

BELAJAR DASAR JAVASCRIPT

1. Pengenalan Dasar JavaScript

- Apa itu JavaScript dan fungsinya dalam pengembangan web
- Menjalankan JavaScript di browser (Console DevTools)
- Sintaks dasar JavaScript
- Variabel (let, const, var)
- Tipe data (string, number, boolean, array, object)

📌 Sumber belajar:

- [MDN JavaScript Guide](#)
 - Coba latihan di **console browser** (tekan F12 di Chrome → Tab "Console")
-

2. Struktur dan Kontrol Program

- Operator (aritmatika, logika, perbandingan)
- Percabangan (if, else, switch)
- Perulangan (for, while, do while)
- Function (fungsi dasar, parameter, return value)

📌 Latihan:

- Buat program sederhana seperti **kalkulator sederhana** atau **cek bilangan ganjil/genap**
-

3. Manipulasi DOM (Document Object Model)

- Cara mengakses elemen HTML dengan JavaScript (`document.getElementById`, `querySelector`)
- Mengubah teks dan atribut elemen HTML
- Menambahkan dan menghapus elemen HTML
- Event listener (`onclick`, `onchange`, `onmouseover`, dll.)

📌 Latihan:

- Buat **tombol yang mengubah warna background** saat diklik
-

4. Array dan Object

- Cara membuat dan mengakses array (push, pop, map, filter)
- Pengenalan objek (key-value, cara mengakses properti)
- Menggunakan metode **array dan object**

📌 Latihan:

- Buat daftar tugas sederhana (To-Do List)
-

6. JavaScript Modern (ES6+)

- Destructuring
- Template literals
- Spread & Rest operator
- Arrow function

📌 Sumber belajar:

- ES6 Cheatsheet

5. Asynchronous JavaScript

- Pengenalan **Callback, Promise, dan Async/Await**
- Cara kerja **setTimeout()** dan **setInterval()**
- Fetch API untuk mengambil data dari server

📌 Latihan:

- Ambil data dari **JSONPlaceholder API** menggunakan **fetch()**
-

🔥 Tips Belajar JavaScript 🔥

- **Banyak praktik!** Jangan hanya membaca teori, langsung coba coding.
- **Buat proyek sederhana**, seperti kalkulator, to-do list, atau website portofolio.
- **Gabung komunitas** seperti forum JavaScript, Discord, atau Stack Overflow untuk bertanya jika mengalami kendala.

Pengenalan JavaScript

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat halaman web lebih interaktif dan dinamis. Dengan JavaScript, kita bisa:

- Mengubah konten HTML dan CSS secara dinamis
- Membuat efek animasi dan transisi
- Mengontrol elemen di dalam halaman web
- Mengambil dan mengirim data dari atau ke server
- Membangun aplikasi web modern (seperti SPA - Single Page Application)

JavaScript berjalan langsung di dalam browser tanpa perlu diinstal secara terpisah. Setiap browser modern seperti **Google Chrome, Mozilla Firefox, Microsoft Edge, dan Safari** sudah memiliki mesin JavaScript bawaan.

Menjalankan JavaScript di Web Browser (Praktikum-1)

Ada beberapa cara menjalankan JavaScript:

1. Menjalankan JavaScript di dalam HTML (Inline JS)

Kita bisa menulis kode JavaScript langsung di dalam elemen HTML menggunakan atribut onclick, onmouseover, dll.

Contoh:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contoh JavaScript</title>
</head>
<body>
    <button onclick="alert('Halo, Selamat Datang!')">Klik Saya</button>
</body>
</html>
```

Penjelasan:

Ketika tombol diklik, JavaScript akan menjalankan fungsi alert() yang menampilkan pesan pop-up.

2. Menjalankan JavaScript di dalam <script> (Internal JS)

Kita juga bisa menulis JavaScript di dalam tag <script> di dalam file HTML.

Contoh:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Internal JavaScript</title>
</head>
<body>

    <h1 id="judul">Halo, Dunia!</h1>
    <button onclick="ubahTeks()">Klik untuk Ubah</button>

    <script>
        function ubahTeks() {
            document.getElementById("judul").innerHTML = "Teks Berubah!";
        }
    </script>
</body>
</html>
```

Penjelasan:

- Fungsi ubahTeks() mengubah teks pada elemen <h1> ketika tombol diklik.

3. Menjalankan JavaScript dari File Eksternal (External JS)

Kita bisa menyimpan kode JavaScript dalam file terpisah dengan ekstensi .js, lalu menghubungkannya ke HTML.

Contoh:

File index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>External JavaScript</title>
    <script src="script.js"></script>
</head>
<body>

    <h1 id="judul">Halo, Dunia!</h1>
    <button onclick="ubahTeks()">Klik untuk Ubah</button>

</body>
</html>
```

File script.js

```
function ubahTeks() {
    document.getElementById("judul").innerHTML = "Teks Berubah!";
}
```

Penjelasan:

- File JavaScript eksternal memisahkan kode dari file HTML, membuatnya lebih rapi dan mudah dikelola.

Sintaks Dasar JavaScript (Praktikum-2)

Sintaks dalam JavaScript mirip dengan bahasa pemrograman lain seperti C atau Java. Berikut beberapa aturan dasar:

1. **Setiap perintah diakhiri dengan tanda titik koma ;**

```
console.log("Halo, dunia!");
```

2. **JavaScript bersifat case-sensitive**

nama dan Nama adalah dua variabel yang berbeda.

3. **JavaScript mengabaikan spasi kosong dan indentasi**

Namun, menulis kode dengan indentasi yang baik membuatnya lebih mudah dibaca.

4. **Menggunakan komentar**

Komentar satu baris: //

Komentar multi-baris:

```
/*
Ini adalah komentar multi-baris
*/
```

Variabel dalam JavaScript (Praktikum-3)

Variabel digunakan untuk menyimpan nilai dalam program. Ada tiga cara mendeklarasikan variabel dalam JavaScript:

1. Menggunakan **var** (cara lama, kurang direkomendasikan)

```
var nama = "Budi";
console.log(nama);
```

Kelemahan **var**

- Bisa dideklarasikan ulang tanpa error.
- Bersifat **global scope**, sehingga mudah terjadi konflik variabel.

2. Menggunakan **let** (disarankan)

```
let umur = 25;
console.log(umur);
```

Kelebihan **let**

- Tidak bisa dideklarasikan ulang.
- Bersifat **block scope**, sehingga lebih aman digunakan.

3. Menggunakan **const** (untuk nilai tetap)

```
const PI = 3.14;
console.log(PI);
```

Kelebihan **const**

- Nilainya tidak bisa diubah setelah deklarasi.

Tipe Data dalam JavaScript (Praktikum-4)

JavaScript memiliki beberapa tipe data utama:

1. String (Teks)

Digunakan untuk menyimpan teks, ditulis dalam tanda kutip (" " atau ' ').

```
let nama = "Andi";
console.log(nama);
```

2. Number (Angka)

Digunakan untuk angka, baik bilangan bulat maupun desimal.

```
let umur = 30;
let harga = 199.99;
console.log(umur, harga);
```

3. Boolean (True/False)

Digunakan untuk nilai benar atau salah.

```
let isActive = true;
console.log(isActive);
```

4. Array (Daftar)

Digunakan untuk menyimpan beberapa nilai dalam satu variabel.

```
let buah = ["Apel", "Jeruk", "Mangga"];
console.log(buah[0]); // Output: Apel
```

5. Object (Data Berstruktur)

Menyimpan data dalam format key-value.

```
let orang = {
  nama: "Siti",
  umur: 25,
  pekerjaan: "Programmer"
};
console.log(orang.nama); // Output: Siti
```

Kesimpulan

1. **JavaScript** adalah bahasa pemrograman yang berjalan di browser.
 2. Bisa dijalankan langsung di **HTML**, di dalam <script>, atau melalui file eksternal.
 3. **Sintaks dasar** JavaScript mirip dengan bahasa pemrograman lain dan menggunakan titik koma ; .
 4. **Variabel** bisa dideklarasikan dengan var, let, atau const.
 5. **Tipe data utama** di JavaScript meliputi string, number, boolean, array, dan object.
-

Struktur dan Kontrol Program dalam JavaScript (Praktikum-5)

Struktur kontrol dalam JavaScript digunakan untuk **mengatur alur program**, seperti melakukan operasi matematika, membuat keputusan (percabangan), atau menjalankan kode secara berulang (looping).

1. Operator dalam JavaScript

Operator digunakan untuk melakukan operasi pada variabel dan nilai.

a) Operator Aritmatika (Matematika)

Digunakan untuk perhitungan matematika dasar.

Operator	Keterangan	Contoh	Hasil
+	Penjumlahan	5 + 3	8
-	Pengurangan	5 - 3	2
*	Perkalian	5 * 3	15
/	Pembagian	6 / 3	2
%	Modulus (Sisa Bagi)	5 % 3	2
**	Pangkat	2 ** 3	8

Contoh Kode:

```
let a = 10;
let b = 3;

console.log(a + b); // Output: 13
console.log(a - b); // Output: 7
console.log(a * b); // Output: 30
console.log(a / b); // Output: 3.333...
console.log(a % b); // Output: 1
console.log(a ** b); // Output: 1000
```

b) Operator Perbandingan

Digunakan untuk membandingkan dua nilai dan menghasilkan `true` atau `false`.

Operator	Keterangan	Contoh	Hasil
==	Sama dengan (tanpa cek tipe data)	5 == "5"	true
===	Sama dengan (cek tipe data)	5 === "5"	false
!=	Tidak sama dengan	5 != 3	true
!==	Tidak sama dengan (cek tipe data)	5 !== "5"	true
>	Lebih besar	5 > 3	true
<	Lebih kecil	5 < 3	false

Operator	Keterangan	Contoh	Hasil
<code>>=</code>	Lebih besar atau sama dengan	<code>5 >= 5</code>	<code>true</code>
<code><=</code>	Lebih kecil atau sama dengan	<code>3 <= 5</code>	<code>true</code>

Contoh Kode:

```
console.log(5 == "5"); // true (karena hanya membandingkan nilai)
console.log(5 === "5"); // false (karena tipe data berbeda)
console.log(10 > 5); // true
console.log(10 < 5); // false
console.log(10 !== "10"); // true
```

c) Operator Logika

Digunakan untuk menghubungkan beberapa kondisi logika.

Operator	Keterangan	Contoh	Hasil
<code>&&</code>	DAN (AND)	<code>(5 > 3) && (8 > 5)</code>	<code>true</code>
<code>^</code>		<code>^</code>	ATAU (OR)
<code>!</code>	NEGASI (NOT)	<code>!(5 > 3)</code>	<code>false</code>

Contoh Kode:

```
let x = 10;
let y = 5;

console.log(x > 5 && y < 10); // true (karena kedua kondisi benar)
console.log(x > 5 || y > 10); // true (karena salah satu kondisi benar)
console.log(!(x > 5)); // false (karena kondisi aslinya true)
```

2. Percabangan (if, else, switch)

Percabangan digunakan untuk menjalankan kode berdasarkan kondisi tertentu.

a) if dan else

Digunakan untuk mengecek suatu kondisi, jika benar (`true`) kode di dalam blok `if` akan dijalankan, jika tidak akan masuk ke `else`.

Contoh Kode:

```
let umur = 18;

if (umur >= 17) {
    console.log("Anda sudah dewasa.");
} else {
    console.log("Anda masih anak-anak.");
}
```

Output:

Anda sudah dewasa.

b) if...else if...else

Digunakan jika ada lebih dari dua kondisi.

Contoh Kode:

```
let nilai = 80;

if (nilai >= 90) {
    console.log("A");
} else if (nilai >= 75) {
    console.log("B");
} else if (nilai >= 60) {
    console.log("C");
} else {
    console.log("D");
}
```

Output:

B

c) switch

Digunakan jika ada banyak kondisi yang perlu dicek.

Contoh Kode:

```
let hari = "Senin";

switch (hari) {
    case "Senin":
        console.log("Hari ini adalah Senin.");
        break;
    case "Selasa":
        console.log("Hari ini adalah Selasa.");
        break;
    default:
        console.log("Hari tidak diketahui.");
}
```

Output:

Hari ini adalah Senin.

3. Perulangan (Looping)

Looping digunakan untuk mengulang suatu kode berkali-kali.

a) for Loop

Digunakan ketika jumlah perulangan sudah diketahui.

Contoh Kode:

```
for (let i = 1; i <= 5; i++) {  
    console.log("Perulangan ke-" + i);  
}
```

Output:

Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-4
Perulangan ke-5

b) **while** Loop

Digunakan ketika jumlah perulangan **belum pasti**.

Contoh Kode:

```
let i = 1;  
  
while (i <= 5) {  
    console.log("Perulangan ke-" + i);  
    i++;  
}
```

Output:

Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-4
Perulangan ke-5

c) **do while** Loop

Mirip dengan **while**, tapi kode akan **dijalankan minimal satu kali**.

Contoh Kode:

```
let j = 1;  
  
do {  
    console.log("Perulangan ke-" + j);  
    j++;  
} while (j <= 5);
```

Output:

Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-4

Perulangan ke-5

4. Function dalam JavaScript

Fungsi digunakan untuk mengelompokkan kode agar bisa digunakan kembali.

a) Fungsi Dasar

Contoh Kode:

```
function sapa() {  
    console.log("Halo, selamat datang!");  
}  
  
sapa(); // Memanggil fungsi
```

b) Fungsi dengan Parameter

Fungsi bisa menerima input melalui parameter.

Contoh Kode:

```
function sapa(nama) {  
    console.log("Halo, " + nama + "!");  
}  
  
sapa("Budi");  
sapa("Siti");
```

Output:

CopyEdit
Halo, Budi!
Halo, Siti!

c) Fungsi dengan Return Value

Fungsi bisa mengembalikan nilai dengan `return`.

Contoh Kode:

```
function tambah(a, b) {  
    return a + b;  
}  
  
let hasil = tambah(5, 3);  
console.log("Hasil: " + hasil);
```

Output:

Hasil: 8

Kesimpulan

- **Operator** digunakan untuk operasi matematika, perbandingan, dan logika.
 - **Percabangan** (`if`, `else`, `switch`) digunakan untuk keputusan dalam program.
 - **Perulangan** (`for`, `while`, `do while`) digunakan untuk menjalankan kode secara berulang.
 - **Fungsi** digunakan untuk menyusun kode agar lebih rapi dan bisa digunakan kembali.
-

Manipulasi DOM (Document Object Model) dengan JavaScript (Praktikum-6)

DOM (Document Object Model) adalah struktur data yang digunakan oleh browser untuk merepresentasikan halaman web. Dengan DOM, JavaScript bisa mengakses dan memanipulasi elemen HTML secara dinamis, seperti mengubah teks, atribut, menambahkan elemen baru, dan menangani event (seperti klik dan hover).

1. Mengakses Elemen HTML dengan JavaScript

Kita bisa mengakses elemen HTML dengan beberapa metode:

a) `document.getElementById()`

Mengambil elemen berdasarkan **id**.

Contoh:

```
<h1 id="judul">Halo, Dunia!</h1>
<script>
  let elemen = document.getElementById("judul");
  console.log(elemen); // Menampilkan elemen di console
</script>
```

b) `document.getElementsByClassName()`

Mengambil elemen berdasarkan **class** (mengembalikan array-like).

Contoh:

```
<p class="teks">Paragraf 1</p>
<p class="teks">Paragraf 2</p>

<script>
  let elemen = document.getElementsByClassName("teks");
  console.log(elemen[0].innerText); // Output: Paragraf 1
</script>
```

c) `document.getElementsByTagName()`

Mengambil elemen berdasarkan **nama tag**.

Contoh:

```
<p>Paragraf pertama</p>
<p>Paragraf kedua</p>

<script>
  let elemen = document.getElementsByTagName("p");
  console.log(elemen[1].innerText); // Output: Paragraf kedua
</script>
```

d) `document.querySelector()` & `document.querySelectorAll()`

- `querySelector()` = Mengambil **elemen pertama** yang cocok.
- `querySelectorAll()` = Mengambil **semua elemen** yang cocok (array-like).

Contoh:

```
<h1 id="judul">Judul</h1>
<p class="teks">Paragraf</p>

<script>
  let elemen1 = document.querySelector("#judul");
  console.log(elemen1.innerText); // Output: Judul

  let elemen2 = document.querySelectorAll(".teks");
  console.log(elemen2[0].innerText); // Output: Paragraf
</script>
```

2. Mengubah Teks dan Atribut Elemen HTML

a) Mengubah teks dengan `innerText` & `innerHTML`

- `innerText` → Mengubah teks saja
- `innerHTML` → Bisa menyisipkan HTML

Contoh:

```
<h1 id="judul">Judul Lama</h1>

<script>
  document.getElementById("judul").innerText = "Judul Baru";
</script>
```

b) Mengubah atribut dengan `setAttribute()` dan `removeAttribute()`

Kita bisa mengubah atau menghapus atribut HTML.

Contoh:

```


<script>
  document.getElementById("gambar").setAttribute("src", "gambar2.jpg");
  document.getElementById("gambar").setAttribute("alt", "Gambar 2");
</script>
```

Untuk menghapus atribut:

```
document.getElementById("gambar").removeAttribute("alt");
```

3. Menambahkan dan Menghapus Elemen HTML

a) Menambahkan Elemen Baru

Langkah-langkah:

1. Buat elemen dengan `document.createElement()`.
2. Tambahkan teks dengan `innerText` atau `innerHTML`.
3. Sisipkan ke dalam dokumen dengan `appendChild()` atau `insertBefore()`.

Contoh: Menambahkan paragraf baru ke dalam `<div>`.

```
<div id="kontainer"></div>

<script>
    let paragraf = document.createElement("p");
    paragraf.innerText = "Ini paragraf baru!";

    document.getElementById("kontainer").appendChild(paragraf);
</script>
```

b) Menghapus Elemen

Gunakan `removeChild()` untuk menghapus elemen.

Contoh:

```
<p id="hapus">Ini akan dihapus</p>

<script>
    let elemen = document.getElementById("hapus");
    elemen.parentNode.removeChild(elemen);
</script>
```

4. Event Listener (onclick, onchange, onmouseover, dll.)

Event listener digunakan untuk menangani aksi pengguna seperti **klik, input, hover, scroll, dsb.**

a) onclick (Ketika Klik)

Contoh:

```
<button id="tombol">Klik Saya</button>

<script>
    document.getElementById("tombol").onclick = function() {
        alert("Tombol diklik!");
    };
</script>
```

b) **onchange** (Ketika Input Berubah)

Contoh:

```
<input type="text" id="input" placeholder="Ketik sesuatu">
<p id="hasil"></p>

<script>
    document.getElementById("input").onchange = function() {
        document.getElementById("hasil").innerText = "Anda mengetik: " +
        this.value;
    };
</script>
```

c) **onmouseover** (Saat Mouse Hover)

Contoh:

```
<p id="teks">Arahkan mouse ke sini!</p>

<script>
    document.getElementById("teks").onmouseover = function() {
        this.innerText = "Mouse di atas!";
    };
</script>
```

d) **addEventListener()** (Cara Modern untuk Event)

Cara terbaik menangani event adalah dengan `addEventListener()` karena lebih fleksibel.

Contoh:

```
<button id="tombol">Klik Saya</button>

<script>
    document.getElementById("tombol").addEventListener("click", function()
    {
        alert("Event Listener bekerja!");
    });
</script>
```

Kesimpulan

- **Mengakses elemen HTML** bisa dengan `getElementById`, `querySelector`, dll.
- **Mengubah teks** dengan `innerText` dan **mengubah atribut** dengan `setAttribute()`.
- **Menambahkan elemen baru** dengan `createElement()` dan menghapusnya dengan `removeChild()`.
- **Event listener** seperti `onclick`, `onchange`, dan `onmouseover` digunakan untuk interaksi pengguna.

Array dan Object dalam JavaScript (Praktikum-7)

Array dan **Object** adalah dua struktur data penting dalam JavaScript.

- **Array** digunakan untuk menyimpan daftar data secara berurutan.
- **Object** digunakan untuk menyimpan data dalam format **key-value**.

1. Array dalam JavaScript

a) Cara Membuat Array

Array bisa dibuat dengan menggunakan [] (square brackets).

Contoh:

```
let buah = [ "Apel", "Mangga", "Jeruk" ];
console.log(buah); // Output: [ "Apel", "Mangga", "Jeruk" ]
```

b) Mengakses Elemen Array

Gunakan indeks (**dimulai dari 0**) untuk mengakses elemen dalam array.

```
console.log(buah[0]); // Output: Apel
console.log(buah[1]); // Output: Mangga
```

c) Metode Dasar Array

Metode	Keterangan	Contoh
push()	Menambahkan elemen di akhir array	buah.push("Pisang")
pop()	Menghapus elemen terakhir	buah.pop()
unshift()	Menambahkan elemen di awal array	buah.unshift("Semangka")
shift()	Menghapus elemen pertama	buah.shift()
length	Menghitung jumlah elemen	buah.length

Contoh Kode:

```
buah.push("Pisang"); // Tambah di akhir
console.log(buah); // Output: [ "Apel", "Mangga", "Jeruk", "Pisang" ]

buah.pop(); // Hapus terakhir
console.log(buah); // Output: [ "Apel", "Mangga", "Jeruk" ]

buah.unshift("Semangka"); // Tambah di awal
console.log(buah); // Output: [ "Semangka", "Apel", "Mangga", "Jeruk" ]

buah.shift(); // Hapus pertama
console.log(buah); // Output: [ "Mangga", "Jeruk" ]

console.log(buah.length); // Output: 3
```

d) Looping pada Array

Gunakan `forEach()` atau `for` loop untuk menampilkan semua elemen.

```
buah.forEach(function(item, index) {  
    console.log(index + ": " + item);  
});
```

Output:

```
0: Apel  
1: Mangga  
2: Jeruk
```

2. Metode `map()`, `filter()`, `find()` pada Array

a) `map()` → Memproses setiap elemen dalam array

Mengembalikan array baru dengan hasil operasi pada setiap elemen.

Contoh:

```
let angka = [1, 2, 3, 4];  
let hasil = angka.map(function(num) {  
    return num * 2;  
});  
console.log(hasil); // Output: [2, 4, 6, 8]
```

b) `filter()` → Menyaring elemen berdasarkan kondisi

Mengembalikan array baru yang memenuhi kondisi tertentu.

Contoh:

```
let angkaBesar = angka.filter(function(num) {  
    return num > 2;  
});  
console.log(angkaBesar); // Output: [3, 4]
```

c) `find()` → Mencari elemen pertama yang memenuhi kondisi

Mengembalikan satu nilai pertama yang ditemukan.

Contoh:

```
let angkaPertama = angka.find(function(num) {  
    return num > 2;  
});  
console.log(angkaPertama); // Output: 3
```

3. Object dalam JavaScript

a) Cara Membuat Object

Object menggunakan **key-value** dalam {} (kurung kurawal).

```
let orang = {
  nama: "Budi",
  umur: 25,
  pekerjaan: "Programmer"
};
```

b) Mengakses Properti Object

Gunakan **dot notation** (.) atau **bracket notation** ([]).

```
console.log(orang.nama); // Output: Budi
console.log(orang["umur"]); // Output: 25
```

c) Menambahkan dan Menghapus Properti

Gunakan . atau delete.

```
orang.alamat = "Jakarta"; // Menambahkan properti baru
console.log(orang.alamat); // Output: Jakarta

delete orang.pekerjaan; // Menghapus properti
console.log(orang); // Output: { nama: "Budi", umur: 25, alamat: "Jakarta" }
```

4. Metode dalam Object

Object bisa memiliki **metode** (fungsi dalam objek).

```
let mobil = {
  merk: "Toyota",
  warna: "Merah",
  nyalakan: function() {
    console.log("Mobil dinyalakan!");
  }
};

mobil.nyalakan(); // Output: Mobil dinyalakan!
```

5. Array of Object

Array bisa berisi object.

```
let mahasiswa = [
  { nama: "Budi", umur: 20 },
  { nama: "Siti", umur: 22 },
```

```
];
```

```
console.log(mahasiswa[0].nama); // Output: Budi
```

Kesimpulan

- **Array** menyimpan banyak nilai, bisa dimanipulasi dengan metode seperti `push()`, `pop()`, `map()`, dan `filter()`.
- **Object** menyimpan data dalam bentuk **key-value** dan bisa memiliki metode sendiri.
- **Array of Object** adalah kombinasi dari array dan object, berguna untuk menyimpan banyak data kompleks.

Asynchronous JavaScript: Callback, Promise, dan Async/Await (Praktikum-8)

Dalam JavaScript, **kode berjalan secara sinkron (sequential/ berurutan)**. Namun, ada banyak situasi di mana kita ingin menjalankan kode secara **asinkron (tidak menunggu proses sebelumnya selesai)**, seperti saat:

- Mengambil data dari API
- Membaca file dari server
- Menjalankan operasi yang memakan waktu

JavaScript memiliki **Callback, Promise, dan Async/Await** untuk menangani operasi asinkron.

1. Callback: Cara Lama untuk Asynchronous

a) Apa itu Callback?

Callback adalah **fungsi yang dipanggil setelah tugas selesai**.

Contoh Sinkron (Tidak Asynchronous):

```
function halo() {  
    console.log("Halo, Dunia!");  
}  
halo();  
console.log("Kode berikutnya berjalan langsung.");
```

Output:

Halo, Dunia!
Kode berikutnya berjalan langsung.

- Karena tidak ada delay, eksekusi berjalan berurutan.
-

b) Contoh Callback Asynchronous (Menggunakan `setTimeout`)

```
console.log("Mulai");  
  
setTimeout(function () {  
    console.log("Proses Asynchronous selesai!");  
}, 2000); // Delay 2 detik  
  
console.log("Selesai");
```

Output:

Mulai
Selesai
Proses Asynchronous selesai! (setelah 2 detik)

- `setTimeout` menunda eksekusi callback selama 2 detik, tetapi kode berikutnya tetap berjalan.
-

c) Callback di dalam Fungsi

```
function prosesData(nama, callback) {
  console.log("Memproses data untuk " + nama);
  setTimeout(() => {
    callback();
  }, 2000);
}

prosesData("Budi", function () {
  console.log("Data berhasil diproses!");
});
```

Output:

Memproses data untuk Budi
(Data diproses selama 2 detik...)
Data berhasil diproses!

✗ Masalah Callback: Callback Hell (Nested Callback)

Jika terlalu banyak callback bersarang, kodenya sulit dibaca.

```
setTimeout(() => {
  console.log("Proses 1");
  setTimeout(() => {
    console.log("Proses 2");
    setTimeout(() => {
      console.log("Proses 3");
    }, 1000);
  }, 1000);
}, 1000);
```

✓ Solusi: Gunakan Promise atau Async/Await.

2. Promise: Cara Lebih Baik dari Callback

a) Apa itu Promise?

Promise adalah **objek yang merepresentasikan hasil dari operasi asynchronous** yang bisa:

1. **Pending** → Sedang dalam proses
2. **Resolved (Fulfilled)** → Berhasil
3. **Rejected** → Gagal

Contoh Promise Sederhana:

```
let janji = new Promise((resolve, reject) => {
  let sukses = true; // Ubah ke `false` untuk melihat hasil reject

  setTimeout(() => {
    if (sukses) {
      resolve("Data berhasil diambil!");
    } else {
      reject("Terjadi kesalahan!");
    }
  }, 2000);
});
```

```
janji
  .then((hasil) => console.log(hasil)) // Jika sukses
  .catch((error) => console.log(error)); // Jika gagal
```

- `.then()` dipanggil jika berhasil.
- `.catch()` dipanggil jika gagal.

b) Contoh Promise dengan Fetch API

`fetch()` adalah fungsi bawaan JavaScript untuk mengambil data dari API.

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then((response) => response.json()) // Mengubah response ke JSON
  .then((data) => console.log(data)) // Menampilkan data
  .catch((error) => console.log("Error:", error));
```

- Jika berhasil, data JSON akan ditampilkan.
- Jika gagal, `.catch()` akan menangani error.

3. Async/Await: Cara Modern dan Mudah

a) Apa itu Async/Await?

Async/Await adalah cara yang lebih mudah dibaca dan ditulis untuk menangani **Promise**.

Tanpa Async/Await (Menggunakan `.then()`)

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.log("Error:", error));
```

- Bisa dibuat lebih sederhana dengan **Async/Await**:

```
async function getData() {
  try {
    let response = await
    fetch("https://jsonplaceholder.typicode.com/posts/1");
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.log("Error:", error);
  }
}

getData();
```

- `await` menunggu hasil Promise sebelum lanjut ke baris berikutnya.
- `try...catch` menangkap error jika terjadi masalah.

b) Contoh Async/Await dalam Fungsi

```
function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            let sukses = true;
            if (sukses) {
                resolve("Data berhasil diambil!");
            } else {
                reject("Terjadi kesalahan!");
            }
        }, 2000);
    })
}

async function getData() {
    try {
        let hasil = await fetchData();
        console.log(hasil);
    } catch (error) {
        console.log(error);
    }
}

getData();
```

- Output jika sukses:** "Data berhasil diambil!"
- Output jika gagal:** "Terjadi kesalahan!"

Kesimpulan

- Callback:** Cara lama untuk menangani async, tapi bisa menyebabkan Callback Hell.
- Promise:** Mempermudah dengan `.then()` dan `.catch()`, lebih mudah dari callback.
- Async/Await:** Cara modern dan lebih mudah dibaca untuk menangani Promise.

Modul dan ES6+ (JavaScript Modern) (Praktikum-9)

JavaScript terus berkembang dengan fitur baru untuk membuat kode lebih **efisien, bersih, dan mudah dibaca**. ES6 (ECMAScript 2015) dan versi setelahnya memperkenalkan banyak fitur penting seperti **Modul, Destructuring, Spread Operator, Arrow Function, Class, dll.**

1. Modul dalam JavaScript

a) Apa Itu Modul?

Modul adalah cara untuk **memisahkan kode menjadi file terpisah** agar lebih rapi dan dapat digunakan kembali.

- Tanpa Modul (Script Biasa):** Semua kode dalam satu file .js.
- Dengan Modul:** Kode dipisah ke beberapa file .js, lalu di-*import* sesuai kebutuhan.

b) Cara Menggunakan Modul (Import & Export)

1. Buat file **math.js** (Modul yang berisi fungsi)

```
// math.js
export function tambah(a, b) {
    return a + b;
}

export function kali(a, b) {
    return a * b;
}
```

2. Gunakan di file lain (**main.js**)

```
// main.js
import { tambah, kali } from "./math.js";

console.log(tambah(5, 3)); // Output: 8
console.log(kali(5, 3)); // Output: 15
```

3. Gunakan **type="module"** di HTML untuk menjalankan file JS

```
<script type="module" src="main.js"></script>
```

- export** digunakan untuk mengirim fungsi/variabel dari file lain.
- import** digunakan untuk mengambil fungsi/variabel dari modul lain.

c) Default Export

Jika hanya ada satu nilai yang diekspor, bisa menggunakan **export default**.

1. Buat file **greeting.js**

```
// greeting.js
export default function greet(nama) {
    return `Halo, ${nama}!`;
```

}

2. Gunakan di file lain (main.js)

```
// main.js
import greet from "./greeting.js";

console.log(greet("Budi")); // Output: Halo, Budi!
```

- export default** hanya bisa digunakan sekali per file.
- Saat **import**, tidak perlu pakai {}.

2. Fitur ES6+ (JavaScript Modern)

a) **let** dan **const** (Pengganti **var**)

Variabel	Bisa diubah?	Berlaku dalam?
Var	Ya	Global/fungsi
Let	Ya	Blok {}
const	Tidak	Blok {}

Contoh:

```
let nama = "Budi"; // Bisa diubah
const umur = 25; // Tidak bisa diubah

nama = "Siti"; // OK
// umur = 30; ✗ Error!
```

b) Template Literal (String Interpolation dengan ``)

```
let nama = "Budi";
console.log(`Halo, nama saya ${nama}`); // Output: Halo, nama saya Budi
```

- Menggunakan **backtick** () dan \${ } untuk menyisipkan variabel.

c) Arrow Function (Fungsi Lebih Ringkas)

```
// Fungsi biasa
function halo(nama) {
    return `Halo, ${nama}`;
}

// Arrow function (lebih singkat)
const halo = (nama) => `Halo, ${nama}`;

console.log(halo("Budi")); // Output: Halo, Budi
```

- Tidak perlu pakai **return** jika hanya satu baris.
- Lexical this** (tidak mengubah konteks **this** dalam objek).

d) Destructuring (Mengambil Data dari Array/Object dengan Mudah)

Contoh pada Array:

```
let angka = [1, 2, 3];

let [a, b, c] = angka; // Destructuring array
console.log(a, b, c); // Output: 1 2 3
```

Contoh pada Object:

```
let orang = { nama: "Budi", umur: 25 };

let { nama, umur } = orang; // Destructuring object
console.log(nama, umur); // Output: Budi 25
```

- Tidak perlu akses `orang.nama`, cukup `nama`.

e) Spread Operator (...)

Digunakan untuk **menyalin atau menggabungkan array/object**.

Contoh: Menyalin Array

```
let angka1 = [1, 2, 3];
let angka2 = [...angka1, 4, 5]; // Gabungkan array

console.log(angka2); // Output: [1, 2, 3, 4, 5]
```

Contoh: Menyalin Object

```
let orang = { nama: "Budi", umur: 25 };
let orangBaru = { ...orang, pekerjaan: "Programmer" };

console.log(orangBaru); // Output: { nama: "Budi", umur: 25, pekerjaan: "Programmer" }
```

f) Rest Parameter (...) dalam Function

Mengubah banyak parameter menjadi array.

```
function jumlah(...angka) {
    return angka.reduce((total, num) => total + num, 0);
}

console.log(jumlah(1, 2, 3, 4)); // Output: 10
```

- `...angka` mengubah input menjadi array `[1, 2, 3, 4]`.

g) Class dalam JavaScript (ES6 OOP)

Sebelumnya, JavaScript tidak memiliki class seperti bahasa pemrograman lain (Java, Python).

Membuat class:

```
class Orang {
  constructor(nama, umur) {
    this.nama = nama;
    this.umur = umur;
  }

  sapa() {
    return `Halo, saya ${this.nama}`;
  }
}

let budi = new Orang("Budi", 25);
console.log(budi.sapa()); // Output: Halo, saya Budi
```

- constructor** adalah fungsi yang otomatis dipanggil saat objek dibuat.
 - Bisa membuat banyak objek dengan struktur yang sama.
-

Kesimpulan

- Modul:** Memecah kode menjadi file kecil agar lebih rapi.
 - let & const:** Pengganti `var`, lebih aman.
 - Template Literal:** String dengan `${ }` lebih mudah dibaca.
 - Arrow Function:** Fungsi lebih ringkas dan praktis.
 - Destructuring:** Ambil nilai dari Array/Object dengan mudah.
 - Spread & Rest Operator:** Menggabungkan atau menyebarluaskan nilai.
 - Class:** Cara modern membuat Object-Oriented Programming (OOP).
-

