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)

What is Class in JavaScript?

Kebanyakan bahasa pemrograman memanfaatkan class dalam penerapan paradigma OOP. Pada JavaScript tidak ada konsep class murni bahkan hingga saat ini. Namun bukan berarti kita tidak bisa menerapkan paradigma OOP sepenuhnya.

Seperti yang kita ketahui, class pada OOP merupakan sebuah blueprint. Jika berbicara mengenai blueprint, JavaScript memiliki konsep untuk membuatnya tanpa melalui class. Konsep tersebut adalah ***prototype***. Sejak awal developer menggunakan konsep ini dalam menerapkan paradigma OOP di JavaScript.

Walaupun dapat menggantikan class, konsep prototype tidak serupa dengan class pada bahasa lain. Terlebih pada penulisan sintaknya. Hal ini menjadikan banyak developer kebingungan khususnya developer yang dasarnya dari bahasa *class basis* (seperti Java, C++, C#, Swift, etc.). Meskipun banyak juga developer yang merasa bahwa class tidak diperlukan pada JavaScript, namun fitur class ini tetap menjadi salah satu yang dikembangkan pada ECMAScript 6.

A Class Before ES6

Sebelum ES6, Hal yang paling mendekati dengan class yaitu membuat sebuah objek menggunakan constructor function dan keyword `new`, lalu untuk menambahkan method kita gunakan konsep `prototype`.

```
function Car(manufacture, color) {
    this.manufacture = manufacture;
    this.color = color;
    this.enginesActive = false;
}

Car.prototype.startEngines = function () {
    console.log('Mobil dinyalakan...');
    this.enginesActive = true;
};

Car.prototype.info = function () {
    console.log("Manufacture: " + this.manufacture);
    console.log("Color: " + this.color);
    console.log("Engines: " + (this.enginesActive ? "Active" : "Inactive"));
}

var johnCar = new Car("Honda", "Red");
johnCar.startEngines();
johnCar.info();

/* output:
Mobil dinyalakan...
```

Pada kode di atas `Car` merupakan *constructor function* yang akan membuat instance `Car` baru setiap kali kode `new Car()` dieksekusi. Melalui `Car.prototype`, method `startEngines()` dan `carInfo()` diwarisi pada setiap *instance* Car yang dibuat, sehingga `johnCar` (sebagai instance `Car`) dapat mengakses kedua method tersebut.

Teknik dasar ini yang digunakan dalam membuat class di JavaScript sebelum ES6.

> "Mengapa method pada instance harus disimpan pada `prototype` atau `__proto__` ? Mengapa tidak disimpan pada `constructor` sama seperti properti?
>
> Alasannya adalah jika kita menyimpan method pada `constructor` maka method tersebut akan selalu dibuat ketika *instance* dibuat. Ini bukan pendekatan yang baik karena jika method memiliki kode yang banyak, maka akan memakan memori yang banyak.
>
> Sedangkan jika menggunakan prototype, method hanya dibuat satu kali. Dan method tersebut diwarisi kepada setiap *instance* yang dibuat."

ES6 Classes

Dengan hadirnya class pada ES6, pembuatan class di JavaScript menjadi lebih mudah dan juga penulisannya mirip seperti bahasa pemrograman lain berbasis class. Pembuatan class pada ES6 menggunakan keyword `class` itu sendiri kemudian diikuti dengan nama class-nya.

```
class Car {

    // Sama seperti function constructor
    constructor(manufacture, color) {
        this.manufacture = manufacture;
        this.color = color;
        this.enginesActive = false;
    }
```

```
    // Sama seperti Car.prototype.startEngine
    startEngines() {
        console.log('Mobil dinyalakan...');
        this.enginesActive = true;
    }

    // Sama seperti car.prototype.info
    info() {
        console.log(`Manufacture: ${this.manufacture}`);
        console.log(`Color:  ${this.color}`);
        console.log(`Engines: ${this.enginesActive ? "Active" : "Inactive"}`);
    }

}
```

Jika Anda terbiasa dengan bahasa pemrograman berbasis class, pasti penulisannya sangat serupa bukan? Walaupun dari segi sintaksis pembuatan class antara keduanya cukup berbeda, namun perilaku dari objek yang dibuat dengan keduanya sama. Inilah mengapa class pada ES6 hanya sebuah *syntactic sugar* dari konsep *prototype* yang sudah ada.

Mari kita bahas class pada ES6 lebih lanjut.

> "Ketika kita hendak membuat sebuah *constructor function* ataupun *class*. Secara *code convention* (aturan penulisan), gunakan PascalCase dalam penamaannya. Contohnya `Car` daripada `car`, `SportCar` daripada `sportCar` atau `Sportcar`"

⬅ KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA ➡