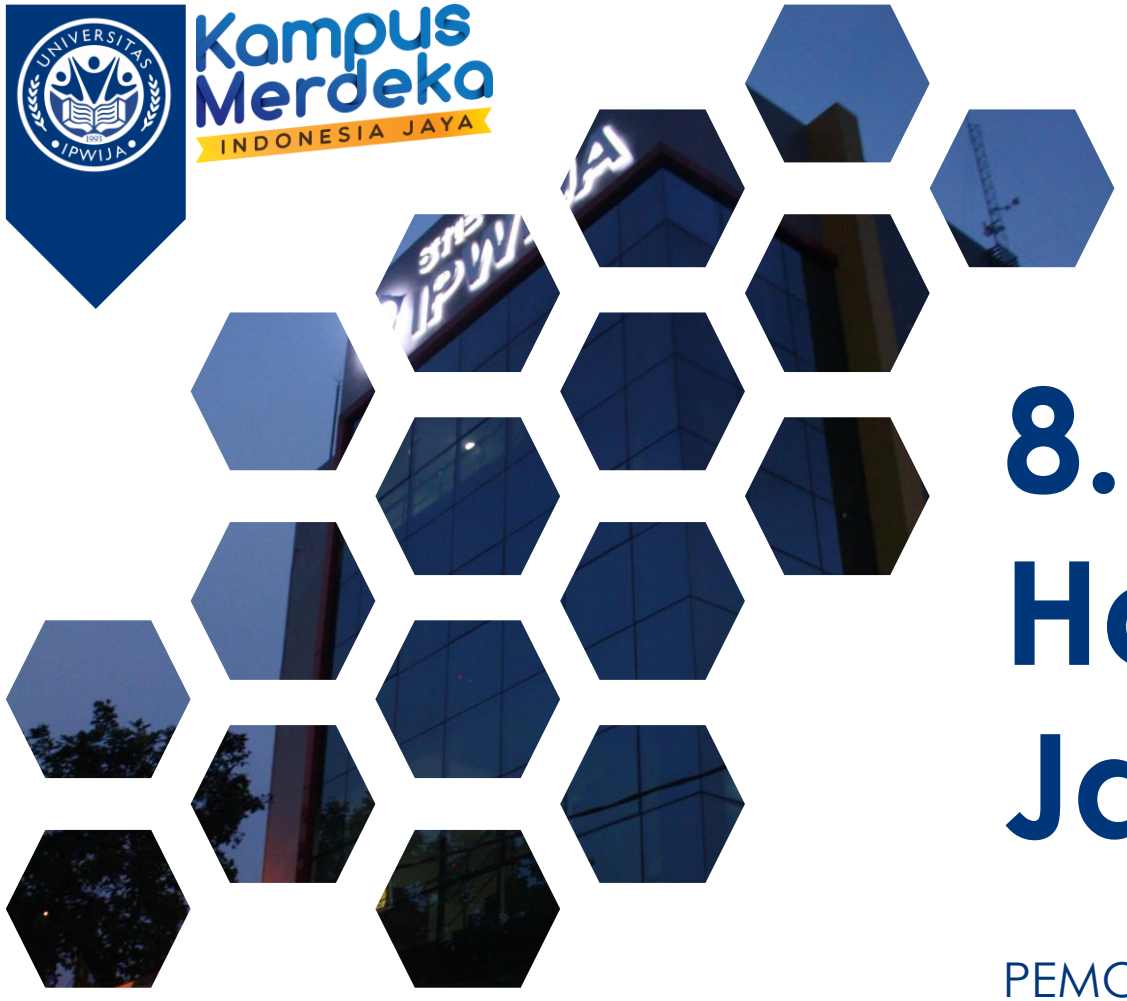


8. Exception Handling dalam JavaScript

PEMOGRAMAN BERORIENTASI OBJEK

1. Konsep Penanganan Kesalahan dalam Pemrograman

Kesalahan dalam pemrograman bisa dikategorikan menjadi beberapa jenis:

1. **Syntax Error** → Kesalahan dalam aturan penulisan kode.
2. **Runtime Error** → Kesalahan yang terjadi saat program berjalan.
3. **Logical Error** → Program berjalan, tapi hasilnya tidak sesuai harapan.
4. **Exception** → Kesalahan yang bisa kita tangani menggunakan mekanisme khusus.

1. console.log("Halo Mahasiswa" // SyntaxError: missing closing parenthesis
2. let hasil = 10 / 0; // Infinity (bukan error, tapi hasil yang tidak terduga)
3. let angka = "5" + 3; // Output: "53" (bukan 8, karena "5" dianggap string)
4. Try-Catch-Finally

2.1. Try-Catch-Finally

Dalam JavaScript, kita bisa menangani error menggunakan blok **try-catch-finally**, di mana:

try → Bagian kode yang dicoba dieksekusi.

catch → Bagian kode yang dijalankan jika ada error.

finally → Bagian kode yang tetap dieksekusi, baik terjadi error atau tidak.

Contoh penggunaan:

```
try {  let mahasiswa = ["Andi", "Budi", "Citra"];
    console.log(mahasiswa[5].toUpperCase());
} catch (error) {
    console.log("Terjadi error: " + error.message);
} finally {  console.log("Proses selesai.");}:
```

Output:

Terjadi error: Cannot read properties of undefined (reading 'toUpperCase')

Proses selesai.

2.2. Penggunaan throw

Mekanisme throw untuk membuat error sendiri

Selain menangani error, kita juga bisa membuat custom error dengan **throw**.

```
function cekUsia(usia) {
    if (usia < 18) {
        throw new Error("Maaf, Anda belum cukup umur!");
    }
    return "Selamat datang!";
}
```

```
try {
    console.log(cekUsia(16));
} catch (error) {
    console.log("Terjadi kesalahan: " + error.message);
}
```

Output:

Terjadi kesalahan: Maaf, Anda belum cukup umur!

3. Custom Error Handling dalam JavaScript

Terkadang, kita ingin membuat jenis error khusus agar lebih mudah ditangani. Kita bisa membuat kelas custom error menggunakan **extends Error**.

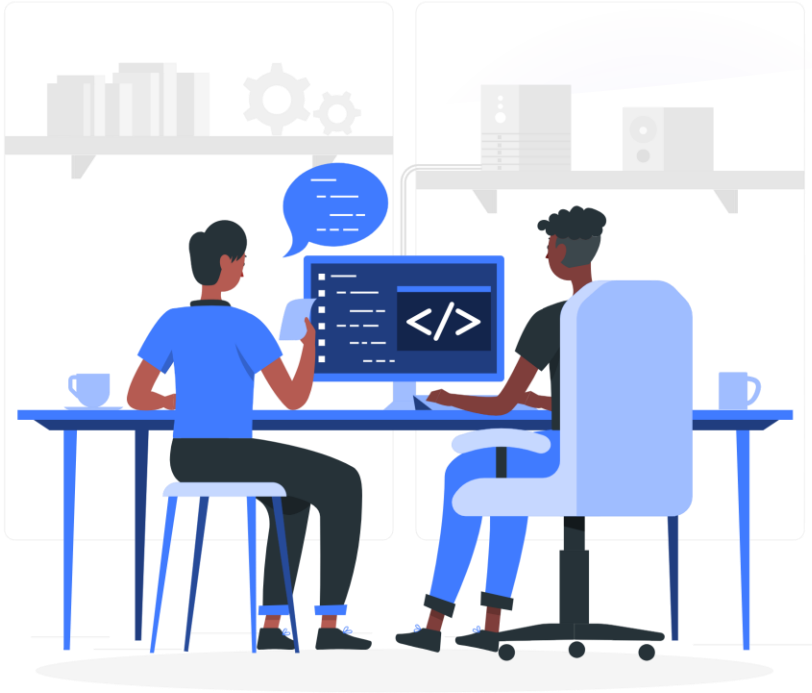
```
class ValidasiError extends Error {
    constructor(pesan) {  super(pesan);
        this.name = "ValidasiError";  }}
function cekNama(nama) {
    if (nama.length < 3) {
        throw new ValidasiError("Nama terlalu pendek!");  }
    return `Nama valid: ${nama}`;}
try {  console.log(cekNama("Jo"));}
catch (error) {
    if (error instanceof ValidasiError) {
        console.log("Terjadi kesalahan validasi: " + error.message);    }
else {  console.log("Kesalahan lain: " + error.message);  }
}
```

Output:

Terjadi kesalahan validasi: Nama terlalu pendek!

Penutup

- Exception Handling membantu program menangani error dengan lebih aman.
- Gunakan try-catch-finally untuk menangani error tanpa membuat program crash.
- Gunakan throw untuk membuat error sendiri, terutama dalam validasi input.
- Bisa membuat custom error dengan class extends Error.



Praktikum:

- Membuat halaman web dinamis dengan PHP