Contents

[**Pengenalan Struktur Data** 1](#_Toc107415309)

**Pengenalan Struktur Data**

Sebelumnya kita sempat mempelajari tentang variabel dan tipe data yang ada di JavaScript. Tipe data yang kita bahas sebelumnya termasuk dalam kategori tipe data primitif yang hanya bisa menyimpan satu nilai.

Pada modul ini kita akan membahas tentang tipe data JavaScript yang dapat menyimpan struktur data yang lebih kompleks. Beberapa jenis struktur data yang akan kita pelajari, antara lain:

* Object
* Array
* Map
* Set

## Object

Kali ini kita akan berkenalan dengan tipe data object. Sebuah tipe data yang sangat berguna dalam pengembangan aplikasi dengan JavaScript. Object mampu menyimpan nilai dari beragam tipe data dan membentuk data yang lebih kompleks.

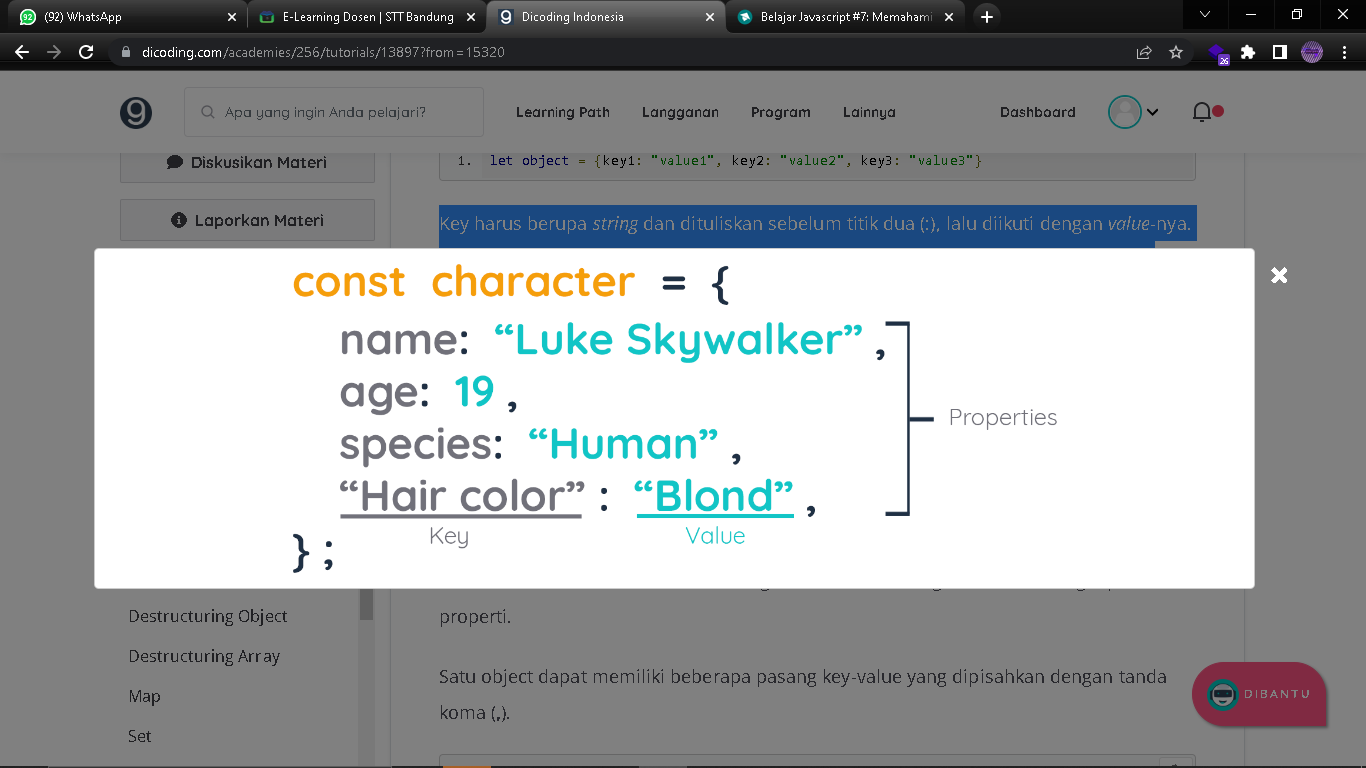
Untuk menetapkan objek pada sebuah variabel kita gunakan tanda kurung kurawal {}.

1. const user = {};

Object berisi pasangan key dan value yang juga dikenal dengan property. Key berperan mirip seperti nama variabel yang menyimpan sebuah nilai. Sementara, value berisi nilai dengan tipe data apa pun termasuk objek lain. Key dan value di dalam object dituliskan seperti berikut:

1. let object = {key1: "value1", key2: "value2", key3: "value3"}

Key harus berupa string dan dituliskan sebelum titik dua (:), lalu diikuti dengan value-nya. Meskipun key merupakan string, kita tidak perlu menuliskan tanda petik kecuali ada karakter khusus seperti spasi.



Tanda koma di akhir properti bersifat opsional. Namun, jika tanda koma tersebut ditulis akan lebih memudahkan ketika kita ingin memindah, mengubah, atau menghapus properti.

Satu object dapat memiliki beberapa pasang key-value yang dipisahkan dengan tanda koma (**,**).

1. const user = {firstName: "Luke", lastName: "Skywalker", age: 19, isJedi: true};

Dalam menuliskan objek, baris baru tidaklah penting dan tidak akan berpengaruh apa pun. Sehingga lebih baik setiap kita menetapkan key-value buatlah baris baru untuk memisahkan antar nilainya. Hal ini akan memudahkan kita dalam membaca dan memahami struktur data dari sebuah object.

1. const user = {
2. firstName: "Luke",
3. lastName: "Skywalker",
4. age: 19,
5. isJedi: true,
6. };

Kemudian untuk mengakses nilai dari properti object, kita dapat memanggil nama object lalu tanda titik dan diikuti nama propertinya. Contoh:

const user = {

firstName: "Luke",

lastName: "Skywalker",

age: 19,

isJedi: true,

};

console.log(`Halo, nama saya ${user.firstName} ${user.lastName}`);

console.log(`Umur saya ${user.age} tahun`);

Selain dot operator, kita juga bisa mengakses properti dari object menggunakan bracket atau tanda kurung siku.

1. user[“home world”];

Untuk mengakses key yang memiliki spasi atau karakter khusus lainnya maka kita perlu menggunakan bracket seperti di atas.

const user = {

firstName: "Luke",

lastName: "Skywalker",

age: 19,

isJedi: true,

"home world": "Tattooine"

};

console.log(`Halo, nama saya ${user.firstName} ${user.lastName}`);

console.log(`Umur saya ${user.age} tahun`);

console.log(`Saya berasal dari ${user["home world"]}`);

/\* output

Halo, nama saya Luke Skywalker

Umur saya 19 tahun

Saya berasal dari Tattooine

Setelah mempelajari bagaimana membuat object dan menampilkan property di dalamnya, selanjutnya kita akan memodifikasi sebuah object. Untuk mengubah nilai properti di dalam object kita gunakan assignment operator (=).

const spaceship = {

name: "Millenium Falcon",

manufacturer: "Corellian Engineering Corporation",

maxSpeed: 1200,

color: "Light gray"

};

spaceship.color = "Glossy red";

spaceship["maxSpeed"] = 1300;

console.log(spaceship);

/\* output

{

name: 'Millenium Falcon',

manufacturer: 'Corellian Engineering Corporation',

maxSpeed: 1300,

color: 'Glossy red'

}

\*/

Tunggu dulu. Object spaceship dideklarasikan sebagai const, tetapi kenapa kita bisa mengubah nilainya?

Yang perlu diperhatikan adalah mengubah nilai berbeda dengan menginisialisasi ulang nilai. Ketika membuat sebuah object, kita tidak terikat dengan properti di dalamnya sehingga kita masih bisa memodifikasi nilainya. Berbeda jika kita menginisialisasi ulang variabel dari object.

1. const spaceship = {
2. name: "Millenium Falcon",
3. manufacturer: "Corellian Engineering Corporation",
4. maxSpeed: 1200,
5. color: "Light gray"
6. };
8. spaceship = { name: "New Millenium Falcon" }; // Error

Ketika kita mengubah object menggunakan assignment operator dan property/key-nya sudah ada, maka nilai di dalamnya akan tergantikan dengan nilai yang baru. Sedangkan, jika property dengan nama key yang ditentukan tidak ditemukan, maka property baru akan ditambahkan ke object.

const spaceship = {

name: "Millenium Falcon",

manufacturer: "Corellian Engineering Corporation",

maxSpeed: 1200,

color: "Light gray"

};

spaceship.color = "Glossy red";

spaceship["maxSpeed"] = 1300;

spaceship.class = "Light freighter";

console.log(spaceship);

/\* output

{

name: 'Millenium Falcon',

manufacturer: 'Corellian Engineering Corporation',

maxSpeed: 1300,

color: 'Glossy red',

class: 'Light freighter'

}

\*/

Kita juga dapat menghapus property pada object menggunakan keyword delete seperti berikut:

const spaceship = {

name: "Millenium Falcon",

manufacturer: "Corellian Engineering Corporation",

maxSpeed: 1200,

color: "Light gray"

};

spaceship.color = "Glossy red";

spaceship["maxSpeed"] = 1300;

delete spaceship.manufacturer;

console.log(spaceship);

/\* output

{ name: 'Millenium Falcon', maxSpeed: 1300, color: 'Glossy red' }

\*/

## Array

Array merupakan tipe data yang dapat mengelompokkan lebih dari satu nilai dan menempatkannya dalam satu variabel. Contoh:

let myArray = ["Cokelat", 42.5, 22, true, "Programming"];

console.log(myArray);

/\* output:

[ 'Cokelat', 42.5, 22, true, 'Programming' ]

\*/

Perbedaan array dengan object adalah data pada array disusun secara berurutan dan diakses menggunakan index. Untuk mengakses nilai di dalam array, kita gunakan tanda kurung siku [] yang di dalamnya berisi angka yang merupakan posisi nilai yang ingin diakses.

1. console.log(myArray[1]);

Jika Anda mengira bahwa konsol akan menampilkan teks ‘Cokelat’, maka tebakan Anda kurang tepat. Dalam sebuah array, indeks dimulai dari 0, sehingga ketika kita mengakses data pada myArray yang berada pada indeks ke-1 artinya data tersebut merupakan data pada posisi ke-2. Jadi nilai yang akan ditampilkan pada konsol adalah 42.5.

let myArray = ["Cokelat", 42.5, 22, true, "Programming"];

console.log(myArray[1]);

/\* output:

42.5

\*/

Lalu, apa yang akan terjadi jika kita berusaha mengakses index di luar ukuran array-nya? Jika kita mengakses nilai array lebih dari index-nya, maka hasilnya akan undefined. Index terakhir array selalu jumlah nilai array - 1.

let myArray = ["Cokelat", 42.5, 22, true, "Programming"];

console.log(myArray[0]);

console.log(myArray[1]);

console.log(myArray[2]);

console.log(myArray[3]);

console.log(myArray[4]);

console.log(myArray[5]);

console.log("Panjang nilai myArray adalah " + myArray.length + ".");

/\* output:

Cokelat

42.5

22

true

Programming

undefined

Panjang nilai myArray adalah 5.

\*/

Sejauh ini kita baru belajar menginisialisasi dan mengakses elemen dari sebuah array. Pastinya Anda bertanya, “Bagaimana kita memanipulasi data pada array tersebut?”

Nah, untuk menambahkan data ke dalam array, kita bisa menggunakan metode push(). Fungsi push ini akan menambahkan data di akhir array.

const myArray = ["Cokelat", 42.5, 22, true, "Programming"];

myArray.push('JavaScript');

console.log(myArray);

/\* output

[ 'Cokelat', 42.5, 22, true, 'Programming', 'JavaScript' ]

\*/

Sedangkan untuk mengeluarkan data atau elemen terakhir dari array, kita bisa gunakan metode pop().

const myArray = ["Cokelat", 42.5, 22, true, "Programming"];

myArray.pop();

console.log(myArray);

/\* output

[ Cokelat, 42.5, 22, true ]

\*/

Metode lain yang bisa kita gunakan untuk memanipulasi data pada array adalah shift() dan unshift(). Metode shift() digunakan untuk mengeluarkan elemen pertama dari array, sementara unshift() digunakan untuk menambahkan elemen di awal array.

const myArray = ["Cokelat", 42.5, 22, true, "Programming"];

myArray.shift();

myArray.unshift("Apple");

console.log(myArray);

/\* output

[ 'Apple', 42.5, 22, true, 'Programming' ]

\*/

Lalu bagaimana jika kita ingin menghapus data dari array? Sama seperti object, kita bisa menggunakan keyword delete.

const myArray = ["Cokelat", 42.5, 22, true, "Programming"];

delete myArray[1];

console.log(myArray);

/\* output

[ 'Cokelat', <1 empty item>, 22, true, 'Programming' ]

\*/

Namun, perhatikan di sini bahwa keyword delete hanya menghapus data pada index yang ditentukan lalu membiarkan posisi tersebut kosong. Untuk menghapus elemen, gunakan metode splice() seperti ini:

const myArray = ["Cokelat", 42.5, 22, true, "Programming"];

myArray.splice(2, 1); // Menghapus dari index 2 sebanyak 1 elemen

console.log(myArray);

/\* output

[ 'Cokelat', 42.5, true, 'Programming' ]

\*/

metode splice() bisa juga untuk menambahkan array baru seperti ini:

const fruits = ["Banana", "Orange", "Apple", "Mango"];

// At position 2, add 2 elements:

fruits.splice(2, 1, "Lemon", "Kiwi");

console.log(fruits);

//Output :

Banana,Orange,Lemon,Kiwi,Mango

## Spread Operator

Masih terkait dengan array, ES6 memiliki fitur menarik untuk membantu pengelolaan array menjadi lebih mudah, yaitu spread operator.

Sesuai namanya “spread”, fitur ini digunakan untuk menyebarkan nilai array atau lebih tepatnya iterable object menjadi beberapa elemen. Spread operator dituliskan dengan tiga titik (...). Mari kita lihat contoh kode berikut:

1. const favorites = ["Seafood", "Salad", "Nugget", "Soup"];
3. console.log(favorites);
5. /\* output
6. [ 'Seafood', 'Salad', 'Nugget', 'Soup' ]
7. \*/

Pada kode tersebut hasil yang dicetak adalah sebuah array (ditunjukkan dengan tanda [ ]), karena memang kita mencetak nilai favorites itu sendiri. Nah, dengan menggunakan spread operator kita dapat menyebarkan nilai-nilai dalam array tersebut.

const favorites = ["Seafood", "Salad", "Nugget", "Soup"];

console.log(...favorites);

/\* output

Seafood Salad Nugget Soup

\*/

Terlihat perbedaannya? Mengapa bisa demikian? Spread operator bekerja seperti meleburkan nilai array menjadi beberapa elemen sesuai panjang nilai array-nya. Sehingga jika kita menuliskan kode seperti ini:

1. console.log(...favorites);

Sama seperti kita menuliskan kode seperti ini:

1. console.log(favorites[0], favorites[1], favorites[2], favorites[3]);

Spread operator dapat digunakan untuk menggabungkan dua buah array ke dalam array baru. Jika tidak menggunakan spread operator ini maka hasilnya akan seperti ini:

const allFavorites = [favorites, others];

console.log(allFavorites);

/\* output

[

[ 'Seafood', 'Salad', 'Nugget', 'Soup' ],

[ 'Cake', 'Pie', 'Donut' ]

]

\*/

Nilai array tidak akan tergabung. Alih-alih menggabungkan nilainya, variabel allFavorites menjadi array baru yang menampung dua array di dalamnya. Nah, lantas bagaimana jika kita mencoba menggunakan spread operator?

const favorites = ["Seafood", "Salad", "Nugget", "Soup"];

const others = ["Cake", "Pie", "Donut"];

const allFavorites = [...favorites, ...others];

console.log(allFavorites);

/\* output

[ 'Seafood', 'Salad', 'Nugget', 'Soup', 'Cake', 'Pie', 'Donut' ]

\*/

Yup, dengan menggunakan spread operator nilai dua array tersebut berhasil tergabung.

Selain array, spread operator juga bisa digunakan untuk object literals. Hal ini memungkinkan kita dapat menggabungkan beberapa object dengan kode yang lebih ringkas.

const obj1 = { firstName: 'Obi-Wan', age: 30 };

const obj2 = { lastName: 'Kenobi', gender: 'M' };

const newObj = { ...obj1, ...obj2 };

console.log(newObj);

/\* output

{ firstName: 'Obi-Wan', age: 30, lastName: 'Kenobi', gender: 'M' }

\*/

## Destructuring Object & Array

Literasi object dan array adalah dua hal yang paling banyak digunakan dalam mengelola data di JavaScript. JSON (JavaScript Object Notation) merupakan format data paling populer yang digunakan dalam transaksi data saat ini.

1. [
2. {
3. "id": 14,
4. "title": "Belajar Fundamental Aplikasi Android",
5. "author": "Google ATP"
6. },
7. {
8. "id": 51,
9. "title": "Belajar Membuat Aplikasi Android untuk Pemula",
10. "author": "Google ATP"
11. },
12. {
13. "id": 123,
14. "title": "Belajar Dasar Pemrograman Web",
15. "author": "Dicoding Indonesia"
16. },
17. {
18. "id": 163,
19. "title": "Belajar Fundamental Front-End Web Development",
20. "author": "Dicoding Indonesia"
21. }

Jika kita lihat pada struktur JSON di atas, kita dapat menyimpulkan struktur tersebut dibangun dari array dan object. Karena kedua hal ini banyak digunakan untuk mengelola data pada JavaScript untuk memudahkan developer, ES6 menambahkan fitur untuk destructuring object dan array.

Apa sebenarnya destructuring object dan array itu? Destructuring dalam JavaScript merupakan sintaksis yang dapat mengeluarkan nilai dari array atau properties dari sebuah object ke dalam satuan yang lebih kecil.

Secara tidak sadar mungkin kita pernah melakukan destructuring. Namun, sebelum ES6 hal tersebut dilakukan dengan cara seperti ini:

1. const foods = ['Pie', 'Cake', 'Honey']
3. const myFood = foods[0]
4. const yourFood = foods[1]
5. const ourFood = foods[2]
7. console.log(myFood, yourFood, ourFood)
9. /\* output:
10. Pie Cake Honey
11. \*/

Perhatikan kode pada destructuring object di atas, Kode tersebut akan mengekstraksi nilai yang berada di dalam object profile dan menyimpannya pada variabel lokal yang memiliki nama sama dengan properti di dalam object profile. Mungkin mengekstraksi nilai dari object dengan langkah ini terlihat mudah, tetapi bayangkan jika object memiliki banyak properti dan harus melakukan hal tersebut secara manual satu persatu. Terlalu banyak kode yang dituliskan berulang, bukan?

Itulah alasan ES6 menambahkan fitur yang memudahkan kita untuk destructuring object maupun array. Ketika kita ingin memecah struktur data menjadi bagian-bagian yang lebih kecil, kita akan dipermudah untuk mendapatkan data yang diinginkan.

Lantas bagaimana cara melakukan destructuring object dan array pada ES6? Mari kita simak materi berikutnya.

## Destructuring Object

Penulisan sintaksis destructuring object pada ES6 menggunakan object literal ({ }) di sisi kiri dari operator assignment.

const profile = {

firstName: "John",

lastName: "Doe",

age: 18

}

const {firstName, lastName, age} = profile;

console.log(firstName, lastName, age);

/\* output:

John Doe 18

\*/

Pada contoh di atas tanda kurung kurawal merepresentasikan object yang akan didestrukturisasi. Di dalamnya terdapat firstName, lastName, dan age yang merupakan variabel untuk menyimpan nilai properti dari object profile. Kita juga perlu perhatikan penamaan variabelnya. Pastikan penamaannya sama seperti properti object-nya. Melalui nama variabel inilah nilai-nilai properti object akan dimasukkan secara otomatis. Sehingga variabel firstName akan berisikan nilai profile.firstName, lastName akan berisikan nilai profile.lastName, begitu juga dengan variabel age akan berisikan nilai profile.age.

Dalam destructuring object, kita bisa menentukan salah satu nilai yang ingin kita desktrukturisasikan. Sehingga kita tidak perlu membuat variabel sebanyak properti yang dimiliki objeknya, contohnya:

1. const {lastName} = profile;

### Destructuring Assignment

Pada contoh sebelumnya, kita telah melakukan destructuring object pada deklarasi variabel. Namun, pada kasus tertentu mungkin kita perlu melakukannya pada variabel yang sudah dideklarasikan.

const profile = {

firstName: "John",

lastName: "Doe",

age: 18

}

let firstName = "Dimas";

let age = 20;

// menginisialisasi nilai baru melalui destructuring object

({firstName, age} = profile);

console.log(firstName);

console.log(age);

/\* output:

John

18

\*/

Saat melakukan destructuring assignment, kita perlu menuliskan destructuring object di dalam tanda kurung. Jika tidak menuliskan tanda kurung, tanda kurung kurawal akan membuat JavaScript mengira kita membuat block statement, sementara block statement tidak bisa berada pada sisi kiri assignment.

1. // tidak bisa karena JavaScript mengira kita membuat block statement
2. // block statement tidak bisa berada pada sisi kiri assignment
3. {firstName, age} = profile;

Nah, inilah fungsinya tanda kurung. Ia akan memberi tahu JavaScript bahwa tanda kurawal di dalamnya bukan sebuah block statement melainkan sebuah expression, sehingga assignment dapat dilakukan.

1. ({firstName, age} = profile);

### Default Values

Ketika kita mendestruksikan objek dan menetapkan variabel dengan nama yang bukan merupakan properti dari objek, maka nilai dari variabel tersebut menjadi undefined. Contohnya: