# Front End Bootcamp
# PT. Mahardika Solusi Teknologi

Ariel Arliyanus, S.Kom

**Chapter 11 – JavaScript Class**

Chapter 11
# JavaScript Class

## Class

Pada materi sebelumnya, telah dipelajari tentang Object Literals di Javascript. Dengan Object kita dapat merepresentasikan segala hal termasuk program yang akan kita buat dengan Javascript. Materi kali ini akan mempelajari object dalam bentuk Class.

Di dalam dunia pemrograman dikenal sebuah konsep dengan nama OOP atau Object Oriented Programming. Secara sederhana, dengan konsep OOP maka segala sesuatu dapat kita anggap sebagai sebuah Object atau cetakan (Blueprint). Contohnya terdapat Object Class bernama “Car” yang berarti mobil. Suatu (cetakan) mobil biasanya memiliki nama dan dibuat oleh sebuah pabrikan otomotif (factory). Selain itu mobil dapat memiliki fungsi untuk berjalan, mengerem, membunyikan klakson, dll.

## Mendefinisikan Sebuah Class

Class sebetulnya adalah sebuah functions yang spesial, dan seperti function yang kita ketahui sebelumnya bahwa functions dapat dideklarasikan dan dipanggil begitu pula dengan Class.

### Deklarasi Class

Cara yang pertama untuk membuat sebuah class yaitu dengan mendeklarasikannya. Caranya adalah tuliskan “Class” diikuti dengan nama class-nya.

```
class Car {
    constructor(brand,factory) {
        this.brand = brand
        this.factory = factory
        this.sound = "honk! honk!vroomvroom"
    }
}
```

## Ekspresi Class

Cara lain untuk membuat sebuah class yaitu dengan cara membuat sebuah variabel. Class tersebut boleh diberikan nama atau tidak diberi nama. contohnya sebagai berikut:

```
// Tidak diberi nama
var Car = class {
    constructor(brand, factory) {
        this.brand = brand
        this.factory = factory
    }
}
console.log(Car.name) // Car

// Diberi nama
var Car = class Car2 {
    constructor(brand, factory) {
        this.brand = brand
        this.factory = factory
    }
}
console.log(Car.name) // Car2
```

Nama sebuah Class biasanya menggunakan kapital pada huruf pertama nya. Jika terdapat dua kata atau lebih maka huruf pertama pada kata yang selanjutnya harus kapital.

```
class Car{} // BENAR
class car{} // SALAH
class SportsCar {} // BENAR
class sportscar {} // SALAH
```

# Method

Sintaks constructor pada class merupakan method khusus, dimana dilakukan inisialisasi properties, yang akan dieksekusi secara otomatis ketika class dibuat, dan ia harus memiliki nama “constructor”. (Jika tidak dituliskan, maka Javascript akan menambahkan method constructor kosong secara otomatis).

Kita juga dapat membuat method sendiri, dengan sintaks yang sudah biasa kita gunakan:

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return "I have a " + this.carname;
  }
}

mycar = new Car("Ford");
console.log(mycar.present()) // I have a Ford
```

Seperti yang terlihat pada contoh di atas, method dapat digunakan dengan cara memanggil nama class dan method nya ditambah dengan kurung buka dan kurung tutup. Parameter bisa dimasukan di dalam tanda kurung jika diperlukan, seperti pada contoh di bawah.

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present(x) {
    return x + ", I have a " + this.carname;
  }
}

mycar = new Car("Ford");
console.log(mycar.present("Hello"));
```

# Static Method

Static methods didefinisikan hanya untuk class itu sendiri. sehingga, jika melihat pada contoh sebelumnya static method hanya bisa diakses melalui Car, dan tidak bisa melalui mycar:

```
 class Car {
  constructor(brand) {
    this.carname = brand;
  }
  static hello() {
    return "Hello!!";
  }
}

mycar = new Car("Ford");

// memanggil 'hello()' pada class Car:
console.log(Car.hello());

// dan tidak bisa pada 'mycar':
// console.log(mycar.hello());
// jika menggunakan sintaks tersebut akan memunculkan error.
```

# Inheritance

Untuk membuat inheritance dari suatu class, gunakan keyword `extends`. Class yang dibuat dengan metode inheritance, akan memiliki method yang sama dengan class asalnya. Contoh berikut adalah class Model yang merupakan inheritance dari class Car:

```
 class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}

mycar = new Model("Ford", "Mustang");
console.log(mycar.show());
```

Method `super()` mengacu pada class asalnya, dimana dengan menggunakan method `super()` di dalam method constructor, kita dapat memanggil constructor class asalnya dan mengakses property dan method nya.

# Getters dan Setters

Pada class juga kita dapat menggunakan getter dan setter. getter dan setter dapat digunakan untuk melakukan proses tertentu pada suatu property, sebelum property tersebut digunakan. Untuk menambahak getter dan setter pada class, gunakan keyword `get` dan `set`. (Meskipun getter merupakan sebuah method, namun dalam menggunakannya tidak digunakan “( )” setelah memanggil method tersebut, seperti pada contoh di bawah).

```
 class Car {
  constructor(brand) {
    this.carname = brand;
  }
  get cnam() {
    return this.carname;
  }
  set cnam(x) {
    this.carname = x;
  }
}

mycar = new Car("Ford");
console.log(mycar.cnam); // Ford
// getter cnam digunakan tanpa "()"
```

Biasanya untuk membedakan method dengan property, property ditulis dengan menggunakan “_” di depan namanya, sementara method (termasuk getter dan setter) tidak.

```
class Car {
  constructor(brand) {
    this._carname = brand;
  }
  get carname() {
    return this._carname;
  }
  set carname(x) {
    this._carname = x;
  }
}

mycar = new Car("Ford");
mycar.carname = "Volvo"; // memanggil setter, mengubah Ford menjadi Volvo
console.log(mycar.carname); // Volvo
```

# Tugas

## *Soal 1*

Terdapat sebuah class Animal yang memiliki sebuah constructor name, default property legs= 4 dan cold_blooded = false.

**Release 0**

Buatlah class Animal yang menerima satu parameter constructor berupa name. Secara default class Animal akan memiliki property yaitu legs (jumlah kaki) yang bernilai 4 dan cold_blooded bernilai false.

Gunakan method getter dan setter untuk mengakses property di dalam class

```
class Animal {
    // Code class di sini
}

var sheep = new Animal("shaun");

console.log(sheep.name) // "shaun"
console.log(sheep.legs) // 4
console.log(sheep.cold_blooded) // false
sheep.legs = 3
console.log(sheep.legs)
```

**Release 1**

Buatlah class Frog dan class Ape yang merupakan inheritance dari class Animal. Perhatikan bahwa Ape (Kera) merupakan hewan berkaki 2, hingga dia tidak menurunkan sifat jumlah kaki 4 dari class Animal. class Ape memiliki function yell() yang menampilkan “Auooo” dan class Frog memiliki function jump() yang akan menampilkan “hop hop”.

```
// Code class Ape dan class Frog di sini

var sungokong = new Ape("kera sakti")
sungokong.yell() // "Auooo"
sungokong.legs = 2
console.log(sungokong.name)
console.log(sungokong.legs)
console.log(sungokong.cold_blooded)

var kodok = new Frog("buduk")
kodok.jump() // "hop hop"
console.log(kodok.name)
console.log(kodok.legs)
console.log(kodok.cold_blooded)
```

## *Soal 2*

Terdapat sebuah class dengan nama `Clock` yang ditulis seperti penulisan pada function, ubahlah fungsi tersebut menjadi class dan pastikan fungsi tersebut tetap berjalan dengan baik. Jalankan fungsi di terminal/console Anda untuk melihat hasilnya. (tekan tombol Ctrl + C pada terminal untuk menghentikan method clock.start())

Hint: Fokus soal ini hanya pada kegiatan mengubah struktur function `Clock` menjadi class. Jangan lupa menambahkan constructor di dalam class, dan ubah function di dalam `Clock` menjadi method pada class.

```
function Clock({ template }) {

  var timer;

  function render() {
    var date = new Date();

    var hours = date.getHours();
    if (hours < 10) hours = '0' + hours;

    var mins = date.getMinutes();
    if (mins < 10) mins = '0' + mins;

    var secs = date.getSeconds();
    if (secs < 10) secs = '0' + secs;

    var output = template
      .replace('h', hours)
      .replace('m', mins)
      .replace('s', secs);

    console.log(output);
  }

  this.stop = function() {
    clearInterval(timer);
  };

  this.start = function() {
    render();
    timer = setInterval(render, 1000);
  };

}

var clock = new Clock({template: 'h:m:s'});
clock.start();
```

function di atas diubah menjadi struktur class seperti berikut:

```
class Clock {
    // Code di sini
}

var clock = new Clock({template: 'h:m:s'});
clock.start();
```

berikut ini contoh output dari tugas 7:

```
----SOAL 1----
----Release 0----
shaun
4
false
3
----Release 1----
Auooo
kera sakti
2
false
hop hop
buduk
4
false

----SOAL 2----
07:52:16
07:52:17
07:52:18
07:52:19
07:52:20
```