BSI BANK SYARIAH INDONESIA Rakamin

Javascript

OOP & DOM

[CONFIDENTIAL]

BSI BANK SYARIAH INDONESIA Rakamin

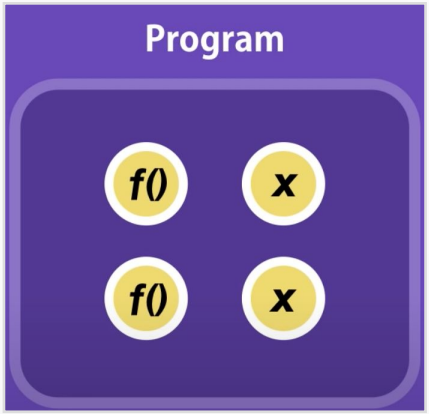
Javascript OOP & DOM

- [ ] OOP in Javascript
- [ ] DOM

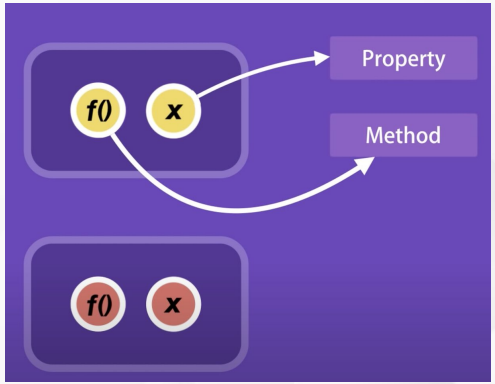
Object Oriented Programming

Object-oriented programming atau OOP adalah suatu metode pemrograman yang berorientasi pada objek.

Objek-objek yang saling berkaitan dan disusun kedalam satu kelompok ini disebut dengan class. Nantinya, objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit.



Procedural Programming



Object Oriented Programming

Sebuah class juga dapat memiliki variable

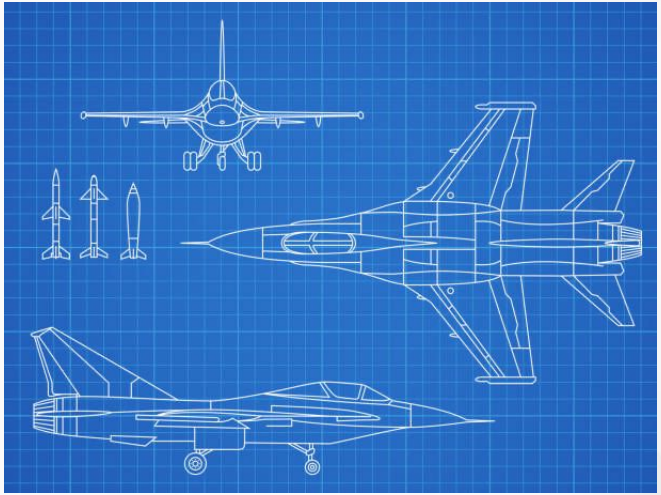
```
class User {
  constructor(name) {
    this.name = name;
  }
  sayHi() {
    console.log(this.name);
  }
}
// Penggunaan:
let user = new User("John");
user.sayHi();
```

Sebuah Class juga dapat memiliki variable

```
class User {
  constructor(name) {
    this.name = name;
  }
  sayHi() {
    console.log(this.name);
  }
}
// Penggunaan:
let user = new User("John");
user.sayHi();
```

Class

Class adalah sebuah prototype, atau blueprint, atau rancangan yang mendefinisikan variable dan method pada seluruh object tertentu



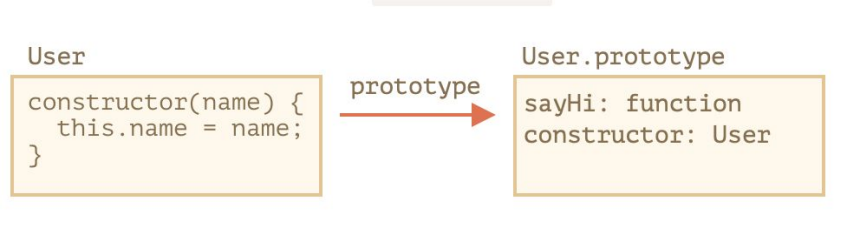
Implementasi Class

```
class User {
  constructor(name) {
    this.name = name;
  }
  sayHi() {

console.log(this.name);
  }
}
// Penggunaan:
let user = new
User("John");
user.sayHi();
```

Ketika new User(“John”) dipanggil:

1. Sebuah Object baru dibuat.
2. constructor secara otomatis dijalankan dengan argumen yang diberikan dan memberikan nilai tersebut ke dalam variable this.name.



Getter & Setter

```
class User {
  . . .
  get name() {
    return this._name;
  }
  . . .
}
```

Getter

```
class User {
  . . .
  set name(value) {
    if (value.length < 4) {
      console.log("Name is too short");
      return;
    }
    this._name = value;
  }  . . .
}
```

Setter

Class Variable

Sebuah class juga dapat memiliki variable

```
class User {
  name = "John";
  sayHi() {
     console.log(`Hello, ${this.name}!`);
  }
}
new User().sayHi(): // Hello, John!
```

Implementasi Class Inheritance

```
class Animal {
  constructor(name) {
    this.speed = 0;
    this.name = name;
  }
  run(speed){
    this.speed = speed;
    console.log(`${this.name} runs with speed ${this.speed}.`);
  }
  stop() {
    this.speed = 0;
    console.log(`${this.name} stands still.`);
  }
}
```

```
class Rabbit extends Animal {
  hide() {
    console.log(`${this.name} hides!`);
  }
}
let rabbit = new Rabbit("White Rabbit");
rabbit.run(5); // White Rabbit runs with speed 5.
rabbit.hide(); // White Rabbit hides!
```

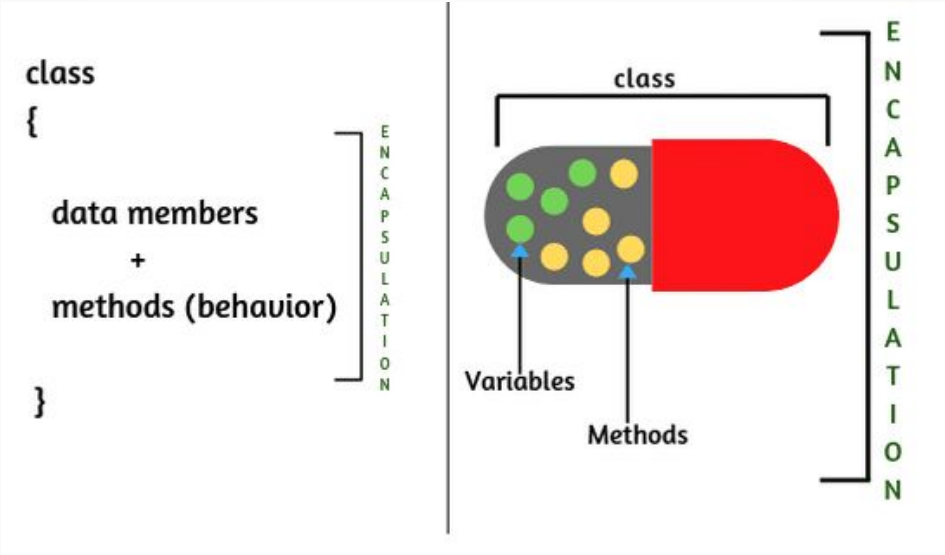
Method Override

```
class Animal {
 . . .
  stop() {
    this.speed = 0;
    console.log(`${this.name} stands still.`);
  }
}
```

```
class Rabbit extends Animal {
  stop() {
    // method ini yang akan digunakan
  }
}
```

Encapsulation

Encapsulation adalah mekanisme membungkus informasi sehingga dapat menyembunyikan informasi yang seharusnya disembunyikan atau tidak.

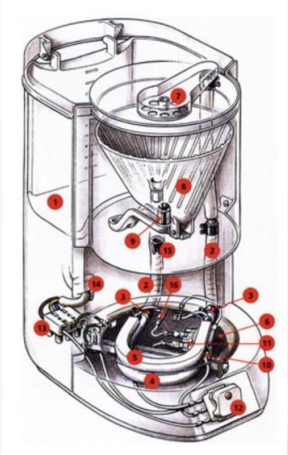


Encapsulation

Membedakan fungsi-fungsi internal dan eksternal.

Misalnya, pengguna tidak perlu tahu apa fungsi-fungsi yang terdapat dalam sebuah mesin kopi. Pengguna hanya perlu menekan tombol start dan kopi akan dibuat.

Jika semua fungsi di dalam mesin kopi dapat diakses oleh pengguna, maka bisa membahayakan pengguna dan mesin kopi itu sendiri



Private Method and Property

- **Internal Interface** — method dan property yang bisa diakses oleh method lain di dalam class, tapi tidak bisa diakses dari luar.
- **External Interface** — method dan property dapat diakses baik dari dalam, maupun dari luar class

Dalam Javascript, terdapat dua tipe object field:

Public: dapat diakses dari mana saja — merepresentasikan External Interface.

Private: hanya dapat diakses dari dalam class — merepresentasikan Internal Interface

Implementasi Encapsulation

```
class CoffeeMachine {
  _waterAmount = 0;
  set waterAmount(value) {
    if (value < 0) {
      value = 0;
    }
    this._waterAmount = value;
  }
  constructor(power) {
    this._power = power;
  }
}
```

```
// membuat mesin kopi
let coffeeMachine = new CoffeeMachine(100);
// menambahkan air
coffeeMachine.waterAmount = -10; // _waterAmount akan menjadi 0, bukan -10
```

Implementasi Encapsulation

Sebuah property yang private harus dimulai dengan #. Property ini hanya boleh diakses dari dalam class.

Sebagai contoh, berikut adalah variable private #waterLimit dan method private #fixWaterAmount:

```
class CoffeeMachine {
  #waterLimit = 200;
  #fixWaterAmount(value) {
    if(value < 0) return 0;
    if (value > this.#waterLimit) return this.#waterLimit;
  }
  setWaterAmount(value) {
    this.#waterLimit = this.#fixWaterAmount(value);
  }
}
let coffeeMachine = new CoffeeMachine();
// tidak bisa mengakses property private dari sebuah class
coffeeMachine.#fixWaterAmount(123); // Error
coffeeMaching.#waterLimit = 1000; // Error
```

DOM

DOM (Document Object Modelling) menyediakan sekumpulan fungsi dan attribute/data yang bisa kita manfaatkan dalam membuat program JavaScript. Fungsi itu dikenal dengan sebutan API (Application Programming Interface).

DOM

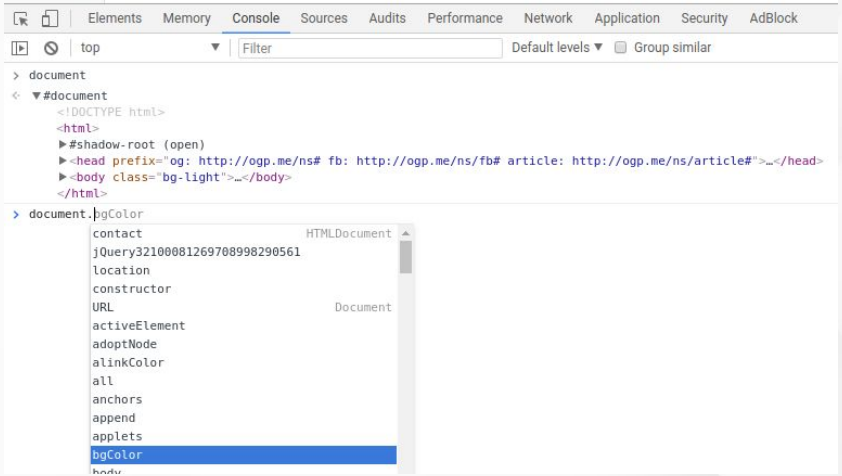
DOM (Document Object Modelling) menyediakan sekumpulan fungsi dan attribute/data yang bisa kita manfaatkan dalam membuat program JavaScript. Fungsi itu dikenal dengan sebutan API (Application Programming Interface).

DOM

Seperti yang sudah kita ketahui, DOM adalah sebuah object buat memodelkan dokumen HTML.

Object DOM di JavaScript bernama document. Object ini berisi segala hal yang kita butuhkan untuk memanipulasi HTML.

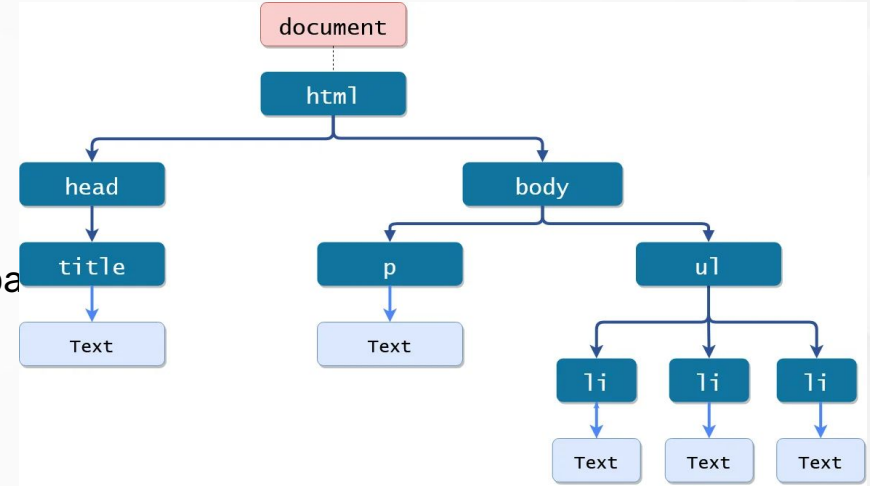
Jadi, jika kita coba ketik document pada console JavaScript seperti contoh di gambar, maka yang akan tampil adalah kode HTML.



DOM

Object document yang ada di DOM itu sebagai representasi model dari dokumen HTML.

Object ini berisi kumpulan fungsi dan atribut berupa object dari elemen HTML yang bisa digambarkan dalam bentuk pohon. Ilustrasinya seperti pada gambar.



Penggunaan DOM

Jika kita ingin mengakses elemen yang spesifik, terdapat beberapa fungsi yang sering digunakan:

Fungsi	Kegunaan
getElementById()	memilih *element* berdasarkan atribut *id*
getElementByName()	memilih *element* berdasarkan atribut *name*
getElementByClassName()	memilih *element* berdasarkan atribut *class*
getElementByTagName()	memilih *element* berdasarkan nama tag dan bisa mengembalikan nilai berupa *array*, mengingat *element* html dengan tag tertentu bisa jadi lebih dari satu
querySelector()	mencari *element* DOM pertama yang sesuai dengan aturan *selector* CSS yang diberikan ke fungsi
querySelectorAll()	sama seperti *querySelector*, tapi mengembalikan semua *element* yang memenuhi aturan (bukan hanya *element* pertama)

Contoh Mengakses Elemen

html Copy code

```html
<!DOCTYPE html>
<html>
<head>
  <title>Contoh DOM</title>
</head>
<body>

<h1 id="judul">Selamat datang!</h1>

<script>
  // Mengakses elemen berdasarkan id
  var judulElemen = document.getElementById("judul");

  // Mengubah teks pada elemen
  judulElemen.innerHTML = "Hello, DOM!";
</script>

</body>
</html>
```

- Menggunakan document.getElementById() untuk mendapatkan referensi ke elemen dengan id "judul".
- Menggunakan innerHTML untuk mengubah teks pada elemen.

Contoh Menambah dan Menghapus Elemen

html Copy code

```html
<!DOCTYPE html>
<html>
<head>
  <title>Contoh DOM</title>
</head>
<body>

<ul id="daftarBuah">
  <li>Apel</li>
  <li>Pisang</li>
</ul>

<button onclick="tambahBuah()">Tambah Buah</button>
<button onclick="hapusBuah()">Hapus Buah</button>
```

```
<script>
  function tambahBuah() {
    // Membuat elemen baru
    var liBaru = document.createElement("li");
    liBaru.textContent = "Jeruk";

    // Menambahkan elemen baru ke dalam daftarBuah
    document.getElementById("daftarBuah").appendChild(liBaru);
  }

  function hapusBuah() {
    // Menghapus elemen terakhir dari daftarBuah
    var daftarBuah = document.getElementById("daftarBuah");
    daftarBuah.removeChild(daftarBuah.lastElementChild);
  }
</script>

</body>
</html>
```

- Membuat fungsi tambahBuah() untuk membuat elemen baru (<li>) dan menambahkannya ke dalam daftar.
- Membuat fungsi hapusBuah() untuk menghapus elemen terakhir dari daftar.

Copyright Rakamin

BSI BANK SYARIAH INDONESIA Rakamin

Thank You

ODP BY LEAD IT BATCH 20 & 21

[CONFIDENTIAL]