

Coding Subuh #17

Error Handling

Ahmad Muhardian

Agenda kita..

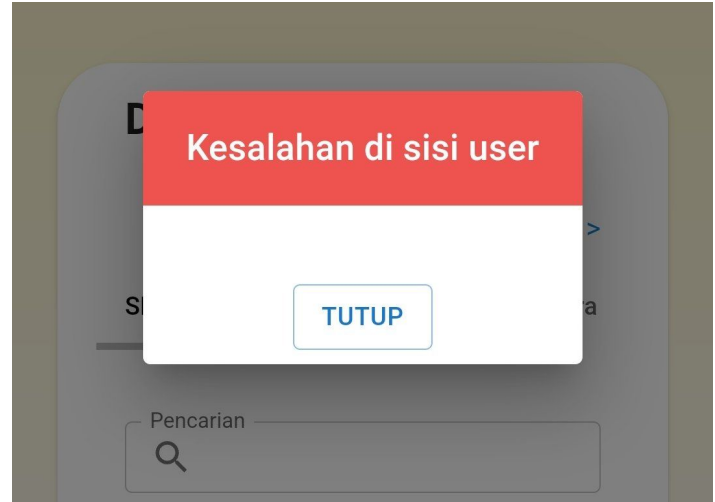
- Error handling?
- Blok try...catch
- Custom Error
- Latihan

Error Handling?

Apa itu Error Handling?

- **Error handling** adalah proses yang digunakan untuk menangani kesalahan atau masalah yang muncul saat kode program dieksekusi.
- Dengan error handling, kita bisa mengantisipasi hal yang tidak terduga (*exception*) terjadi saat program dijalankan di sisi user

Error yang terjadi saat coding adalah tanggung jawab programmer untuk memperbaikinya, tetapi error yang terjadi di sisi user adalah ...



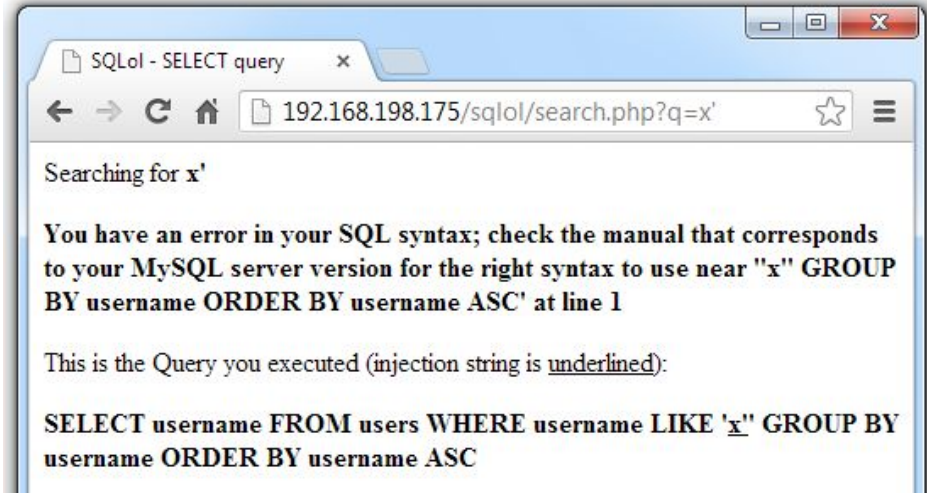
Mengapa kita butuh Error Handling?

- Buat mencegah aplikasi crash/force close.
- Memberikan feedback ke user kalau ada error.
- **Keamanan:** biar pesan error gak dibaca Hacker 🤔

Unfortunately, Force Closes
has stopped.

Report

OK



Error Handling di JavaScript


Block **try..catch**

- Kita bisa handle error di JS dengan blok **try..catch**.
- Blok **try** berisi kode program yang akan dites. Jika ada error di dalam blok **try**, maka kode yang ada di dalam blok **catch** akan dieksekusi.
- Blok catch punya parameter **error** yang berisi informasi dari error yang terjadi

```
try {  
    // kode program yang  
    // akan dites  
} catch (error){  
    // kode di blok ini akan  
    // dieksekusi kalau ada error  
}
```

Contoh: try...catch

Kode ini akan error, karena variabel firstName dan lastName belum dibuat



```
try {  
  let fullName = firstName + " " + lastName;  
} catch (error){  
  console.log(error.message);  
  alert("Ups! terjadi kendala teknis");  
}
```


Menggunakan **finally**

- Blok **finally** adalah blok yang akan selalu dieksekusi dalam kondisi error maupun tidak.
- Biasanya blok finally dipakai untuk menjalankan kode tertentu seperti:
 - Menutup koneksi ke database
 - Mencatat log yang terjadi
 - Menghapus cache
 - Dan sebagainya..

```
try {  
    // kode program yang  
    // akan dites  
} catch (error){  
    // kode di blok ini akan  
    // dieksekusi kalau ada error  
} finally {  
    // kode ini akan selalu  
    // dikerjakan walaupun error  
    // dan tidak error  
}
```

Contoh: try...catch...finally

Apapun yang terjadi blok kode di finally akan tetap dieksekusi

```
try {  
    authenticate(username, password);  
} catch (error){  
    alert("Ups, terjadi kesalahan");  
} finally {  
    console.log("LOG: user login attempt");  
}
```

Melempar Error?

Membuat Custom Error dengan throw

Pesan error

- Kita bisa membuat custom error dengan melempar object error
- Gunakan kata kunci **throw** diikuti dengan object error-nya

```
function bagi(a,b){  
  if(b === 0){  
    throw new Error("AAAAAA Error, gak bisa bagi nol!");  
  }  
  
  return a / b;  
}
```

```
try{  
  bagi(3,0);  
} catch (error){  
  console.log(error.message);  
}
```

Custom Error dengan Turunan Class

- Kita bisa menambahkan properti untuk object error dengan membuat class turunannya

```
class MyError extends Error {  
    constructor(message) {  
        super(message);  
        this.name = "MyError";  
    }  
}
```

Penggunaan class MyError()
untuk membuat object error
dan melemparnya



```
throw new MyError("Ohh my error!");
```

Contoh: Custom Error dengan Turunan Class

```
class BalanceError extends Error {  
  constructor(message) {  
    super(message);  
    this.name = "BalanceError";  
  }  
}
```



```
class Wallet {  
  constructor(initialBalance){  
    this.balance = initialBalance;  
  }  
  
  topUp(amount){  
    if (amount < 0 || isNaN(amount)){  
      throw new BalanceError("Ups! jumlah saldo salah.");  
    }  
    this.balance += amount;  
  }  
  
  withdraw(amount){  
    if (amount < 0 || isNaN(amount) || amount > this.balance){  
      throw new BalanceError("Ups! jumlah saldo salah.");  
    }  
    this.balance -= amount;  
  }  
}
```

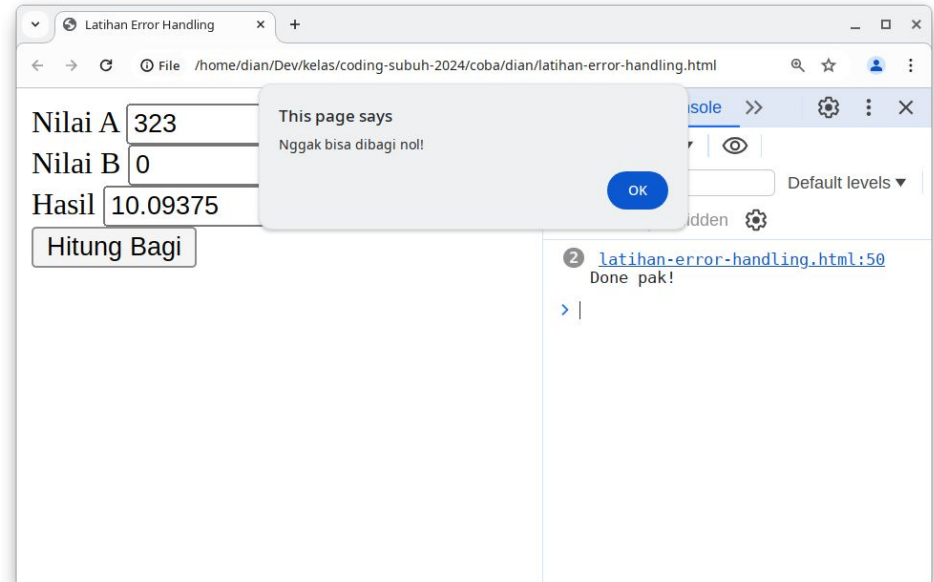
```
try {  
  const wallet = new Wallet(1000);  
  wallet.withdraw(9999999999);  
} catch (error) {  
  if(error instanceof BalanceError){  
    console.log("Hoooo ini error karena saldo salah!");  
    console.log("Error", error.message);  
  } else {  
    console.log("Error", error.message);  
  }  
}
```



Latihan 

Latihan: Error Handling

- Buat aplikasi pembagian nilai seperti gambar di samping
- Gunakan throw untuk melempar error
- Gunakan try..catch..finally



Latihan: Custom Error

- Buat custom error dengan class turunan dari class **Error()**
- Bisa coba contoh kasus aplikasi Wallet, topup dan withdraw.

Saldo:

Rp 10.000,00



Top up (+)



Withdraw (-)



Selamat

Kita sudah selesai hari ini..

Resources & Referensi

- [Control flow and error handling - JavaScript | MDN](#)