MODUL

WEB PROGRAMMER (REACT JS)

Silabus

Berikut ini silabus Mobile Apps Development menggunakan Flutter:

1. **JavaScript Programming**
2. **Dasar-dasar React**
3. **React Component**
4. **Bekerja dengan REST**
5. **React Router**
6. **Mini Project**
7. **Case Study**

JavaScript Programming

- Pengenalan JavaScript
- Variabel dan Tipe Data
- Operator
- Struktur Kendali
- Array dan method Array: map, filter, reduce, forEach
- Fungsi
- Class dan Object
- Asynchronous JavaScript
- Exception Handling

JavaScript Programming

- JavaScript adalah bahasa pemrograman yang umumnya digunakan di dalam pengembangan web untuk membuat halaman web menjadi interaktif dan dinamis. Dengan JavaScript, Anda dapat membuat fitur seperti:
  - Validasi formulir pada browser (misalnya mengecek input sebelum dikirim).
  - Animasi pada elemen (misalnya efek hover atau slideshow).
  - Menangani peristiwa seperti klik, input, dan hover.
  - Manipulasi halaman HTML dan CSS (Document Object Model atau DOM).
- JavaScript bekerja baik di sisi klien (browser) maupun di sisi server (Node.js). Di browser, JavaScript mengendalikan bagaimana halaman merespons dan berinteraksi dengan pengguna, sedangkan di server (dengan Node.js), JavaScript bisa digunakan untuk menangani permintaan jaringan, akses database, dan logika bisnis.

JavaScript Programming

- Ketika halaman web dimuat, browser membaca dan mengeksekusi kode JavaScript. Setiap halaman HTML dapat menyertakan JavaScript dengan dua cara:
- Internal: Menyisipkan JavaScript langsung di dalam file HTML menggunakan tag `<script>`.

internal.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh JavaScript</title>
  </head>
  <body>
    <h1>Hello, JavaScript!</h1>
    <script>
      console.log("Kode JavaScript ini berjalan di browser!");
    </script>
  </body>
</html>
```

JavaScript Programming

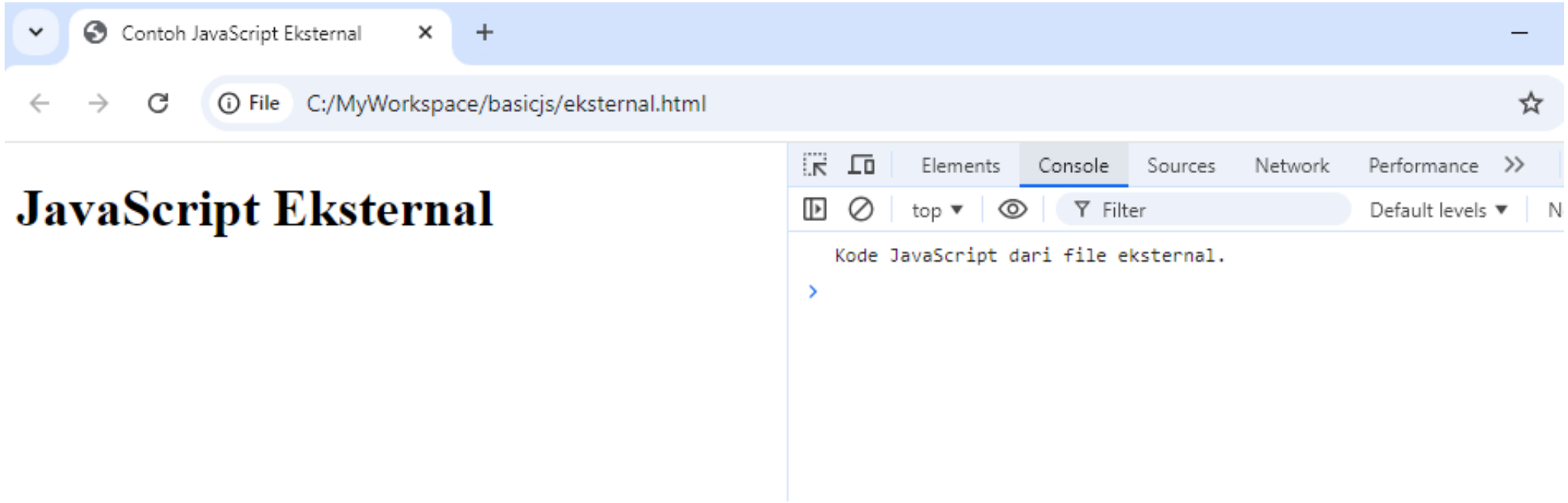
- Eksternal: Menggunakan file JavaScript eksternal (.js) yang dihubungkan ke file HTML.

eksternal.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh JavaScript Eksternal</title>
    <script src="script.js"></script>
  </head>
  <body>
    <h1>JavaScript Eksternal</h1>
  </body>
</html>
```

Debuging JavaScript

- Buka menggunakan web browser (Chrome/Edge/Firefox)
- Untuk Chrome tekan CTRL + SHIF + J



JavaScript Programming

Aturan Penulisan Kode JavaScript

- JavaScript case-sensitive atau peka terhadap huruf besar/kecil. Artinya, variabel `myVariable` dan `MyVariable` dianggap sebagai dua variabel yang berbeda. Nama variabel dan fungsi sebaiknya ditulis dengan konsisten.
- Variabel dan Fungsi: Gunakan camelCase untuk nama variabel dan fungsi. Ini berarti kata pertama ditulis dengan huruf kecil, dan kata-kata selanjutnya ditulis dengan huruf kapital di awal.
- Class: Gunakan PascalCase untuk nama kelas, yaitu setiap kata dimulai dengan huruf kapital. Contoh: MyClass
- Setiap pernyataan sebaiknya diakhiri dengan titik koma (`;`).

JavaScript Programming

Reserved word (kata-kata yang sudah didefinisikan oleh bahasa pemrograman dan tidak dapat diubah dan digunakan sebagai nama variabel)

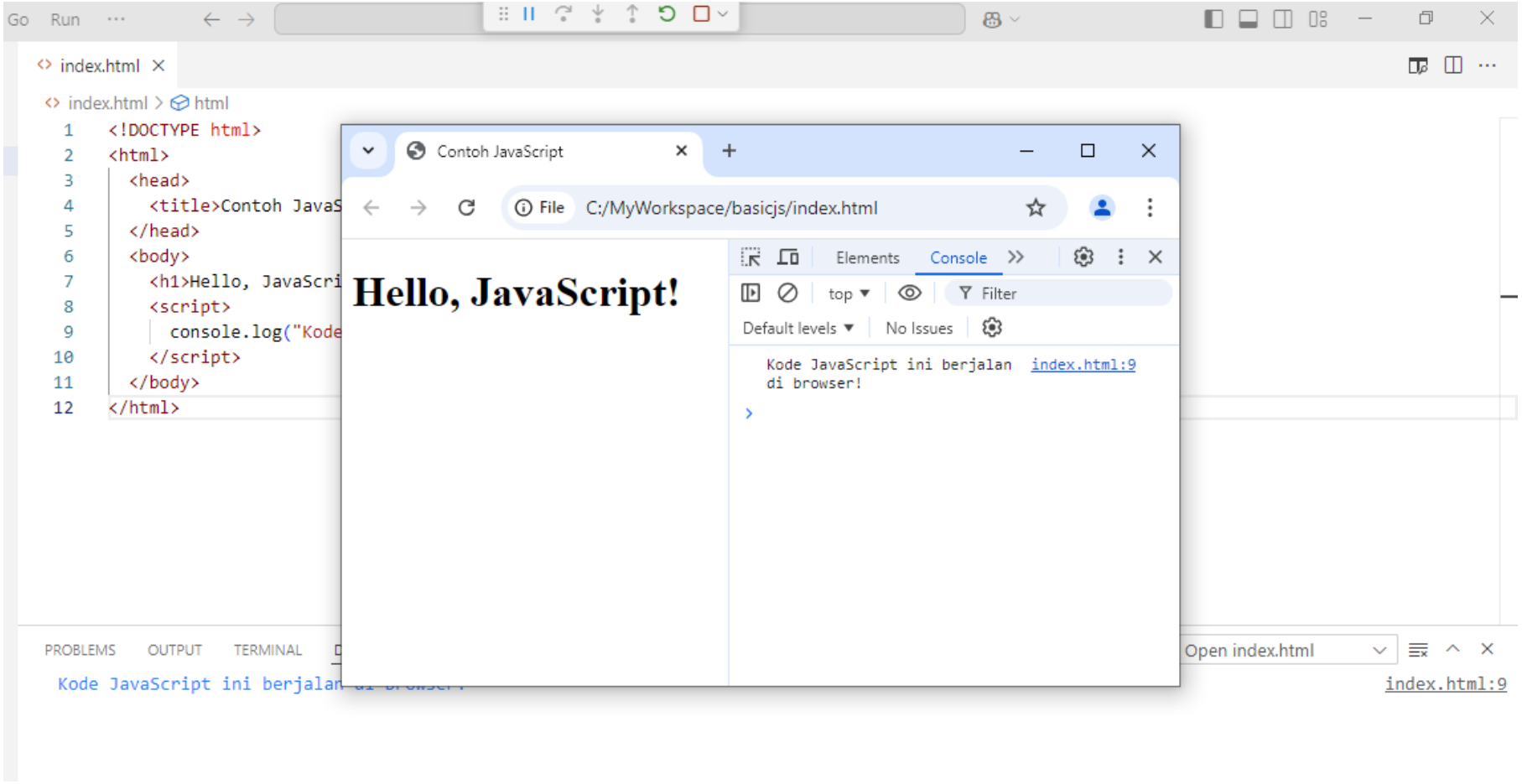
https://www.instagram.com/coding_dev_/p/CkVdMU5rH9l/

Praktik Membuat project

1. Buat Folder misal di D:\belajarjs
2. Buka Visual Studio Code, File -> Open Folder
3. Buat file, File -> New File, ketik index.html
4. Ketik kode dasar HTML
5. Run di VSCode, Run -> Run without debuging -> Chrome (atau buka langsung file dengan browser)

Praktik

Tampilan index.html



Variabel dan Tipe Data

- Di JavaScript, tipe data dapat dikelompokkan ke dalam dua kategori utama: **Primitive Types** dan **Non-Primitive Types** (Object Types).
- Berbeda dengan beberapa bahasa lain, JavaScript tidak memiliki tipe khusus untuk `int` atau `double`. Sebaliknya, ia menggunakan tipe Number untuk semua angka, baik bilangan bulat maupun desimal.
- JavaScript juga memiliki beberapa tipe data lainnya seperti String, Boolean, dan tipe Object yang mencakup berbagai struktur data seperti Array (mirip dengan List), Set, dan Map.

Variabel dan Tipe Data

1. **Number (Integer dan Float/Double)**

   JavaScript menggunakan tipe data `Number` untuk mewakili semua angka, baik bilangan bulat (integer) maupun angka desimal (float/double).

```
// Bilangan bulat (Integer)
let age = 25;
console.log(age); // Output: 25

// Bilangan desimal (Float/Double)
let price = 99.99;
console.log(price); // Output: 99.99
```

Variabel dan Tipe Data

2. String

`String` adalah tipe data yang digunakan untuk menyimpan teks. String dapat diapit dengan tanda kutip tunggal (`'...'`), tanda kutip ganda (`"..."`), atau template literal (`` `...` ``) untuk menyisipkan variabel.

```
// Menggunakan tanda kutip tunggal atau ganda
let firstName = 'John';
let lastName = "Doe";

// Menggunakan template literal
let fullName = `${firstName} ${lastName}`;
console.log(fullName); // Output: John Doe
```

Variabel dan Tipe Data

3. Boolean

`Boolean` adalah tipe data yang hanya memiliki dua nilai: `true` atau `false`. Tipe ini sering digunakan untuk kondisi logika.

```
let isActive = true;
let isLoggedIn = false;

console.log(isActive);   // Output: true
console.log(isLoggedIn); // Output: false
```

Variabel dan Tipe Data

4. **Undefined**

   Variabel yang dideklarasikan tapi belum diinisialisasi akan memiliki nilai `undefined`.

```
let notAssigned;
console.log(notAssigned); // Output: undefined
```

5. **Null**

   `Null` adalah tipe data yang secara eksplisit menunjukkan bahwa variabel tidak memiliki nilai.

```
let emptyValue = null;
console.log(emptyValue); // Output: null
```

Variabel dan Tipe Data

6. Dynamic Typing (Variabel Tanpa Tipe Tetap)

JavaScript adalah dynamically typed language, yang berarti bahwa variabel dapat menyimpan berbagai tipe data selama program berjalan. Dengan kata lain, tipe data dari variabel dapat berubah.

```
let dynamicVariable = 5;          // Number
dynamicVariable = "Hello";        // String
dynamicVariable = true;           // Boolean
console.log(dynamicVariable);     // Output: true
```

Variabel dan Tipe Data

7. Var, Let, dan Const

JavaScript menyediakan tiga cara untuk mendeklarasikan variabel, yaitu `var`, `let`, dan `const`.

- var: Deklarasi variabel lama yang memiliki cakupan fungsi (function-scoped).
- let: Variabel yang memiliki cakupan blok (block-scoped), ideal untuk variabel yang bisa diubah.
- const: Variabel konstan yang tidak bisa diubah setelah dideklarasikan.

```
var oldVar = "Ini var lama";
let myVariable = "Ini let baru";
const constantVariable = "Ini const";

console.log(oldVar);
console.log(myVariable);
console.log(constantVariable);
```

Variabel dan Tipe Data

8. Array (Mirip dengan List)

Di JavaScript, Array digunakan untuk menyimpan kumpulan data yang berurutan. Tipe data ini mirip dengan List di bahasa pemrograman lain.

```
let fruits = ["apple", "banana", "orange"];
console.log(fruits[0]); // Output: apple

// Menambahkan elemen
fruits.push("mango");
console.log(fruits); // Output: ["apple", "banana", "orange", "mango"]
```

Variabel dan Tipe Data

9. Set

Set adalah struktur data yang mirip dengan Array tetapi hanya menyimpan nilai unik (tidak ada duplikat).

```
let uniqueNumbers = new Set([1, 2, 3, 3, 4]);
console.log(uniqueNumbers); // Output: Set {1, 2, 3, 4}

// Menambahkan elemen
uniqueNumbers.add(5);
console.log(uniqueNumbers); // Output: Set {1, 2, 3, 4, 5}
```

Variabel dan Tipe Data

10. Map

Map adalah struktur data yang menyimpan pasangan kunci-nilai (key-value pairs). Berbeda dengan objek biasa, Map mempertahankan urutan elemen saat ditambahkan.

```
let user = new Map();
user.set("name", "Alice");
user.set("age", 25);

console.log(user.get("name")); // Output: Alice
console.log(user.get("age"));  // Output: 25

// Menampilkan semua pasangan kunci-nilai
for (let [key, value] of user) {
    console.log(`${key}: ${value}`);
}
// Output:
// name: Alice
// age: 25
```

Variabel dan Tipe Data

11. Object (Mirip dengan Map)

Object adalah tipe data khusus di JavaScript yang juga menyimpan pasangan kunci-nilai, tetapi kuncinya selalu berupa `string` atau `symbol`.

```
let person = {
    name: "John",
    age: 30,
    isEmployed: true,
};

console.log(person.name);       // Output: John
console.log(person["age"]);     // Output: 30
```

Variabel dan Tipe Data

12. Tipe Data `Function`

JavaScript memperlakukan fungsi sebagai tipe data, jadi Anda bisa menyimpan fungsi dalam variabel, mengirimnya sebagai parameter, atau mengembalikannya dari fungsi lain. Ini dikenal sebagai First-Class Functions.

```
// Menyimpan fungsi dalam variabel
let greet = function(name) {
    return `Hello, ${name}!`;
};

console.log(greet("Alice")); // Output: Hello, Alice!

// Fungsi sebagai parameter
function performOperation(a, b, operation) {
    return operation(a, b);
}

let result = performOperation(5, 3, (x, y) => x + y); // Passing a function as argument
console.log(result); // Output: 8
```

Variabel dan Tipe Data

13. Destructuring

JavaScript mendukung Destructuring, yang memungkinkan Anda memecah atau mengekstrak nilai dari objek atau array secara langsung ke dalam variabel.

```
// Array Destructuring
let colors = ["red", "green", "blue"];
let [firstColor, secondColor] = colors;

console.log(firstColor); // Output: red
console.log(secondColor); // Output: green

// Object Destructuring
let user = { name: "Alice", age: 25 };
let { name, age } = user;

console.log(name); // Output: Alice
console.log(age);  // Output: 25
```

Variabel dan Tipe Data

14. Optional Chaining and Nullish Coalescing

Dua fitur baru ini memungkinkan penulisan kode yang lebih aman dan ringkas ketika mengakses properti yang mungkin `null` atau `undefined`.

- Optional Chaining (`?.`): Mengakses properti hanya jika objek atau properti sebelumnya tidak `null` atau `undefined`.

```
let user = { profile: { name: "Alice" } };
console.log(user?.profile?.name); // Output: Alice
console.log(user?.address?.city); // Output: undefined
```

Variabel dan Tipe Data

- Nullish Coalescing (`??`): Mengembalikan nilai kanan hanya jika nilai kiri adalah `null` atau `undefined`.

```
let username = null;
let defaultName = "Guest";
console.log(username ?? defaultName); // Output: Guest
```

Variabel dan Tipe Data

15. Template Literals dan Tagged Templates

Penjelasan sebagai berikut:

- Template Literals menggunakan tanda backticks (`` `...` ``) untuk menyusun string yang lebih fleksibel, memungkinkan sisipan variabel dengan `${}` dan pemformatan multi-baris.

```
let name = "Alice";
let greeting = `Hello, ${name}!`;
console.log(greeting); // Output: Hello, Alice!
```

Variabel dan Tipe Data

- Tagged Templates: Fungsi dapat digunakan untuk memproses template literal sebelum menghasilkan string akhir.

```
function tag(strings, ...values) {
    return strings[0] + values[0].toUpperCase();
}

let name = "alice";
let result = tag`Hello, ${name}!`;
console.log(result); // Output: Hello, ALICE!
```

Variabel dan Tipe Data

16. Promise dan Async/Await (Tipe Data Asynchronous)

JavaScript mendukung `Promise` untuk menangani operasi asynchronous, seperti permintaan jaringan atau operasi I/O.

```
// Menggunakan Promise
let fetchData = new Promise((resolve, reject) => {
    setTimeout(() => resolve("Data fetched"), 1000);
});

fetchData.then((data) => console.log(data)); // Output setelah 1 detik: Data fetched

// Menggunakan Async/Await
async function getData() {
    let result = await fetchData;
    console.log(result); // Output: Data fetched
}
```

Deklarasi variabel Global dan Lokal

JavaScript terdapat **variabel global** dan **variabel lokal**, yang bergantung pada di mana dan bagaimana variabel tersebut dideklarasikan. Berikut adalah penjelasan lengkap mengenai variabel global dan lokal di JavaScript:

- **Global**: Variabel global adalah variabel yang dapat diakses dari mana saja dalam kode, baik dari dalam fungsi maupun luar fungsi. Variabel ini biasanya dideklarasikan di luar semua fungsi atau blok kode.
  - Jika variabel dideklarasikan tanpa kata kunci `let`, `const`, atau `var` di luar atau dalam fungsi, maka ia menjadi variabel global.
  - Variabel yang dideklarasikan menggunakan `var` di luar fungsi juga menjadi global, tetapi ini tidak berlaku untuk `let` atau `const`.

Deklarasi variabel Global dan Lokal

```
// Variabel global tanpa `let`, `const`, atau `var`
globalVar = "This is a global variable";

// Variabel global dengan `var`
var globalVar2 = "This is another global variable";

function displayGlobal() {
    console.log(globalVar);   // Output: This is a global variable
    console.log(globalVar2);  // Output: This is another global variable
}

displayGlobal();
```

Deklarasi variabel Global dan Lokal

- **Lokal**: Variabel lokal adalah variabel yang hanya bisa diakses dari dalam fungsi atau blok tempat variabel itu dideklarasikan. Variabel lokal tidak dapat diakses di luar cakupannya (scope).
  - Variabel yang dideklarasikan dengan `let` atau `const` di dalam blok kode `{}` bersifat block-scoped.
  - Variabel yang dideklarasikan dengan `var` di dalam fungsi bersifat function-scoped.
-

Deklarasi variabel Global dan Lokal

```
function myFunction() {
    let localVar = "This is a local variable";
    const localConst = "This is a local constant";
    console.log(localVar);    // Output: This is a local variable
    console.log(localConst);  // Output: This is a local constant
}

myFunction();

// Mengakses variabel lokal di luar fungsi akan menyebabkan error
console.log(localVar); // Error: localVar is not defined
console.log(localConst); // Error: localConst is not defined
```

Scope dalam JavaScript

Ada dua jenis scope utama dalam JavaScript:

- Global Scope: Variabel yang dideklarasikan di luar fungsi atau blok menjadi bagian dari global scope.
- Local Scope: Variabel yang dideklarasikan di dalam fungsi atau blok memiliki cakupan lokal yang terbatas pada fungsi atau blok tersebut.

Scope dalam JavaScript

```
let globalVar = "I'm a global variable";

function outerFunction() {
    let outerVar = "I'm a variable in outerFunction";

    function innerFunction() {
        let innerVar = "I'm a variable in innerFunction";
        console.log(globalVar);  // Output: I'm a global variable
        console.log(outerVar);   // Output: I'm a variable in outerFunction
        console.log(innerVar);   // Output: I'm a variable in innerFunction
    }

    innerFunction();
    console.log(innerVar); // Error: innerVar is not defined
}

outerFunction();
```

this dalam JavaScript

this adalah kata kunci khusus dalam JavaScript yang merujuk pada objek yang sedang digunakan pada konteks tertentu. Nilai `this` bergantung pada context di mana ia dipanggil.

- Dalam fungsi biasa, `this` merujuk pada objek global (di browser, ini adalah `window`).
- Dalam metode objek, `this` merujuk pada objek tempat metode tersebut dipanggil.
- Dalam konstruktor atau class, this merujuk pada instance dari objek yang sedang dibuat.
- Arrow functions: tidak memiliki `this` sendiri, melainkan mewarisi nilai `this` dari konteks di mana fungsi tersebut dibuat.

this dalam JavaScript

```
// Contoh dalam fungsi biasa
function showThis() {
    console.log(this);
}
showThis(); // Di browser: Output: window (global object)

// Contoh dalam objek
const person = {
    name: "Alice",
    greet: function() {
        console.log(this.name);
    }
};
person.greet(); // Output: Alice

// Contoh dengan arrow function
const greetArrow = () => {
    console.log(this);
};
greetArrow(); // Output: this akan merujuk ke global object (window)
```

Tipe Data `NaN` dan `Infinity`

- `NaN` (Not a Number): Merupakan nilai yang menunjukkan bahwa operasi matematika yang tidak valid atau tidak terdefinisi telah dilakukan.
- `Infinity`: Merupakan nilai khusus yang mewakili bilangan yang sangat besar, lebih besar dari apa pun yang dapat ditangani JavaScript.

```
let invalidNumber = 0 / 0; // Akan menghasilkan NaN
console.log(invalidNumber); // Output: NaN

let largeNumber = 1 / 0; // Akan menghasilkan Infinity
console.log(largeNumber); // Output: Infinity

console.log(NaN === NaN); // Output: false (NaN tidak pernah sama dengan NaN)
```

Konversi Tipe Data (Type Coercion)

- JavaScript memiliki konsep type coercion, di mana JavaScript secara otomatis mengkonversi tipe data menjadi tipe yang diperlukan saat operasi dilakukan. Misalnya, saat melakukan operasi matematika dengan string, JavaScript akan mengkonversi string menjadi angka secara otomatis jika memungkinkan.

```
let num = "123";
let convertedNum = Number(num); // Mengkonversi string ke number
console.log(convertedNum); // Output: 123

let bool = String(true); // Mengkonversi boolean ke string
console.log(bool); // Output: "true"
```

Spread Operator (`...`)

- Operator spread memungkinkan Anda untuk menyebarkan elemen dari array atau properti dari objek ke dalam struktur lainnya. Ini digunakan untuk menggabungkan array, menyalin array atau objek, dan bahkan menggabungkan properti objek.

```
let arr1 = [1, 2, 3];
let arr2 = [...arr1, 4, 5];
console.log(arr2); // Output: [1, 2, 3, 4, 5]
```

```
let obj1 = { name: "Alice", age: 25 };
let obj2 = { ...obj1, country: "USA" };
console.log(obj2); // Output: { name: "Alice", age: 25, country: "USA" }
```

Operator

- Dalam JavaScript, operator adalah simbol yang digunakan untuk melakukan operasi pada satu atau lebih operand (nilai).
- JavaScript menyediakan berbagai jenis operator yang dapat digunakan untuk melakukan berbagai operasi seperti aritmatika, perbandingan, logika, dan lainnya.
- Berikut adalah penjelasan mengenai berbagai jenis operator yang tersedia dalam JavaScript:

Operator

1. Operator Aritmatika

   Operator aritmatika digunakan untuk melakukan operasi matematika pada angka.

Operator	Nama	Contoh	Deskripsi
`+`	Penjumlahan	`a + b`	Menambahkan dua operand
`-`	Pengurangan	`a - b`	Mengurangkan satu operand dari yang lain
`*`	Perkalian	`a * b`	Mengalikan dua operand
`/`	Pembagian	`a / b`	Membagi satu operand dengan yang lain
`%`	Modulus (Sisa)	`a % b`	Mendapatkan sisa bagi dua operand
`++`	Increment	`a++` atau `++a`	Menambahkan 1 ke operand
`--`	Decrement	`a--` atau `--a`	Mengurangi 1 dari operand

Operator

Contoh Operator Aritmatika:

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

// Penjumlahan
console.log(a + b); // Output: 13

// Pengurangan
console.log(a - b); // Output: 7

// Perkalian
console.log(a * b); // Output: 30

// Pembagian
console.log(a / b); // Output: 3.333...
```

Operator

2. Operator Perbandingan

   Operator perbandingan digunakan untuk membandingkan dua nilai dan mengembalikan true atau false.

Operator	Nama	Contoh	Deskripsi
`==`	Sama dengan	`a == b`	Memeriksa apakah nilai operand sama tanpa memeriksa tipe
`===`	Strict equality	`a === b`	Memeriksa apakah nilai dan tipe operand sama
`!=`	Tidak sama dengan	`a != b`	Memeriksa apakah nilai operand berbeda
`!==`	Strict inequality	`a !== b`	Memeriksa apakah nilai atau tipe operand berbeda
`>`	Lebih besar	`a > b`	Memeriksa apakah operand kiri lebih besar dari kanan
`<`	Lebih kecil	`a < b`	Memeriksa apakah operand kiri lebih kecil dari kanan
`>=`	Lebih besar atau sama	`a >= b`	Memeriksa apakah operand kiri lebih besar atau sama dengan kanan
`<=`	Lebih kecil atau sama	`a <= b`	Memeriksa apakah operand kiri lebih kecil atau sama dengan kanan

Operator

Contoh Operator Perbandingan:

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

// Sama dengan
console.log(x == y); // Output: false

// Sama dengan (strict, memperhatikan tipe data)
console.log(x === y); // Output: false

// Tidak sama dengan
console.log(x != y); // Output: true
```

Operator

3. Operator Logika

   Operator logika digunakan untuk melakukan operasi logika pada dua nilai boolean.

Operator	Nama	Contoh	Deskripsi
`&&`	AND	`a && b`	Mengembalikan `true` jika kedua operand adalah `true`
`	`	`	`	OR
`!`	NOT	`!a`	Mengembalikan kebalikan dari nilai boolean operand

Operator

Contoh Operator Logika:

```
let p = true;
let q = false;

// AND
console.log(p && q); // Output: false

// OR
console.log(p || q); // Output: true

// NOT
console.log(!p); // Output: false
```

Operator

4. Operator Penugasan

   Operator penugasan digunakan untuk menetapkan nilai ke variabel.

Operator	Nama	Contoh	Deskripsi
`=`	Penugasan	`a = b`	Menetapkan nilai dari operand kanan ke operand kiri
`+=`	Penugasan tambah	`a += b`	Menambahkan operand kanan ke kiri, kemudian menetapkan hasil ke operand kiri (`a = a + b`)
`-=`	Penugasan kurang	`a -= b`	Mengurangi operand kanan dari kiri, kemudian menetapkan hasil ke operand kiri (`a = a - b`)
`*=`	Penugasan kali	`a *= b`	Mengalikan operand kiri dengan kanan, kemudian menetapkan hasil ke operand kiri (`a = a * b`)
`/=`	Penugasan bagi	`a /= b`	Membagi operand kiri dengan kanan, kemudian menetapkan hasil ke operand kiri (`a = a / b`)
`%=`	Penugasan modulus	`a %= b`	Menetapkan sisa bagi operand kiri dengan kanan ke operand kiri (`a = a % b`)

Operator

Contoh Operator Penugasan:

```
let m = 5;
let n = 2;

// Penugasan dasar
m = n;
console.log(m); // Output: 2

// Penugasan tambah
m += n; // Sama dengan m = m + n
console.log(m); // Output: 4

// Penugasan kurang
m -= n; // Sama dengan m = m - n
console.log(m); // Output: 2
```

Operator

5. Operator ternary

   Operator ternary adalah operator kondisional yang digunakan untuk mengevaluasi kondisi dalam satu baris. Operator ini menggunakan simbol ? dan :.

```
kondisi ? nilaiJikaTrue : nilaiJikaFalse;
```

   Contoh:

```
let age = 18;
let canVote = age >= 18 ? "Eligible to vote" : "Not eligible to vote";
console.log(canVote); // Output: "Eligible to vote"
```

Latihan Operator

1. Buat program untuk menghitung luas segi tiga dari alas dan tinggi
2. Buat program menghitung luas lingkaran

Struktur Kendali

- Struktur kendali (control structure) adalah bagian penting dalam pemrograman yang digunakan untuk mengontrol alur eksekusi program.
- Dalam JavaScript, terdapat beberapa struktur kendali yang umum digunakan, termasuk pernyataan bersyarat (if, else, switch) dan perulangan (for, while, do-while).
- Berikut adalah penjelasan detail mengenai masing-masing struktur kendali ini.

Struktur Kendali

1. Pernyataan Bersyarat
   a. If dan else

      Pernyataan if digunakan untuk mengeksekusi blok kode tertentu jika kondisi yang diberikan benar (true). Jika kondisi salah (false), blok kode dalam else dapat dieksekusi.

```
let age = 16;

if (age >= 18) {
  console.log("You are eligible to vote.");
} else {
  console.log("You are not eligible to vote.");
}
// Output: You are not eligible to vote.
```

Struktur Kendali

b. If else if else if

Kita dapat menggunakan beberapa kondisi dengan else if untuk memeriksa kondisi tambahan.

```
let score = 85;

if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else if (score >= 70) {
  console.log("Grade: C");
} else {
  console.log("Grade: D");
}
// Output: Grade: B
```

Struktur Kendali

c. switch

Pernyataan switch digunakan untuk memeriksa sebuah nilai terhadap beberapa kemungkinan nilai yang berbeda.

Ini merupakan alternatif untuk menggunakan banyak if-else yang membuat kode lebih bersih.

```
let day = 3;
let dayName;

switch (day) {
  case 1:
    dayName = "Monday";
    break;
  case 2:
    dayName = "Tuesday";
    break;
  case 3:
    dayName = "Wednesday";
    break;
  case 4:
    dayName = "Thursday";
    break;
  case 5:
    dayName = "Friday";
    break;
  case 6:
    dayName = "Saturday";
    break;
  case 7:
    dayName = "Sunday";
    break;
  default:
    dayName = "Invalid day";
}
console.log(dayName); // Output: Wednesday
```

Struktur Kendali

1. Perulangan

   Perulangan digunakan untuk mengeksekusi blok kode berulang kali selama kondisi tertentu terpenuhi.

   a. for

      Struktur perulangan for digunakan ketika jumlah iterasi sudah diketahui.

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

Struktur Kendali

b. while

Perulangan while akan terus berjalan selama kondisi yang diberikan bernilai benar.

```
let count = 1;

while (count <= 5) {
  console.log("Count:", count);
  count++;
}
// Output:
// Count: 1
// Count: 2
// Count: 3
// Count: 4
// Count: 5
```

Struktur Kendali

c. do-while

Perulangan do-while mirip dengan while, tetapi memastikan bahwa blok kode dieksekusi setidaknya satu kali, bahkan jika kondisi awalnya salah.

```
let num = 1;

do {
  console.log("Number:", num);
  num++;
} while (num <= 5);
// Output:
// Number: 1
// Number: 2
// Number: 3
// Number: 4
// Number: 5
```

Struktur Kendali

d. `for...of` Loop

Digunakan untuk mengiterasi elemen-elemen dari objek yang dapat diiterasi, seperti array atau string.

```
let fruits = ["apple", "banana", "cherry"];

for (let fruit of fruits) {
  console.log(fruit);
}
// Output:
// apple
// banana
// cherry
```

Struktur Kendali

e. `for...in` Loop

Digunakan untuk mengiterasi properti-properti dari suatu objek.

```
let person = { name: "Alice", age: 25, city: "New York" };

for (let key in person) {
  console.log(key + ": " + person[key]);
}
```

Break dan Continues

- `break` digunakan untuk keluar dari perulangan atau `switch` secara langsung.
- `continue` digunakan untuk melewatkan iterasi saat ini dan melanjutkan ke iterasi berikutnya.

```
// Contoh penggunaan `break`
for (let i = 1; i <= 5; i++) {
  if (i === 3) break;
  console.log(i);
}
// Output:
// 1
// 2

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

map, filter, reduce, forEach

Dalam JavaScript, metode `map`, `filter`, `reduce`, dan `forEach` adalah fungsi array yang umum digunakan untuk manipulasi dan pengolahan data dalam array. Berikut penjelasan dan contoh kode untuk masing-masing metode:

1. map

   `map` membuat array baru dengan hasil pemanggilan fungsi tertentu pada setiap elemen dalam array.

```
let numbers = [1, 2, 3, 4];
let doubledNumbers = numbers.map(num => num * 2);
console.log(doubledNumbers); // Output: [2, 4, 6, 8]
```

map, filter, reduce, forEach

2. filter
   `filter` membuat array baru dengan semua elemen yang memenuhi kondisi dalam fungsi callback.

```
let numbers = [1, 2, 3, 4];
let greaterThanTwo = numbers.filter(num => num > 2);
console.log(greaterThanTwo); // Output: [3, 4]
```

map, filter, reduce, forEach

3. reduce
   `reduce` mengakumulasi semua elemen dalam array menjadi satu nilai, sesuai dengan fungsi yang kita berikan.

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce((accumulator, currentValue) => accumulator + currentValue, 0);
console.log(sum); // Output: 10
```

map, filter, reduce, forEach

4. foreach
   `forEach` mengeksekusi fungsi tertentu sekali untuk setiap elemen dalam array, tetapi tidak mengembalikan array baru (tidak seperti `map`, `filter`, atau `reduce`).

```
let numbers = [1, 2, 3, 4];
numbers.forEach(num => console.log(num));
// Output:
// 1
// 2
// 3
// 4
```

map, filter, reduce, forEach

Metode	Mengembalikan Array Baru	Mengubah Array Asli	Deskripsi
`map`	Ya	Tidak	Menerapkan fungsi ke setiap elemen dan mengembalikan array baru
`filter`	Ya	Tidak	Mengembalikan array baru yang hanya mengandung elemen yang memenuhi kondisi
`reduce`	Tidak	Tidak	Menggabungkan semua elemen dalam array menjadi satu nilai
`forEach`	Tidak	Tidak	Menjalankan fungsi untuk setiap elemen, tetapi tidak mengembalikan atau mengubah array

Fungsi

- Di JavaScript, **function** atau fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi memungkinkan kita mengorganisir, mengelompokkan, dan menggunakan kembali kode untuk berbagai keperluan tanpa harus menulis ulang logika yang sama.
- Jenis-jenis function di JavaScript
  1. Function Declaration (Deklarasi Fungsi)
     Ini adalah cara mendeklarasikan fungsi yang paling umum di JavaScript. Nama fungsi dan blok kode didefinisikan, dan fungsi ini bisa dipanggil kapan saja di program.

```
function tambah(a, b) {
  return a + b;
}

let hasil = tambah(3, 5);
console.log(hasil); // Output: 8
```

Fungsi

2. Function Expression (Ekspresi Fungsi)
   Fungsi juga bisa didefinisikan sebagai ekspresi dan disimpan dalam variabel. Fungsi yang didefinisikan dengan cara ini dikenal sebagai function expression.

```
const kali = function(x, y) {
  return x * y;
};

console.log(kali(4, 5)); // Output: 20
```

Fungsi

3. Arrow Function (Fungsi Panah)
   Arrow functions adalah cara yang lebih ringkas untuk mendefinisikan fungsi, dan sering digunakan dalam JavaScript modern, terutama dalam konteks `map`, `filter`, atau `reduce`.

```
const kuadrat = x => x * x;


console.log(kuadrat(6)); // Output: 36
```

   Jika fungsi hanya memiliki satu parameter dan satu baris kode, tanda kurung dan kurung kurawal dapat dihilangkan, seperti pada contoh di atas.

Fungsi

4. Anonymous Function (Fungsi Tanpa Nama)
   Anonymous function adalah fungsi yang tidak memiliki nama dan biasanya digunakan sebagai argumen untuk fungsi lain, seperti `setTimeout`, `forEach`, atau `map`.

```
setTimeout(function() {
  console.log("Hello, world!");
}, 1000);
// Output: Hello, world! (setelah 1 detik)
```

Fungsi

Paramater Default : JavaScript juga mendukung parameter default, yaitu nilai yang akan digunakan jika argumen tidak diberikan saat memanggil fungsi.

```
function greet(name = "Guest") {
  console.log("Hello, " + name);
}

greet("Alice"); // Output: Hello, Alice
greet(); // Output: Hello, Guest
```

Fungsi

Fungsi dengan Rest Parameter: Rest parameter (`...`) memungkinkan kita menangkap sejumlah argumen tak terbatas dalam sebuah array.

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

console.log(sum(1, 2, 3, 4, 5)); // Output: 15
```

Fungsi

5. Immediately Invoked Function Expression (IIFE) (Ekspresi Fungsi yang Langsung Dipanggil)
   IIFE adalah fungsi yang langsung dieksekusi setelah didefinisikan. Ini berguna untuk membuat variabel dan fungsi yang hanya ada dalam cakupan (scope) fungsi tersebut dan tidak terpengaruh atau mempengaruhi bagian lain dari kode.

```
(function() {
  let message = "Hello from IIFE!";
  console.log(message);
})();
// Output: Hello from IIFE!
```

Fungsi

- Di JavaScript, **function** atau fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi memungkinkan kita mengorganisir, mengelompokkan, dan menggunakan kembali kode untuk berbagai keperluan tanpa harus menulis ulang logika yang sama.
- Jenis-jenis function di JavaScript
  1. Function Declaration (Deklarasi Fungsi)
  2. Function Expression (Ekspresi Fungsi)
  3. Arrow Function (Fungsi Panah)
  4. Anonymous Function (Fungsi Tanpa Nama)
  5. Immediately Invoked Function Expression (IIFE) (Ekspresi Fungsi yang Langsung Dipanggil)

Fungsi

- Di JavaScript, **function** atau fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi memungkinkan kita mengorganisir, mengelompokkan, dan menggunakan kembali kode untuk berbagai keperluan tanpa harus menulis ulang logika yang sama.
- Jenis-jenis function di JavaScript
  1. Function Declaration (Deklarasi Fungsi)
  2. Function Expression (Ekspresi Fungsi)
  3. Arrow Function (Fungsi Panah)
  4. Anonymous Function (Fungsi Tanpa Nama)
  5. Immediately Invoked Function Expression (IIFE) (Ekspresi Fungsi yang Langsung Dipanggil)

Class dan Object

- Di JavaScript, class (kelas) dan object (objek) adalah konsep penting dalam pemrograman berorientasi objek (OOP) yang memungkinkan kita membuat struktur data yang kompleks dengan cara yang terorganisir.
- Class adalah cetak biru atau template untuk membuat object yang memiliki properti dan metode tertentu.
    1. class: adalah template atau blueprint yang digunakan untuk membuat objek. Di dalam class, kita bisa mendefinisikan properti (atribut) dan metode (fungsi) yang dimiliki oleh setiap objek yang dibuat dari class tersebut.

Class dan Object

Contoh class:

```
class Mobil {
  constructor(merk, warna) {
    this.merk = merk;
    this.warna = warna;
  }

  tampilkanInfo() {
    console.log(`Mobil ini adalah ${this.merk} berwarna ${this.warna}.`);
  }
}
```

Class dan Object

- Constructor (`constructor(merk, warna)`) adalah metode khusus dalam class yang akan dipanggil saat membuat objek baru. Di sini, kita memberikan dua parameter `merk` dan `warna` yang akan disimpan sebagai properti objek (`this.merk` dan `this.warna`).
- Method (`tampilkanInfo`) adalah fungsi yang bisa digunakan oleh objek yang dibuat dari class ini.

Class dan Object

2. Object adalah instance dari class. Setelah kita memiliki class, kita bisa membuat objek berdasarkan class tersebut. Setiap objek yang dibuat akan memiliki properti dan metode yang didefinisikan dalam class.

```
// Membuat objek baru dari class Mobil
let mobil1 = new Mobil("Toyota", "Merah");
let mobil2 = new Mobil("Honda", "Biru");

// Menggunakan metode tampilkanInfo
mobil1.tampilkanInfo(); // Output: Mobil ini adalah Toyota berwarna Merah.
mobil2.tampilkanInfo(); // Output: Mobil ini adalah Honda berwarna Biru.
```

Inheritance

JavaScript memungkinkan kita membuat class baru yang mewarisi properti dan metode dari class lain. Ini disebut Inheritance.

```
// Membuat class MobilBalap yang mewarisi dari Mobil
class MobilBalap extends Mobil {
  constructor(merk, warna, kecepatan) {
    super(merk, warna); // Memanggil constructor dari class parent (Mobil)
    this.kecepatan = kecepatan;
  }

  // Metode baru khusus untuk MobilBalap
  tampilkanKecepatan() {
    console.log(`Mobil ini bisa mencapai kecepatan ${this.kecepatan} km/jam.`);
  }
}

// Membuat objek dari class MobilBalap
let mobilBalap = new MobilBalap("Ferrari", "Merah", 350);
mobilBalap.tampilkanInfo();        // Output: Mobil ini adalah Ferrari berwarna Merah.
mobilBalap.tampilkanKecepatan();   // Output: Mobil ini bisa mencapai kecepatan 350 km/jam
```

Asynchronous JavaScript

- Dalam JavaScript, asynchronous mengacu pada proses yang tidak harus berjalan secara berurutan. JavaScript biasanya mengeksekusi kode secara synchronous (berurutan, satu per satu), tetapi dengan asynchronous programming, kita bisa menjalankan kode tanpa harus menunggu proses yang lama selesai. Ini memungkinkan kita untuk menangani tugas-tugas seperti pengambilan data dari API, pembacaan file, atau operasi lainnya tanpa mengganggu alur program.
- JavaScript mengatur proses asynchronous melalui Event Loop, yang menangani callback atau tugas asynchronous lainnya. Ada beberapa teknik untuk menangani asynchronous programming di JavaScript, yaitu:
  1. Callback
  2. Promises
  3. Async/Await

Asynchronous JavaScript

1. **Callback**: adalah fungsi yang dikirim sebagai argumen ke fungsi lain dan dipanggil setelah operasi asynchronous selesai. Namun, penggunaan callback yang berlebihan bisa menyebabkan "callback hell"—kode yang sulit dibaca dan dikelola.

```
function fetchData(callback) {
  setTimeout(() => {
    console.log("Data diambil dari server");
    callback();
  }, 2000); // Simulasi penundaan 2 detik
}

function processData() {
  console.log("Data sedang diproses");
}

fetchData(processData);

// Output setelah 2 detik:
// Data diambil dari server
// Data sedang diproses
```

Asynchronous JavaScript

2. **Promise**: adalah objek yang mewakili keberhasilan atau kegagalan dari operasi asynchronous. Sebuah Promise dapat berada dalam tiga state:
    1. Pending: Menunggu hasil.
    2. Fulfilled: Berhasil menyelesaikan tugas.
    3. Rejected: Gagal menyelesaikan tugas.

Asynchronous JavaScript

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const success = true; // Ubah ke false untuk melihat perbedaan
      if (success) {
        resolve("Data berhasil diambil dari server");
      } else {
        reject("Gagal mengambil data dari server");
      }
    }, 2000);
  });
}

fetchData()
  .then((result) => {
    console.log(result); // Jika sukses: "Data berhasil diambil dari server"
  })
  .catch((error) => {
    console.error(error); // Jika gagal: "Gagal mengambil data dari server"
  });
```

Asynchronous JavaScript

3. **Async/Await**:adalah cara yang lebih modern untuk menangani kode asynchronous di JavaScript, yang memperbaiki keterbacaan kode Promise. `async` digunakan untuk mendeklarasikan fungsi asynchronous, dan `await` digunakan di dalamnya untuk menunggu Promise selesai sebelum melanjutkan eksekusi.

Asynchronous JavaScript

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const success = true;
      if (success) {
        resolve("Data berhasil diambil dari server");
      } else {
        reject("Gagal mengambil data dari server");
      }
    }, 2000);
  });
}

async function processData() {
  try {
    const result = await fetchData(); // Menunggu fetchData selesai
    console.log(result);
  } catch (error) {
    console.error(error); // Menangani error jika Promise ditolak
  }
}

processData();
// Output setelah 2 detik:
// Data berhasil diambil dari server
```

Asynchronous JavaScript

Metode	Deskripsi	Kelebihan	Kekurangan
Callback	Fungsi dikirim sebagai argumen dan dipanggil setelah tugas selesai	Simpel dan mudah dimengerti untuk kasus sederhana	Dapat menyebabkan callback hell
Promises	Menyediakan cara menangani hasil asynchronous dengan `.then()` dan `.catch()`	Membantu menghindari callback hell	Kode masih bisa terlihat kompleks
Async/Await	Sintaks modern yang membuat kode asynchronous tampak lebih mirip kode synchronous	Mudah dibaca dan lebih bersih	Hanya dapat digunakan dalam fungsi `async`

Asynchronous JavaScript

1. **Callback**: Cocok untuk operasi sederhana dan cepat, namun hindari penggunaan yang berlebihan.
2. **Promises**: Cocok untuk menghindari callback hell dan menangani asynchronous secara lebih terstruktur.
3. **Async/Await**: Direkomendasikan untuk menulis kode asynchronous yang lebih bersih dan mudah dibaca, terutama jika menggunakan Promises yang saling terkait.

• Exception Handling

Exception Handling adalah mekanisme dalam pemrograman untuk menangani kesalahan (error) yang terjadi saat program dijalankan. Di JavaScript, penanganan kesalahan dilakukan menggunakan blok `try...catch`. Mekanisme ini memungkinkan kita untuk menangkap dan menangani kesalahan tanpa menghentikan eksekusi program secara keseluruhan.

```
function bacaFile() {
  try {
    // Simulasi pembacaan file
    console.log("Membaca file...");
    throw new Error("Gagal membaca file"); // Simulasi error
  } catch (error) {
    console.log("Error: " + error.message);
  } finally {
    console.log("Menutup koneksi file...");
  }
}
```

Silabus

Berikut ini silabus Mobile Apps Development menggunakan Flutter:

1. JavaScript Programming
2. Dasar-dasar React
3. React Component
4. Bekerja dengan REST
5. React Router
6. Mini Project
7. Case Study

React

The library for web and native user interfaces

Learn React | API Reference

Dasar-dasar React

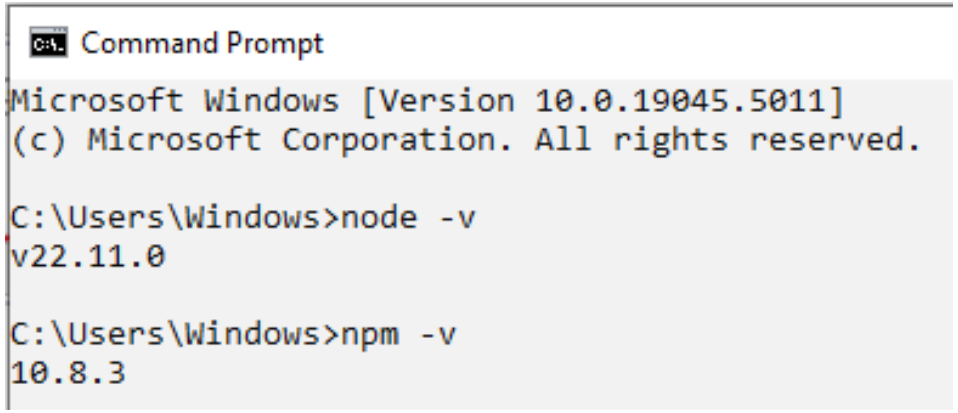
- React adalah sebuah library JavaScript yang digunakan untuk membangun antarmuka pengguna (UI) dinamis dan interaktif. React dikembangkan oleh Facebook dan dirilis pada tahun 2013. React memungkinkan pengembang untuk membuat aplikasi berbasis komponen yang dapat dipakai kembali (reusable components). React menggunakan konsep virtual DOM untuk meningkatkan performa, yang membuatnya lebih cepat daripada pendekatan tradisional.
- React memungkinkan pengembang untuk membangun aplikasi satu halaman (SPA - Single Page Application) yang responsif dan cepat, di mana seluruh aplikasi dapat dimuat hanya sekali dan navigasi antar halaman dapat dilakukan tanpa melakukan refresh halaman penuh.

Dasar-dasar React

1. Install Node.js dan npm
2. Membuat Proyek React Baru
3. Struktur Proyek React
4. Menjalankan Proyek React
5. Menambahkan Komponen React

Dasar-dasar React

1. Install Node.js dan npm
   Sebelum mulai bekerja dengan React, pastikan Anda sudah menginstal Node.js dan npm (Node Package Manager), yang akan digunakan untuk mengelola dependensi dan menjalankan server pengembangan.
   1. Node.js: Ini adalah runtime JavaScript yang digunakan untuk menjalankan aplikasi JavaScript di server.
   2. npm: Ini adalah package manager yang digunakan untuk mengelola pustaka (library) eksternal.



Dasar-dasar React

2. Membuat Proyek React Baru
    1. Buka cmd arahkan ke folder misal d:\belajar_react
    2. cmd ketik: **npx create-react-app hello-word**
    3. Buka VSCode, open folder hello-world

Dasar-dasar React

3. Struktur Project:

```
my-app/
├── node_modules/          # Folder untuk dependensi eksternal (pustaka) yang diinstal
├── public/                # Folder untuk file statis seperti index.html, favicon.ico, dll.
│   ├── index.html         # File HTML utama
│   └── ...
├── src/                   # Folder untuk kode sumber aplikasi React
│   ├── App.css            # Gaya untuk komponen App
│   ├── App.js             # Komponen utama aplikasi
│   ├── index.css          # Gaya global
│   ├── index.js           # Titik masuk aplikasi React
│   └── ...
├── .gitignore             # File untuk mengecualikan file atau folder dari kontrol versi
├── package.json           # File konfigurasi proyek dan dependensi
├── README.md              # Dokumentasi proyek
└── package-lock.json      # File yang mengunci versi dependensi
```

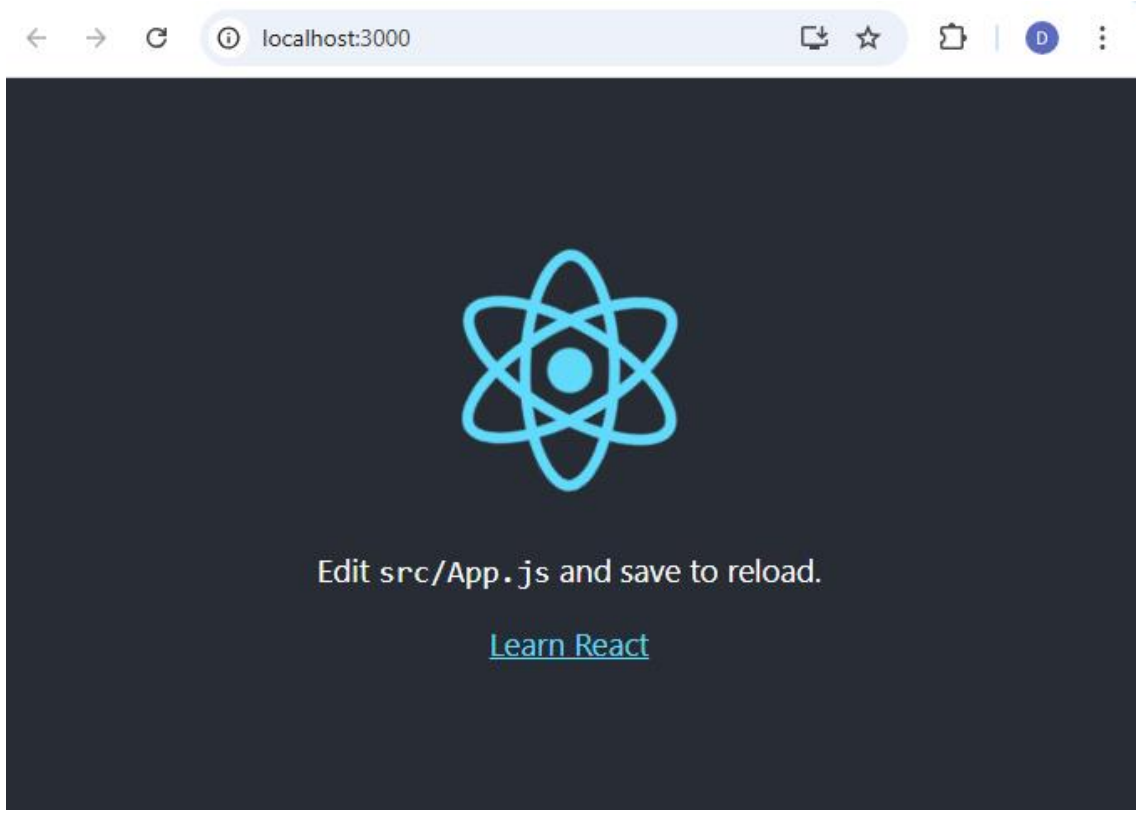
Dasar-dasar React

Struktur Project:

- **public/index.html**: File HTML dasar yang digunakan oleh aplikasi React.
- **src/index.js**: Titik masuk (entry point) aplikasi React. Di sini, ReactDOM digunakan untuk merender aplikasi ke dalam elemen HTML di file `index.html`.
- **src/App.js**: Komponen utama aplikasi yang akan dirender di browser.

Dasar-dasar React

4. Menjalankan Proyek React
    1. Di VSCode pilih menu View -> Terminal
    2. Ketik **npm start**



Dasar-dasar React

Ubah App.js

JS App.js M ×

src > JS App.js > ...

```
1   import './App.css';
2
3   function App() {
4     return (
5       <div className="App">
6         <header className="App-header">
7           <h1>Selamat datang di Aplikasi React!</h1>
8         </header>
9       </div>
10    );
11  }
12
13  export default App;
```

React Component

1. Buat file di src Counter.js
2. Ketik file berikut di :

Counter.js U

src > Counter.js > default

```
1  function Counter() {
2    return (
3      <div>
4        <button>Increment</button>
5      </div>
6    );
7  }
8  export default Counter;
```

3. Penggunaan di App.js

App.js M

src > App.js > ...

```
1  import './App.css';
2
3  function App() {
4    return (
5      <div className="App">
6        <header className="App-header">
7          <h1>Selamat datang di Aplikasi React!</h1>
8        </header>
9      </div>
10   );
11 }
12
13 export default App;
```

React Component

4. Modifikasi Counter.js

JS Counter.js U ×

src > JS Counter.js > ...

```
1   // src/Counter.js
2   import React, { useState } from 'react';
3
4   function Counter() {
5     const [count, setCount] = useState(0); // state untuk menyimpan nilai counter
6
7     const handleClick = () => {
8       setCount(count + 1); // update nilai counter
9     };
10
11    return (
12      <div>
13        <h1>Counter: {count}</h1>  {/* Menampilkan nilai counter */}
14        <button onClick={handleClick}>Increment</button> {/* Tombol untuk menambah nilai */}
15      </div>
16    );
17  }
18
19  export default Counter;
```

React Component

- Penjelasan Counter.js
  Penjelasan kode sebagai berikut:
  - useState(0):Hook ini digunakan untuk mendeklarasikan state count yang dimulai dengan nilai 0. setCount digunakan untuk mengubah nilai count.
  - handleClick: Fungsi ini dipanggil ketika tombol diklik, yang akan menambah nilai count sebesar 1.
  - Tombol (<button>): Ketika tombol ini diklik, handleClick akan dipanggil dan nilai count akan bertambah.
  - export berarti mengekspor elemen (seperti fungsi, variabel, atau objek) dari suatu modul sehingga elemen tersebut bisa diakses dan digunakan oleh modul lain.

React Component

5. Menambah CSS dan gunakan dalam Counter.js

Counter.css U

src > # Counter.css > ...

```css
/* src/Counter.css */
.counter {
    text-align: center;
    font-family: Arial, sans-serif;
    margin-top: 50px;
}

.counter h1 {
    color: #4caf50;
}

.counter button {
    padding: 10px 20px;
    font-size: 16px;
    background-color: #008CBA;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.counter button:hover {
    background-color: #007B9A;
}
```

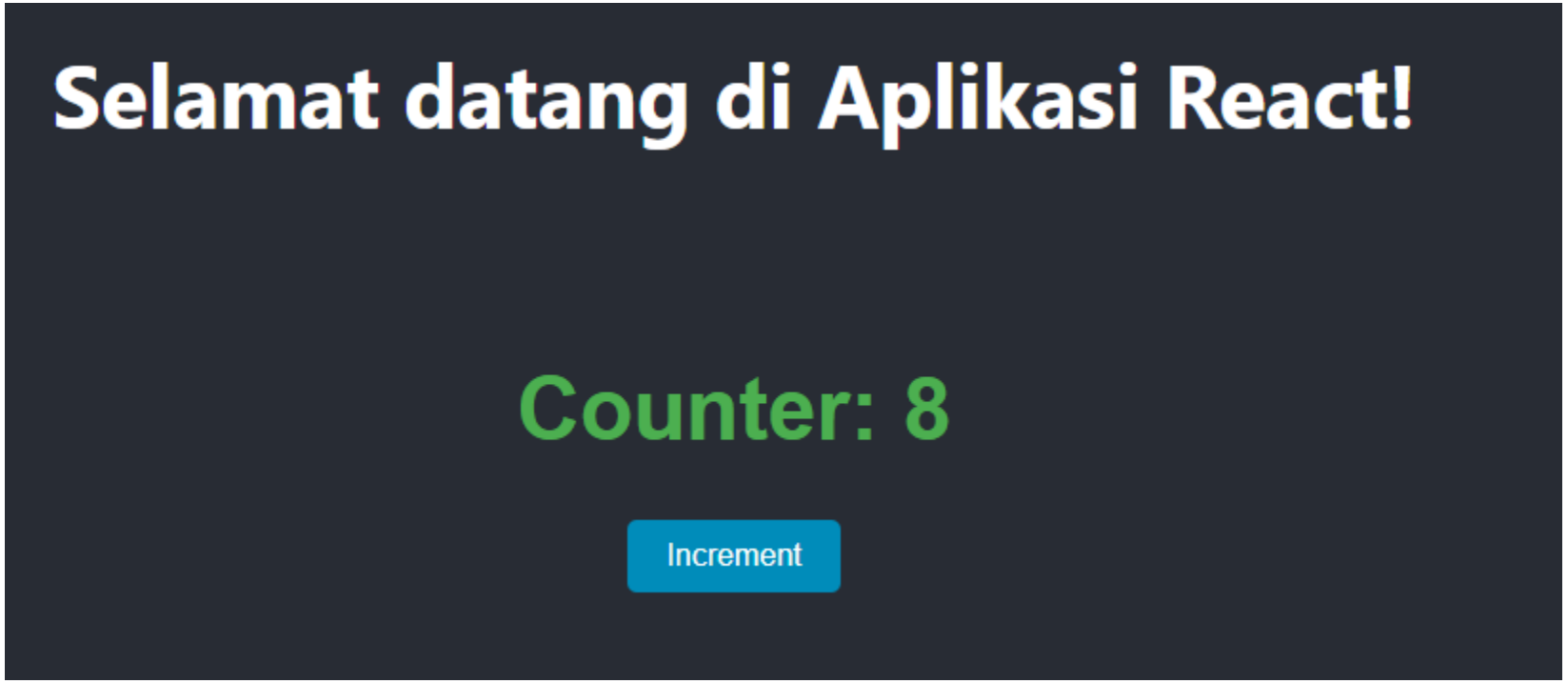
Counter.js U

src > Counter.js > ...

```jsx
// src/Counter.js
import React, { useState } from 'react';
import './Counter.css'; // Mengimpor file CSS

function Counter() {
  const [count, setCount] = useState(0); // state untuk menyimpan nilai counter

  const handleClick = () => {
    setCount(count + 1); // update nilai counter
  };

  return (
    <div className="counter"> {/* Menambahkan kelas CSS */}
      <h1>Counter: {count}</h1>  {/* Menampilkan nilai counter */}
      <button onClick={handleClick}>Increment</button> {/* Tombol untuk menambah nilai */}
    </div>
  );
}

export default Counter;
```

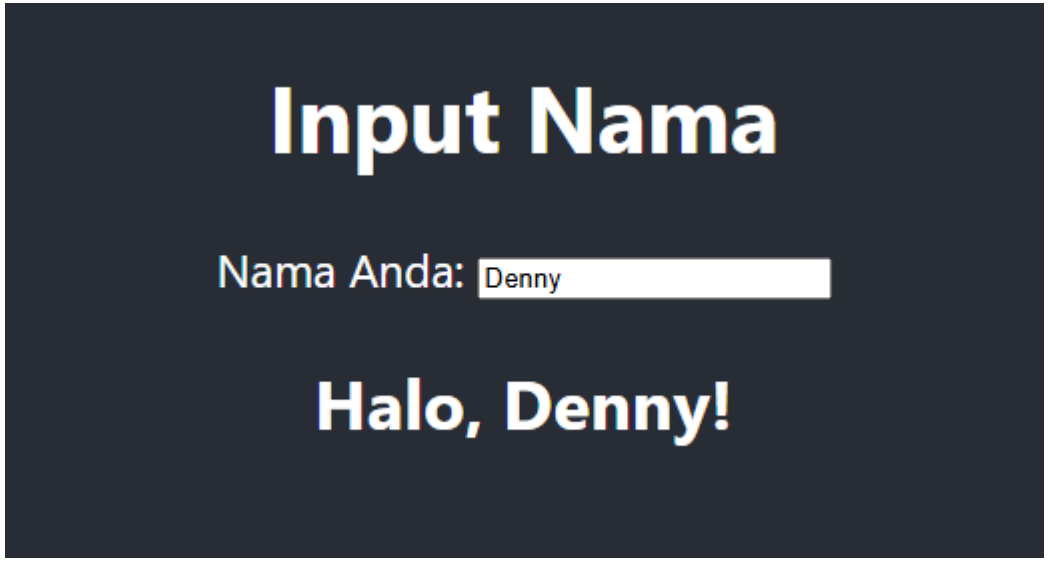
React Component

6. Hasil



Latihan

1. Buat file InputNama.js dengan output

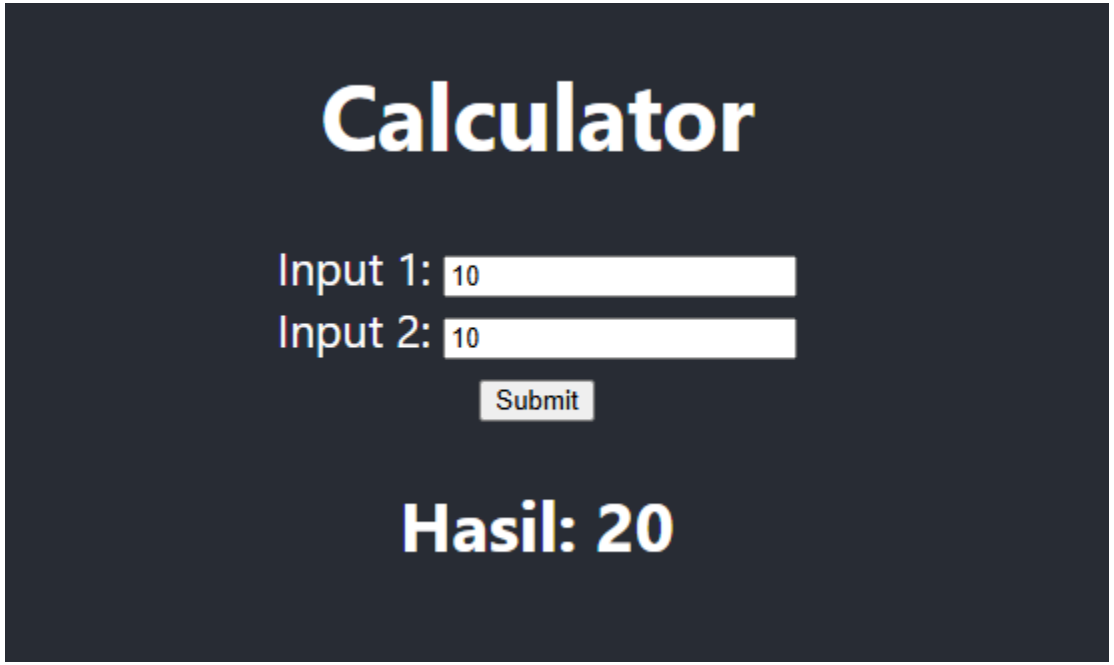


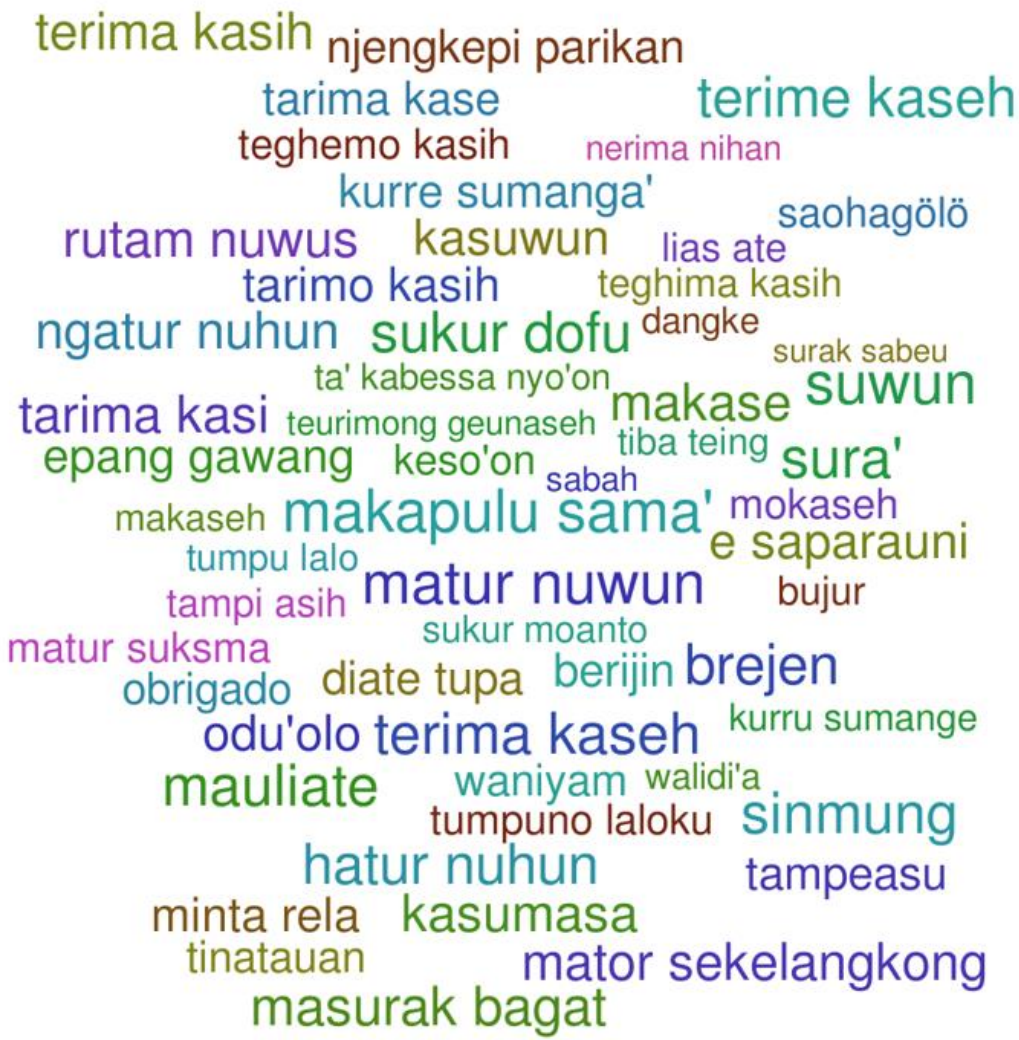
Latihan

```jsx
import React, { useState } from 'react';

function InputNama() {
  const [nama, setNama] = useState('');

  return (
    <div>
      <h1>Input Nama</h1>
      <form>
        <div>
          <label>Nama Anda: </label>
          <input
           value={nama}
            type="text"
            onChange={(e) => setNama(e.target.value)}
            placeholder="Masukkan nama"
          />
        </div>
      </form>
      {nama && (
        <div>
          <h2>Halo, {nama}!</h2>
        </div>
      )}
    </div>
  );
}

export default InputNama;
```

Latihan

2. Buat seperti berikut





Telkom digiUp 2024

ESCALATING DIGITAL SKILLS COMPETENCIES FOR FUTURE DIGITAL TALENT

BUMN UNTUK INDONESIA

Telkom Indonesia

the world in your hand