JS

JavaScript

untuk Pemula

Panduan Lengkap Belajar JavaScript dari Nol

Pemula Friendly | **Contoh Kode Lengkap** | **Latihan Praktis**

Edisi 2025 · Lengkap & Terstruktur

Daftar Isi

BAB 1

Pengenalan JavaScript

JavaScript adalah bahasa pemrograman tingkat tinggi yang awalnya dibuat untuk membuat halaman web menjadi interaktif. Saat ini, JavaScript berkembang menjadi bahasa yang sangat powerful dan dapat digunakan di mana saja — mulai dari browser, server (Node.js), aplikasi mobile, hingga IoT.

1.1 Apa itu JavaScript?

JavaScript (JS) adalah bahasa pemrograman yang berjalan langsung di browser tanpa perlu dikompilasi. Ia adalah salah satu dari tiga pilar teknologi web:

- **HTML** — Struktur/konten halaman web
- **CSS** — Tampilan dan gaya halaman web
- **JavaScript** — Perilaku dan interaktivitas halaman web

■■ Fakta Menarik

JavaScript bukan Java! Meskipun namanya mirip, keduanya adalah bahasa yang sama sekali berbeda. Nama 'JavaScript' dipilih karena alasan pemasaran pada tahun 1995.

1.2 Sejarah Singkat

JavaScript diciptakan oleh Brendan Eich pada tahun 1995 hanya dalam 10 hari saat ia bekerja di Netscape. Awalnya bernama 'Mocha', lalu 'LiveScript', sebelum akhirnya menjadi 'JavaScript'. Kini JavaScript distandarisasi oleh ECMAScript (ES) dan terus berkembang pesat.

Tahun	Versi	Fitur Penting
1995	JavaScript 1.0	Lahirnya JavaScript di Netscape
1997	ES1	Standar pertama ECMAScript
2009	ES5	JSON, Array methods, strict mode
2015	ES6/ES2015	let/const, Arrow fn, Class, Promise
2017	ES8	async/await, Object.entries
2023+	ESNext	Fitur modern terus bertambah

1.3 Cara Menjalankan JavaScript

Ada beberapa cara mudah untuk menjalankan JavaScript:

- **Browser Console** — Tekan F12 di Chrome/Firefox, pilih tab 'Console'
- **File HTML** — Buat file .html dan sisipkan tag <script>
- **Node.js** — Jalankan JS di luar browser via terminal
- **Online Editor** — Gunakan CodePen, JSFiddle, atau Replit

1.4 Hello World!

Tradisi pertama dalam belajar bahasa pemrograman adalah menampilkan teks 'Hello World!'. Berikut caranya di JavaScript:

hello_world.js

```
// Cara 1: Menampilkan di Console
console.log("Hello, World!");

// Cara 2: Menampilkan di halaman web
document.write("Hello, World!");

// Cara 3: Menampilkan dialog popup
alert("Hello, World!");
```

hello_world.js

```
// Cara 1: Menampilkan di Console
console.log("Hello, World!");

// Cara 2: Menampilkan di halaman web
document.write("Hello, World!");

// Cara 3: Menampilkan dialog popup
alert("Hello, World!");
```

■ Output

```
Hello, World!
```

■ Tips Pemula

Gunakan console.log() sesering mungkin saat belajar. Ini adalah cara terbaik untuk melihat nilai variabel dan men-debug kode kamu.

BAB 2

Variabel & Tipe Data

Variabel adalah 'kotak' penyimpanan data dalam program. Di JavaScript, kita bisa menyimpan berbagai jenis data seperti angka, teks, dan lainnya ke dalam variabel.

2.1 Cara Mendeklarasikan Variabel

JavaScript memiliki tiga kata kunci untuk membuat variabel: **var**, **let**, dan **const**.

variabel.js

```
// var – cara lama (hindari di kode modern)
var nama = "Budi";

// let – untuk variabel yang bisa berubah
let umur = 25;
umur = 26; // boleh diubah

// const – untuk nilai yang tidak berubah (konstanta)
const PI = 3.14159;
// PI = 3.14; // ERROR! const tidak bisa diubah
```

variabel.js

```
// var – cara lama (hindari di kode modern)
var nama = "Budi";

// let – untuk variabel yang bisa berubah
let umur = 25;
umur = 26; // boleh diubah

// const – untuk nilai yang tidak berubah (konstanta)
const PI = 3.14159;
// PI = 3.14; // ERROR! const tidak bisa diubah
```

Keyword	Bisa Diubah?	Scope	Kapan Digunakan?
var	■ Ya	Function	Kode lama (hindari)
let	■ Ya	Block	Variabel yang berubah
const	■ Tidak	Block	Nilai tetap / konstanta

2.2 Tipe Data Primitif

JavaScript memiliki 7 tipe data primitif yang merupakan tipe data paling dasar:

tipe_data.js

```
// 1. String — teks (gunakan kutip)
let nama = "Siti Aisyah";
let kota = 'Jakarta';
let salam = `Halo, ${nama}!`; // template literal

// 2. Number — angka (bulat maupun desimal)
let usia = 20;
let tinggi = 165.5;

// 3. Boolean — true atau false
let aktif = true;
let lulus = false;

// 4. Undefined — variabel belum diberi nilai
let nilai;
console.log(nilai); // undefined

// 5. Null — sengaja dikosongkan
let data = null;

// 6. BigInt — angka sangat besar
let bigNum = 9007199254740991n;

// 7. Symbol — nilai unik (tingkat lanjut)
let id = Symbol("id");
```

tipe_data.js

```
// 1. String — teks (gunakan kutip)
let nama = "Siti Aisyah";
let kota = 'Jakarta';
let salam = `Halo, ${nama}!`; // template literal

// 2. Number — angka (bulat maupun desimal)
let usia = 20;
let tinggi = 165.5;

// 3. Boolean — true atau false
let aktif = true;
let lulus = false;

// 4. Undefined — variabel belum diberi nilai
let nilai;
console.log(nilai); // undefined

// 5. Null — sengaja dikosongkan
let data = null;
```

```
// 6. BigInt — angka sangat besar
let bigNum = 9007199254740991n;

// 7. Symbol — nilai unik (tingkat lanjut)
let id = Symbol("id");
```

2.3 typeof — Mengecek Tipe Data

Gunakan operator **typeof** untuk mengecek tipe data suatu nilai:

typeof.js

```
console.log(typeof "Hello"); // "string"
console.log(typeof 42); // "number"
console.log(typeof true); // "boolean"
console.log(typeof undefined); // "undefined"
console.log(typeof null); // "object" (quirk JS!)
console.log(typeof [1,2]); // "object"
```

typeof.js

```
console.log(typeof "Hello"); // "string"
console.log(typeof 42); // "number"
console.log(typeof true); // "boolean"
console.log(typeof undefined); // "undefined"
console.log(typeof null); // "object" (quirk JS!)
console.log(typeof [1,2]); // "object"
```

■ Output

```
string
number
boolean
undefined
object
object
```

■■ Perhatian

typeof null mengembalikan 'object' bukan 'null'. Ini adalah bug lama di JavaScript yang sengaja dipertahankan untuk kompatibilitas. Untuk mengecek null, gunakan: nilai === null

BAB 3

Operator

Operator adalah simbol yang digunakan untuk melakukan operasi pada nilai (operand). JavaScript memiliki berbagai jenis operator.

3.1 Operator Aritmatika

aritmatika.js

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

console.log(a + b); // 13 — Penjumlahan
console.log(a - b); // 7 — Pengurangan
console.log(a * b); // 30 — Perkalian
console.log(a / b); // 3.33— Pembagian
console.log(a % b); // 1 — Modulus (sisa bagi)
console.log(a ** b); // 1000— Pangkat (10^3)

// Increment & Decrement
let x = 5;
x++; // x menjadi 6 (tambah 1)
x--; // x menjadi 5 lagi (kurang 1)
```

aritmatika.js

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

console.log(a + b); // 13 — Penjumlahan
console.log(a - b); // 7 — Pengurangan
console.log(a * b); // 30 — Perkalian
console.log(a / b); // 3.33— Pembagian
console.log(a % b); // 1 — Modulus (sisa bagi)
console.log(a ** b); // 1000— Pangkat (10^3)

// Increment & Decrement
let x = 5;
x++; // x menjadi 6 (tambah 1)
x--; // x menjadi 5 lagi (kurang 1)
```

3.2 Operator Perbandingan

Operator perbandingan menghasilkan nilai **boolean** (true/false):

perbandingan.js

```
let x = 5, y = "5";

console.log(x == y); // true — sama nilai (tipe diabaikan)
console.log(x === y); // false — sama nilai DAN tipe (strict)
console.log(x != y); // false
console.log(x !== y); // true — tidak sama (strict)
console.log(x > 3); // true — lebih dari
console.log(x < 3); // false — kurang dari
console.log(x >= 5); // true — lebih dari atau sama
console.log(x <= 4); // false — kurang dari atau sama
```

perbandingan.js

```
let x = 5, y = "5";

console.log(x == y); // true — sama nilai (tipe diabaikan)
console.log(x === y); // false — sama nilai DAN tipe (strict)
console.log(x != y); // false
console.log(x !== y); // true — tidak sama (strict)
console.log(x > 3); // true — lebih dari
console.log(x < 3); // false — kurang dari
console.log(x >= 5); // true — lebih dari atau sama
console.log(x <= 4); // false — kurang dari atau sama
```

■ Selalu Gunakan === (Strict Equality)

Hindari == karena bisa menyebabkan hasil yang tidak terduga akibat type coercion. Biasakan menggunakan === untuk perbandingan yang aman dan tepat.

3.3 Operator Logika

logika.js

```
// && (AND) — true jika KEDUA kondisi true
console.log(true && true); // true
console.log(true && false); // false

// || (OR) — true jika SALAH SATU kondisi true
console.log(false || true); // true
console.log(false || false); // false

// ! (NOT) — membalik nilai boolean
console.log(!true); // false
console.log(!false); // true

// Contoh nyata:
```

```
let usia = 20, punyaKTP = true;
let bisaDaftar = usia >= 17 && punyaKTP;
console.log(bisaDaftar); // true
```

logika.js

```
// && (AND) — true jika KEDUA kondisi true
console.log(true && true); // true
console.log(true && false); // false

// || (OR) — true jika SALAH SATU kondisi true
console.log(false || true); // true
console.log(false || false); // false

// ! (NOT) — membalik nilai boolean
console.log(!true); // false
console.log(!false); // true

// Contoh nyata:
let usia = 20, punyaKTP = true;
let bisaDaftar = usia >= 17 && punyaKTP;
console.log(bisaDaftar); // true
```

■ Output

```
true
false
true
false
false
true
true
```

BAB 4

Struktur Kontrol

Struktur kontrol memungkinkan program mengambil keputusan berdasarkan kondisi tertentu. Seperti memilih jalan di persimpangan — program akan menjalankan kode yang berbeda tergantung kondisi.

4.1 Pernyataan if / else if / else

if_else.js

```
let nilai = 78;

if (nilai >= 90) {
console.log("Grade A – Sangat Baik!");
} else if (nilai >= 80) {
console.log("Grade B – Baik");
} else if (nilai >= 70) {
console.log("Grade C – Cukup");
} else {
console.log("Grade D – Perlu Belajar Lebih Giat");
}
```

if_else.js

```
let nilai = 78;

if (nilai >= 90) {
console.log("Grade A – Sangat Baik!");
} else if (nilai >= 80) {
console.log("Grade B – Baik");
} else if (nilai >= 70) {
console.log("Grade C – Cukup");
} else {
console.log("Grade D – Perlu Belajar Lebih Giat");
}
```

■ Output

```
Grade C – Cukup
```

4.2 Switch-Case

Switch digunakan ketika ada banyak kemungkinan nilai yang perlu diperiksa:

switch.js

```
let hari = "Senin";

switch (hari) {
case "Senin":
console.log("Awal pekan, semangat!");
break;
case "Jumat":
console.log("Hampir weekend!");
break;
case "Sabtu":
case "Minggu":
console.log("Selamat liburan!");
break;
default:
console.log("Hari kerja biasa");
}
```

switch.js

```
let hari = "Senin";

switch (hari) {
case "Senin":
console.log("Awal pekan, semangat!");
break;
case "Jumat":
console.log("Hampir weekend!");
break;
case "Sabtu":
case "Minggu":
console.log("Selamat liburan!");
break;
default:
console.log("Hari kerja biasa");
}
```

■ Output

```
Awal pekan, semangat!
```

4.3 Operator Ternary

Ternary adalah cara singkat menulis if-else dalam satu baris:

ternary.js

```
// Sintaks: kondisi ? nilaiJikaTrue : nilaiJikaFalse

let usia = 20;

let status = usia >= 18 ? "Dewasa" : "Anak-anak";

console.log(status); // "Dewasa"

// Setara dengan:

let status2;

if (usia >= 18) { status2 = "Dewasa"; }

else { status2 = "Anak-anak"; }
```

ternary.js

```
// Sintaks: kondisi ? nilaiJikaTrue : nilaiJikaFalse

let usia = 20;

let status = usia >= 18 ? "Dewasa" : "Anak-anak";

console.log(status); // "Dewasa"

// Setara dengan:

let status2;

if (usia >= 18) { status2 = "Dewasa"; }

else { status2 = "Anak-anak"; }
```

■ Output

```
Dewasa
```

BAB 5

Perulangan (Loop)

Loop digunakan untuk menjalankan blok kode berulang kali. Bayangkan kamu harus mencetak angka 1-100 — tanpa loop kamu harus menulis 100 baris kode!

5.1 for Loop

for loop digunakan ketika kamu tahu berapa kali perulangan harus berjalan:

for_loop.js

```
// Sintaks: for (inisialisasi; kondisi; update) { ... }

// Cetak angka 1 sampai 5
for (let i = 1; i <= 5; i++) {
console.log(`Angka: ${i}`);
}

// Loop mundur
for (let i = 5; i >= 1; i--) {
console.log(i);
}
```

for_loop.js

```
// Sintaks: for (inisialisasi; kondisi; update) { ... }

// Cetak angka 1 sampai 5
for (let i = 1; i <= 5; i++) {
console.log(`Angka: ${i}`);
}

// Loop mundur
for (let i = 5; i >= 1; i--) {
console.log(i);
}
```

Output

```
Angka: 1
Angka: 2
Angka: 3
Angka: 4
Angka: 5
```

5.2 while & do…while

while.js

```
// while — cek kondisi SEBELUM eksekusi
let angka = 1;
while (angka <= 3) {
console.log("While: " + angka);
angka++;
}
// do...while — eksekusi MINIMAL SEKALI dulu
let n = 10;
do {
console.log("do-while: " + n); // tetap jalan meski n=10 > 3
n++;
} while (n <= 3);
```

while.js

```
// while — cek kondisi SEBELUM eksekusi
let angka = 1;
while (angka <= 3) {
console.log("While: " + angka);
angka++;
}
// do...while — eksekusi MINIMAL SEKALI dulu
let n = 10;
do {
console.log("do-while: " + n); // tetap jalan meski n=10 > 3
n++;
} while (n <= 3);
```

■ Output

```
While: 1
While: 2
While: 3
do-while: 10
```

5.3 for…of & for…in

for_of_in.js

```
// for...of — iterasi nilai dalam Array/String
```

```
const buah = ["Apel", "Mangga", "Pisang"];
for (const item of buah) {
console.log(item);
}
// for...in — iterasi KEY pada Object
const siswa = { nama: "Andi", umur: 17, kelas: "XI" };
for (const key in siswa) {
console.log(`${key}: ${siswa[key]}`);
}
```

for_of_in.js

```
// for...of — iterasi nilai dalam Array/String
const buah = ["Apel", "Mangga", "Pisang"];
for (const item of buah) {
console.log(item);
}
// for...in — iterasi KEY pada Object
const siswa = { nama: "Andi", umur: 17, kelas: "XI" };
for (const key in siswa) {
console.log(`${key}: ${siswa[key]}`);
}
```

Output

```
Apel
Mangga
Pisang
nama: Andi
umur: 17
kelas: XI
```

5.4 break & continue

break_continue.js

```
// break — keluar dari loop sepenuhnya
for (let i = 1; i <= 10; i++) {
if (i === 4) break; // berhenti di angka 4
console.log(i);
}
// Output: 1, 2, 3
// continue — skip iterasi saat ini, lanjut ke berikutnya
```

```
for (let i = 1; i <= 5; i++) {
if (i === 3) continue; // skip angka 3
console.log(i);
}
// Output: 1, 2, 4, 5
```

break_continue.js

```
// break — keluar dari loop sepenuhnya
for (let i = 1; i <= 10; i++) {
if (i === 4) break; // berhenti di angka 4
console.log(i);
}
// Output: 1, 2, 3

// continue — skip iterasi saat ini, lanjut ke berikutnya
for (let i = 1; i <= 5; i++) {
if (i === 3) continue; // skip angka 3
console.log(i);
}
// Output: 1, 2, 4, 5
```

BAB 6

Fungsi (Function)

Fungsi adalah blok kode yang dirancang untuk menjalankan tugas tertentu dan bisa dipanggil berulang kali. Fungsi membuat kode lebih terorganisir, dapat digunakan ulang, dan mudah dipelihara.

6.1 Deklarasi Fungsi

fungsi.js

```
// Mendefinisikan fungsi
function sapa(nama) {
console.log(`Halo, ${nama}! Selamat datang.`);
}
// Memanggil fungsi
sapa("Dewi");
sapa("Raka");
// Fungsi dengan nilai kembalian (return)
function tambah(a, b) {
return a + b;
}
let hasil = tambah(5, 3);
console.log("Hasil: " + hasil); // Hasil: 8
```

fungsi.js

```
// Mendefinisikan fungsi
function sapa(nama) {
console.log(`Halo, ${nama}! Selamat datang.`);
}
// Memanggil fungsi
sapa("Dewi");
sapa("Raka");
// Fungsi dengan nilai kembalian (return)
function tambah(a, b) {
return a + b;
}
let hasil = tambah(5, 3);
console.log("Hasil: " + hasil); // Hasil: 8
```

■ Output

```
Halo, Dewi! Selamat datang.
Halo, Raka! Selamat datang.
Hasil: 8
```

6.2 Arrow Function (ES6)

Arrow function adalah cara singkat menulis fungsi yang diperkenalkan di ES6:

arrow_function.js

```
// Fungsi biasa
function kali(a, b) { return a * b; }

// Arrow function — setara dengan di atas
const kali = (a, b) => a * b;

// Arrow function dengan satu parameter (tanpa kurung)
const kuadrat = x => x * x;
console.log(kuadrat(4)); // 16

// Arrow function dengan body blok (pakai return eksplisit)
const greet = (nama) => {
const pesan = `Selamat pagi, ${nama}!`;
return pesan;
};
console.log(greet("Budi"));
```

arrow_function.js

```
// Fungsi biasa
function kali(a, b) { return a * b; }

// Arrow function — setara dengan di atas
const kali = (a, b) => a * b;

// Arrow function dengan satu parameter (tanpa kurung)
const kuadrat = x => x * x;
console.log(kuadrat(4)); // 16

// Arrow function dengan body blok (pakai return eksplisit)
const greet = (nama) => {
const pesan = `Selamat pagi, ${nama}!`;
return pesan;
};
console.log(greet("Budi"));
```

■ Output

```
16
Selamat pagi, Budi!
```

6.3 Parameter Default & Rest

parameter.js

```
// Parameter default
function perkenalan(nama, kota = "Indonesia") {
console.log(`${nama} dari ${kota}`);
}
perkenalan("Ayu"); // Ayu dari Indonesia
perkenalan("Budi", "Bandung"); // Budi dari Bandung

// Rest parameter (...args) — menerima banyak argumen
function jumlahkan(...angka) {
return angka.reduce((total, n) => total + n, 0);
}
console.log(jumlahkan(1,2,3,4,5)); // 15
```

parameter.js

```
// Parameter default
function perkenalan(nama, kota = "Indonesia") {
console.log(`${nama} dari ${kota}`);
}
perkenalan("Ayu"); // Ayu dari Indonesia
perkenalan("Budi", "Bandung"); // Budi dari Bandung

// Rest parameter (...args) — menerima banyak argumen
function jumlahkan(...angka) {
return angka.reduce((total, n) => total + n, 0);
}
console.log(jumlahkan(1,2,3,4,5)); // 15
```

■ Output

```
Ayu dari Indonesia
Budi dari Bandung
15
```

BAB 7

Array

Array adalah struktur data yang menyimpan banyak nilai dalam satu variabel. Bayangkan array seperti rak buku — setiap buku punya posisi (index) mulai dari 0.

7.1 Membuat & Mengakses Array

array_dasar.js

```
// Membuat array
const buah = ["Apel", "Manga", "Pisang", "Jeruk"];
// index: 0 1 2 3

// Mengakses elemen (index mulai dari 0)
console.log(buah[0]); // "Apel"
console.log(buah[2]); // "Pisang"

// Jumlah elemen
console.log(buah.length); // 4

// Mengubah nilai
buah[1] = "Mangga";
console.log(buah); // ["Apel","Mangga","Pisang","Jeruk"]
```

array_dasar.js

```
// Membuat array
const buah = ["Apel", "Manga", "Pisang", "Jeruk"];
// index: 0 1 2 3

// Mengakses elemen (index mulai dari 0)
console.log(buah[0]); // "Apel"
console.log(buah[2]); // "Pisang"

// Jumlah elemen
console.log(buah.length); // 4

// Mengubah nilai
buah[1] = "Mangga";
console.log(buah); // ["Apel","Mangga","Pisang","Jeruk"]
```

■ Output

```
Apel
Pisang
4
```

```
["Apel","Mangga","Pisang","Jeruk"]
```

7.2 Method Array Penting

array_methods.js

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

// push – tambah di akhir
arr.push(4); // [1,2,3,4]

// pop – hapus dari akhir
arr.pop(); // [1,2,3]

// unshift – tambah di awal
arr.unshift(0); // [0,1,2,3]

// shift – hapus dari awal
arr.shift(); // [1,2,3]

// indexOf – cari posisi elemen
console.log(arr.indexOf(2)); // 1

// includes – cek apakah ada
console.log(arr.includes(5)); // false

// join – gabung jadi string
console.log(arr.join(" - ")); // "1 - 2 - 3"
```

array_methods.js

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

// push – tambah di akhir
arr.push(4); // [1,2,3,4]

// pop – hapus dari akhir
arr.pop(); // [1,2,3]

// unshift – tambah di awal
arr.unshift(0); // [0,1,2,3]

// shift – hapus dari awal
arr.shift(); // [1,2,3]

// indexOf – cari posisi elemen
console.log(arr.indexOf(2)); // 1

// includes – cek apakah ada
console.log(arr.includes(5)); // false

// join – gabung jadi string
console.log(arr.join(" - ")); // "1 - 2 - 3"
```

7.3 map, filter, reduce

Tiga method ini sangat powerful untuk mengolah data dalam array:

map_filter_reduce.js

```
const angka = [1, 2, 3, 4, 5];

// map — transformasi setiap elemen
const dikali2 = angka.map(x => x * 2);
console.log(dikali2); // [2, 4, 6, 8, 10]

// filter — ambil elemen yang memenuhi syarat
const genap = angka.filter(x => x % 2 === 0);
console.log(genap); // [2, 4]

// reduce — akumulasi nilai
const total = angka.reduce((acc, x) => acc + x, 0);
console.log(total); // 15
```

map_filter_reduce.js

```
const angka = [1, 2, 3, 4, 5];

// map — transformasi setiap elemen
const dikali2 = angka.map(x => x * 2);
console.log(dikali2); // [2, 4, 6, 8, 10]

// filter — ambil elemen yang memenuhi syarat
const genap = angka.filter(x => x % 2 === 0);
console.log(genap); // [2, 4]

// reduce — akumulasi nilai
const total = angka.reduce((acc, x) => acc + x, 0);
console.log(total); // 15
```

■ Output

```
[2, 4, 6, 8, 10]
[2, 4]
15
```

BAB 8

Object

Object adalah kumpulan pasangan key-value (properti). Jika array seperti daftar, object lebih seperti formulir data — setiap informasi punya label/nama yang jelas.

8.1 Membuat & Mengakses Object

object.js

```
// Membuat object
const mahasiswa = {
nama: "Rizky",
nim: "202312345",
ipk: 3.75,
aktif: true,
// Method (fungsi dalam object)
perkenalan() {
console.log(`Nama saya ${this.nama}, IPK: ${this.ipk}`);
}
};

// Dot notation
console.log(mahasiswa.nama); // "Rizky"

// Bracket notation
console.log(mahasiswa["ipk"]); // 3.75

// Memanggil method
mahasiswa.perkenalan();
```

object.js

```
// Membuat object
const mahasiswa = {
nama: "Rizky",
nim: "202312345",
ipk: 3.75,
aktif: true,
// Method (fungsi dalam object)
perkenalan() {
console.log(`Nama saya ${this.nama}, IPK: ${this.ipk}`);
}
```

```
};

// Dot notation
console.log(mahasiswa.nama); // "Rizky"

// Bracket notation
console.log(mahasiswa["ipk"]); // 3.75

// Memanggil method
mahasiswa.perkenalan();
```

Output

```
Rizky
3.75
Nama saya Rizky, IPK: 3.75
```

8.2 Destructuring Object & Array

Destructuring memungkinkan kamu 'membongkar' nilai dari object/array ke variabel terpisah secara elegan:

destructuring.js

```
// Object Destructuring
const { nama, ipk } = mahasiswa;
console.log(nama, ipk); // "Rizky" 3.75

// Destructuring dengan rename
const { nama: studentName, nim: id } = mahasiswa;
console.log(studentName); // "Rizky"

// Array Destructuring
const warna = ["merah", "hijau", "biru"];
const [pertama, kedua, ketiga] = warna;
console.log(pertama); // "merah"
```

destructuring.js

```
// Object Destructuring
const { nama, ipk } = mahasiswa;
console.log(nama, ipk); // "Rizky" 3.75

// Destructuring dengan rename
const { nama: studentName, nim: id } = mahasiswa;
console.log(studentName); // "Rizky"

// Array Destructuring
const warna = ["merah", "hijau", "biru"];
const [pertama, kedua, ketiga] = warna;
```

```
console.log(pertama); // "merah"
```

8.3 Spread Operator (...)

spread.js

```
// Spread pada Array — menyalin/menggabung
const a = [1, 2, 3];
const b = [4, 5];
const c = [...a, ...b]; // [1,2,3,4,5]
const salinan = [...a]; // [1,2,3] — salinan baru

// Spread pada Object — menyalin/menggabung
const info = { kota: "Surabaya", jurusan: "TI" };
const profil = { ...mahasiswa, ...info };
console.log(profil.kota); // "Surabaya"
console.log(profil.nama); // "Rizky"
```

spread.js

```
// Spread pada Array — menyalin/menggabung
const a = [1, 2, 3];
const b = [4, 5];
const c = [...a, ...b]; // [1,2,3,4,5]
const salinan = [...a]; // [1,2,3] — salinan baru

// Spread pada Object — menyalin/menggabung
const info = { kota: "Surabaya", jurusan: "TI" };
const profil = { ...mahasiswa, ...info };
console.log(profil.kota); // "Surabaya"
console.log(profil.nama); // "Rizky"
```

BAB 9

Manipulasi DOM

DOM (Document Object Model) adalah representasi struktur HTML sebagai objek di JavaScript. Dengan DOM, kamu bisa mengubah konten, style, dan struktur halaman web secara dinamis menggunakan JavaScript.

■■ Apa itu DOM?

DOM adalah pohon (tree) objek yang merepresentasikan dokumen HTML. Setiap tag HTML menjadi 'node' yang bisa diakses dan dimanipulasi melalui JavaScript.

9.1 Memilih Elemen HTML

dom_selector.js

```
// Pilih 1 elemen berdasarkan ID
const judul = document.getElementById("judul");

// Pilih 1 elemen dengan CSS selector
const tombol = document.querySelector(".btn-primary");
const input = document.querySelector("#username");

// Pilih SEMUA elemen (NodeList)
const semua = document.querySelectorAll(".card");
semua.forEach(el => console.log(el.textContent));

// Pilih berdasarkan tag
const paragraf = document.getElementsByTagName("p");
```

dom_selector.js

```
// Pilih 1 elemen berdasarkan ID
const judul = document.getElementById("judul");

// Pilih 1 elemen dengan CSS selector
const tombol = document.querySelector(".btn-primary");
const input = document.querySelector("#username");

// Pilih SEMUA elemen (NodeList)
const semua = document.querySelectorAll(".card");
semua.forEach(el => console.log(el.textContent));

// Pilih berdasarkan tag
const paragraf = document.getElementsByTagName("p");
```

9.2 Mengubah Konten & Style

dom_manipulasi.js

```
// Mengubah teks
judul.textContent = "Judul Baru";

// Mengubah HTML (hati-hati XSS!)
judul.innerHTML = "<span>Judul <b>Bold</b></span>";

// Mengubah style langsung
judul.style.color = "#F7A41D";
judul.style.fontSize = "2rem";

// Mengelola class CSS
judul.classList.add("aktif"); // tambah class
judul.classList.remove("lama"); // hapus class
judul.classList.toggle("show"); // toggle class
judul.classList.contains("aktif"); // cek class → true

// Mengubah atribut
input.setAttribute("placeholder", "Masukkan nama...");
console.log(input.getAttribute("type"));
```

dom_manipulasi.js

```
// Mengubah teks
judul.textContent = "Judul Baru";

// Mengubah HTML (hati-hati XSS!)
judul.innerHTML = "<span>Judul <b>Bold</b></span>";

// Mengubah style langsung
judul.style.color = "#F7A41D";
judul.style.fontSize = "2rem";

// Mengelola class CSS
judul.classList.add("aktif"); // tambah class
judul.classList.remove("lama"); // hapus class
judul.classList.toggle("show"); // toggle class
judul.classList.contains("aktif"); // cek class → true

// Mengubah atribut
input.setAttribute("placeholder", "Masukkan nama...");
console.log(input.getAttribute("type"));
```

9.3 Membuat & Menghapus Elemen

dom_create.js

```
// Membuat elemen baru
const li = document.createElement("li");
li.textContent = "Item Baru";
li.classList.add("list-item");

// Tambahkan ke DOM
const ul = document.querySelector("ul");
ul.appendChild(li); // tambah di akhir
ul.prepend(li); // tambah di awal

// Hapus elemen
li.remove(); // hapus elemen
ul.removeChild(ul.lastChild); // hapus child
```

dom_create.js


```
// Membuat elemen baru
const li = document.createElement("li");
li.textContent = "Item Baru";
li.classList.add("list-item");

// Tambahkan ke DOM
const ul = document.querySelector("ul");
ul.appendChild(li); // tambah di akhir
ul.prepend(li); // tambah di awal

// Hapus elemen
li.remove(); // hapus elemen
ul.removeChild(ul.lastChild); // hapus child
```

BAB 10

Event & Interaktivitas

Event adalah kejadian yang terjadi di halaman web, seperti klik tombol, input teks, atau scroll halaman. JavaScript dapat 'mendengarkan' event ini dan merespons dengan menjalankan fungsi tertentu.

10.1 addEventListener

event_listener.js

```
// Sintaks: element.addEventListener("event", fungsi)
const tombol = document.getElementById("myBtn");
// Klik biasa
tombol.addEventListener("click", function() {
alert("Tombol diklik!");
});
// Dengan arrow function + Event Object
tombol.addEventListener("click", (e) => {
console.log("Target:", e.target); // elemen yang diklik
console.log("Tipe:", e.type); // "click"
});
```

event_listener.js

```
// Sintaks: element.addEventListener("event", fungsi)
const tombol = document.getElementById("myBtn");
// Klik biasa
tombol.addEventListener("click", function() {
alert("Tombol diklik!");
});
// Dengan arrow function + Event Object
tombol.addEventListener("click", (e) => {
console.log("Target:", e.target); // elemen yang diklik
console.log("Tipe:", e.type); // "click"
});
```


10.2 Event yang Sering Digunakan

Event	Kapan Terjadi
`click`	Elemen diklik
`dblclick`	Elemen double-klik
`mouseover`	Mouse masuk ke area elemen
`keydown`	Tombol keyboard ditekan
`keyup`	Tombol keyboard dilepas
`input`	Input field diubah (real-time)
`change`	Input selesai diubah
`submit`	Form dikirim
`load`	Halaman/resource selesai dimuat
`DOMContentLoaded`	HTML selesai di-parse
`scroll`	Halaman di-scroll
`resize`	Ukuran window berubah

10.3 Proyek Mini: Kalkulator Sederhana

Saatnya menggabungkan semua yang telah dipelajari! Berikut kalkulator sederhana menggunakan HTML + JavaScript:

kalkulator.html

```
<!-- HTML -->
<input id="angka1" type="number" placeholder="Angka 1">
<input id="angka2" type="number" placeholder="Angka 2">
<select id="operator">
<option value="+">+</option>
<option value="-">-</option>
<option value="*">×</option>
<option value="/">÷</option>
</select>
<button id="hitung">Hitung</button>
<p id="hasil"></p>
```

kalkulator.html

```
<!-- HTML -->
<input id="angka1" type="number" placeholder="Angka 1">
<input id="angka2" type="number" placeholder="Angka 2">
```

```
<select id="operator">
<option value="+">+</option>
<option value="-">-</option>
<option value="*">×</option>
<option value="/">÷</option>
</select>
<button id="hitung">Hitung</button>
<p id="hasil"></p>
```

kalkulator.js

```
// JavaScript
document.getElementById("hitung")
.addEventListener("click", () => {
const a = parseFloat(document.getElementById("angka1").value);
const b = parseFloat(document.getElementById("angka2").value);
const op = document.getElementById("operator").value;

let hasil;
if (op === "+") hasil = a + b;
else if (op === "-") hasil = a - b;
else if (op === "*") hasil = a * b;
else hasil = b !== 0 ? a / b : "Error: bagi nol";

document.getElementById("hasil").textContent =
`Hasil: ${a} ${op} ${b} = ${hasil}`;
});
```

kalkulator.js

```
// JavaScript
document.getElementById("hitung")
.addEventListener("click", () => {
const a = parseFloat(document.getElementById("angka1").value);
const b = parseFloat(document.getElementById("angka2").value);
const op = document.getElementById("operator").value;

let hasil;
if (op === "+") hasil = a + b;
else if (op === "-") hasil = a - b;
else if (op === "*") hasil = a * b;
else hasil = b !== 0 ? a / b : "Error: bagi nol";

document.getElementById("hasil").textContent =
`Hasil: ${a} ${op} ${b} = ${hasil}`;
});
```

■ Langkah Selanjutnya

Setelah menguasai dasar-dasar di buku ini, kamu bisa lanjutkan belajar: Async/Await & Fetch API, ES Modules, Framework (React/Vue), Node.js untuk backend, dan penanganan error dengan try/catch.

Selamat! Kamu telah menyelesaikan e-book ini. Kunci sukses belajar programming adalah **konsistensi dan praktik**. Buat proyek kecilmu sendiri, eksplorasi, dan jangan takut salah — error adalah teman terbaik seorang programmer! ■

Sumber Belajar Lanjutan
• MDN Web Docs — developer.mozilla.org
• JavaScript.info — javascript.info
• freeCodeCamp — freecodecamp.org
• Eloquent JavaScript — eloquentjavascript.net (Gratis!)