Coding Subuh #14

Object Oriented Programming

Ahmad Muhardian

Agenda kita..

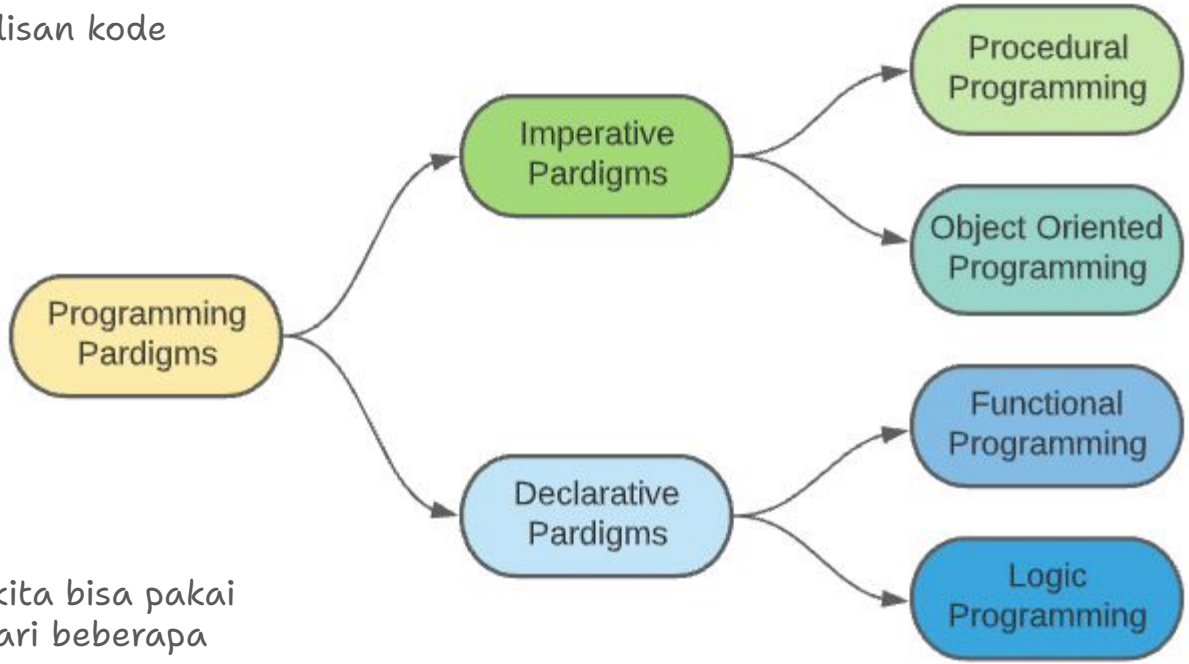
- Kenalan dengan OOP
- OOP di Javascript
- Latihan

Object Oriented Programming (OOP)

Programming Paradigms

Teknik/cara/gaya penulisan kode dalam pemrograman

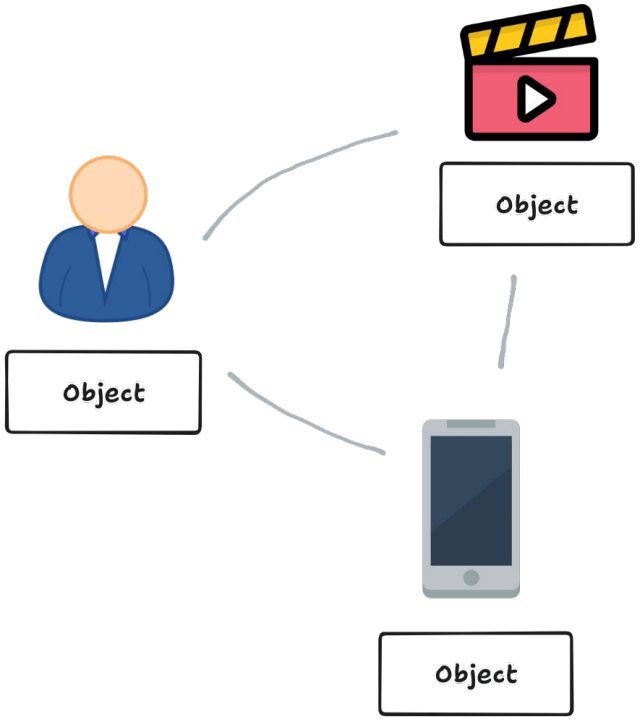
Simpelnya:
anggap paradigma ini seperti aliran dalam perguruan bela diri :)



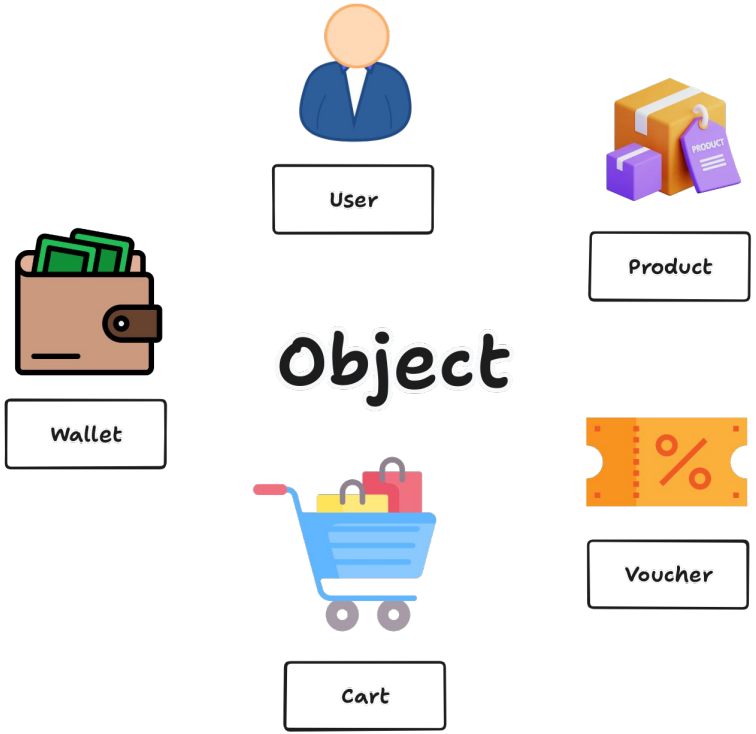
Dalam pemrograman, kita bisa pakai satu atau gabungan dari beberapa paradigma

Apa itu OOP?

- OOP adalah teknik/paradigma pemrograman yang menggunakan **object**.
- Objek dapat dianggap sebagai model dari suatu entitas dalam dunia nyata
- Object-object dapat saling berinteraksi sehingga membentuk sebuah program.



Contohnya:



Dalam sistem e-commerce, kita bisa membuat object-object untuk memodelkan beberapa object di dunia nyata seperti produk, keranjang, wallet, voucher, dll.

OOP

- Pendekatan berorientasikan object untuk menyelesaikan masalah
- Data dan fungsi dibungkus dalam object
- OOP cocok dipakai untuk project skala besar dan kompleks karena mudah dikelola

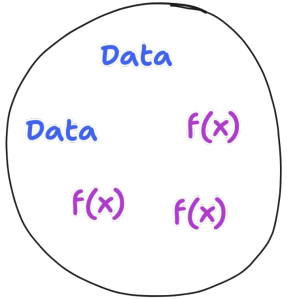
Prosedural

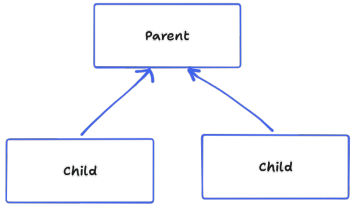
- Pendekatan menggunakan pendekatan langkah demi langkah atau prosedural
- Kode diorganisir secara sequence dengan serangkaian prosedur/fungsi
- Prosedural kurang bagus untuk untuk proyek skala besar

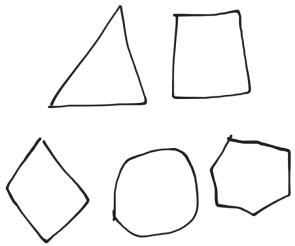
Dalam sebuah project, kadang beberapa paradigma digunakan bersamaan. Misalnya OOP, prosedural, event-driven, functional.

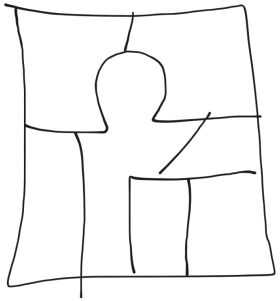
4 Pilar Konsep dalam OOP

1. **Enkapsulasi** (Class dan object) -> Pembungkusan fungsi dan data dalam class/object;
2. **Inheritance** -> pewarisan class
3. **Polimorfisme** -> method yang sama tapi beda bentuk/isi
4. **Abstraction** -> menyembunyikan detail implementasi/proses









1. Enkapsulasi

- Enkapsulasi -> pembungkusan data dan fungsi dalam satu unit class/object.

```
const iPhone = {
  name: "iPhone 11 Pro",
  battery: 93,
  charging: _ => this.battery++,
  discharging: _ => this.battery--,
  getCurrentBattery: _ => this.battery
}
```

```
class Phone {
  constructor(name, battery){
    this._name = name;
    this._battery = battery;
  }

  getCurrentBattery(){
    return this._battery;
  }

  charging(){
    this._battery++;
  }

  discharging(){
    this._battery--;
  }
}
```

2. Inheritance

- Inheritance -> Pewarisan class. Class bisa mewarisi properti dan method dari induknya.

Sub Class

```
class Tablet extends Phone {
  constructor(wifiOn){
    this._wifiOn = wifiOn
  }
}
```

Super Class

```
class Phone {
  constructor(name, battery){
    this._name = name;
    this._battery = battery;
  }

  getCurrentBattery(){
    return this._battery;
  }

  charging(){
    this._battery++;
  }

  discharging(){
    this._battery--;
  }
}
```

3. Polimorfisme

- Polimorfisme -> method yang sama, tapi isinya beda.

Pada contoh di samping, terdapat tiga class. Masing-masing class punya method speak(), tetapi isi methodnya berbeda-beda.

JavaScript

```
class Animal {
  speak() {
    return 'Some generic sound';
  }
}

class Cat extends Animal {
  speak() {
    return 'Meow';
  }
}

class Dog extends Animal {
  speak() {
    return 'Woof';
  }
}
```

4. Abstraksi

- Abstraksi -> menyembunyikan detail implementasi/proses

Fungsi _calculateWDfee() tidak bisa diakses dari luar class, cuma bisa dipakai di dalam class karena di set private.

```
class Wallet {
  constructor(balance){
    this._balance = balance;
  }

  _calculateWDfee(trx){
    return trx * 0.1;
  }

  wd(amount){
    this._balance -= amount + this._calculateWDfee();
  }
}
```

OOP di Javascript

Object Prototype

- Pembuatan class di Javascript awalnya menggunakan fungsi dengan object ***prototype***.
- Object instance dibuat dengan kata kunci ***new***.
- Kita bisa buat lebih dari satu object dengan class yang sama.

```
// membuat class
function Phone(name, sim1, sim2){
  this.name = name;
  this.sim1 = sim1;
  this.sim2 = sim2;
}

// tambah method ke class Phone
Phone.prototype.setSim1 = function(newSim){
  this.sim1 = newSim;
}

// membuat object instance
const iPhone = new Phone("iPhone", "XL", "Telkomsel");
```

Class (ES6)

- Class baru ditambahkan pada Javascript versi ES6
- Properti dalam class bisa diinisialisasi di dalam fungsi `constructor()`
- Fungsi `constructor()` adalah fungsi yang akan dieksekusi ketika object dibuat.
- Fungsi/method di dalam class dibuat tanpa menggunakan kata kunci ***function***.

```
class Phone {
  constructor(name, battery){
    this._name = name;
    this._battery = battery;
  }

  getCurrentBattery(){
    return this._battery;
  }

  charging(){
    this._battery++;
  }

  discharging(){
    this._battery--;
  }
}


const oppo = new Phone("Oppo Find X", 100);
```

Aturan nama class

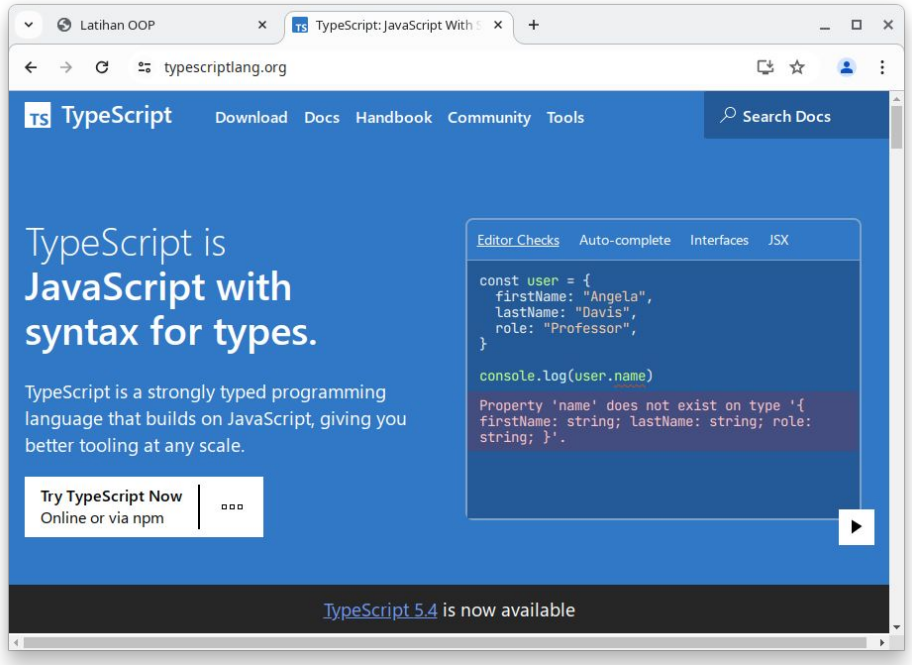
- Huruf pertama disarankan kapital untuk membedakannya dengan variabel
- Sisanya sama seperti variabel, tidak boleh pakai simbol operator dan angka di depan
- Aturan nama method sama seperti nama fungsi
- Untuk properti dan method private, biasanya pakai _ di depannya.

```
class Phone {
  constructor(name, battery){
    this._name = name;
    this._battery = battery;
  }

  getCurrentBattery(){
    return this._battery;
  }

  charging(){
    this._battery++;
  }

  discharging(){
    this._battery--;
  }
}


const oppo = new Phone("Oppo Find X", 100);
```

Di Javascript belum ada..

- Modifier untuk member class seperti ***private***, ***public***, ***protected***
- ***Interface*** dan ***class Abstract***
- Generic

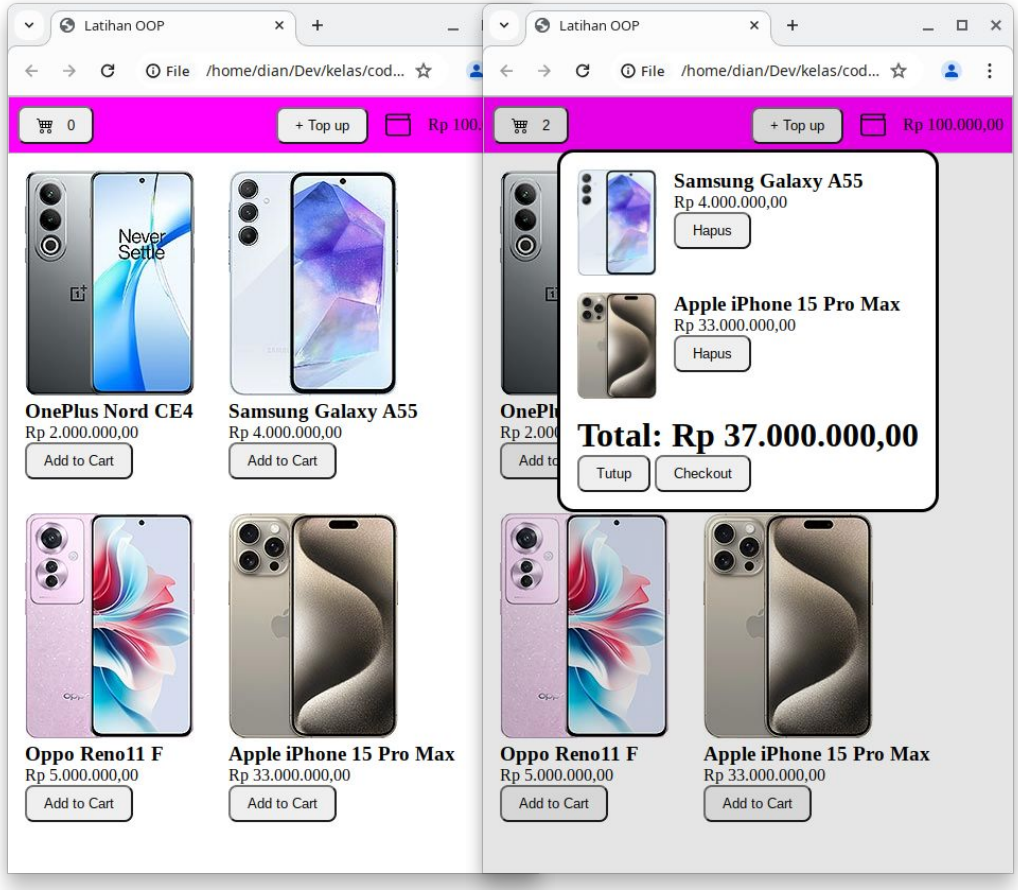
Tapi tenang saja, kita bisa pakai Typescript untuk memprogram secara OOP di tingkat lanjut



Latihan 💪

Latihan: Toko Hp

- Buat class Product, Cart, dan Wallet dan buat object instance-nya di main.js
- Struktur Folder:
  - index.html
  - style.css
  - main.js
  - Product.js
  - Cart.js
  - Wallet.js



Latihan: Toko Hp

- Buat class Product, Cart, dan Wallet dan buat object instance-nya di main.js
- Struktur Folder:
  - index.html
  - style.css
  - main.js
  - Product.js
  - Cart.js
  - Wallet.js

main.js

pertemuan-14 > _dian > main.js > ...

```
// coba buat object product dan cart
const keranjang = new Cart();

// buat object Wallet dengan saldo awal 10000
const goPay = new Wallet(10000);

// buat object product
const iPhone = new Product(
  "iPhone 15 Pro Max",
  33000000,
  "https://fdn2.gsmarena.com/vv/bigpic/apple-iphone-15-pro-max.jpg",
);
const samsung = new Product(
  "Samsung Galaxy A55",
  6000000,
  "https://fdn2.gsmarena.com/vv/bigpic/samsung-galaxy-a55.jpg",
);

// tambahkan iPhone ke keranjang
keranjang.addItem(iPhone);

// topup 5jt
goPay.topUp(5000000);

console.log(goPay.getCurrentBalanceIDR());
```

Selamat

Kita sudah selesai hari ini..

Resources & Referensi

- Icon: https://getbootstrap.com/docs/5.0/extend/icons/
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes