order', makeCoffee);
10.
11. // Memicu event 'coffee-order' terjadi.
12. myEventEmitter.emit('coffee-order', { name: 'Tubruk' });
13.
14. /**
15.  * output:
16.  * Kopi Tubruk telah dibuat!
17.  */
```

Fungsi `emit()` menerima nilai argumen sebanyak apa pun yang Anda mau, namun nilai yang pertama merupakan nama dari event yang akan

nama nilai yang pertama merupakan nama dan event yang akan dibangkitkan, argumen kedua dan seterusnya adalah nilai yang akan digunakan untuk menjadi dari parameter fungsi listener.

Anda juga bisa mendaftarkan lebih dari satu fungsi listener pada sebuah event menggunakan fungsi `on`.

```
8.
9.  const makeBill = ({ price }) => {
10.    console.log(`Bill sebesar ${price} telah dibuat!`);
11.  }
12.
13.  myEventEmitter.on('coffee-order', makeCoffee);
14.  myEventEmitter.on('coffee-order', makeBill);
15.
16.  myEventEmitter.emit('coffee-order', { name: 'Tubruk', price: 15000 });
17.
18.  /**
19.   * output:
20.   * Kopi Tubruk telah dibuat!
21.   * Bill sebesar 15000 telah dibuat!
22.   */
```

Atau Anda bisa membuat satu fungsi khusus untuk menangani event. Biasanya fungsi ini memiliki nama 'handler' atau 'listener' pada akhir penamaanya.

```
12.
```

```
13. const onCoffeeOrderedListener = ({ name, price }) => {
14.     makeCoffee(name);
15.     makeBill(price);
16. }
17.
18. myEventEmitter.on('coffee-order', onCoffeeOrderedListener);
19.
20. myEventEmitter.emit('coffee-order', { name: 'Tubruk', price: 15000 });
21.
22. /**
23.  * output:
24.  * Kopi Tubruk telah dibuat!
25.  * Bill sebesar 15000 telah dibuat!
26.  */
```

Latihan: Events

Ayo kita latihan apa yang sudah kita pelajari!

Silakan buat folder baru bernama **events** dan di dalamnya buat berkas JavaScript baru bernama **index.js**.

▼ NODEJS-BASIC

▼ events

JS index.js

Tuliskan starter code berikut di dalam **index.js**:


Salin starter code berikut di dalam `index.js`.

```
1. // TODO 1
2.
3. const birthdayEventListener = (name) => {
4.   console.log(`Happy birthday ${name}!`);
5. }
6.
7. // TODO 2
8.
9. // TODO 3
10.
11. // TODO 4
```

Wah cukup banyak juga yah TODO yang harus dikerjakan. Simak penjelasannya:

- **TODO 1** : Buat atau impor variabel `EventEmitter` dari core module `events`.
- **TODO 2** : Buat variabel `myEmitter` yang merupakan instance dari `EventEmitter`.
- **TODO 3** : Tentukan `birthdayEventListener` sebagai aksi ketika event `'birthday'` dibangkitkan pada `myEmitter`.
- **TODO 4** : Bangkitkanlah event `'birthday'` pada `myEmitter` dengan method `emit()` dan beri nilai argumen listener dengan nama Anda.

Bila Anda selesai mengerjakan TODO yang ada, eksekusi berkas `index.js` dengan perintah:



```
1. node ./events/index.js
```

Maka console akan menampilkan output seperti berikut:

```
C:\javascript-projects\nodejs-basic>node ./events/index.js  
Happy birthday Dimas!
```

Mengalami kesulitan dalam menyelesaikan latihan?

Cobalah untuk ulas kembali materi yang diberikan atau tanyakan kesulitan yang Anda alami pada [forum diskusi](#). Hindari melihat atau membandingkan [kode solusi](#) pada latihan event. sebelum Anda mencobanya sendiri.



Events

