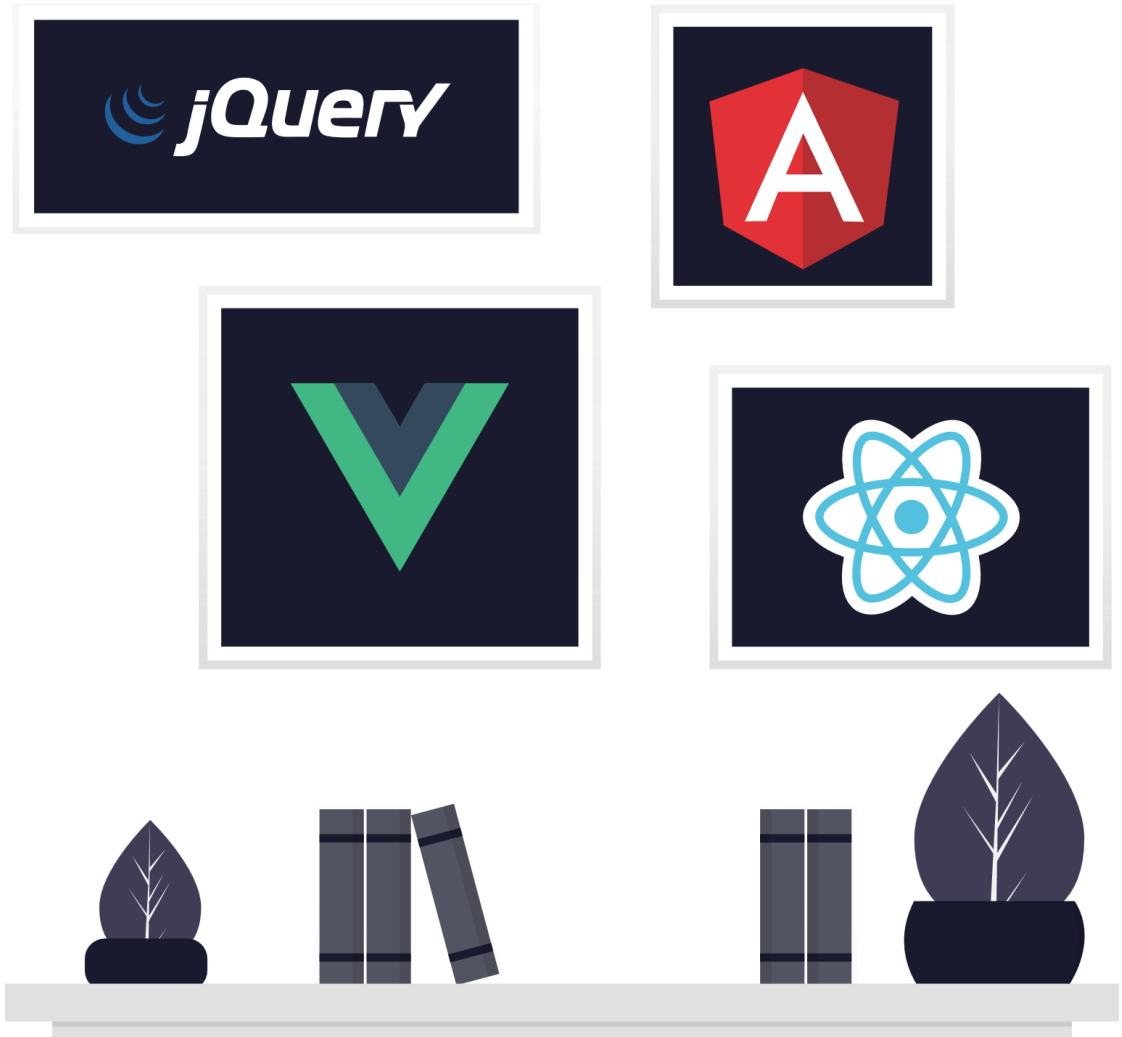
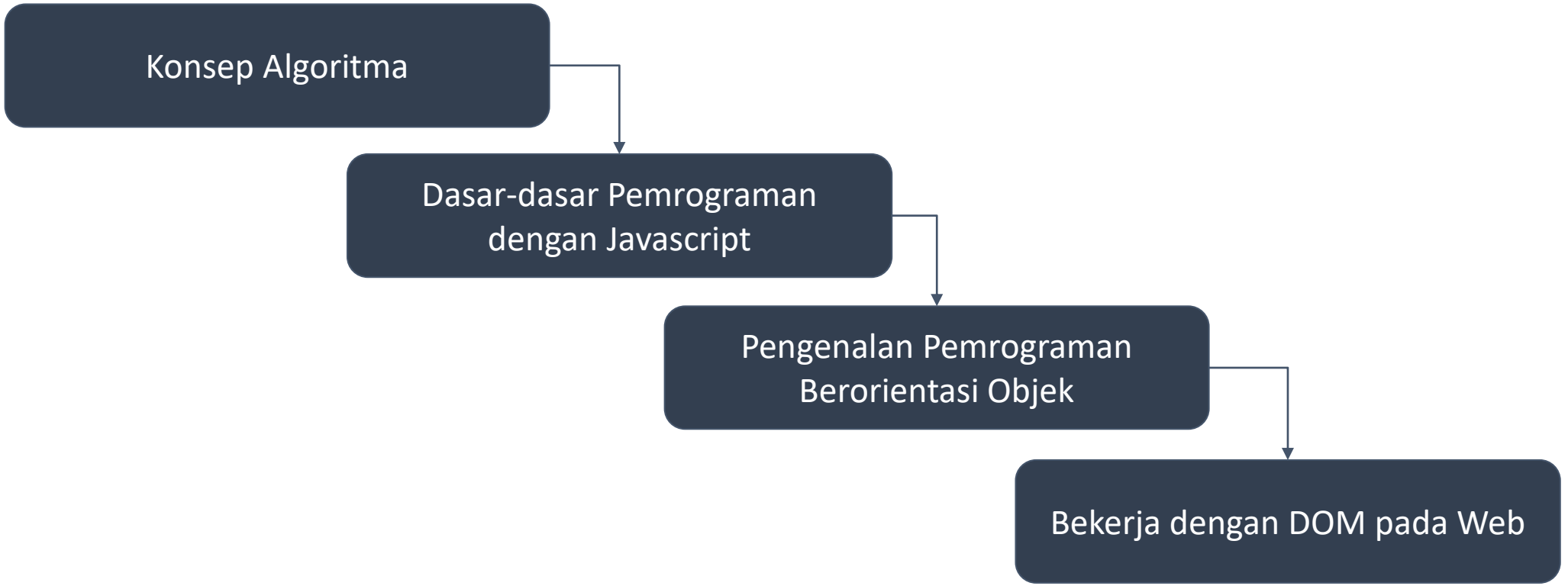
PEMROGRAMAN JAVASCRIPT



- **KODE** : MK03
- **MATAKULIAH** : PEMROGRAMAN JAVASCRIPT
- **SKS** : 4 SKS
- **SEMESTER** : 1 (SATU)


Konsep Algoritma
Dasar-dasar Pemrograman dengan Javascript
Pengenalan Pemrograman Berorientasi Objek
Bekerja dengan DOM pada Web

RENUNGAN ISLAMI

"Barangsiapa yang tidak duduk bersama orang yang berjaya, mana mungkin dia akan berjaya.“

- Habib Umar bin Hafidz -

OBJECT ORIENTED PROGRAMMING

Object Oriented Programming

OOP atau Object Oriented Programming merupakan salah satu pattern pada programming yang sangat umum digunakan oleh developer di dunia. Kalau berbicara mengenai OOP pasti kaitannya sangat erat dengan bahasa pemograman Java. Ya secara bahasa Java itu adalah bahasa yang pure mengusung pattern OOP.

KONSEP OOP

JavaScript sebagai bahasa yang terpopuler didunia saat ini tentu saja tidak mau kalah. Karena sekarang pada Javascript kita juga bisa menerapkan pattern OOP.

Paradigma OOP

Paradigma OOP berdasarkan pada konsep objek yang memiliki atribut serta dapat melakukan operasi atau prosedur tertentu.

Contoh : Objek Kucing

- Memiliki atribut : Warna, Ras, Panjang ekor, dll.
- Memiliki fungsi/prosedur : Berjalan, Bersuara, Makan.

Class JavaScript

Class adalah definisi dari objek yang nantinya akan dibentuk

Analoginya sama seperti blueprint sebuah gedung.

Class = blueprint/rancangan gedung

Instance/Objek = gedung

Deklarasi Class Kucing

```
class Kucing {
        constructor() {
                this.warna = 'Hitam';
                this.ras = 'Domestik'
        }

        makan( food ) {
                console.log(`Kucingnya hari ini makan`, food);
        };
}
```

CLASS JAVASCRIPT

Membuat Instansi/Objek dari Class

Membuat instance dari class menggunakan keyword new

```
Manis = new Kucing();
```

Setelah dibuat, fungsi dan atribut dari objek tersebut dapat dipanggil

- `Manis.makan("ikan");`
- `Manis.warna = "Putih";`

CLASS JAVASCRIPT

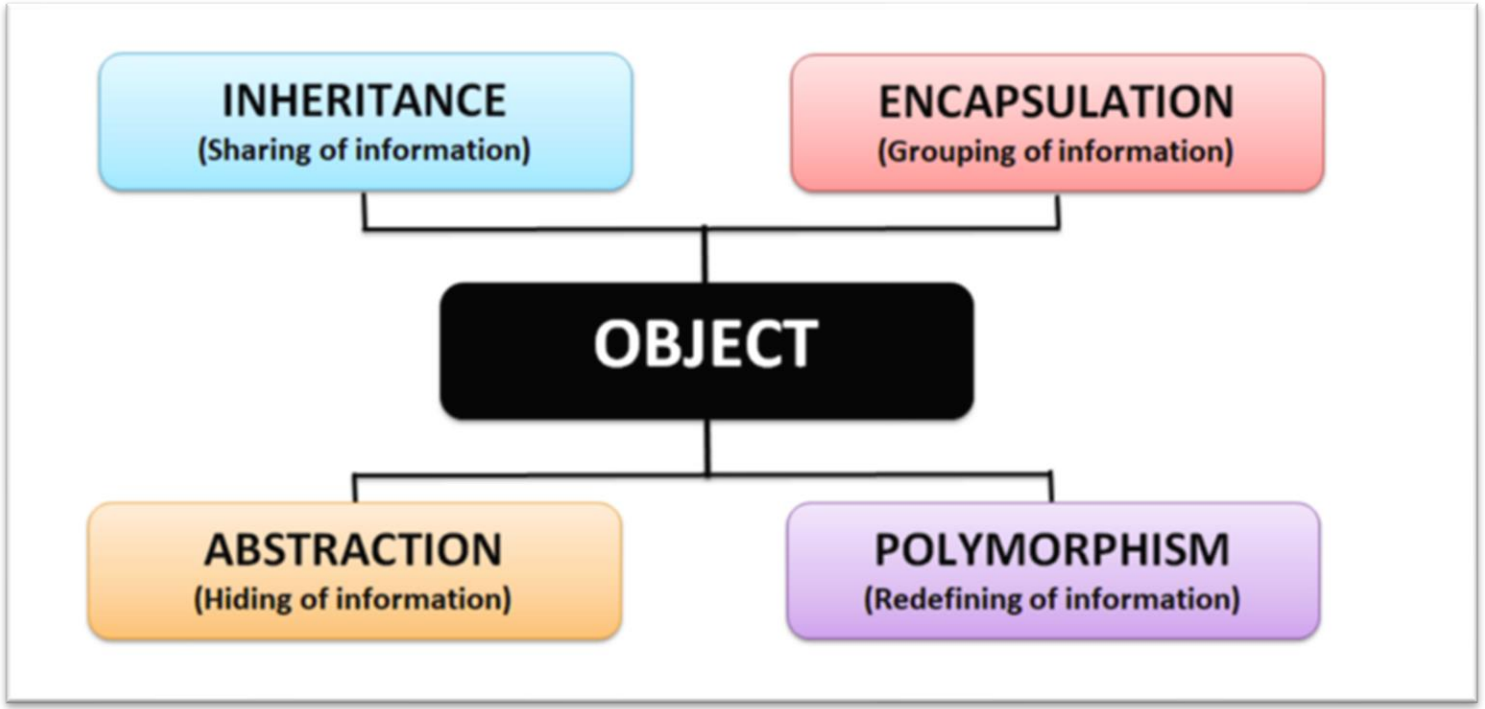
```
> class Kucing {
      constructor() {
              this.warna = 'Hitam';
          this.ras = 'Domestik'
      }

      makan( food ) {
              console.log(`Kucingnya makan`, food);
      };
  }
<· undefined
> Manis = new Kucing();
<· ▶ Kucing {warna: "Hitam", ras: "Domestik"}
>
```

PILAR OOP

4 Pilar OOP :

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism



Encapsulation

Encapsulation atau enkapsulasi adalah cara untuk membatasi akses langsung ke properti atau method dari sebuah objek. Enkapsulasi mencakup gagasan bahwa data(property atau method) suatu objek tidak boleh langsung diekspos.

Inheritance

Inheritance dalam OOP adalah sebuah proses dimana sebuah class mewariskan property dan methodnya ke class lain atau childnya.

Polymorhpism

Dari asal katanya, poly berarti banyak dan morphe berarti bentuk.

Dengan kata lain polymorphism berarti banyak bentuk.

Polymorphism adalah kemampuan untuk membuat variabel, fungsi, atau objek yang memiliki banyak bentuk.

Abstraction

Abstraction adalah sebuah teknik untuk menyembunyikan detail tertentu dari sebuah objek dan hanya menampilkan fungsionalitas atau fitur penting dari objek tersebut.

LEKSIKAL THIS

Objek biasanya dibuat untuk merepresentasikan benda yang ada di dunia nyata, seperti user, pesanan, dll. Objek tersebut bisa memanggil atribut pada dirinya sendiri, contoh :

```
let user = {
        name: "John",
        age: 30,
        sayHi() { alert( `Hi, I am ${user.name} ); }
};
```

LEKSIKAL THIS

Tapi cara tersebut tidak reliabel karena variabel user yang digunakan bisa hilang. Alternatifnya, agar objeknya menjalankan fungsi dan memanggil atribut dari objek tersebut, digunakan leksikal this.

```
let user = {
        name: "John",
        age: 30,
        sayHi() { alert(this.name); }
};
```

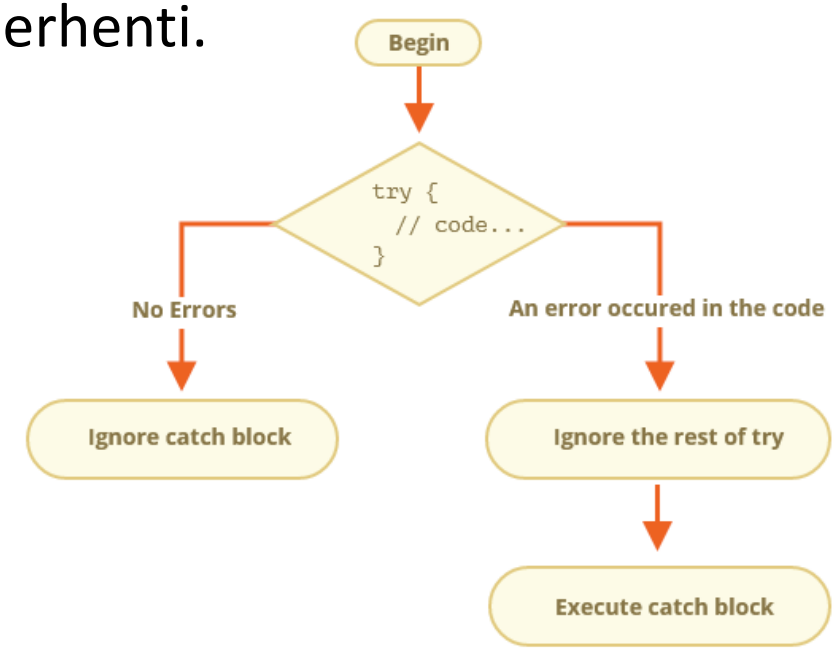
ERROR HANDLING

Error bisa terjadi karena banyak hal : kesalahan kita (programmer), user input yang tidak terduga, server respon yang tidak semestinya, dan alasan lainnya. Seberapa hebatpun programmer, tidak akan luput dari kesalahan. Yang penting, bagaimana kesalahan ini tidak mengganggu jalannya program yang dibuat. Biasanya jika ada error, maka script akan berhenti dan mengeluarkan pesan di console. Dengan penanganan yang tepat, meski error terjadi, program tetap bisa berjalan.

Try Catch

- Sintaks try… catch membantu kita menangkap error (jika terjadi) sehingga scriptnya dapat melakukan hal lain alih-alih berhenti.
- try {

```
    // code...
} catch (err) {
    // error handling
}
```



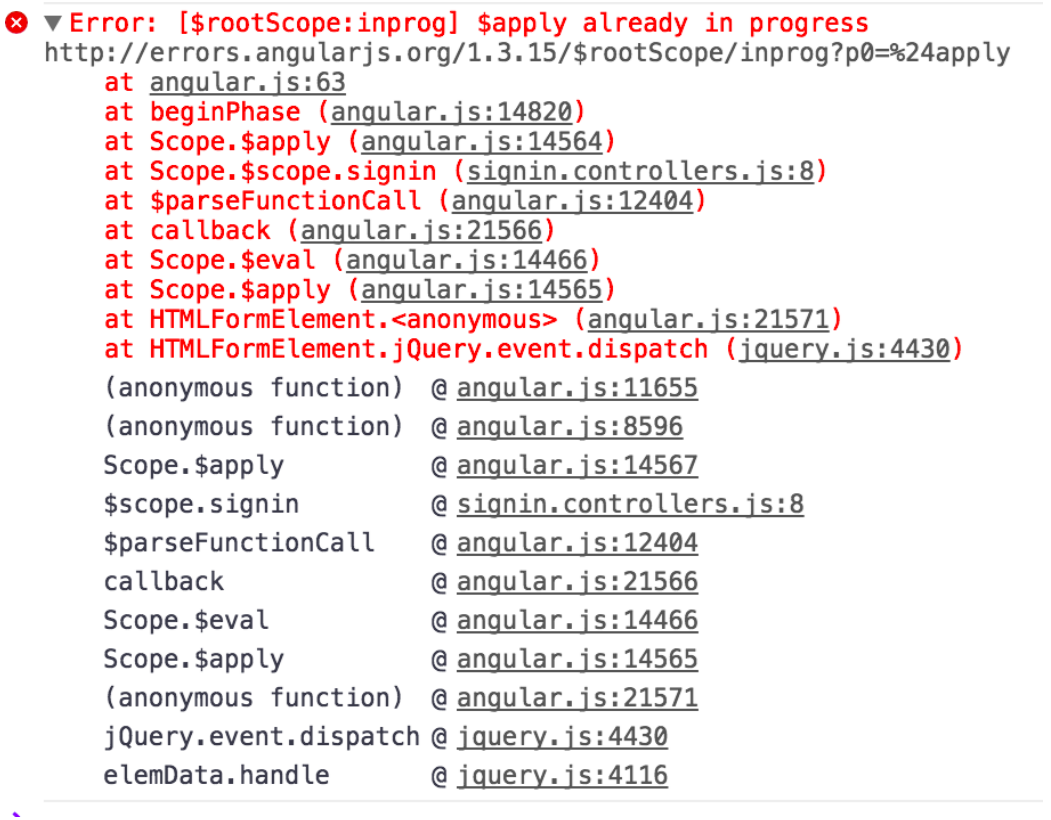
Try Catch

- Saat terjadi error, Javascript membuat sebuah objek yang menyimpan informasi detail terkait error tersebut. Objek tersebut dimasukkan ke dalam argumen fungsi catch()
- catch (err)
- Variabelnya dapat berupa apapun (tidak harus err seperti di contoh)

ERROR HANDLING

Stack

Stack yang muncul pada error adalah informasi tentang rangkaian pemanggilan fungsi yang menyebabkan error terjadi. Dapat digunakan untuk tujuan debugging.



ERROR HANDLING

Jenis-jenis Error

- **EvalError** - Error yang muncul terkait dengan fungsi eval()
- **RangeError** - Error saat variabel numerik atau parameternya ada di luar range yang valid
- **ReferenceError** - Error yang muncul saat men-dereference suatu referensi yang invalid
- **SyntaxError** - Error karena suatu kesalahan penulisan syntax
- **TypeError** - Error yang muncul saat variabel atau parameter tipenya tidak sesuai
- **URIError** - Error karena parameter yang diberikan ke fungsi encodeURI() atau decodeURI() invalid
- **AggregateError** - Beberapa error yang dikemas dalam sebuah error saat dilaporkan oleh operasi yang berjalan
- **InternalError** - Error yang muncul karena kesalahan internal javascript enginenya

TUGAS JAVASCRIPT

1. Buatlah class hewan yang menyimpan nama, jumlah kaki dan dapat berjalan
2. Buatlah object dari class hewan.
3. Simpan dalam file index.js
4. Upload pada akun github kalian dengan nama repository “Pertemuan-9-JS”

Terima Kasih
Syukron Katsiron
Arigatou Gozaimasu