



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA



Project Based  
Learning



J.620100.017.02

# MENGIMPLEMENTASIKAN PEMROGRAMAN TERSTRUKTUR

Program Pelatihan :  
Pengembangan Web dengan Node.js dan React

Smart Creative Skills Team ©

# ABSENSI

# SAFETY INDUCTION

Membangun budaya keselamatan dan tanggung jawab digital.

Mencegah risiko kecelakaan, kerusakan alat, dan pelanggaran data.



**Keamanan Fisik  
Peralatan TIK**



**Keamanan Data  
dan Informasi**

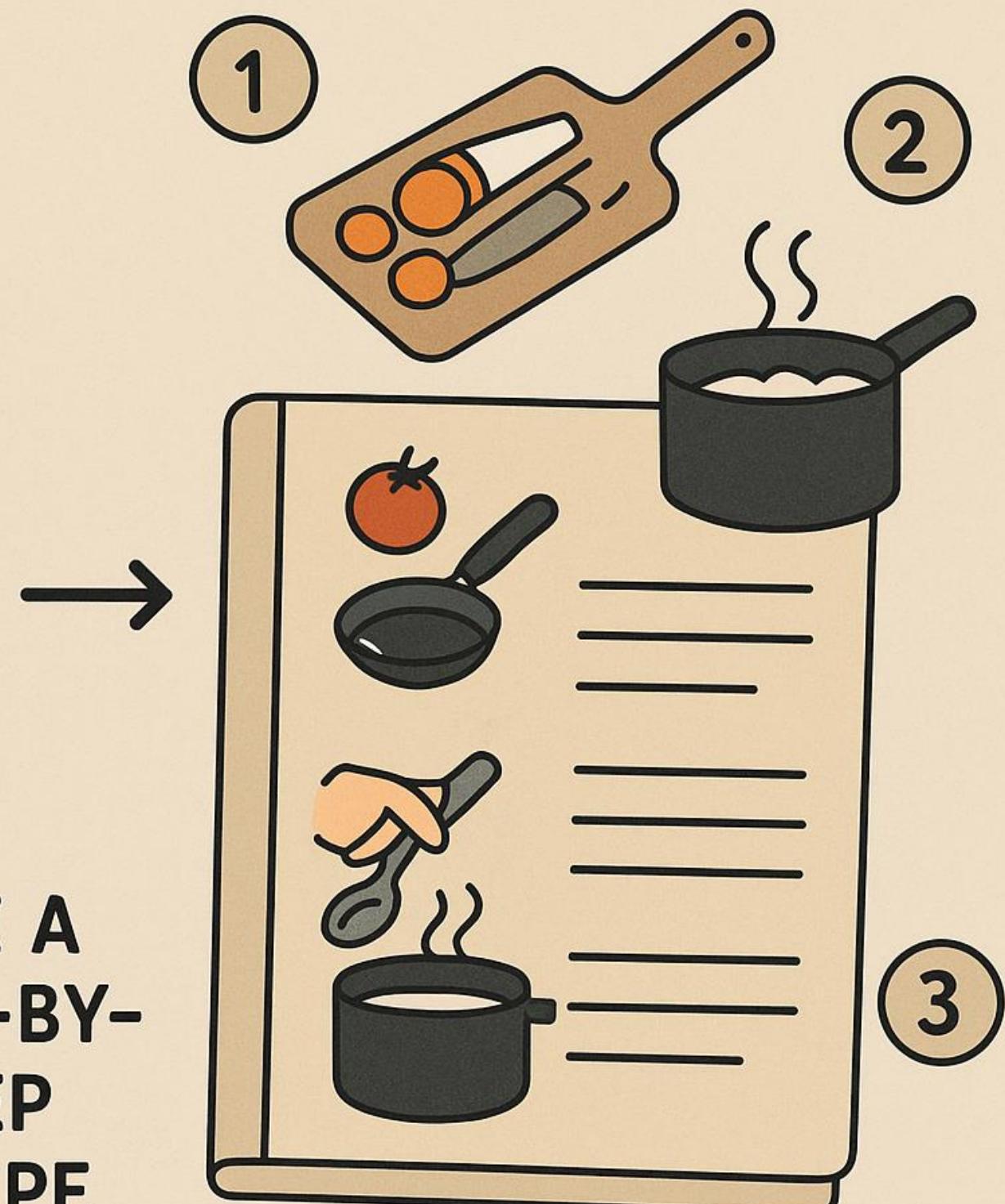


**Prosedur  
Darurat**

# STRUCTURED PROGRAMMING



LIKE A  
STEP-BY-  
STEP  
RECIPE

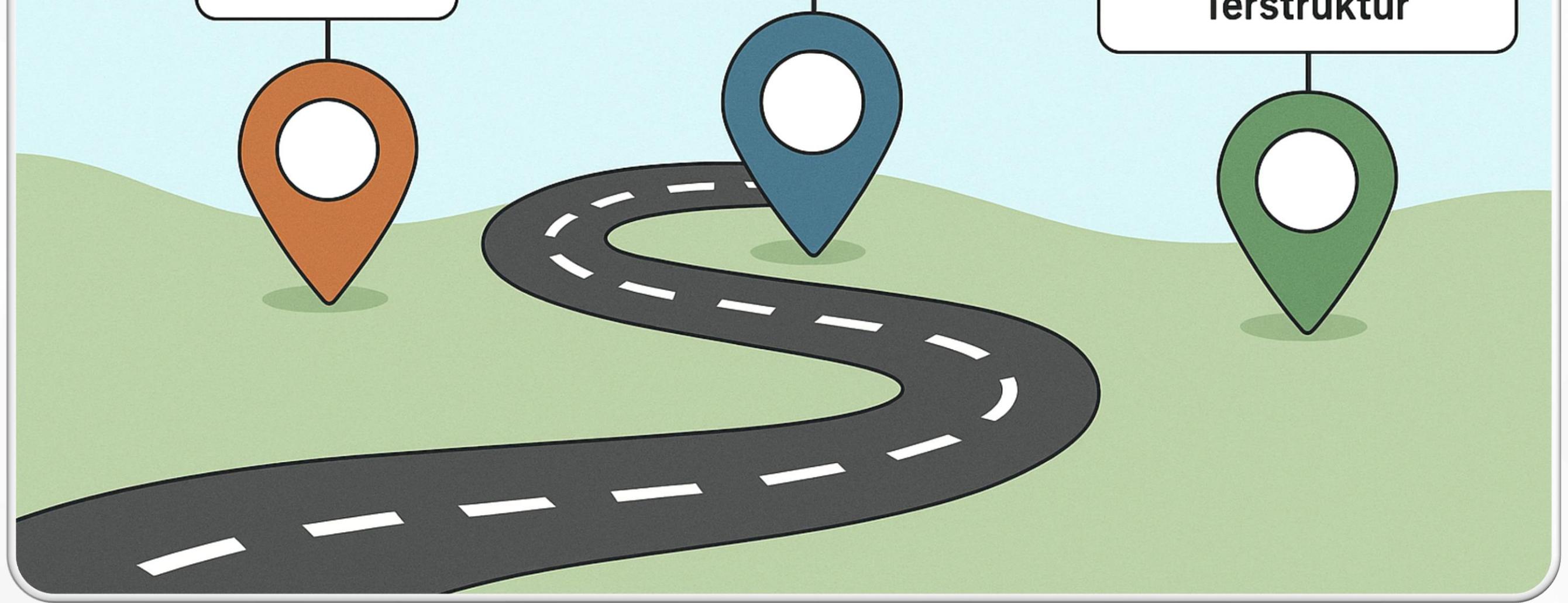


# TRAINING ROADMAP

**TNA**  
Training Needs  
Analysis

**Pengembangan Web  
dengan Node.js  
dan React**

**Mengimplementasikan  
Pemrograman  
Terstruktur**



# TUJUAN

Setelah mengikuti pelatihan ini, peserta pelatihan dapat mengimplementasikan pemrograman terstruktur dalam proyek aplikasi web sederhana menggunakan Node.js dan React tanpa kesalahan sintaks saat dijalankan.

# MANFAAT

Agar peserta terbiasa berpikir logis dan terstruktur, serta berpengalaman langsung menggunakan Node.js dan React untuk menghadapi kebutuhan dunia kerja/membuat proyek digital sendiri





KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

**EK 1**

# **Menggunakan Tipe Data dan Control Program**



Smart Creative Skills Team ©

# Tipe Data

Sekumpulan informasi yang memiliki nilai dan karakteristik tertentu

## Tipe Data Primitif

**Tipe data dasar**, nilainya **langsung disimpan**, dan **immutable** (tidak bisa diubah secara langsung)

## Tipe Data Non Primitif

Tipe data ini menyimpan **referensi** ke memori, dan nilainya bisa diubah (**mutable**)

# Tipe Data Primitif

Tipe	Penjelasan	Contoh Kode
String	Kumpulan karakter/text	<code>let nama = "Ali";</code>
Number	Angka bulat/desimal	<code>let umur = 20;</code> <code>let tinggi = 175.5;</code>
Boolean	Bernilai <code>true</code> atau <code>false</code>	<code>let aktif = true;</code>
Null	Nilai kosong yang sengaja dikosongkan	<code>let nilai = null;</code>
Undefined	Variabel yang belum diberi nilai	<code>let alamat;</code>
Symbol	Nilai unik, sering untuk properti objek privat	<code>let id = Symbol("id");</code>
BigInt	Untuk menyimpan angka besar melebihi Number biasa	<code>let angkaBesar =</code> <code>12345678901234567890n;</code>

## Contoh :

```

let nama = "Ali";           // String
let umur = 20;              // Number
let lulus = true;            // Boolean
let nilaiUjian = null;       // Null
let jurusan;                // Undefined
let kode = Symbol("kode");   // Symbol
let populasi = 987654321987654321n; // BigInt

```

# Tipe Data Non Primitif

Tipe	Penjelasan	Contoh Kode
Array	Kumpulan data berurutan	<pre>let buah = ["apel", "jeruk", "mangga"];</pre>
Object	Kumpulan pasangan key dan value	<pre>let siswa = { nama: "Ani", umur: 17 };</pre>
Function	Blok kode yang bisa dipanggil ulang	<pre>function sapa() { return "Halo";} }</pre>

## Contoh :

```
let buah = ["apel", "jeruk", "mangga"]; // Array  
  
let siswa = {  
    nama: "Ani",  
    umur: 17,  
    lulus: true  
};  
  
function sapa() {  
    return "Halo Dunia!";  
} // Function
```

# Variable

Suatu blok data untuk menampung sekumpulan data dengan berbagai tipe data apapun

**let**

**const**

**var**

# let

Untuk variable yang **nilainya bisa berubah**

```
let usia = 17;  
usia = 18;
```

# const

Untuk variable yang **nilainya tetap** (konstanta)

```
const pi = 3.14;  
  
// pi = 3.14159; ✗ Error!
```

# var

Untuk kode javascript lama, tidak disarankan digunakan

```
var nama = "Ali";
```

# Komentar

Digunakan untuk menjelaskan kode atau menonaktifkan baris kode sementara.

Komentar tidak dijalankan oleh Javascript.

Beberapa jenis komentar :

**Komentar Satu Baris**

**Komentar Banyak Baris**

# Komentar Satu Baris (//)

Digunakan untuk memberi catatan pendek di satu baris

```
// Ini komentar satu baris  
let nama = "Ali"; // Komentar setelah kode
```

# Komentar Banyak Baris (/\* ... \*/)

Digunakan untuk menjelaskan bagian kode panjang / memblokir beberapa baris.

```
/*
Ini adalah komentar
Lebih dari satu baris
Biasanya untuk penjelasan panjang
*/
let umur = 17;
```

# Operator

Simbol atau kata khusus untuk melakukan operasi pada nilai (data)

Beberapa jenis operator :

**Operator Aritmatika**

**Operator Assignment**

**Operator Perbandingan**

**Operator Kondisional**



# Operator Aritmatika

Operator yang melibatkan operasi matematika, diantaranya

Operator	Nama	Contoh	Hasil
+	Penjumlahan	5 + 3	8
-	Pengurangan	5 - 3	2
*	Perkalian	5 * 3	15
/	Pembagian	6 / 2	3
%	Modulus (sisa)	5 % 2	1
**	Pangkat	2 ** 3	8
++	Increment	let x=1; x++	2
--	Decrement	let x=2; x--	1

# Operator Aritmatika

Contoh :

```
1 const a = 20;  
2 const b = 3;  
3  
4 console.log("Penjumlahan:", a + b);  
5 console.log("Pengurangan:", a - b);  
6 console.log("Perkalian:", a * b);  
7 console.log("Pembagian:", a / b);  
8 console.log("Sisa Bagi:", a % b);  
9 console.log("Pangkat:", a ** b);
```

Output :

- \$ node aritmatika.js  
Penjumlahan: 23  
Pengurangan: 17  
Perkalian: 60  
Pembagian: 6.666666666666667  
Sisa Bagi: 2  
Pangkat: 8000

# Operator Assignment

Operator untuk menyimpan suatu nilai ke dalam suatu variable

Operator	Arti	Contoh	Sama dengan
=	Penugasan	x = 5	—
+=	Tambah lalu simpan	x += 2	x = x + 2
-=	Kurangi lalu simpan	x -= 2	x = x - 2
*=	Kali lalu simpan	x *= 3	x = x * 3
/=	Bagi lalu simpan	x /= 2	x = x / 2
%=	Modulus lalu simpan	x %= 2	x = x % 2

# Operator Assignment

Contoh :

```
1 let x = 10;
2
3 console.log("Nilai awal x:", x);
4
5 x += 5;
6 console.log("Setelah x += 5:", x);
7
8 x -= 3;
9 console.log("Setelah x -= 3:", x);
10
11 x *= 2;
12 console.log("Setelah x *= 2:", x);
13
14 x /= 4;
15 console.log("Setelah x /= 4:", x);
16
17 x %= 4;
18 console.log("Setelah x %= 4:", x);
19
20 x **= 3;
21 console.log("Setelah x **= 3:", x);
```

Output :

- \$ node assignment.js  
Nilai awal x: 10  
Setelah x += 5: 15  
Setelah x -= 3: 12  
Setelah x \*= 2: 24  
Setelah x /= 4: 6  
Setelah x %= 4: 2  
Setelah x \*\*= 3: 8

# Operator Perbandingan

Operator yang membandingkan suatu nilai dengan nilai yang lain. Hasilnya berupa boolean (true atau false).

Operator	Arti	Contoh
<code>==</code>	Sama nilai (tidak peduli tipe)	<code>5 == '5'</code> → <code>true</code>
<code>===</code>	Sama nilai dan tipe	<code>5 === '5'</code> → <code>false</code>
<code>!=</code>	Tidak sama (nilai)	<code>5 != '5'</code> → <code>false</code>
<code>!==</code>	Tidak sama (nilai/tipenya)	<code>5 !== '5'</code> → <code>true</code>
<code>&gt;</code>	Lebih besar	<code>7 &gt; 5</code> → <code>true</code>
<code>&lt;</code>	Lebih kecil	<code>7 &lt; 5</code> → <code>false</code>
<code>&gt;=</code>	Lebih besar atau sama	<code>5 &gt;= 5</code> → <code>true</code>
<code>&lt;=</code>	Lebih kecil atau sama	<code>4 &lt;= 5</code> → <code>true</code>

# Operator Perbandingan

Contoh :

```
1 const a = 10;
2 const b = '10';
3 const c = 5;
4
5 console.log("a == b:", a == b);
6 console.log("a === b:", a === b);
7
8 console.log("a != b:", a != b);
9 console.log("a !== b:", a !== b);
10
11 console.log("a > c:", a > c);
12 console.log("a < c:", a < c);
13 console.log("a >= 10:", a >= 10);
14 console.log("c <= 5:", c <= 5);
```

Output :

```
$ node perbandingan.js
a == b: true
a === b: false
a != b: false
a !== b: true
a > c: true
a < c: false
a >= 10: true
c <= 5: true
```

# Operator Kondisional / Logika

Operator yang digunakan untuk operasi kondisi (true/false)

Operator	Keterangan
&&	Dan
	Atau
!	Kebalikan

- && (AND) : hanya **true** jika **semua** kondisi benar
- || (OR) : cukup **satu** kondisi benar, maka hasil **true**
- ! (NOT) : membalikkan nilai **true jadi false**, dan sebaliknya

# Operator Kondisional / Logika

Contoh :

```
1 const nilai = 85;
2 const lulus = nilai >= 75;
3 const hadir = true;
4
5 // AND: harus dua-duanya benar
6 if (lulus && hadir) {
7   console.log("Siswa lulus dan hadir.");
8 }
9
10 // OR: salah satu cukup
11 if (lulus || hadir) {
12   console.log("Siswa dianggap aktif.");
13 }
14
15 // NOT: membalikkan kondisi
16 console.log("Tidak hadir:", !hadir);
```

Output :

- \$ node logika.js  
Siswa lulus dan hadir.  
Siswa dianggap aktif.  
Tidak hadir: **false**



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

EK 2

# **Membuat Program Sederhana (Kondisi/Conditional) pada Javascript**



Smart Creative Skills Team ©

# Kondisi

... Cara yang digunakan dalam program komputer untuk mengambil keputusan dari kemungkinan benar (true) atau salah (false) terhadap beberapa kondisi

Jenis – jenis kondisi :

**If**

**If Else**

**If Else If**

**Switch Case**

# If

## Syntax :

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

## Contoh :

```
1 Let hour = 14;  
2 Let greeting;  
3  
4 if (hour < 18) {  
5     greeting = "Good Day";  
6     console.log(greeting);  
7 }
```

## Output :

```
$ node if.js  
Good Day
```

# If - Else

## Syntax :

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

## Contoh :

```
1 let hour = 20;  
2 let greeting;  
3  
4 if (hour < 18) {  
5     greeting = "Good Day";  
6     console.log(greeting);  
7 } else {  
8     greeting = "Good Evening";  
9     console.log(greeting);  
10 }
```

## Output :

```
$ node if-else.js  
Good Evening
```

# If – Else - If

## Syntax :

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false and condition2 is false  
}
```

## Contoh :

```
1 Let hour = 9;  
2 Let greeting;  
3  
4 if (hour < 10) {  
5     greeting = "Good Morning";  
6     console.log(greeting);  
7 } else if (hour < 20) {  
8     greeting = "Good Day";  
9     console.log(greeting);  
10 } else {  
11     greeting = "Good Evening";  
12 }
```

## Output :

- \$ node if-else-if.js  
**Good Morning**

# Switch Case

## Syntax :

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

# Switch - Case

Contoh :

```
1 let day;
2 switch (new Date().getDay()) {
3     case 0:
4         day = "Sunday";
5         break;
6     case 1:
7         day = "Monday";
8         break;
9     case 2:
10        day = "Tuesday";
11        break;
12    case 3:
13        day = "Wednesday";
14        break;
15    case 4:
16        day = "Thursday";
17        break;
18    case 5:
19        day = "Friday";
20        break;
21    case 6:
22        day = "Saturday";
23        break;
24    default:
25        day = "Unknown Day";
26    }
27 console.log("Today is: " + day);
```

Output :

```
● $ node switch-case.js
Today is: Monday
```



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

EK 2

# **Membuat Program Sederhana (Perulangan/Looping) pada Javascript**



Smart Creative Skills Team ©

# Perulangan

Tindakan mengulang/merepetisi sebuah proses, dengan tujuan untuk mendapatkan deret hasil atau dengan tujuan mendapatkan hasil tertentu dengan repetisi.

Jenis – jenis perulangan :

**For Loop**

**While Loop**

# For Loop

## Syntax :

```
for([Inisialisasi]; [Kondisi]; [Incremental/Decremental]) {  
    [Proses] // Merupakan proses yang akan dijalankan dalam satu iterasi  
}
```

- **Inisialisasi** : nilai awal dari variable
- **Kondisi** : diperiksa tiap pengulangan, selama true, loop akan berjalan
- **Incremental/Decremental** : dieksekusi setiap akhir pengulangan

# For Loop

Contoh :

```
1  for (let i = 5; i >= 1; i--) {  
2    |   console.log("Hitung mundur : " + i);  
3 }
```

Output :

- \$ node for.js  
Hitung mundur : 5  
Hitung mundur : 4  
Hitung mundur : 3  
Hitung mundur : 2  
Hitung mundur : 1

# While Loop

## Syntax While :

while akan menjalankan kode selama kondisinya bernilai true

## Syntax Do - While :

do – while akan menjalankan kode 1 kali, lalu mengecek kondisinya

```
while (condition) {  
    // code block to be executed  
}
```

```
do {  
    // code block to be executed  
}  
while (condition);
```

# While Loop

## Contoh While :

```
1 let i = 1;  
2  
3 while (i <= 5) {  
4     console.log("Angka ke-" + i);  
5     i++;  
6 }
```

```
1 let i = 6;  
2  
3 while (i <= 5) {  
4     console.log("Angka ke-" + i);  
5     i++;  
6 }
```

## Output While :

- \$ node while.js  
Angka ke-1  
Angka ke-2  
Angka ke-3  
Angka ke-4  
Angka ke-5

- \$ node while.js

# Do - While

## Contoh Do - While :

```
1 let i = 1;  
2  
3 do {  
4     console.log("Angka ke-" + i);  
5     i++;  
6 } while (i <= 5);
```

```
1 let i = 6;  
2  
3 do {  
4     console.log("Angka ke-" + i);  
5     i++;  
6 } while (i <= 5);
```

## Output While :

- \$ node do-while.js  
Angka ke-1  
Angka ke-2  
Angka ke-3  
Angka ke-4  
Angka ke-5

- \$ node do-while.js  
Angka ke-6



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

EK 3

# **Membuat Program Menggunakan Prosedur dan Fungsi pada Javascript**



Smart Creative Skills Team ©

# Prosedur

Prosedur adalah fungsi **yang tidak mengembalikan nilai** (return), dan digunakan untuk **menjalankan suatu aksi langsung**, seperti mencetak ke layar atau menampilkan pesan

## Ciri – Ciri Prosedur :

Ciri - Ciri	Keterangan
Ditulis dengan function	Seperti fungsi biasa
Tidak menggunakan return	Fokusnya pada <b>aksi</b> , bukan nilai hasil
Sering pakai console.log()	Untuk menampilkan output langsung

# Prosedur

Contoh :

```
1 function tampilkanPesan() {  
2     console.log("Selamat datang di JavaScript!");  
3 }  
4  
5 tampilkanPesan();
```

Output :

- \$ node prosedur.js  
*Selamat datang di JavaScript!*

# Prosedur dengan Parameter

Contoh :

```
1 function sapa(nama) {  
2     console.log("Halo, " + nama + "!");  
3 }  
4  
5 sapa("Emily");  
6 sapa("Mark");
```

Output :

- \$ node prosedur-param.js  
Halo, Emily!  
Halo, Mark!

# Fungsi

... Fungsi adalah **blok kode yang bisa dipanggil ulang** untuk menjalankan tugas tertentu

Struktur Dasar Fungsi :

```
function namaFungsi(parameter1, parameter2) {  
    // kode yang dijalankan  
    return hasil;  
}
```

# Fungsi

Contoh :

```
1 function tampilanWaktu() {  
2     let jam = new Date().getHours();  
3     console.log("Sekarang jam: " + jam);  
4 }  
5 tampilanWaktu();
```

Output :

```
$ node fungsi.js  
Sekarang jam: 15
```

# Fungsi dengan Parameter

Contoh :

```
1 function hitungLuasPersegi(sisi) {  
2     return sisi * sisi;  
3 }  
4 let luas = hitungLuasPersegi(4);  
5 console.log("Luas: " + luas);
```

Output :

```
$ node fungsi-param.js  
Luas: 16
```

# Perbedaan Fungsi dan Prosedur

Aspek	Fungsi	Prosedur
return	Ada Menghitung, menghasilkan	Tidak Ada
Tujuan	Bisa disimpan dalam variabel	Menjalankan aksi
Hasil		Hanya efek langsung (output)



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

EK 4

# **Membuat Program Menggunakan Array pada Javascript**



Smart Creative Skills Team ©

# Array

Array adalah **struktur data** yang menyimpan **banyak nilai** dalam satu variable.

**Dimensi Array**

**Tipe Data Array**

**Panjang Array**

**Method Array**

# Array Satu Dimensi (1D)

Array sederhana yang menyimpan data  
**dalam satu baris**

**Contoh :**

```
1 Let angka = [10, 20, 30, 40];
2 console.log(angka[0]);
```

**Output :**

```
$ node array-1d.js
10
```



**Dimensi  
Array**

# Array Dua Dimensi (2D)

Array sederhana yang menyimpan data  
**dalam satu baris**

Contoh :

```
1 < let matriks = [
2   [1, 2, 3],
3   [4, 5, 6],
4 ];
5
6 console.log(matriks[0][1]);
7 console.log(matriks[1][2]);
```



## Dimensi Array

Output :

```
$ node array-2d.js
2
6
```

# Tipe Data Homogen (Isi Sama)

Array of Number

```
let angka = [10, 20, 30, 40];
```

Array of String

```
let nama = ["Ali", "Budi", "Cici"];
```

Array of Boolean

```
let status = [true, false, true];
```



## Tipe Data Array

# Tipe Data Heterogen (Campuran)

```
let campuran = ["Ali", 25, true, null];
```



## Tipe Data Array

# Array of Object

Object adalah **tipe data kompleks** yang menyimpan **data dalam bentuk pasangan kunci – nilai (key – value)**

Mirip seperti tabel dengan **nama kolom (key)** dan **isi data (value)**

Struktur Umum :

```
let namaArray = [  
    { key1: value1, key2: value2 },  
    { key1: value3, key2: value4 },  
    ...  
];
```

Contoh :

```
let siswa = [  
    { nama: "Ali", nilai: 90 },  
    { nama: "Budi", nilai: 85 },  
    { nama: "Cici", nilai: 88 }  
];
```

## Tipe Data Array

# Array of Object

Cara mengakses nilai object :

```
1  Let siswa = [
2    { nama: "Ali", nilai: 90 },
3    { nama: "Budi", nilai: 85 },
4    { nama: "Cici", nilai: 88 },
5  ];
6  console.log(siswa[0].nama);
7  console.log(siswa[1].nilai);
```

## Tipe Data Array

- \$ node array-object.js
- Ali
- 85

Untuk mengetahui **berapa banyak elemen** di dalam sebuah array, menggunakan property **.length**

Contoh :

```
1 Let buah = ["apel", "jeruk", "mangga", "pir"];
2 console.log(buah.length);
```

Output :

- \$ node panjang-array.js

4

## Panjang Array

# Array Iteration

Array iteration merupakan metode dalam array yang digunakan untuk melakukan **perulangan terhadap elemen – elemen array**.

3 metode yang digunakan :

**forEach()**

**map()**

**filter()**

**Method  
Array**

# forEach()

Digunakan untuk **menjalankan fungsi pada setiap elemen array**, tapi **tidak mengembalikan array baru**.

Sintaks :

```
array.forEach(function(elemen, index, array) {  
    // kode yang dijalankan  
});
```

Contoh :

```
1  let angka = [10, 20, 30];  
2  
3  angka.forEach(function (nilai, index) {  
4      console.log("Index ke-" + index + ": " + nilai);  
5  });
```

## Method Array

Output :

- \$ node for-each.js  
Index ke-0: 10  
Index ke-1: 20  
Index ke-2: 30

# map()

Digunakan untuk **mengubah nilai setiap elemen** dalam array dan **mengembalikan array baru**.

Sintaks :

```
let hasil = array.map(function(elemen, index, array) {  
    return elemenBaru;  
});
```

Contoh :

```
1  let angka = [1, 2, 3];  
2  let kuadrat = angka.map(function (nilai) {  
3      return nilai * nilai;  
4  });  
5  
6  console.log(kuadrat);
```

## Method Array

Output :

```
● $ node map.js  
[ 1, 4, 9 ]
```

# filter()

Digunakan untuk **menyaring elemen array** berdasarkan kondisi tertentu, dan **mengembalikan array baru** berisi elemen yang lolos.

Sintaks :

```
let hasil = array.filter(function(elemen) {  
    return kondisi;  
});
```

Contoh :

```
1  let nilai = [70, 80, 45, 90];  
2  let lulus = nilai.filter(function (n) {  
3      return n >= 75;  
4  });  
5  
6  console.log(lulus);
```

## Method Array

Output :

```
$ node filter.js  
[ 80, 90 ]
```

# Menambah/Menghapus Elemen

## Method

## Fungsi

`push()`

Menambah elemen di akhir array

`pop()`

Menghapus elemen terakhir

`unshift()`

Menambah elemen di awal array

`shift()`

Menghapus elemen pertama

`splice()`

Menambah/menghapus elemen di posisi tertentu

`slice()`

Mengambil sebagian array (tidak mengubah array asli)

## Method Array

# Menambah/Menghapus Elemen

Contoh :

```
1 //menambah elemen di akhir
2 let buah = ['apel', 'jeruk'];
3 buah.push('mangga');
4 console.log(buah);
5
6 //menghapus elemen terakhir
7 let buah1 = ['apel', 'jeruk', 'mangga'];
8 buah.pop();
9 console.log(buah);
10
11 //menambah elemen di awal
12 let buah3 = ['jeruk', 'mangga'];
13 buah.unshift('apel');
14 console.log(buah);
15
16 //menghapus elemen pertama
17 let buah4 = ['apel', 'jeruk', 'mangga'];
18 buah.shift();
19 console.log(buah);
```

## Method Array

Output :

```
$ node tambah-elemen.js
[ 'apel', 'jeruk', 'mangga' ]
[ 'apel', 'jeruk' ]
[ 'apel', 'apel', 'jeruk' ]
[ 'apel', 'jeruk' ]
```

# Menambah/Menghapus Elemen

Contoh :

```
21 //menambah elemen di posisi tertentu
22 let angka = [1, 2, 4, 5];
23 angka.splice(2, 0, 3);
24 console.log(angka);

25

26 //menghapus elemen di posisi tertentu
27 let angka1 = [1, 2, 3, 4, 5];
28 angka1.splice(2, 2);
29 console.log(angka1);

30

31 //mengambil sebagian array
32 let angka3 = [1, 2, 3, 4, 5];
33 let potong = angka3.slice(1, 4);
34 console.log(potong);
35 console.log(angka3);
```

## Method Array

Output :

```
[ 1, 2, 3, 4, 5 ]
[ 1, 2, 5 ]
[ 2, 5 ]
[ 1, 2, 5 ]
```

# Pengurutan Array

Method .sort() untuk **mengurutkan elemen array**. .sort() mengurutkan elemen sebagai **string (bukan angka)**

**Pengurutan String (Alfabetis) :**

Contoh :

```
1  let nama = ["Emily", "Mark", "Ashley", "James"];
2  nama.sort();
3  console.log(nama);
```

Method  
Array

Output :

- \$ node urut-string.js  
[ 'Ashley', 'Emily', 'James', 'Mark' ]

# Pengurutan Angka Turun (descending) :

Contoh :

```
1 let angka = [40, 100, 1, 5, 25, 10];
2 angka.sort(function (a, b) {
3     return b - a;
4 });
5 console.log(angka);
```

## Pengurutan Array

Output :

```
● $ node urut-angka-desc.js
[ 100, 40, 25, 10, 5, 1 ]
```

# Pengurutan Angka Naik (ascending) :

Contoh :

```
1  Let angka = [40, 100, 1, 5, 25, 10];
2
3  angka.sort(function(a, b) {
4      return a - b;
5  });
6  console.log(angka);
```

## Pengurutan Array

Output :

- \$ node urut-angka.js  
[ 1, 5, 10, 25, 40, 100 ]

# Custom sort berdasarkan property array of object

Contoh :

```
1 let siswa = [
2   { nama: "Ali", nilai: 85 },
3   { nama: "Budi", nilai: 92 },
4   { nama: "Cici", nilai: 75 }
5 ];
6
7 siswa.sort((a, b) => b.nilai - a.nilai);
8 console.log(siswa);
9 // Urutan dari nilai tertinggi ke terendah
```

Output :

- \$ node custom-sort.js

```
[  
  { nama: 'Budi', nilai: 92 },  
  { nama: 'Ali', nilai: 85 },  
  { nama: 'Cici', nilai: 75 }  
]
```

# Pengurutan Array



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

EK 5

# Membuat Program untuk Akses File

Smart Creative Skills Team ©

# Akses File

Proses untuk membuka, membaca, menulis, atau memodifikasi file yang disimpan di media penyimpanan melalui program komputer.

: : +



## Membuka File

Menyiapkan file agar bisa diakses oleh program

## Membaca File

Mengambil isi file dan digunakan dalam program

## Menulis File

Menyimpan data baru atau mengganti data dalam file

## Menutup File

Mengakhiri akses dan melepaskan resource sistem

## Menghapus File

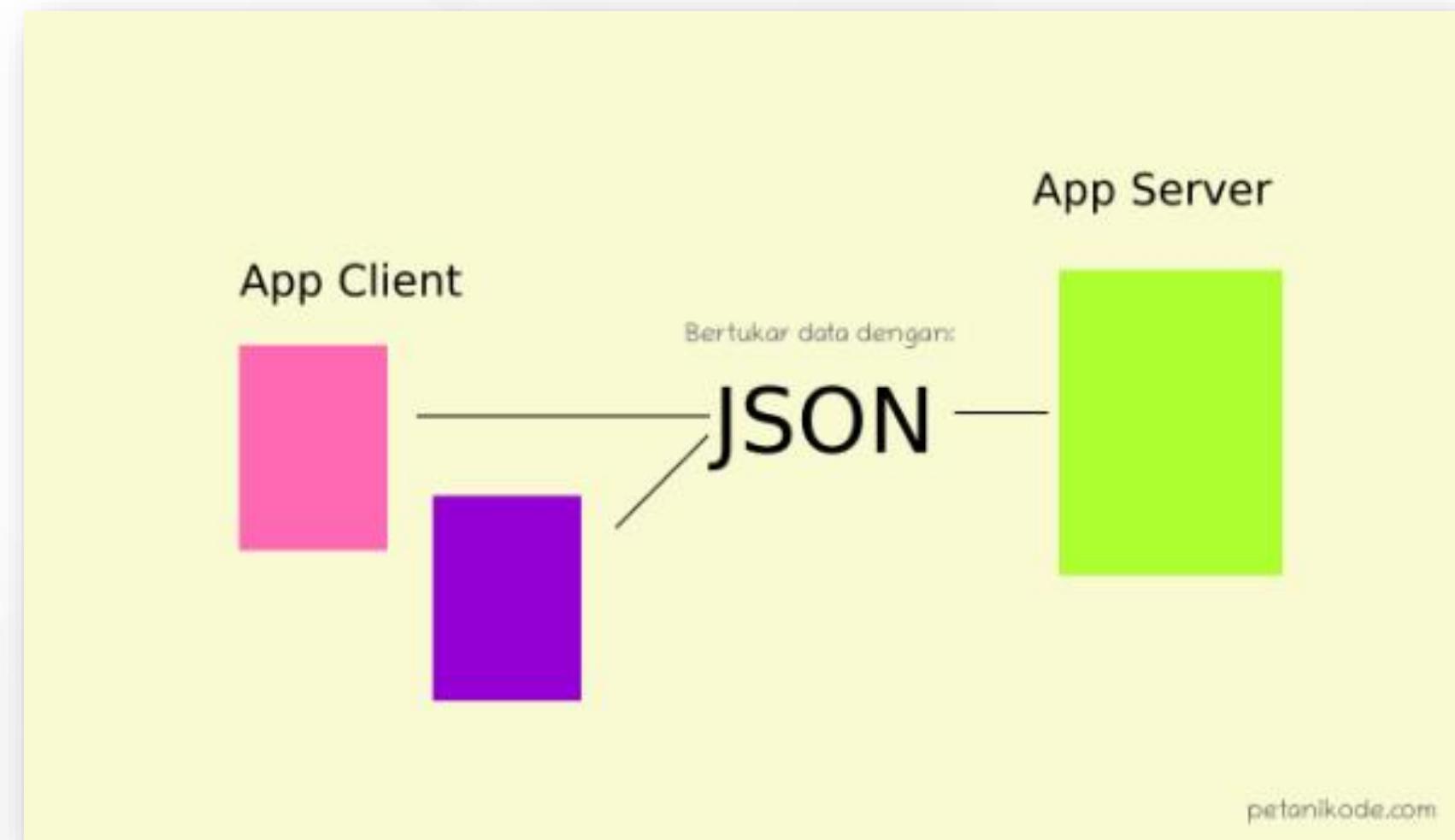
Menghilangkan file dari sistem

JSON (JavaScript Object Notation) adalah **format penyimpanan data** yang :

1. Ringkas
2. Bisa dibaca manusia
3. Digunakan untuk pertukaran data antara sistem

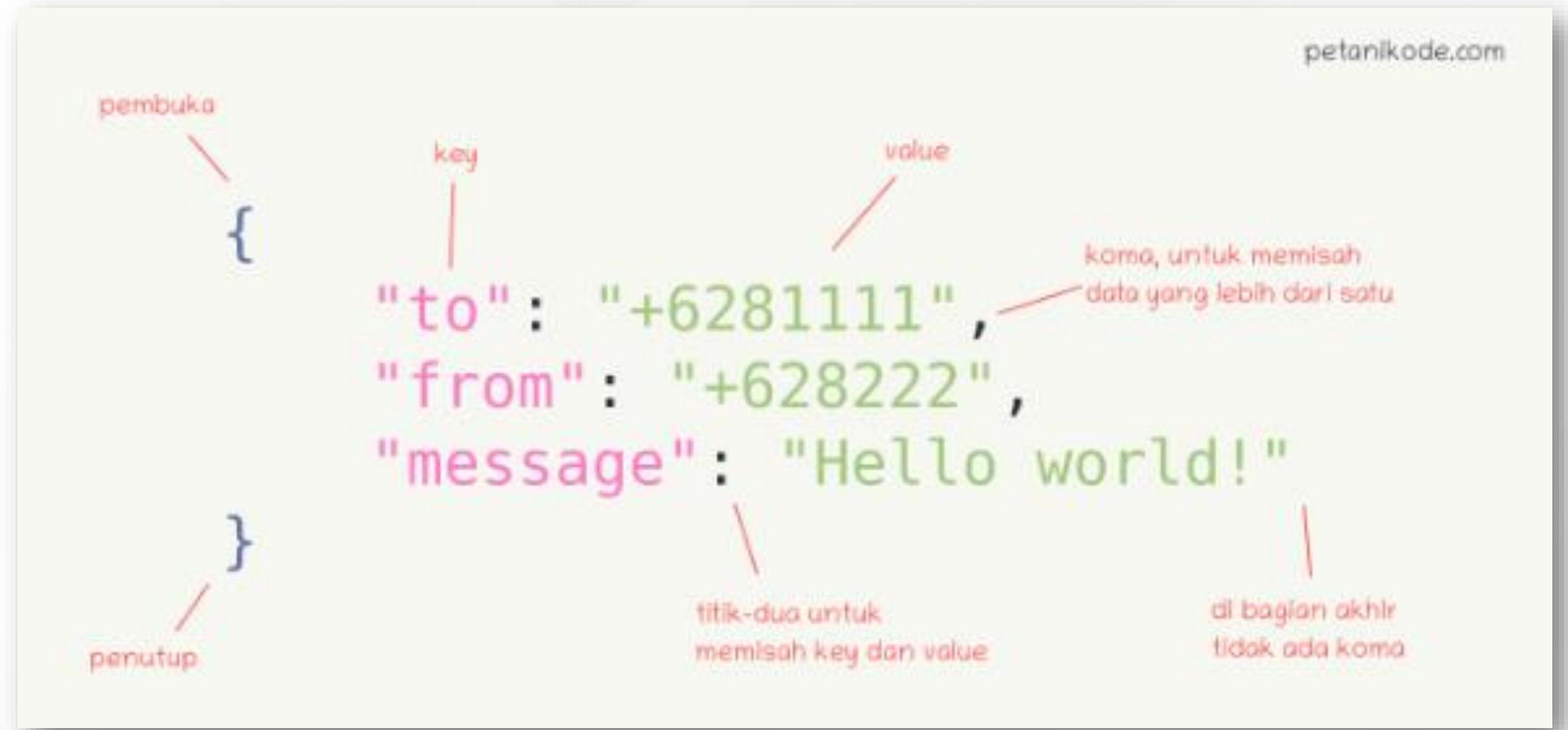


JSON



# Struktur Dasar JSON

JSON selalu dimulai dengan tanda kurung kurawal {...}



# JSON

# Struktur Dasar JSON dalam Array [...]

```
{  
    "name": "Dian",  
    "hobbies": ["Coding", "Blogging", "Drawing"]  
}
```



JSON

```
[  
    { "nama": "Ali", "usia": 20 },  
    { "nama": "Budi", "usia": 22 }  
]
```

# Struktur Dasar JSON dalam Object {...}

```
{  
    "name": "petanikode",  
    "url": "https://www.petanikode.com",  
    "rank": 1,  
    "socialmedia": {  
        "facebook": "petanikode",  
        "twitter": "petanikode",  
        "instagram": "petanikode",  
        "youtube": "petanikode",  
        "github": "petanikode"  
    }  
}
```

Object

JSON

```
{  
    "nama": "Ali",  
    "usia": 20  
}
```

# Contoh Akses File JSON :

## File data.json

```
json > {} data.json > ...
1  < [ 
2    {"id": 1, "nama": "Emily", "usia": 20},
3    {"id": 2, "nama": "Mark", "usia": 25},
4    {"id": 3, "nama": "Sherly", "usia": 21}
5  ]
```

```
json > JS akses-json.js > ...
1  const fs = require("fs");
2
3  //Membuka & membaca file JSON
4  //otomatis membuka
5  const teks = fs.readFileSync("data.json", "utf-8");
6  const data = JSON.parse(teks);
7
8  console.log("Isi Awal : ", data);
9
10 //Menulis data baru ke array
11 data.push({ id: 4, nama: "Jeremy", usia: 28 });
12
13 //Menyimpan (menulis ulang)
14 //otomatis menutup
15 fs.writeFileSync("data.json", JSON.stringify(data, null, 2));
16
17 console.log("Data berhasil ditambahkan dan disimpan ulang");
18 console.log("Isi Akhir : ", data);
```



# JSON

## File akses-json.js

# Contoh Akses File JSON :

Output :

```
● $ node akses-json.js
Isi Awal : [
  { id: 1, nama: 'Emily', usia: 20 },
  { id: 2, nama: 'Mark', usia: 25 },
  { id: 3, nama: 'Sherly', usia: 21 }
]
Data berhasil ditambahkan dan disimpan ulang
Isi Akhir : [
  { id: 1, nama: 'Emily', usia: 20 },
  { id: 2, nama: 'Mark', usia: 25 },
  { id: 3, nama: 'Sherly', usia: 21 },
  { id: 4, nama: 'Jeremy', usia: 28 }
]
```

Hasil akhir file  
data.json :

```
json > {} data.json > ...
1  [
2   {
3     "id": 1,
4     "nama": "Emily",
5     "usia": 20
6   },
7   {
8     "id": 2,
9     "nama": "Mark",
10    "usia": 25
11 },
12  {
13    "id": 3,
14    "nama": "Sherly",
15    "usia": 21
16 },
17  {
18    "id": 4,
19    "nama": "Jeremy",
20    "usia": 28
21 }
22 ]
```

# JSON



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

**EK 6**

# **Mengkompilasi Program**



Smart Creative Skills Team ©

# Pengertian Kompilasi

**Kompilasi** adalah proses mengubah kode sumber (source code) menjadi bentuk lain yang bisa dijalankan oleh komputer.

Javascript bersifat interpreted, proses pengecekan dan eksekusi terjadi saat program dijalankan

# Peran Kompiler

Pada javascript, kompiler digantikan oleh **engine interpreter** (misal : v8 di node.js) yang fungsinya untuk :

1. Membaca dan mengeksekusi kode baris per baris
2. Tetap mengecek kesalahan sintaks saat eksekusi awal

# Proses Kompilasi

1. Membuat file .js
2. Menjalankan program via terminal :

```
node nama-file.js
```

3. Node.js akan memproses kode :
  - Jika ada **kesalahan sintaks, program gagal jalan** dan muncul error
  - Jika tidak ada, program akan dijalankan normal

# Cara Kompilasi

File kompilasi.js :

```
1 const nama = "Emily";
2 console.log("Hai,", nama);
```

Jalankan program + output jika tidak ada kesalahan :

```
● $ node kompilasi.js
Hai, Emily
```

# Cara Kompilasi

File kompilasi.js :

```
1 const nama = "Emily";
2 console.log("Hai,", nama);
```

Jalankan program + outputnya jika ada kesalahan sintaks :

```
② $ node kompilasi.js
③ I:\2025\ToT Web Development\Materi UK Mengimplementasikan Pemrograman Terstruktur\Coding\kompilasi.js:2
  console.log("Hai,", nama;
               ^
               ^
               ^
               ^
SyntaxError: missing ) after argument list
```

# Jenis Pesan/Kesalahan Kompilasi

Jenis Kesalahan	Contoh	Penjelasan
SyntaxError	Unexpected token	Penulisan kode tidak sesuai aturan
ReferenceError	x is not defined	Variabel belum dideklarasikan
TypeError	Undefined is not a function	Kesalahan tipe data
RangeError	Maximum call stack size exceeded	Fungsi rekursif berlebihan
Module not found	Saat require() file yang salah	Gagal menemukan module yang dipanggil

# Try ... Catch

Penanganan kesalahan dengan try .... catch

```
try {  
    // Kode yang bisa menimbulkan error  
} catch (error) {  
    // Menangani error  
}  
:  
:+
```

Fungsi :

1. Mencegah program berhenti tiba – tiba
2. Menangkap dan menangani error secara aman

# Try ... Catch

Contoh :

```
1 try {  
2     let hasil = x + 10; // x belum didefinisikan  
3     console.log("Hasil:", hasil);  
4 } catch (error) {  
5     console.error("Kesalahan:", error.message);  
6 }
```

Output :

```
$ node try-catch.js  
Kesalahan: x is not defined
```



# Kesimpulan

## Menggunakan tipe data dan *control program*

- Tipe data penting untuk menentukan jenis nilai (seperti string, number, boolean, array, dan object) dalam program.
- *Control program* (if, switch, for, while) digunakan untuk mengatur alur logika eksekusi program.
- Kombinasi tipe data dan *control program* memungkinkan pembuatan logika dinamis dan efisien.





# Kesimpulan

## Membuat program sederhana kondisi dan perulangan pada Javascript

- Kondisi (if, else, switch) digunakan untuk pengambilan Keputusan.
- Perulangan (for, while) digunakan untuk mengulang kode.
- Kombinasi keduanya membentuk logika program yang dinamis dan efisien.





# Kesimpulan

## Membuat program menggunakan prosedur dan fungsi pada Javascript

- Fungsi (*function*) adalah blok kode terstruktur untuk menyelesaikan tugas tertentu dan dapat dipanggil berulang kali.
- Prosedur mirip fungsi tetapi biasanya tidak mengembalikan nilai.
- Penggunaan fungsi meningkatkan modularitas, efisiensi, dan keterbacaan kode.



# Kesimpulan

## Membuat program menggunakan *array* pada Javascript

- *Array* menyimpan kumpulan data dalam satu variabel dan mendukung indeks numerik.
- Metode seperti *push*, *pop*, *shift*, *unshift*, *map*, *filter*, dan *sort* mempermudah manipulasi data.
- Penggunaan *array* memungkinkan pengolahan data secara massal dan efisien.





# Kesimpulan

## Membuat program untuk akses *file*

- Javascript di Node.js dapat membaca, menulis, dan memodifikasi *file* menggunakan modul fs.
- Akses *file* berguna untuk menyimpan data secara permanen di luar memori program.
- Format JSON sering digunakan untuk menyimpan dan memproses data *file* karena kesesuaiannya dengan struktur data Javascript.



# Kesimpulan

## Mengkompilasi program

- Mengkompilasi program dengan CLI, digunakan untuk mengubah dan membundel kode Javascript agar efisien dan kompatibel, menggunakan tools seperti Babel dan Webpack lewat command line.
- Jenis – jenis error pada Javascript : Syntax error, Runtime error, Logical error.
- Try – Catch, digunakan untuk menangani error saat runtime agar program tidak langsung berhenti dan bisa memberi respon yang sesuai.

**Ikuti instruksi pada lembar tugas  
praktek yang dibagikan**



**Aplikasi**

# EVALUASI

Buat program Javascript sederhana untuk mengelola data nilai mahasiswa. Fitur dari program adalah sebagai berikut :

1. Input data mahasiswa : nama, mata kuliah, dan nilai
2. Tentukan keterangan kelulusan berdasarkan nilai :  
Nilai  $\geq 75$  : Lulus  
Nilai  $< 75$  : Tidak Lulus
3. Simpan data ke dalam array (berisi objek mahasiswa)
4. Tampilkan semua data siswa dalam format rapi
5. Gunakan fungsi untuk memisahkan tugas – tugas berikut :
  - a. Fungsi input data
  - b. Fungsi menghitung kelulusan
  - c. Fungsi menampilkan data

# Tugas

# Tugas

- Kode program di screenshot dan di copy paste ke word.
- Nama *file* : noabsen\_nama.docx
- Kirim ke email, dengan subjek Tugas1\_Kelas, sebelum pertemuan berikutnya.



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA

# SEKIAN DAN TERIMA KASIH

Sampai jumpa di pertemuan selanjutnya

**Salam Kompeten !**



2025



Smart Creative Skills Team ©