


Job  DISKUSIKAN MATERI

LAPORKAN MATERI

Developer ▾

Daftar Modul

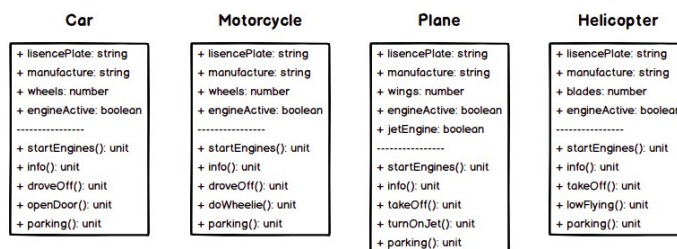
 Izmila Prastika ▾ Masukkan kata kunci

	
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)	✓

Inheritance

Dalam gambaran dunia nyata, banyak objek yang berbeda tetapi punya kesamaan atau kemiripan tertentu. Contohnya mobil dengan motor memiliki banyak kesamaan karena objek tersebut merupakan kendaraan. Mobil merupakan kendaraan darat begitu juga dengan motor. Mungkin yang membedakan objek tersebut adalah jumlah roda dan kapasitas penumpang yang dapat ditampung.

Sama halnya pada OOP, beberapa objek yang berbeda bisa saja memiliki kesamaan dalam hal tertentu. Di situlah konsep inheritance atau pewarisan harus diterapkan. Pewarisan dapat mencegah kita melakukan perulangan kode. Untuk lebih memahaminya lihatlah contoh bagan pada sebuah kelas berikut:



Pada bagan di atas kita dapat lihat class `Car`, `Motorcycle`, `Plane`, dan `Helicopter` memiliki banyak properti yang sama seperti `licensePlate`, `manufacture`, dan `engineActive`. Kemudian memiliki beberapa method yang sama seperti `startEngines()`, `info()`, dan `parking()`.

Jika kita ubah diagram class `Car` di atas menjadi sebuah kode maka kode tampak seperti ini:

```
1. class Car {
2.   constructor(licensePlate, manufacture, wheels) {
3.     this.licensePlate = licensePlate;
4.     this.manufacture = manufacture;
5.     this.wheels = wheels;
6.     this.engineActive = false;
7.   }
8.
9.   startEngines() {
10.    console.log(`Mesin Kendaraan ${this.licensePlate} dinyalakan!`);
11.  }
12.
13.   info() {
14.    console.log(`Nomor Kendaraan: ${this.licensePlate}`);
```

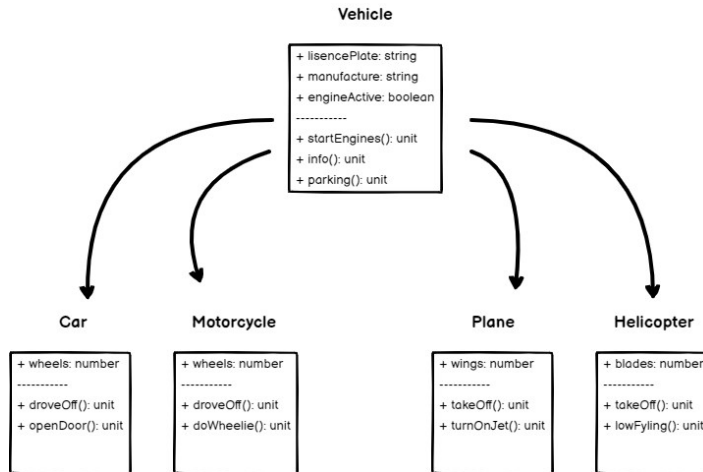
```

15.     console.log(`Manufacture: ${this.manufacture}`);
16.     console.log(`Mesin: ${this.engineActive ? "Active": "Inactive"}`);
17. }
18.
19. droveOff() {
20.     console.log(`Kendaraan ${this.licensePlate} melaju!`);
21. }
22.
23. openDoor() {

```

Tidak ada masalah dengan kode tersebut, tetapi jika kita akan membuat kelas lainnya seperti **Motorcycle**, **Plane**, dan **Helicopter** maka kita harus menuliskan *properti* dan *method* yang sama secara berulang.

Dengan teknik *inheritance*, kita bisa mengelompokkan properti dan method yang sama. Caranya dengan membuat sebuah kelas baru yang nantinya akan diturunkan sifatnya pada class lain:



Ketika class Vehicle telah dibuat, kelas lainnya dapat melakukan **extends** pada kelas tersebut untuk mewarisi sifatnya. Dalam pewarisan, class Vehicle dapat disebut sebagai super atau parent class. Kelas yang mewarisi sifat dari parent class disebut dengan child class.

Pada JavaScript jika kita ingin mewariskan sifat class, lakukan dengan keyword **extends** seperti berikut:

```

1. class ChildClass extends ParentClass {
2.
3. }

```

Sebagai contoh mari kita buat class **Vehicle** yang nantinya akan kita gunakan sebagai parent class.

```

1. class Vehicle {
2.     constructor(licensePlate, manufacture) {
3.         this.licensePlate = licensePlate;
4.         this.manufacture = manufacture;
5.         this.engineActive = false;
6.     }
7.
8.     startEngines() {
9.         console.log(`Mesin kendaraan ${this.licensePlate} dinyalakan!`);
10.    }
11.
12.    info() {
13.        console.log(`Nomor Kendaraan: ${this.licensePlate}`);
14.        console.log(`Manufacture: ${this.manufacture}`);
15.        console.log(`Mesin: ${this.engineActive ? "Active": "Inactive"}`);
16.    }
17.
18.    parking() {
19.        console.log(`Kendaraan ${this.licensePlate} parkir!`);
20.    }
21. }

```

Kemudian kita bisa membuat class **Car** sebagai child class dari **Vehicle**.

```

1. class Car extends Vehicle {
2.     constructor(licensePlate, manufacture, wheels) {
3.         super(licensePlate, manufacture);
4.         this.wheels = wheels;
5.     }
6.
7.     droveOff() {
8.         console.log(`Kendaraan ${this.licensePlate} melaju!`);
9.     }
10.
11.    openDoor() {
12.        console.log(`Membuka pintu!`);
13.    }
14. }

```

Dengan begitu selain properti dan method yang terdapat di dalamnya, class **Car** juga dapat

mengakses seluruh properti dan method yang terdapat pada class **Vehicle**.

```
1. class Vehicle {
2.   constructor(licensePlate, manufacture) {
3.     this.licensePlate = licensePlate;
4.     this.manufacture = manufacture;
5.     this.engineActive = false;
6.   }
7.
8.   startEngines() {
9.     console.log(`Mesin kendaraan ${this.licensePlate} dinyalakan!`);
10.  }
11.
12.   info() {
13.     console.log(`Nomor Kendaraan: ${this.licensePlate}`);
14.     console.log(`Manufacture: ${this.manufacture}`);
15.     console.log(`Mesin: ${this.engineActive ? "Active": "Inactive"}`);
16.   }
17.
18.   parking() {
19.     console.log(`Kendaraan ${this.licensePlate} parkir!`);
20.   }
21. }
22.
23. class Car extends Vehicle {
```

Oiya pada **constructor** class **Car**, kita melihat penggunaan **super()**, apa itu maksudnya?

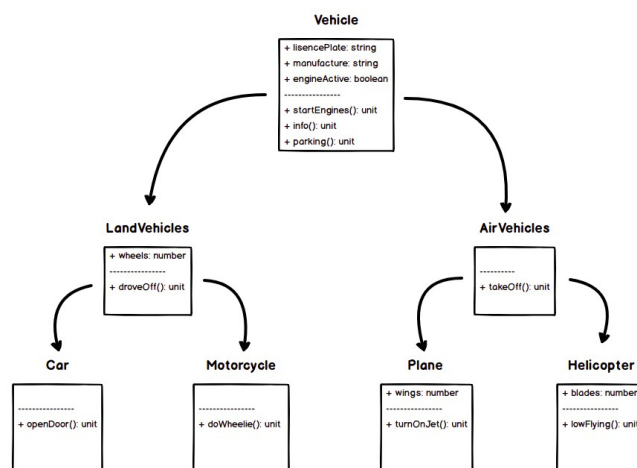
Keyword **super** digunakan untuk mengakses properti dan method yang ada pada induk class ketika berada pada *child class*. Jadi **super(licensePlate, manufacture)** di atas berarti kita mengakses **constructor** dari *parent class* dan mengirimkan **licensePlate**, dan **manufacture** sebagai data yang dibutuhkan olehnya agar objek (*instance*) **Car** berhasil dibuat.

Penggunaan **super** sangat berguna ketika kita hendak menjalankan **method overriding** pada method parent. Contohnya kita akan melakukan **method overriding** pada method **info()** dengan menambahkan informasi jumlah roda pada mobil, maka kita dapat melakukannya dengan seperti ini:

```
1. class Car extends Vehicle {
2.   constructor(licensePlate, manufacture, wheels) {
3.     super(licensePlate, manufacture);
4.     this.wheels = wheels;
5.   }
6.
7.   droveOff() {
8.     console.log(`Kendaraan ${this.licensePlate} melaju!`);
9.   }
10.
11.   openDoor() {
12.     console.log(`Membuka pintu!`);
13.   }
14.
15.   /* overriding method info dari parent class */
16.   info() {
17.     super.info();
18.     console.log(`Jumlah roda: ${this.wheels}`);
19.   }
20. }
21.
22. const johnCar = new Car("H121S", "Honda", 4);
23. johnCar.info();
```

Dalam melakukan pewarisan kelas, tidak ada tingkatan yang membatasinya. Maksudnya, kita dapat mewariskan sifat kelas A pada kelas B, lalu kelas B mewarisi sifatnya kembali pada kelas C dan selanjutnya. Sama halnya dengan Nenek kita mewarisi sifatnya kepada orang tua kita kemudian orang tua kita mewarisi sifatnya kepada kita.

Sehingga jika dilihat dari bagan sebelumnya, class tersebut masih bisa dikelompokkan kembali menjadi seperti ini:



← 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

Copyright © 2020 - Dicoding Indonesia. All rights reserved.

[Terms](#) [Privacy](#)

