

Materi JS

From Basic to Advanced

Bahrul Rozak - Kelas Software Developer

MIT License

Copyright (c) 2024 Bahrul Rozak

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE

Pengantar JavaScript

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat halaman web interaktif. Dibuat oleh Brendan Eich pada tahun 1995, JavaScript awalnya digunakan untuk membuat efek interaktif pada halaman web. Namun, seiring perkembangan web, JavaScript menjadi salah satu bahasa pemrograman paling populer untuk pengembangan web dan aplikasi berbasis web.

1. Penggunaan Utama dari JavaScript

JavaScript digunakan untuk membuat halaman web interaktif dengan menambahkan efek, animasi, validasi formulir, dan interaksi pengguna lainnya.

Saat ini, JavaScript juga digunakan untuk pengembangan aplikasi web berbasis klien (front-end) dan server (back-end).

2. Kelebihan JavaScript

Kemampuan untuk membuat halaman web interaktif tanpa perlu me-refresh halaman.

Memiliki dukungan dari berbagai peramban web utama seperti Chrome, Firefox, dan Safari.

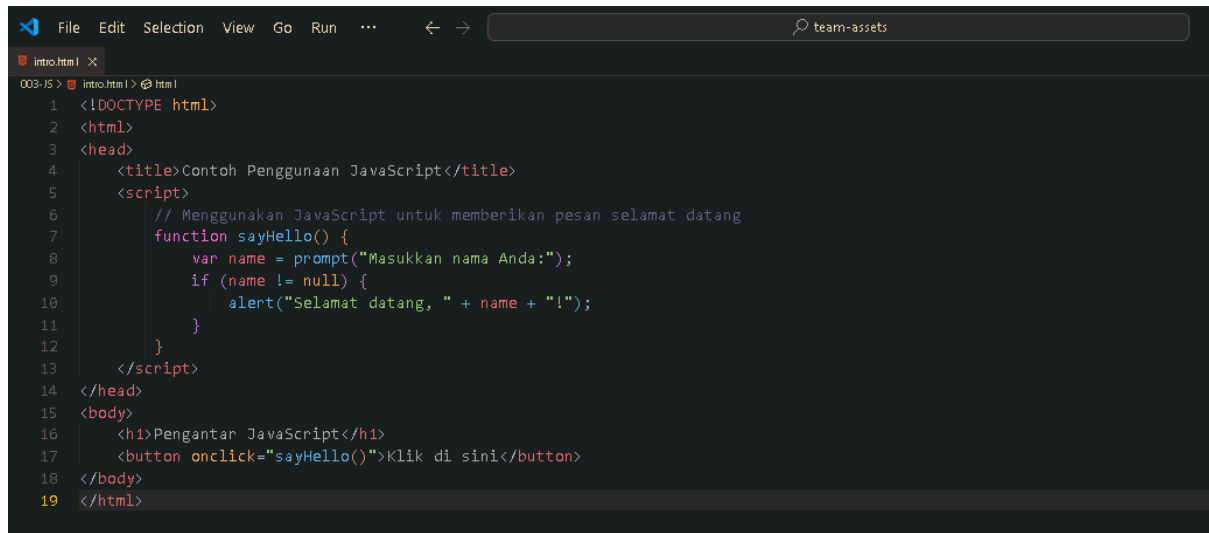
Memiliki banyak framework dan library yang memudahkan pengembangan aplikasi web.

3. Kerugian JavaScript

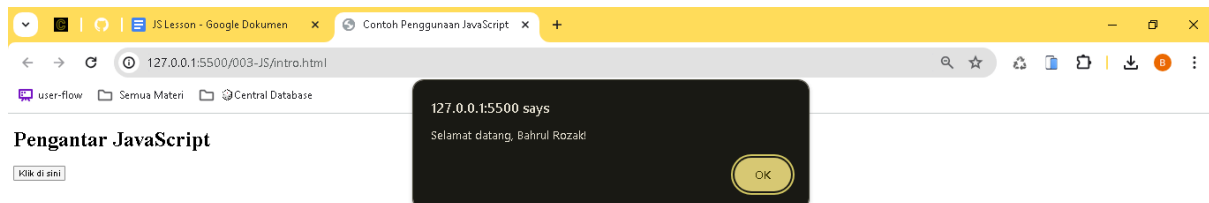
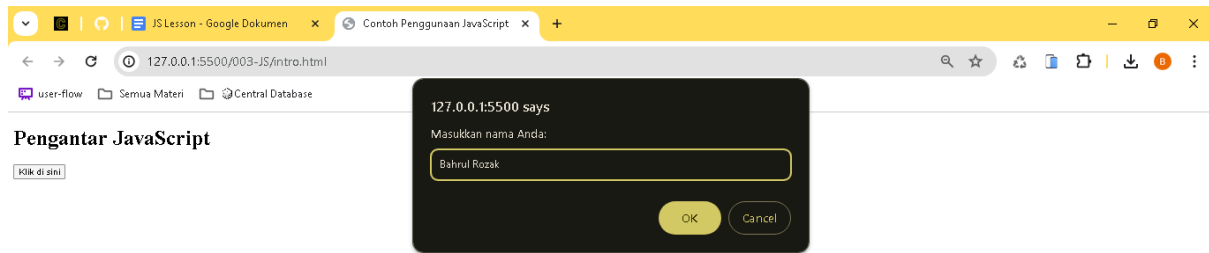
Ketergantungan pada peramban web, sehingga hasil yang berbeda dapat terjadi di peramban yang berbeda.

Keamanan: JavaScript dapat disalahgunakan untuk melakukan serangan keamanan seperti injeksi skrip (script injection) dan XSS (Cross-Site Scripting).

Berikut adalah contoh sederhana penggunaan JavaScript dalam halaman web:



```
003-JS > intro.html > @html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Contoh Penggunaan JavaScript</title>
5   <script>
6     // Menggunakan JavaScript untuk memberikan pesan selamat datang
7     function sayHello() {
8       var name = prompt("Masukkan nama Anda:");
9       if (name != null) {
10        alert("Selamat datang, " + name + "!");
11      }
12    }
13  </script>
14 </head>
15 <body>
16   <h1>Pengantar JavaScript</h1>
17   <button onclick="sayHello()">Klik di sini</button>
18 </body>
19 </html>
```



Dalam contoh di atas, sebuah halaman web sederhana dibuat dengan judul "Pengantar JavaScript". Terdapat sebuah tombol yang jika diklik akan memunculkan kotak dialog untuk memasukkan nama pengguna. Setelah pengguna memasukkan namanya, sebuah pesan selamat datang akan muncul. Ini adalah contoh sederhana penggunaan JavaScript untuk membuat halaman web interaktif.

Sintaks Dasar JavaScript

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat website interaktif. Untuk memahami konsep dasar JavaScript, penting untuk memahami sintaks dasarnya. Berikut adalah penjelasan mengenai sintaks dasar JavaScript beserta contoh source code:

Variabel dan Tipe Data

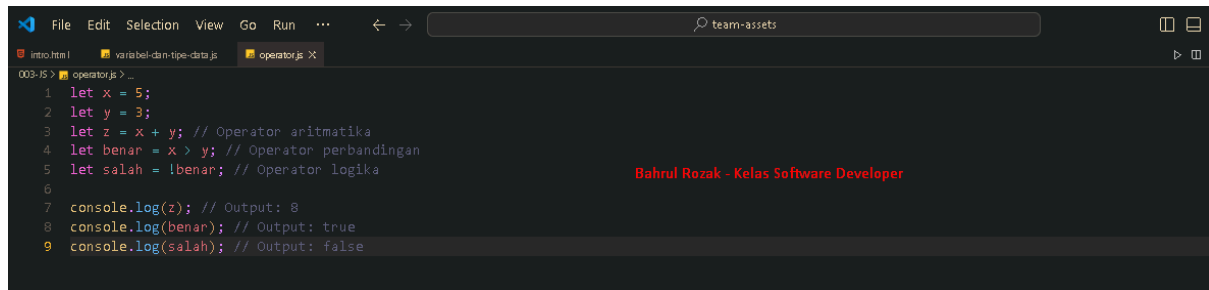
Variabel digunakan untuk menyimpan nilai. Tipe data dalam JavaScript terdiri dari tipe data primitif dan non-primitif. Tipe data primitif meliputi number, string, boolean, null, dan undefined.

```
003-JS > variabel-dan-tipe-data.js > _
1 // Mendeklarasikan variabel
2 let x = 5;
3 let nama = "John";
4 let benar = true;
5 let kosong = null;
6 let tidakTerdefinisi;
7
8 // Menampilkan nilai variabel
9 console.log(x); // Output: 5
10 console.log(nama); // Output: John
11 console.log(benar); // Output: true
12 console.log(kosong); // Output: null
13 console.log(tidakTerdefinisi); // Output: undefined
```

Bahrul Rozak - Kelas Software Developer

Operator

Operator digunakan untuk melakukan operasi pada variabel dan nilai. Contoh operator termasuk aritmatika (+, -, *, /), perbandingan (==, !=, >, <), dan logika (&&, ||, !).

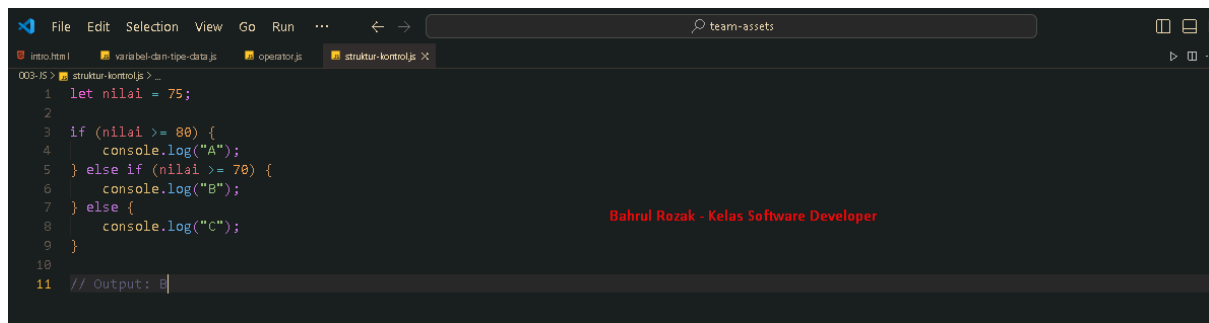


```
File Edit Selection View Go Run ... team-assets
003-JS > operator.js > ...
1 let x = 5;
2 let y = 3;
3 let z = x + y; // Operator aritmatika
4 let benar = x > y; // Operator perbandingan
5 let salah = !benar; // Operator logika
6
7 console.log(z); // Output: 8
8 console.log(benar); // Output: true
9 console.log(salah); // Output: false
```

Bahrul Rozak - Kelas Software Developer

Struktur Kontrol

Struktur kontrol digunakan untuk mengontrol alur program. Contoh struktur kontrol meliputi if-else dan switch-case.

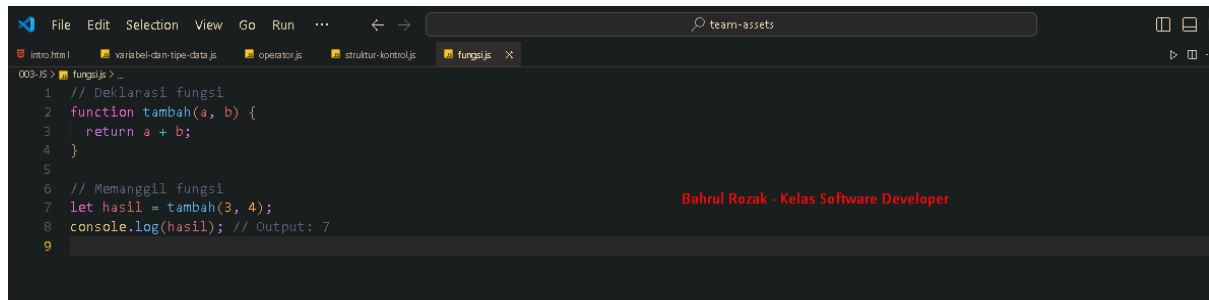


```
File Edit Selection View Go Run ... team-assets
003-JS > struktur-kontrol.js > ...
1 let nilai = 75;
2
3 if (nilai >= 80) {
4   console.log("A");
5 } else if (nilai >= 70) {
6   console.log("B");
7 } else {
8   console.log("C");
9 }
10
11 // Output: B
```

Bahrul Rozak - Kelas Software Developer

Fungsi

Fungsi digunakan untuk menjalankan blok kode tertentu. Fungsi dapat menerima parameter dan mengembalikan nilai.



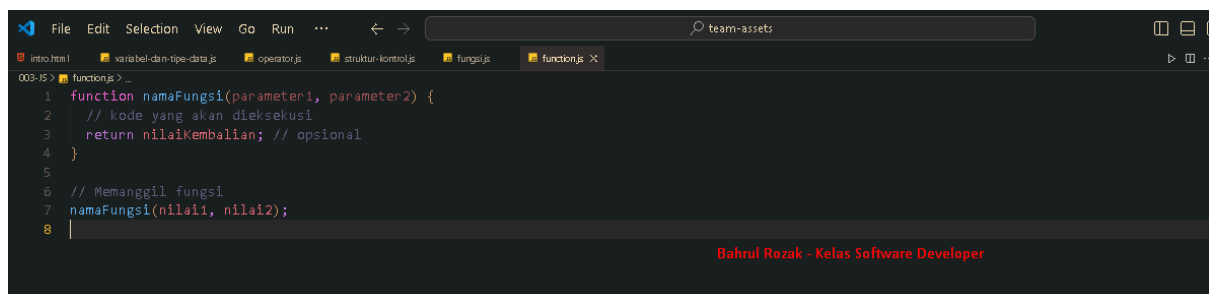
```
File Edit Selection View Go Run ... team-assets
003-JS > fungsi.js > ...
1 // Deklarasi fungsi
2 function tambah(a, b) {
3   return a + b;
4 }
5
6 // Memanggil fungsi
7 let hasil = tambah(3, 4);
8 console.log(hasil); // Output: 7
9
```

Bahrul Rozak - Kelas Software Developer

Sintaks dasar JavaScript ini membentuk dasar pemrograman yang penting untuk dipahami sebelum melangkah ke konsep yang lebih lanjut.

Pengenalan Fungsi dalam JavaScript

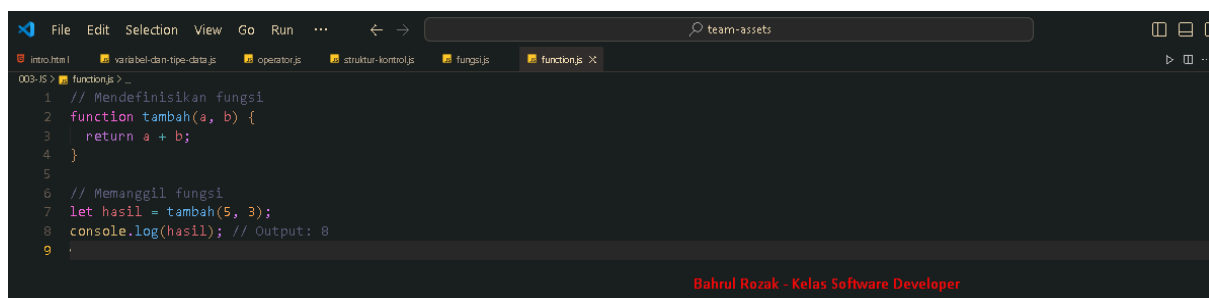
Fungsi dalam JavaScript adalah blok kode yang dirancang untuk melakukan tugas tertentu atau menghitung nilai. Fungsi dapat digunakan untuk mengorganisir kode, menghindari pengulangan, dan membuat kode lebih mudah dipahami. Dalam JavaScript, fungsi dapat didefinisikan dan dipanggil berulang kali sesuai kebutuhan.



```
003-JS > function.js > ...
1 function namaFungsi(parameter1, parameter2) {
2   // kode yang akan dieksekusi
3   return nilaiKembalian; // opsional
4 }
5
6 // Memanggil fungsi
7 namaFungsi(nilai1, nilai2);
8
```

Bahrul Rozak - Kelas Software Developer

Contoh



```
003-JS > function.js > ...
1 // Mendefinisikan fungsi
2 function tambah(a, b) {
3   return a + b;
4 }
5
6 // Memanggil fungsi
7 let hasil = tambah(5, 3);
8 console.log(hasil); // Output: 8
9
```

Bahrul Rozak - Kelas Software Developer

Penjelasan Lebih Mendetail dari kode XD

1. **Definisi Fungsi**
Fungsi didefinisikan menggunakan kata kunci `function`, diikuti dengan nama fungsi yang diinginkan, dan daftar parameter dalam tanda kurung. Parameter adalah nilai yang akan diterima oleh fungsi.
2. **Parameter dan Argumen**
Parameter adalah variabel yang digunakan dalam definisi fungsi. Argumen adalah nilai yang dilewatkan ke fungsi saat dipanggil. Jumlah parameter dalam definisi fungsi dapat bervariasi, dan parameter dapat memiliki nilai default.
3. **Return Statement**
Fungsi dapat mengembalikan nilai menggunakan pernyataan `return`. Nilai yang dikembalikan dapat digunakan di tempat yang memanggil fungsi.
4. **Memanggil Fungsi**
Fungsi dipanggil dengan menuliskan nama fungsi diikuti dengan tanda kurung dan nilai argumen yang diperlukan, jika ada.
5. **Fungsi Sebagai Nilai**

Dalam JavaScript, fungsi dianggap sebagai nilai. Ini berarti fungsi dapat disimpan dalam variabel, dikirimkan sebagai argumen ke fungsi lain, dan dikembalikan dari fungsi lain.

6. Fungsi Anonim

Fungsi anonim adalah fungsi tanpa nama. Fungsi ini sering digunakan dalam situasi di mana fungsi hanya digunakan sekali atau sebagai argumen ke fungsi lain.

```
9
10 // Fungsi anonim sebagai argumen
11 setTimeout(function () {
12   console.log("Hello!");
13 }, 1000);
14
```

Bahrul Rozak - Kelas Software Developer

7. Arrow Function

Arrow function adalah cara baru untuk menulis fungsi dalam JavaScript, yang lebih singkat dan ekspresif. Menggunakan tanda panah => untuk mendefinisikan fungsi.

```
15 // Contoh penggunaan arrow function
16 let sayHello = () => {
17   console.log("Hello!");
18 };
19 sayHello();
20
```

Bahrul Rozak - Kelas Software Developer

Fungsi adalah konsep penting dalam JavaScript yang memungkinkan pengorganisasian yang baik dari kode dan meningkatkan kemampuan untuk mengelola logika aplikasi dengan lebih efisien.

Objek dan Array dalam JavaScript

Objek (Object)

Objek adalah struktur data yang dapat menyimpan berbagai jenis nilai (primitif dan non-primitif) dalam bentuk pasangan kunci-nilai. Dalam JavaScript, objek didefinisikan dengan menggunakan kurung kurawal {} dan berisi satu atau lebih pasangan properti dan nilai.

1. Properti objek adalah kunci (key) yang bersifat unik dan tidak boleh duplikat.
2. Nilai properti bisa berupa tipe data apapun, termasuk objek lain atau fungsi (metode).
3. Objek dapat diakses dan dimanipulasi propertinya menggunakan dot notation (objek.properti) atau bracket notation (objek['properti']).

Contoh:

```
File Edit Selection View Go Run ... team-assets
003-15 > object.js > ...
1 // Membuat objek 'mobil' dengan beberapa properti
2 let mobil = {
3   merk: "Toyota",
4   model: "Avanza",
5   tahun: 2020,
6   warna: "putih",
7 };
8
9 // Mengakses properti objek menggunakan dot notation
10 console.log(mobil.merk); // Output: Toyota
11
12 // Mengakses properti objek menggunakan bracket notation
13 console.log(mobil["tahun"]); // Output: 2020
14
15 // Menambahkan properti baru ke objek
16 mobil.harga = 200000000;
17 console.log(mobil); // Output: { merk: 'Toyota', model: 'Avanza', tahun: 2020, warna: 'putih', harga: 200000000 }
18
```

Bahrul Rozak - Kelas Software Developer

Array

Array adalah tipe data yang digunakan untuk menyimpan sejumlah nilai dalam urutan tertentu. Array dalam JavaScript dapat berisi berbagai jenis nilai dan memiliki metode bawaan untuk memanipulasi data di dalamnya.

1. Array didefinisikan dengan menggunakan kurung siku [] dan nilai-nilainya dipisahkan oleh koma.
2. Indeks array dimulai dari 0 (nol) untuk elemen pertama, 1 untuk elemen kedua, dan seterusnya.
3. Metode bawaan seperti push(), pop(), shift(), unshift(), splice(), dan lainnya digunakan untuk menambah, menghapus, atau mengubah elemen array.

Contoh:

```
003-IS> array.js > ...
1 // Membuat array 'buah' dengan beberapa nilai
2 let buah = ['apel', 'pisang', 'mangga', 'jeruk'];
3
4 // Mengakses nilai array berdasarkan indeks
5 console.log(buah[0]); // Output: apel
6
7 // Menambahkan nilai baru ke array
8 buah.push('anggur');
9 console.log(buah); // Output: ['apel', 'pisang', 'mangga', 'jeruk', 'anggur']
10
11 // Menghapus nilai terakhir dari array
12 buah.pop();
13 console.log(buah); // Output: ['apel', 'pisang', 'mangga', 'jeruk']
```

Bahrul Rozak - Kelas Software Developer

DOM (Document Object Model)

Pengertian DOM DOM (Document Object Model) adalah representasi struktural dari dokumen HTML (atau XML) yang memungkinkan JavaScript untuk mengakses dan memanipulasi elemen dalam dokumen tersebut. DOM merepresentasikan dokumen sebagai pohon struktur, di mana setiap elemen, atribut, dan teks dalam dokumen direpresentasikan sebagai objek dalam pohon.

Akses Elemen dengan DOM Dengan menggunakan DOM, kita dapat mengakses elemen-elemen dalam dokumen HTML dan mengubahnya sesuai kebutuhan. Berikut adalah cara umum untuk mengakses elemen dengan DOM:

1. `document.getElementById()`: Metode ini digunakan untuk mendapatkan elemen berdasarkan ID-nya.

Contoh:

```
let element = document.getElementById('idElement');
```

Bahrul Rozak - Kelas Software Developer

2. `document.getElementsByClassName()`: Metode ini digunakan untuk mendapatkan elemen berdasarkan kelas CSS-nya.

Contoh:

```
let elements = document.getElementsByClassName('className');
```

Bahrul Rozak - Kelas Software Developer

3. `Document.getElementsByTagName()`: Metode ini digunakan untuk mendapatkan elemen berdasarkan tag HTML-nya.

Contoh:

```
let elements = document.getElementsByTagName('tagName');
```

Bahrul Rozak - Kelas Software Developer

4. `Document.querySelector()`: Metode ini digunakan untuk mendapatkan elemen berdasarkan selector CSS.

Contoh:

```
let element = document.querySelector('.className');
```

Bahrul Rozak - Kelas Software Developer

5. `Document.querySelectorAll()`: Metode ini digunakan untuk mendapatkan semua elemen yang cocok dengan selector CSS.

Contoh:

```
let elements = document.querySelectorAll('.className');
```

Bahrul Rozak - Kelas Software Developer

Manipulasi Elemen dengan DOM Selain mengakses elemen, DOM juga memungkinkan kita untuk memanipulasi elemen tersebut. Berikut adalah beberapa contoh manipulasi elemen dengan DOM:

1. Mengubah Teks Elemen:

```
element.textContent = 'Teks Baru';
```

Bahrul Rozak - Kelas Software Developer

2. Menambahkan dan Menghapus Elemen:

```
// Menambahkan elemen baru
let newElement = document.createElement('div');
document.body.appendChild(newElement);

// Menghapus elemen
document.body.removeChild(newElement);
```

Bahrul Rozak - Kelas Software Developer

3. Mengubah Atribut Elemen:

```
element.setAttribute('namaAtribut', 'nilaiAtribut');
```

Bahrul Rozak - Kelas Software Developer

4. Menambahkan dan Menghapus Kelas CSS:

```
// Menambahkan kelas
element.classList.add('namaKelas');

// Menghapus kelas
element.classList.remove('namaKelas');
```

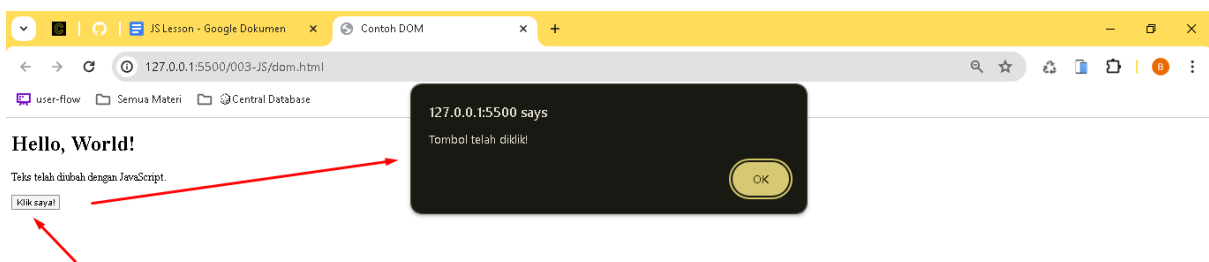
Bahrul Rozak - Kelas Software Developer

Contoh Penggunaan DOM dalam Praktek Misalkan kita memiliki dokumen HTML sebagai berikut:

```
003-JS > dom.html > @html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Contoh DOM</title>
5 </head>
6 <body>
7   <div id="container">
8     <h1>Hello, World!</h1>
9     <p id="paragraf">Ini adalah contoh paragraf.</p>
10    <button id="tombol">Klik saya!</button>
11  </div>
12  <script src="dom.js"></script>
13 </body>
14 </html>
```

Kita dapat menggunakan JavaScript untuk mengubah teks pada paragraf dan menambahkan event listener pada tombol:

```
003-JS > dom.js > _
1 // Mengubah teks pada paragraf
2 let paragraf = document.getElementById('paragraf');
3 paragraf.textContent = 'Teks telah diubah dengan JavaScript.';
4
5 // Menambahkan event listener pada tombol
6 let tombol = document.getElementById('tombol');
7 tombol.addEventListener('click', function() {
8   alert('Tombol telah diklik!');
9 });
```



ES6+ (ECMAScript 2015 dan seterusnya)

ES6 atau ECMAScript 2015 adalah versi JavaScript yang diperkenalkan pada tahun 2015. Ini adalah pembaruan besar dari standar JavaScript yang sebelumnya dan membawa banyak fitur baru dan perbaikan sintaksis yang membuat penulisan kode lebih mudah dan lebih ekspresif. Berikut adalah beberapa fitur utama dari ES6:

1. Let dan Const

let digunakan untuk mendeklarasikan variabel yang nilainya dapat diubah.
const digunakan untuk mendeklarasikan variabel yang nilainya tetap.

```
let x = 10;
x = 20; // Nilai x sekarang adalah 20

const y = 5;
y = 10; // Error, nilai const tidak dapat diubah
```

2. Arrow Functions

Bentuk singkat untuk menulis fungsi.
Tidak memiliki this, arguments, super, atau new.target.

```
const add = (a, b) => a + b;
```

3. Template Literals

Memungkinkan penggunaan string multi-baris dan interpolasi nilai variabel

```
const name = 'John';  
const greeting = `Hello, ${name}!`;
```

Bahrul Rozak - Kelas Software Developer

4. Destructuring

Memecah struktur data menjadi bagian-bagian yang lebih kecil.

```
const person = { name: 'Alice', age: 30 };  
const { name, age } = person;
```

Bahrul Rozak - Kelas Software Developer

5. Spread dan Rest Operators

Spread digunakan untuk menggabungkan array atau objek.

Rest digunakan untuk mengumpulkan elemen sisa menjadi array.

```
const arr1 = [1, 2, 3];  
const arr2 = [...arr1, 4, 5, 6];  
  
const sum = (...numbers) => numbers.reduce((acc, num) => acc + num, 0);
```

Bahrul Rozak - Kelas Software Developer

6. Classes

Penulisan kelas yang lebih dekat dengan paradigma pemrograman berorientasi objek.

```
class Animal {  
  constructor(name) {  
    this.name = name;  
  }  
  
  speak() {  
    console.log(`${this.name} makes a noise.`);  
  }  
}  
  
const dog = new Animal('Dog');  
dog.speak(); // Output: Dog makes a noise.
```

Bahrul Rozak - Kelas Software Developer

7. Modul

Memungkinkan pemisahan kode menjadi modul yang terpisah.

```
// math.js  
export const sum = (a, b) => a + b;  
  
// app.js  
import { sum } from './math';  
console.log(sum(1, 2)); // Output: 3
```

Bahrul Rozak - Kelas Software Developer

8. Promises

Digunakan untuk menangani operasi asynchronous.

```
const fetchData = () => {  
  return new Promise((resolve, reject) => {  
    // Lakukan operasi asynchronous  
    if (success) {  
      resolve(data);  
    } else {  
      reject(error);  
    }  
  });  
};  
  
fetchData()  
  .then(data => console.log(data))  
  .catch(error => console.error(error));
```

Bahrul Rozak - Kelas Software Developer

Asynchronous JavaScript

JavaScript adalah bahasa pemrograman yang bersifat single-threaded, artinya ia hanya dapat melakukan satu tugas pada satu waktu. Namun, dalam pengembangan web, seringkali kita perlu melakukan operasi yang bersifat asinkron, seperti mengambil data dari server, menjalankan animasi, atau menanggapi input pengguna tanpa menghentikan operasi lainnya.

Untuk menangani operasi-operasi ini, JavaScript menyediakan mekanisme asinkron, yang memungkinkan kita untuk menjalankan beberapa tugas sekaligus tanpa harus menunggu tugas sebelumnya selesai. Terdapat beberapa cara untuk mengimplementasikan asinkronitas dalam JavaScript, seperti menggunakan callback, promises, dan async/await.

1. Callback:

Callback adalah salah satu cara paling sederhana untuk mengimplementasikan asinkronitas dalam JavaScript. Sebuah callback adalah sebuah fungsi yang diberikan sebagai argumen kepada fungsi lainnya, dan akan dipanggil setelah operasi asinkron selesai dilakukan.

```
function fetchData(callback) {
  setTimeout(() => {
    callback("Data berhasil diambil");
  }, 2000);
}

function main() {
  console.log("Memulai pengambilan data");

  fetchData(function(data) {
    console.log(data);
    console.log("Pengambilan data selesai");
  });

  console.log("Melakukan operasi lainnya");
}

main();
```

Bahrul Rozak - Kelas Software Developer

2. Promises:

Promises adalah mekanisme yang lebih baik untuk mengelola asinkronitas dibandingkan callback. Sebuah promise mewakili nilai yang belum tentu diketahui saat pembuatan promise tersebut, tetapi akan diketahui pada suatu saat di masa depan. Sebuah promise memiliki dua state, yaitu pending (belum diketahui nilainya), fulfilled (berhasil), dan rejected (gagal)

Contoh:

```

function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve("Data berhasil diambil");
        }, 2000);
    });
}

function main() {
    console.log("Memulai pengambilan data");

    fetchData()
        .then((data) => {
            console.log(data);
            console.log("Pengambilan data selesai");
        })
        .catch((error) => {
            console.error("Terjadi kesalahan:", error);
        });

    console.log("Melakukan operasi lainnya");
}

main();

```

Bahrul Rozak - Kelas Software Developer

3. Async/Await:

Async/Await adalah fitur yang diperkenalkan dalam ECMAScript 2017 untuk membuat kode asinkron terlihat seperti kode synchronous biasa. async digunakan untuk mendefinisikan sebuah fungsi sebagai asynchronous, sementara await digunakan untuk menunggu hasil dari sebuah promise.

```

function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve("Data berhasil diambil");
        }, 2000);
    });
}

async function main() {
    console.log("Memulai pengambilan data");

    try {
        const data = await fetchData();
        console.log(data);
        console.log("Pengambilan data selesai");
    } catch (error) {
        console.error("Terjadi kesalahan:", error);
    }

    console.log("Melakukan operasi lainnya");
}

main();

```

Bahrul Rozak - Kelas Software Developer

Dengan menggunakan mekanisme asinkron seperti callback, promises, dan async/await, kita dapat mengelola asinkronitas dalam JavaScript dengan lebih efektif dan membuat aplikasi web menjadi lebih responsif dan interaktif. Mantap : v

JSON dan AJAX

JSON adalah format pertukaran data yang ringan dan mudah dibaca oleh manusia. Format ini sering digunakan untuk mentransmisikan data antar server dan aplikasi web. JSON bersifat independen terhadap bahasa pemrograman, artinya bisa digunakan dalam berbagai bahasa pemrograman termasuk JavaScript, Python, PHP, dll.

```
{
  "nama": "John Doe",
  "umur": 30,
  "alamat": {
    "jalan": "Jl. Jendral Sudirman",
    "kota": "Jakarta"
  },
  "hobi": ["berenang", "membaca"]
}
```

Bahrul Rozak - Kelas Software Developer

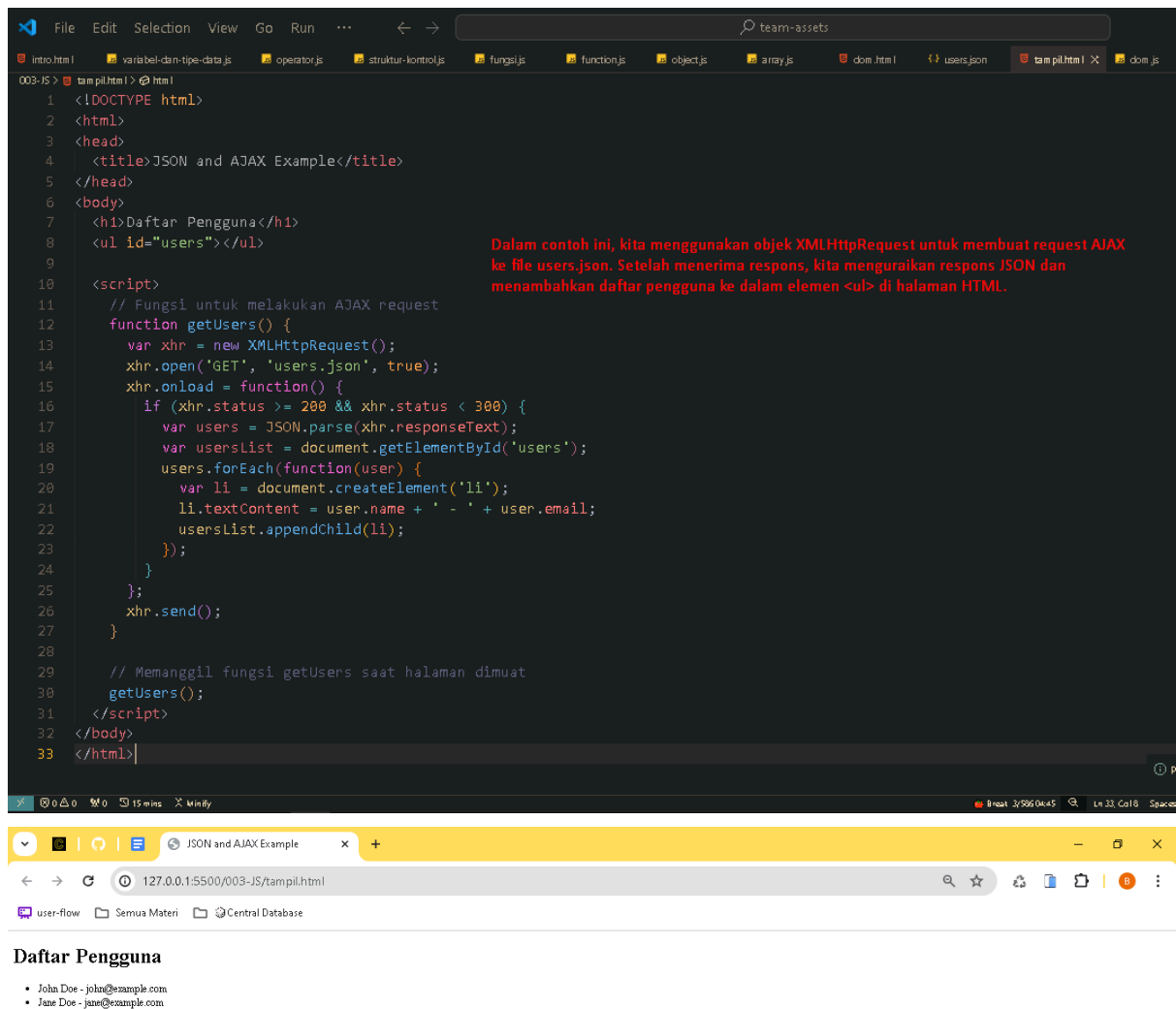
AJAX adalah teknik pengembangan web yang digunakan untuk membuat aplikasi web menjadi lebih interaktif dengan melakukan pertukaran data secara asynchronous antara browser dan server. Dengan AJAX, halaman web dapat memperbarui konten tanpa perlu me-refresh seluruh halaman.

Menggunakan AJAX untuk Mengambil Data JSON

Pertama, kita akan membuat file JSON sederhana yang berisi daftar pengguna dalam file bernama users.json

```
003-JS > {} users.json > ...
1  [
2    {
3      "id": 1,
4      "name": "John Doe",
5      "email": "john@example.com"
6    },
7    {
8      "id": 2,
9      "name": "Jane Doe",
10     "email": "jane@example.com"
11   }
12 ]
13 ,
```

Kemudian, kita akan membuat file HTML dan JavaScript untuk mengambil dan menampilkan data JSON menggunakan AJAX:



Pengelolaan Kesalahan (Error Handling) dalam JavaScript

Error handling adalah proses mengelola kesalahan (error) yang terjadi dalam sebuah program. Dalam JavaScript, terdapat beberapa mekanisme untuk mengelola kesalahan, salah satunya adalah menggunakan try-catch. Berikut adalah penjelasan mengenai masing-masing komponen error handling:

1. **try:** Blok kode yang potensial menghasilkan error ditempatkan di dalam blok try. Jika error terjadi di dalam blok try, eksekusi kode akan berhenti, dan kontrol akan diteruskan ke blok catch.
2. **catch:** Blok catch digunakan untuk menangkap error yang dihasilkan oleh blok try. Anda dapat menentukan jenis error yang ingin ditangkap, atau menggunakan generic catch tanpa argumen untuk menangkap semua jenis error.
3. **finally:** Blok finally opsional dan selalu dieksekusi terlepas dari apakah error terjadi atau tidak. Biasanya digunakan untuk membersihkan sumber daya atau mengeksekusi kode akhir.

Berikut adalah contoh penggunaan try-catch dalam JavaScript:

```
try {
  // Kode yang potensial menghasilkan error
  console.log(undefinedVariable); // Menimbulkan ReferenceError
} catch (error) {
  // Blok ini akan dieksekusi jika error terjadi
  console.error('Terjadi kesalahan:', error.message);
} finally {
  // Blok finally akan dieksekusi selalu
  console.log('Proses error handling selesai.');
```

Bahrul Rozak - Kelas Software Developer

Misalkan kita memiliki sebuah fungsi untuk membagi dua angka, tetapi ingin menghindari terjadinya error jika pengguna memasukkan nilai yang tidak valid (misalnya, string atau pembagian dengan nol). Berikut adalah contoh implementasinya:

```
function divide(a, b) {
  try {
    if (typeof a !== 'number' || typeof b !== 'number') {
      throw new Error('Argumen harus berupa angka.');
```

Bahrul Rozak - Kelas Software Developer

```
    }
    if (b === 0) {
      throw new Error('Pembagian dengan nol tidak diizinkan.');
```

Dalam contoh di atas, fungsi divide akan memeriksa apakah kedua argumen yang diberikan adalah angka dan apakah pembagian dengan nol dihindari. Jika terjadi kesalahan, fungsi akan mengembalikan null dan menampilkan pesan kesalahan.

Storage di Browser

Storage di browser digunakan untuk menyimpan data secara lokal di komputer pengguna. Terdapat beberapa jenis storage di browser, yaitu:

1. **LocalStorage:** Digunakan untuk menyimpan data dalam bentuk key-value pair tanpa batas waktu. Data localStorage akan tetap tersimpan bahkan setelah browser ditutup.
2. **SessionStorage:** Digunakan untuk menyimpan data dalam bentuk key-value pair selama sesi browser aktif. Data sessionStorage akan terhapus saat browser ditutup.

Berikut adalah contoh penggunaan localStorage dan sessionStorage dalam JavaScript:


```
// Menyimpan data ke localStorage
localStorage.setItem('nama', 'John Doe');

// Mengambil data dari localStorage
const nama = localStorage.getItem('nama');
console.log(nama); // Output: John Doe

// Menghapus data dari localStorage
localStorage.removeItem('nama');

// Menyimpan data ke sessionStorage
sessionStorage.setItem('kota', 'Jakarta');

// Mengambil data dari sessionStorage
const kota = sessionStorage.getItem('kota');
console.log(kota); // Output: Jakarta

// Menghapus data dari sessionStorage
sessionStorage.removeItem('kota');
```

Bahrul Rozak - kelas Software Developer

Studi Kasus Sederhana: Aplikasi To-Do List

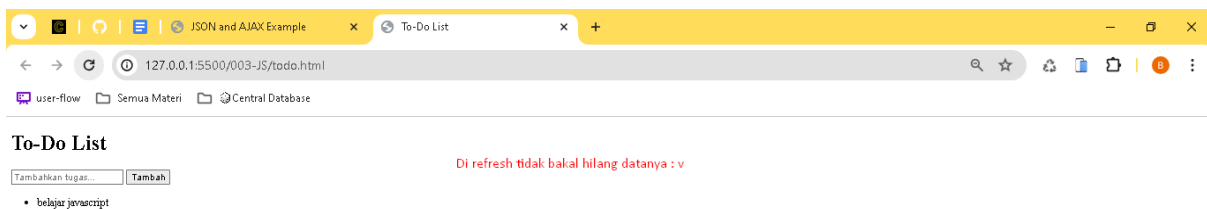
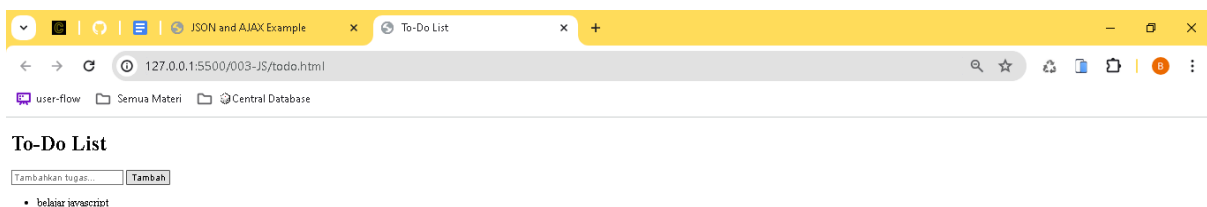
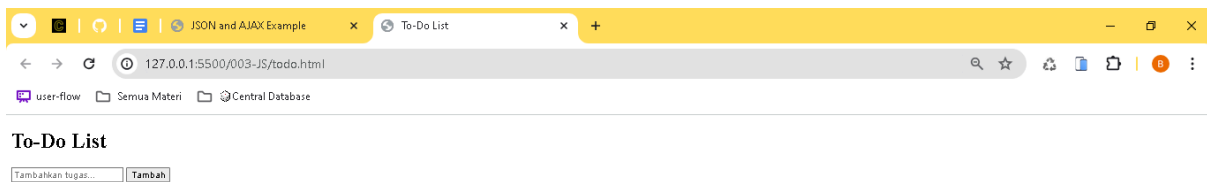
Berikut adalah contoh penerapan storage di browser dalam membuat aplikasi to-do list sederhana dengan JavaScript

```
003-JS > todo.html @ html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>To-Do List</title>
5 </head>
6 <body>
7   <h1>To-Do List</h1>
8   <input type="text" id="taskInput" placeholder="Tambahkan tugas...">
9   <button onclick="addTask()">Tambah</button>
10  <ul id="taskList"></ul>
11
12  <script src="script.js"></script>
13 </body>
14 </html>
```

Bahrul Rozak - Kelas Software Developer

```
File Edit Selection View Go Run ... team-assets
/html | variabel-dan-tipe-data.js | operator.js | struktur-kontrol.js | fungsi.js | function.js | object.js | array.js | dom.html | users.json | tampil.html | todo.html
003-JS > script.js > saveTask
1 const taskInput = document.getElementById('taskInput');
2 const taskList = document.getElementById('taskList');
3
4 // Memuat tugas dari storage
5 document.addEventListener('DOMContentLoaded', () => {
6     const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
7     tasks.forEach(task => {
8         addTaskToDOM(task);
9     });
10 });
11
12 function addTask() {
13     const taskText = taskInput.value.trim();
14     if (taskText !== '') {
15         const task = { id: Date.now(), text: taskText };
16         addTaskToDOM(task);
17         saveTask(task);
18         taskInput.value = '';
19     }
20 }
21
22 function addTaskToDOM(task) {
23     const taskItem = document.createElement('li');
24     taskItem.textContent = task.text;
25     taskList.appendChild(taskItem);
26 }
27
28 function saveTask(task) {
29     const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
30     tasks.push(task);
31     localStorage.setItem('tasks', JSON.stringify(tasks));
32 }
```

Bahrul Rozak - Kelas Software Developer



Dalam contoh ini, setiap kali pengguna menambahkan tugas baru, tugas tersebut akan disimpan dalam localStorage. Saat halaman dimuat, tugas-tugas yang sudah disimpan akan dimuat kembali ke dalam daftar tugas. Dengan menggunakan storage di browser, aplikasi to-do list ini dapat menyimpan tugas-tugas yang sudah ditambahkan oleh pengguna, bahkan setelah halaman direfresh atau browser ditutup.

Modularisasi dalam JavaScript

Modularisasi dalam JavaScript adalah pendekatan untuk memecah kode menjadi bagian-bagian yang lebih kecil dan terpisah, yang disebut modul. Modul memungkinkan Anda untuk mengorganisasi kode secara lebih terstruktur, memungkinkan penggunaan ulang kode, serta mempermudah pemeliharaan dan pengembangan kode.

Dalam JavaScript, modularisasi dapat dilakukan menggunakan fitur seperti import dan export yang diperkenalkan dalam ECMAScript 6 (ES6) atau versi JavaScript yang lebih baru.

Berikut adalah contoh penggunaan modularisasi dalam JavaScript dengan menggunakan ES6:

1. Module A (moduleA.js)

```
// moduleA.js
export const hello = () => {
  return "Hello, ";
};
```

Bahrul Rozak - Kelas Software Developer

2. Module B (moduleB.js)

```
// moduleB.js
export const world = () => {
  return "World!";
};
```

Bahrul Rozak - Kelas Software Developer

3. Main Module (main.js)

```
// main.js
import { hello } from './moduleA.js';
import { world } from './moduleB.js';

const message = hello() + world();
console.log(message);
```

Bahrul Rozak - Kelas Software Developer

Studi Kasus Sederhana: Aplikasi To-Do List

Dalam studi kasus ini, kita akan menggunakan modularisasi untuk memisahkan logika aplikasi to-do list menjadi beberapa modul terpisah. Modul-modul ini akan mencakup logika untuk menambah, menghapus, dan menampilkan daftar tugas.

1. Modul untuk Menambah Tugas (addTask.js)

```
003-JS > addTask.js > ...
1 export const addTask = (taskList, newTask) => {
2   taskList.push(newTask);
3 };
```

2. Modul untuk Menghapus Tugas (deleteTask.js)

```

003-JS > deleteTask.js > ...
1 export const deleteTask = (taskList, taskIndex) => {
2   taskList.splice(taskIndex, 1);
3 };
4

```

3. Modul untuk Menampilkan Daftar Tugas (showTasks.js)

```

003-JS > showTasks.js > ...
1 export const showTasks = (taskList) => {
2   taskList.forEach((task, index) => {
3     console.log(`${index + 1}. ${task}`);
4   });
5 };
6

```

4. Main Module (main.js)

```

003-JS > main.js > ...
1 import { addTask } from './addTask.js';
2 import { deleteTask } from './deleteTask.js';
3 import { showTasks } from './showTasks.js';
4
5 let tasks = ["Belajar JavaScript", "Mengerjakan Tugas"];
6
7 addTask(tasks, "Menyusun Presentasi");
8 showTasks(tasks);
9
10 deleteTask(tasks, 0);
11 showTasks(tasks);

```

Dalam contoh ini, kita telah berhasil memisahkan logika aplikasi to-do list menjadi modul-modul terpisah, yang membuat kode lebih mudah dimengerti, dikelola, dan dikembangkan.

Testing JavaScript

Testing adalah proses penting dalam pengembangan perangkat lunak untuk memastikan bahwa kode yang ditulis berfungsi dengan baik dan sesuai dengan yang diharapkan. Dalam konteks JavaScript, terdapat beberapa jenis tes yang umum digunakan:

1. Unit Testing: Tes unit digunakan untuk menguji fungsi atau komponen kecil dalam isolasi. Hal ini memungkinkan pengembang untuk memastikan bahwa setiap bagian dari kode berfungsi seperti yang diinginkan.
- 2.
3. Integration Testing: Tes integrasi menguji bagaimana komponen-komponen dalam sistem bekerja bersama. Ini memastikan bahwa semua bagian sistem berfungsi dengan baik saat digabungkan.
- 4.
5. End-to-End Testing: Tes end-to-end menguji aplikasi dari awal hingga akhir untuk memastikan bahwa aplikasi berperilaku seperti yang diharapkan oleh pengguna.
- 6.
7. Static Analysis: Analisis statis menggunakan alat seperti ESLint atau JSHint untuk mengidentifikasi potensi bug atau masalah dalam kode JavaScript tanpa harus menjalankan kode.

Jest: Jest adalah kerangka pengujian JavaScript yang populer dan kuat yang biasanya digunakan untuk tes unit dan integrasi. Berikut adalah contoh penggunaan Jest untuk menguji fungsi sederhana:

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

module.exports = { add };
```

Bahrul Rozak - Kelas Software Developer

```
// math.test.js
const { add } = require('./math');

test('adds 1 + 2 to equal 3', () => {
  expect(add(1, 2)).toBe(3);
});
```

Bahrul Rozak - Kelas Software Developer

Dalam contoh ini, kita mendefinisikan fungsi `add` di file `math.js`, dan kemudian kita menguji fungsi tersebut menggunakan Jest di file `math.test.js`. Kita memastikan bahwa fungsi `add` mengembalikan hasil yang benar.

Best Practices dan Performance Optimization in JavaScript

Best Practices:

1. Menulis kode yang mudah dibaca dan dipahami oleh orang lain.
2. Menggunakan nama variabel yang deskriptif.
3. Menggunakan komentar untuk menjelaskan kode yang kompleks.
4. Menggunakan fitur baru dari JavaScript (seperti arrow functions dan destructuring) untuk meningkatkan kejelasan dan kecepatan penulisan kode.

Performance Optimization:

1. Menghindari penggunaan fungsi yang berat secara berulang-ulang dalam loop.
2. Menggunakan metode looping yang efisien (misalnya, `forEach()` untuk array).
3. Meminimalkan akses ke DOM dalam loop.
4. Menggunakan mekanisme caching untuk data yang sering digunakan.

```
// Membuat fungsi untuk menghitung faktorial
function factorial(n) {
  if (n === 0 || n === 1) {
    return 1;
  }
  return n * factorial(n - 1);
}

// Memanggil fungsi faktorial dengan input 5
console.log(factorial(5));

// Contoh penggunaan looping yang efisien
const array = [1, 2, 3, 4, 5];
array.forEach((item) => {
  console.log(item);
});

// Contoh penggunaan caching untuk data yang sering digunakan
function getData() {
  if (!getData.cache) {
    getData.cache = {
      // Data yang sering digunakan disimpan di sini
      name: "John Doe",
      age: 30,
      // ...
    };
  }
  return getData.cache;
}

// Memanggil fungsi getData() untuk mendapatkan data
console.log(getData().name);
console.log(getData().age);
```

Bahrul Rozak - Kelas Software Developer

Studi Kasus Sederhana: Optimasi Fungsi Penjumlahan

Kita akan membuat fungsi sederhana untuk menjumlahkan angka dari 1 hingga n dengan dua cara: menggunakan looping `for` dan menggunakan formula matematika. Kemudian, kita akan mengukur waktu eksekusi keduanya untuk melihat mana yang lebih efisien.

```
// Fungsi untuk menjumlahkan angka dari 1 hingga n menggunakan looping
function sumWithLoop(n) {
  let sum = 0;
  for (let i = 1; i <= n; i++) {
    sum += i;
  }
  return sum;
}

// Fungsi untuk menjumlahkan angka dari 1 hingga n menggunakan formula matematika
function sumWithFormula(n) {
  return (n * (n + 1)) / 2;
}

// Mengukur waktu eksekusi fungsi sumWithLoop
console.time('sumWithLoop');
console.log(sumWithLoop(1000000));
console.timeEnd('sumWithLoop');

// Mengukur waktu eksekusi fungsi sumWithFormula
console.time('sumWithFormula');
console.log(sumWithFormula(1000000));
console.timeEnd('sumWithFormula');
```

Bahrul Rozak - Kelas Software Developer

Studi Kasus 1 : Aplikasi To-Do List: Teori dan Implementasi

Aplikasi To-Do List adalah aplikasi sederhana yang memungkinkan pengguna untuk mencatat dan mengelola daftar tugas yang harus dilakukan. Aplikasi ini biasanya terdiri dari fitur-fitur seperti menambah tugas baru, menandai tugas sebagai selesai, dan menghapus tugas. Dalam pengembangan aplikasi To-Do List menggunakan JavaScript, terdapat beberapa konsep yang perlu dipahami:

1. DOM Manipulation: Untuk menambah, mengedit, atau menghapus elemen HTML.
2. Event Handling: Untuk menangani interaksi pengguna seperti klik tombol atau tekan tombol keyboard.
3. Local Storage: Untuk menyimpan data tugas secara lokal di browser.

Berikut adalah contoh implementasi sederhana dari aplikasi To-Do List menggunakan HTML, CSS, dan JavaScript:

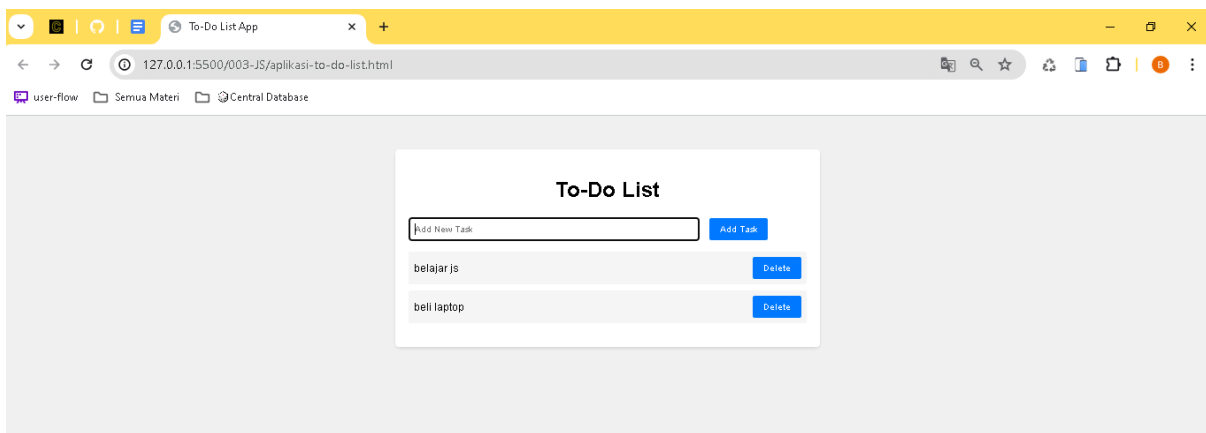
```
003-JS > aplikasi-to-do-list.html > @html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>To-Do List App</title>
7   <link rel="stylesheet" href="todo.css">
8 </head>
9 <body>
10   <div class="container">
11     <h1>To-Do List</h1>
12     <input type="text" id="taskInput" placeholder="Add New Task">
13     <button id="addTaskBtn">Add Task</button>
14     <ul id="taskList"></ul>
15   </div>
16   <script src="script.js"></script>
17 </body>
18 </html>
```

```
File Edit Selection View Go Run ... team-assets
aplikasi-to-do-list.html | todoscss | ... | todoscss |
003-JS > todoscss > ...
1 body {
2   font-family: Arial, sans-serif;
3   margin: 0;
4   padding: 0;
5   background-color: #f4f4f4;
6 }
7
8 .container {
9   max-width: 600px;
10  margin: 50px auto;
11  padding: 20px;
12  background-color: #fff;
13  border-radius: 5px;
14  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
15 }
16
17 h1 {
18   text-align: center;
19 }
20
21 input[type="text"] {
22   width: 70%;
23   padding: 8px;
24   margin-right: 10px;
25   border-radius: 3px;
26   border: 1px solid #ccc;
27 }
28
29 button {
30   padding: 8px 16px;
31   border-radius: 3px;
32   background-color: #007bff;
33   color: #fff;
34   border: none;
35 }
36
37 button:hover {
38 }
39
40
41
42 ul {
43   list-style-type: none;
44   padding: 0;
45 }
46
47 li {
48   margin-bottom: 10px;
49   padding: 8px;
50   background-color: #f9f9f9;
51   border-radius: 3px;
52   display: flex;
53   align-items: center;
54   justify-content: space-between;
55 }
56
57 .completed {
58   text-decoration: line-through;
59   color: #999;
60 }
61
```

```
003-JS > todojs > ...
1 // Get DOM elements
2 const taskInput = document.getElementById('taskInput');
3 const addTaskBtn = document.getElementById('addTaskBtn');
4 const taskList = document.getElementById('taskList');
5
6 // Load tasks from Local Storage
7 let tasks = JSON.parse(localStorage.getItem('tasks')) || [];
8
9 // Function to render tasks
10 function renderTasks() {
11   taskList.innerHTML = '';
12   tasks.forEach((task, index) => {
13     const li = document.createElement('li');
14     li.innerHTML = `
15       <span>${task.text}</span>
16       <button class="deleteBtn" data-index="${index}">Delete</button>
17     `;
18     if (task.completed) {
19       li.classList.add('completed');
20     }
21     taskList.appendChild(li);
22   });
23 }
24
25 // Add task
26 function addTask() {
27   const text = taskInput.value.trim();
28   if (text !== '') {
29     tasks.push({ text, completed: false });
30     localStorage.setItem('tasks', JSON.stringify(tasks));
31     renderTasks();
32     taskInput.value = '';
33   }
34 }
35
```



```
003-JS > | 003-JS > ...
36 // Delete task
37 function deleteTask(index) {
38     tasks.splice(index, 1);
39     localStorage.setItem('tasks', JSON.stringify(tasks));
40     renderTasks();
41 }
42
43 // Toggle task completion
44 function toggleTaskCompletion(index) {
45     tasks[index].completed = !tasks[index].completed;
46     localStorage.setItem('tasks', JSON.stringify(tasks));
47     renderTasks();
48 }
49
50 // Event listeners
51 addTaskBtn.addEventListener('click', addTask);
52 taskInput.addEventListener('keypress', function(event) {
53     if (event.key === 'Enter') {
54         addTask();
55     }
56 });
57 taskList.addEventListener('click', function(event) {
58     if (event.target.classList.contains('deleteBtn')) {
59         const index = event.target.dataset.index;
60         deleteTask(index);
61     } else {
62         const index = event.target.parentElement.dataset.index;
63         toggleTaskCompletion(index);
64     }
65 });
66
67 // Initial render
68 renderTasks();
```

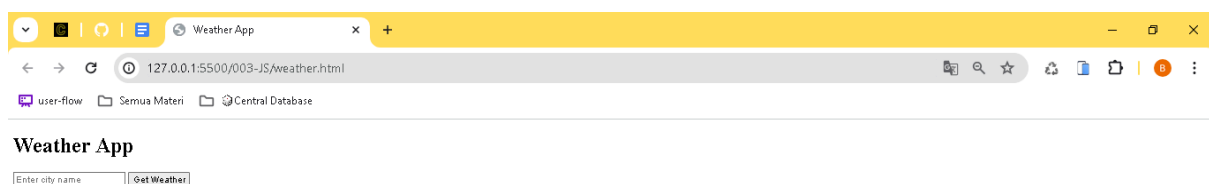


Studi Kasus 2 : Aplikasi Cuaca

Aplikasi Cuaca adalah aplikasi yang menampilkan informasi cuaca saat ini maupun perkiraan cuaca di masa depan. Aplikasi ini biasanya mengambil data dari layanan cuaca melalui API (Application Programming Interface) dan menampilkannya dalam bentuk yang mudah dipahami oleh pengguna, seperti ikon cuaca, suhu, kelembaban, dan lain-lain.

Berikut adalah contoh sederhana aplikasi cuaca menggunakan JavaScript dan API dari OpenWeatherMap. Aplikasi ini akan menampilkan suhu dan deskripsi cuaca berdasarkan kota yang dimasukkan pengguna.

```
003-JS> @ weather.html > @ html
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport"
6              content="width=device-width, initial-scale=1.0">
7          <title>Weather App</title>
8      </head>
9      <body>
10         <h1>Weather App</h1>
11         <input type="text" id="cityInput" placeholder="Enter city name">
12         <button onclick="getWeather()">Get Weather</button>
13         <div id="weatherInfo"></div>
14
15         <script>
16         function getWeather() {
17             const city = document.getElementById('cityInput').value;
18             const apiKey = 'YOUR_API_KEY'; // Replace with your OpenWeatherMap API key
19             const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;
20
21             fetch(apiUrl)
22                 .then(response => response.json())
23                 .then(data => {
24                     const weatherInfo = document.getElementById('weatherInfo');
25                     weatherInfo.innerHTML = `
26                         <h2>${data.name}, ${data.sys.country}</h2>
27                         <p>Temperature: ${data.main.temp}°C</p>
28                         <p>Weather: ${data.weather[0].description}</p>
29                     `;
30                 })
31                 .catch(error => {
32                     console.error('Error fetching data:', error);
33                 });
34         }
35         </script>
36     </body>
37 </html>
```



Studi Kasus 3 : E-commerce Website

E-commerce Website adalah website yang memungkinkan pengguna untuk melakukan transaksi jual-beli secara online. Pada aplikasi E-commerce, pengguna dapat melihat daftar produk, menambahkan produk ke dalam keranjang belanja, melakukan pembayaran, dan melacak status pengiriman barang. Pengembangan E-commerce Website melibatkan integrasi dengan API untuk mendapatkan informasi produk, serta implementasi sistem pembayaran dan manajemen keranjang belanja.

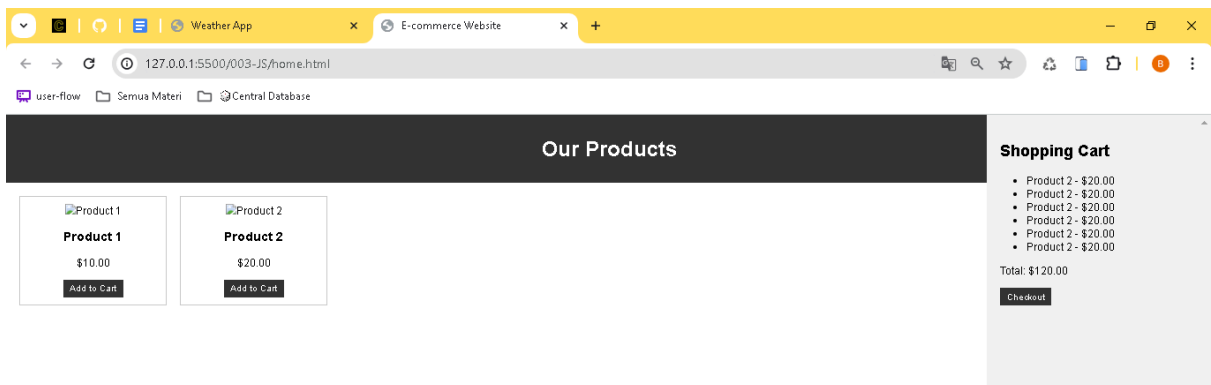
Berikut adalah contoh studi kasus sederhana untuk E-commerce Website menggunakan HTML, CSS, dan JavaScript. Kami akan membuat halaman sederhana yang menampilkan daftar produk dan memungkinkan pengguna untuk menambahkan produk ke dalam keranjang belanja.

```
003-jS > home.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>E-commerce Website</title>
8      <link rel="stylesheet" href="styles.css">
9  </head>
10 <body>
11     <header>
12         <h1>Our Products</h1>
13     </header>
14     <div class="products">
15         <div class="product">
16             
17             <h3>Product 1</h3>
18             <p>$10.00</p>
19             <button class="add-to-cart" data-id="1">Add to Cart</button>
20         </div>
21         <div class="product">
22             
23             <h3>Product 2</h3>
24             <p>$20.00</p>
25             <button class="add-to-cart" data-id="2">Add to Cart</button>
26         </div>
27         <!-- More products here -->
28     </div>
29     <aside>
30         <h2>Shopping Cart</h2>
31         <ul id="cart-items">
32             <!-- Cart items will be added here dynamically -->
33         </ul>
34         <p>Total: <span id="cart-total">0.00</span></p>
35         <button id="checkout">Checkout</button>
36     </aside>
37     <script src="script.js"></script>
38 </body>
39 </html>
```

```
File Edit Selection View Go Run ... team-assets
0045: @ homecss>
1 body {
2   font-family: Arial, sans-serif;
3   margin: 0;
4   padding: 0;
5 }
6
7 header {
8   background-color: #333;
9   color: white;
10  text-align: center;
11  padding: 10px 0;
12 }
13
14 .products {
15   display: flex;
16   flex-wrap: wrap;
17   gap: 20px;
18   padding: 20px;
19 }
20
21 .product {
22   width: 200px;
23   border: 1px solid #ccc;
24   padding: 10px;
25   text-align: center;
26 }
27
28 aside {
29   background-color: #f4f4f4;
30   padding: 20px;
31   position: fixed;
32   top: 0;
33   right: 0;
34   width: 300px;
35   height: 100%;
36   overflow-y: scroll;
37 }
38
```

```
homejs X
003-JS> homejs>
1 const products = [
2   { id: 1, name: "Product 1", price: 10.0 },
3   { id: 2, name: "Product 2", price: 20.0 },
4   // Add more products here
5 ];
6
7 const cartItems = [];
8
9 function renderProducts() {
10   const productsContainer = document.querySelector(".products");
11   productsContainer.innerHTML = "";
12   products.forEach((product) => {
13     const productElement = document.createElement("div");
14     productElement.classList.add("product");
15     productElement.innerHTML = `
16       
17       <h3>${product.name}</h3>
18       <p>${product.price.toFixed(2)}</p>
19       <button class="add-to-cart" data-id="${
20         product.id
21       }">Add to Cart</button>
22     `;
23     productsContainer.appendChild(productElement);
24   });
25 }
26
27 function renderCart() {
28   const cartItemsElement = document.getElementById("cart-items");
29   const cartTotalElement = document.getElementById("cart-total");
30   cartItemsElement.innerHTML = "";
31   let total = 0;
32   cartItems.forEach((item) => {
33     const itemElement = document.createElement("li");
34     itemElement.textContent = `${item.name} - ${item.price.toFixed(2)}`;
35     cartItemsElement.appendChild(itemElement);
36     total += item.price;
37   });
38   cartTotalElement.textContent = total.toFixed(2);
39 }
40
```

```
File Edit Selection View Go Run ... team-ass
homejs x
003-JS > homejs > .
41 document.addEventListener("DOMContentLoaded", () => {
44   document.addEventListener("click", (event) => {
45     if (event.target.classList.contains("add-to-cart")) {
46       const productId = parseInt(event.target.getAttribute("data-id"));
47       const product = products.find((p) => p.id === productId);
48       if (product) {
49         cartItems.push(product);
50         renderCart();
51       }
52     }
53   });
54
55   document.getElementById("checkout").addEventListener("click", () => {
56     alert("Checkout not implemented in this example!");
57   });
58 });
59
```



Dalam contoh ini, kita membuat halaman sederhana yang menampilkan daftar produk dan keranjang belanja. Pengguna dapat menambahkan produk ke dalam keranjang belanja dengan mengklik tombol "Add to Cart". Namun, fungsi checkout tidak diimplementasikan dalam contoh ini. Anda dapat mengembangkan fitur ini lebih lanjut dengan menambahkan integrasi API untuk pembayaran dan manajemen keranjang belanja yang lebih kompleks.

Selesai sudah materi javascript, admin harap teman-teman setelah mengikuti tutorial ini sudah bisa menggunakan javascript untuk berbagai studi kasus nyata