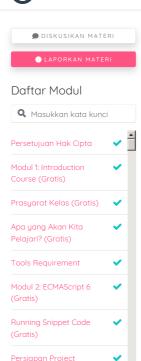
~



Exporting and importing single value (Default Export)

Pada Node.js sebelumnya tidak ada perbedaan antara exporting multiple value dan single value. Semua nilai yang akan diekspor, dijadikan nilai dari properti *module.exports*. Pada ES6 module, jika kita hanya mengekspor satu nilai pada sebuah berkas JavaScript baik itu primitive value, function, array, object ataupun class, kita gunakan keyword export default. Contohnya seperti ini:

```
1. const coffeeStock = {
2. arabica: 100,
3. robusta: 150,
4. liberica: 200
5. }
6.
7. export default coffeeStock;
```

Lalu bagaimana cara untuk mengimpor nilainya? Kita dapat melakukannya dengan menggunakan keyword *import* ... *from* seperti berikut ini:

```
    import coffeeStock from "./state.js";
```

Berbeda dengan gaya Node,js, kita gunakan keyword *import* untuk menggantikan *const*, *let*, ataupun *var* dalam mendeklarasi variabel yang diimpor. Lalu di sana juga kita menggunakan from dalam menspesifikasikan lokasi berkas JavaScript-nya.

Ketika menggunakan export default, kita dapat menggunakan penamaan apa saja ketika mendeklarasikan variabel dalam mengimpor nilainya.

```
1. // Kita dapat mengubah penamaan coffeeStock sesuai kebutuhan kita.
2. import stock from "./state.js";
```

Hal tersebut aman untuk dilakukan karena dengan menggunakan *export default*, dapat dipastikan hanua satu nilai uana diekspor pada satu berkas JavaScript.

Setelah kita berhasil mendapatkan nilai yang diekspor, kita dapat menggunakan nilainya layaknya variabel lokal biasa.

```
index.html app.js state.js
                                                                                               c

    import coffeeStock from "./state.js";

 2.
 3.
4. const displayStock = stock => {
 5. const coffeeStockListElement = document.querySelector("#coffee-stock-list");
      for(const type in stock) {
 6.
       const coffeeStockItemElement = document.createElement("li");
        coffeeStockItemElement.innerText = `${type}: ${stock[type]}`;
 9.
        coffeeStockListElement.appendChild(coffeeStockItemElement);
10
11. }
12.
13.
displayStock(coffeeStock);
```

Exporting and Importing Multiple Value (Named Export/Import)

Jika sebelumnya kita hanya melakukan ekspor satu nilai pada berkas JavaScript menggunakan default export, pada materi kali ini kita akan membahas bagaimana cara melakukan ekspor banyak nilai dalam satu berkas JavaScript dengan menggunakan ES6.

Named export digunakan untuk mengekspor banyak nilai dalam berkas JavaScript. Cara kerjanya mirip seperti pada Node.js. Nilai yang akan diekspor dituliskan di dalam objek literals, seperti ini:

```
1. const coffeeStock = {
2.    arabica: 100,
3.    robusta: 150,
4.    liberica: 200
5.    }
6.
7.    const isCoffeeMakerReady = true;
8.
9.    export { coffeeStock, isCoffeeMakerReady };
```

Lalu untuk mendapatkan nilai yang diekspor menggunakan named export, kita gunakan teknik destructuring object.

```
    import { coffeeStock, isCoffeeMakerReady } from "./state.js";
    console.log(coffeeStock);
```

```
4. console.log(isCoffeeMakerReady);
5.
6. /* output:
7. { arabica: 100, robusta: 150, liberica: 200 }
8. true
9. */
```

Karena named import menggunakan teknik destructuring object untuk mendapatkan nilainya, maka pastikan penamaan variabel sesuai dengan nama variabel yang diekspor. Jika terjadi kesalahan penulisan, maka akan terjadi eror seperti berikut:

Namun jika kita tetap ingin mengubah penamaan variabel dari named import, kita bisa melakukannya dengan menambahkan keyword as setelah penamaan variabelnya.

```
    import { coffeeStock as stock, isCoffeeMakerReady } from "./state.js";
    console.log(stock);
    console.log(isCoffeeMakerReady);
    /* output:
    { arabica: 100, robusta: 150, liberica: 200 }
    true
    */
```

Setelah kita berhasil mendapatkan nilai yang diekspor, kita dapat menggunakan nilainya layaknya variabel lokal biasa.

```
index.html app.js state.js
                                                                                               •

    import { coffeeStock, isCoffeeMakerReady } from "./state.js";

                                                                                                _
  4. const displayStock = stock => {
  5. const coffeeStockListElement = document.querySelector("#coffee-stock-list");

    for (const type in stock) {
    const coffeeStockItemElement = document.createElement("li");

        coffeeStockItemElement.innerText = `${type}: ${stock[type]}`;
  8.

    coffeeStockListElement.appendChild(coffeeStockItemElement);

 10.
 11. }
 12.
 13.
 14. const coffeeOrder = (type, miligrams) => {
 15. return new Promise((resolve, reject) => {
        if (isCoffeeMakerReady) {
 16.
 17. if (coffeeStock[type] >= miligrams) {
            resolve("Kopi berhasil dipesan!");
 18.
 19. } else {
            reject("Maaf Stock kopi habis!")
 21. }
 22.
        } else {
         reject("Maaf mesin sedang rusak!")
 23.
← KEMBALI KE MATERI SEBELUMNYA
                                                          LANJUTKAN KE MATERI BERIKUTNYA →
```





PERUSAHAAN	PROGRAM	SUPPORT
Tentang Kami	Academy	Bantuan
Blog	Challenge	FAQ
Berita Terbaru	Event	Hubungi Kami
	Job	
	Rewards	

