

Laporan Praktikum IV

Desain Web



Oleh:

Andi Ode

Larios

4523210015

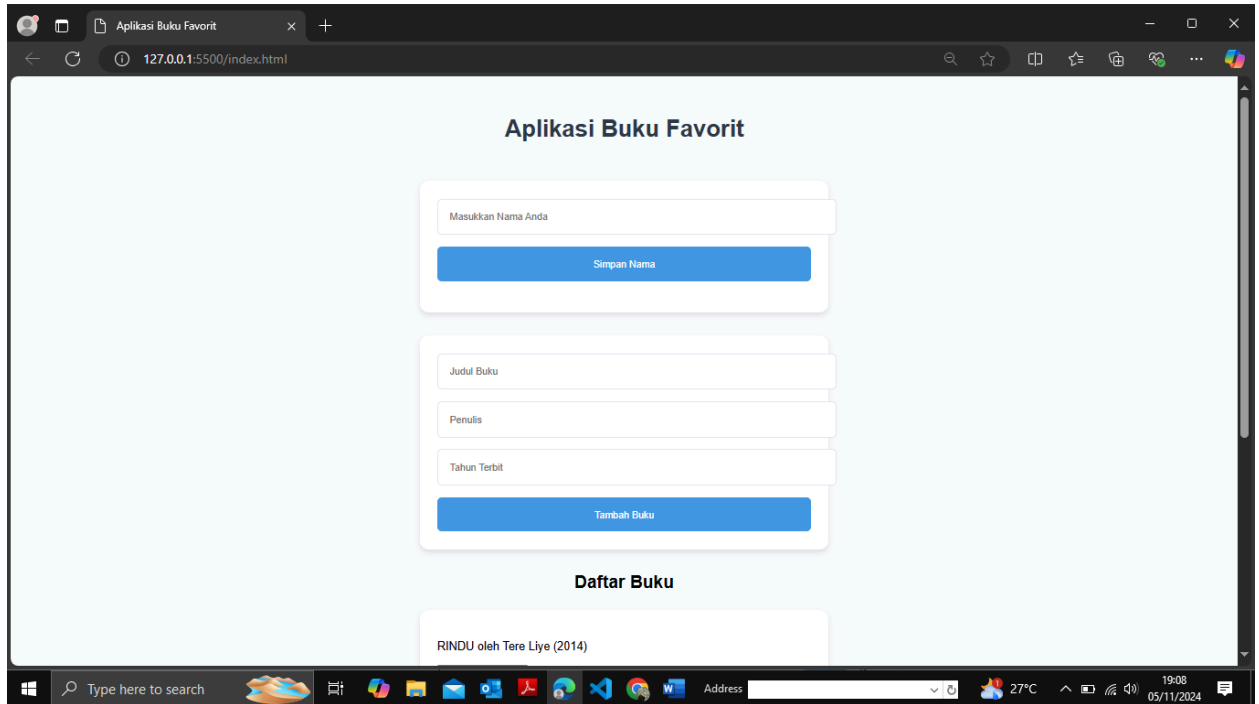
Dosen:

Adi Wahyu Pribadi , S.Si., M.Kom

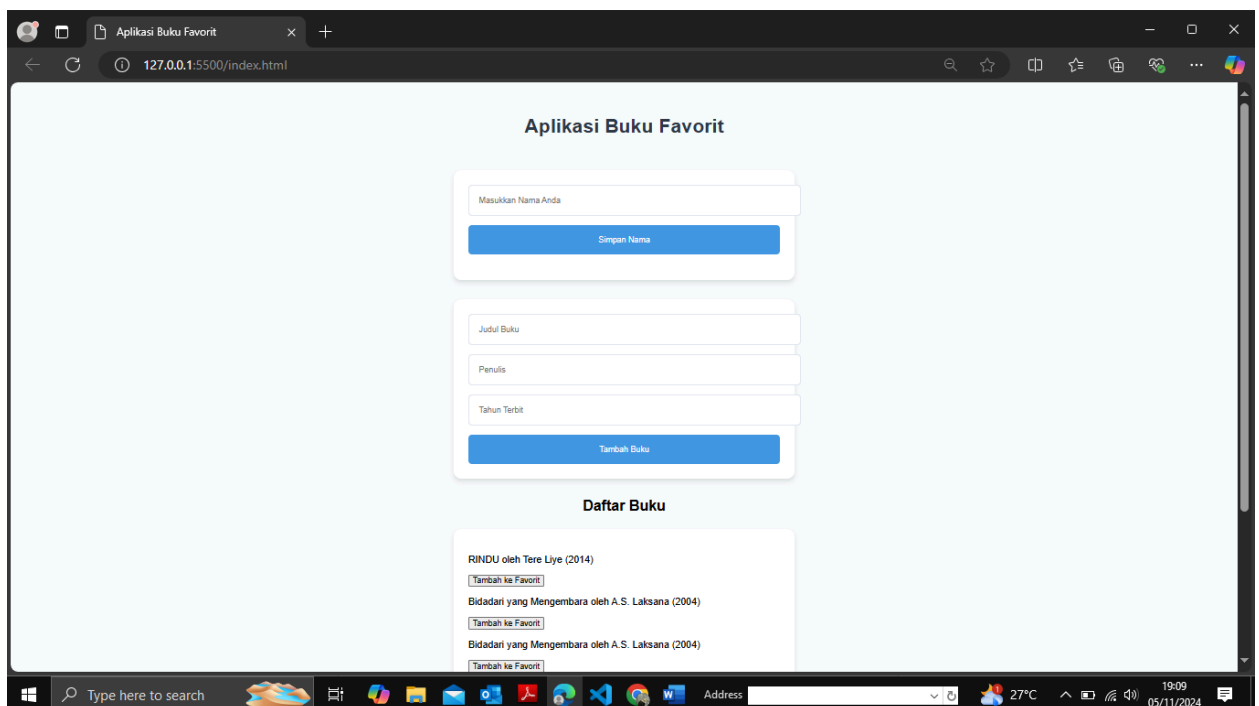
**S1-TEKNIK
INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS
PANCASILA 2023/2024**

Tugas Uji Aplikasi

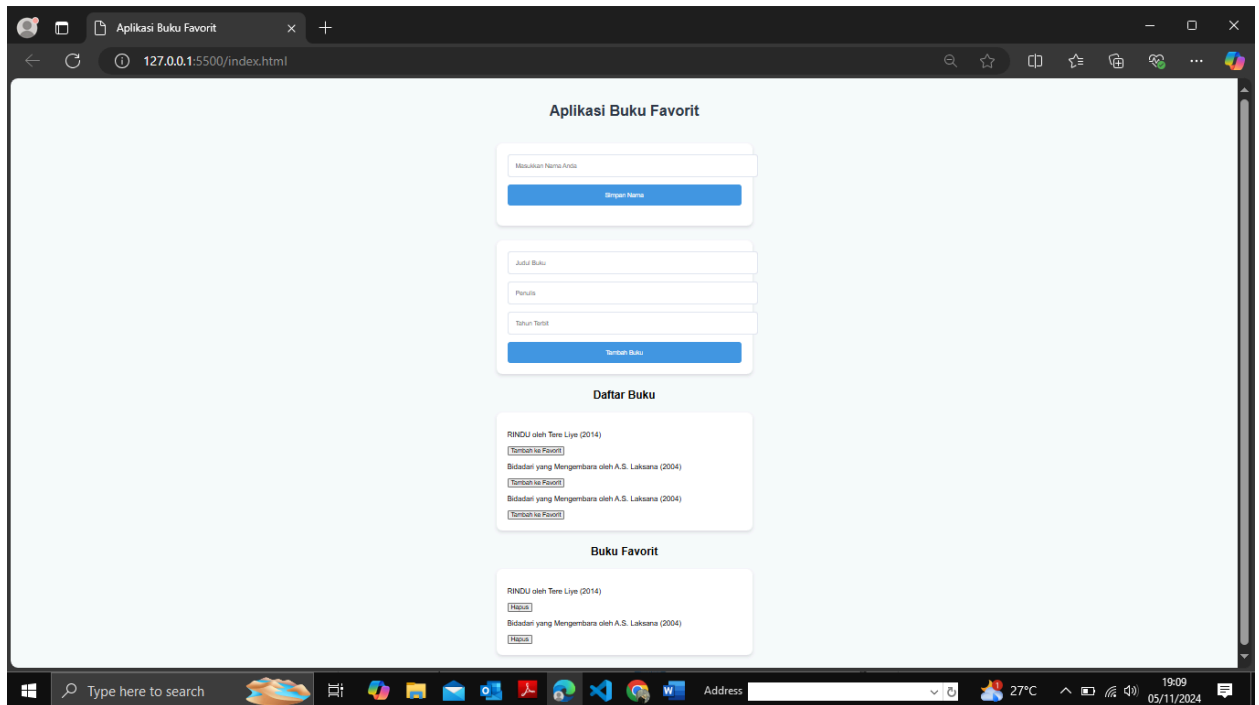
1. Buka index.html di browser



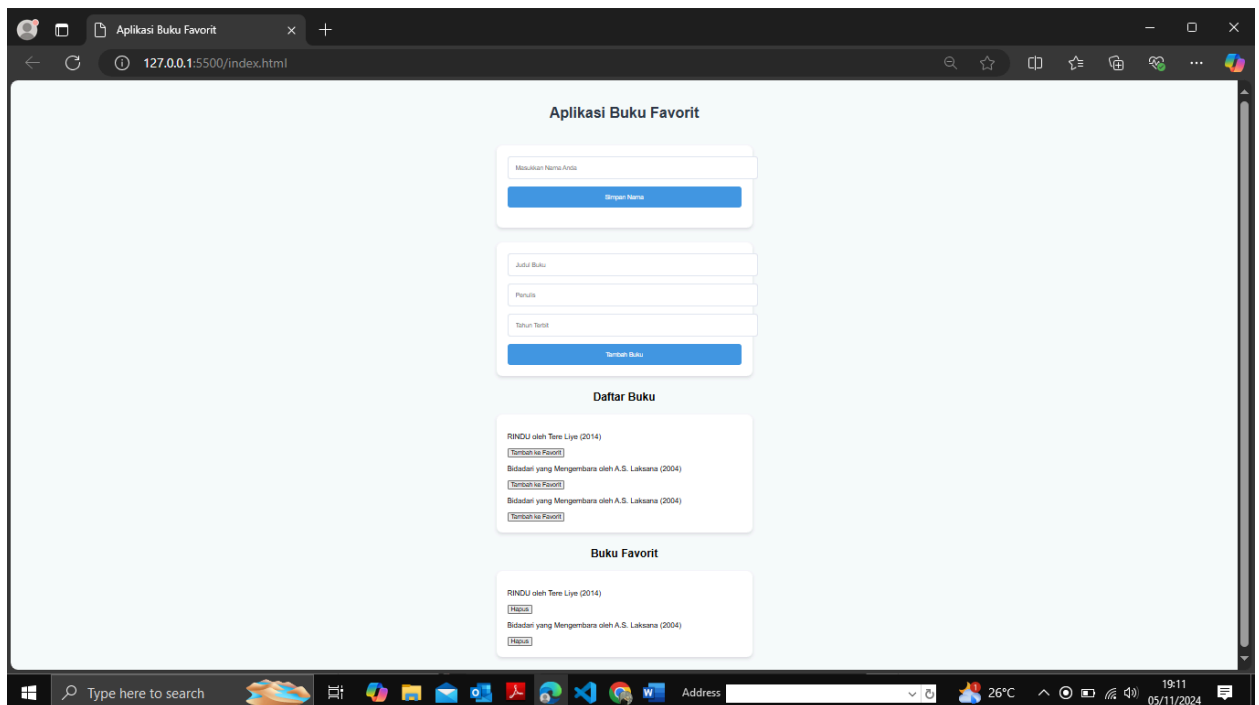
2. Tambahkan beberapa buku.



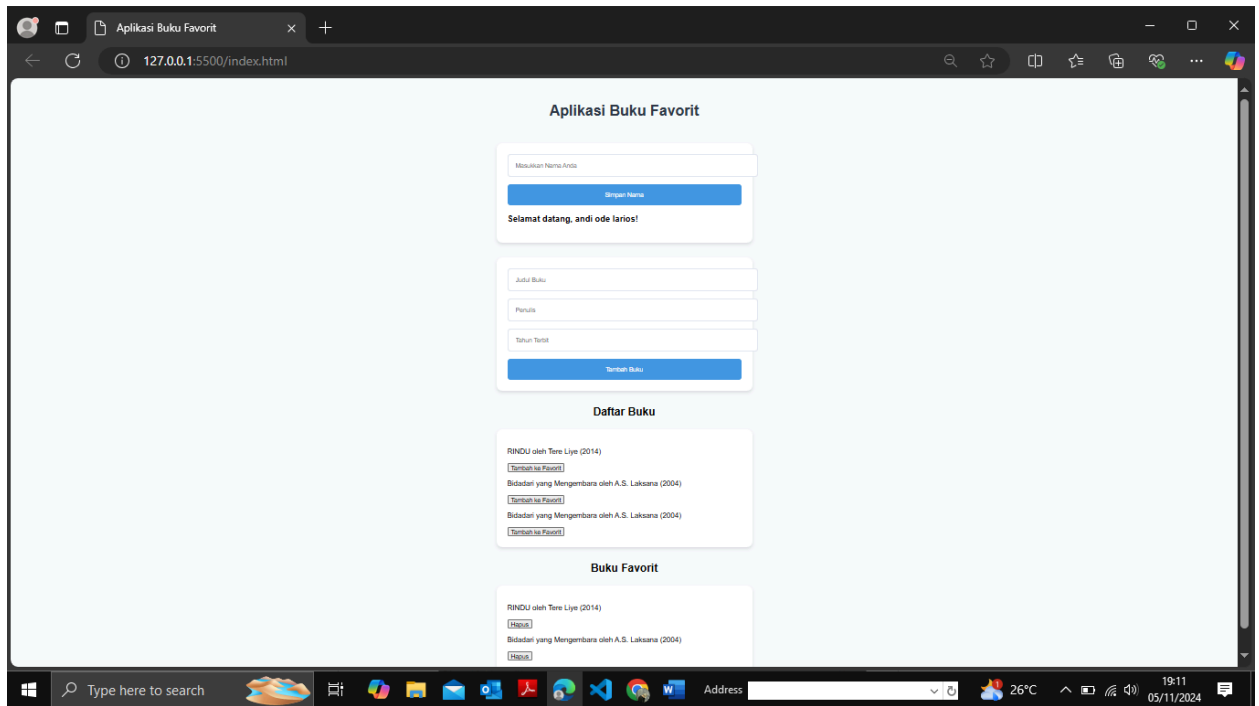
3. Tandai beberapa buku sebagai favorit



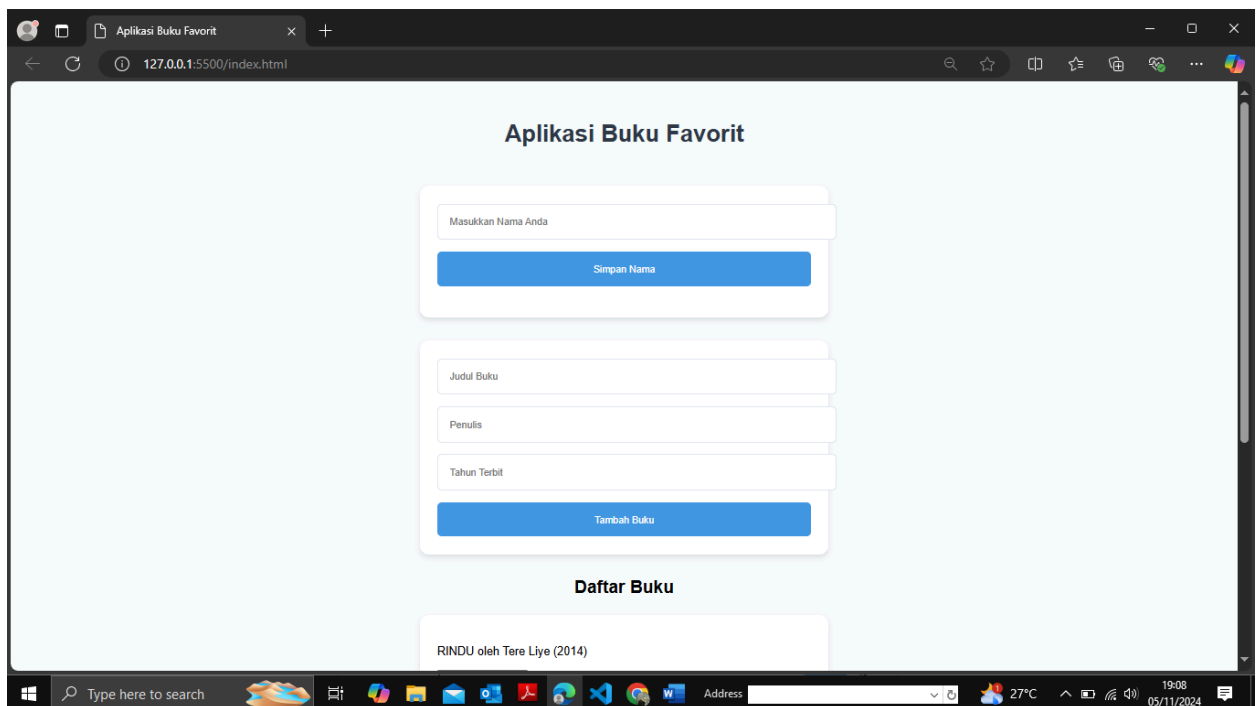
4. Refresh halaman dan pastikan buku favorit tetap ada



5. Masukkan nama pengguna dan pastikan nama tersebut ditampilkan



6. Tutup tab atau browser, lalu buka kembali dan cek apakah nama pengguna masih ada (seharusnya tidak, karena menggunakan session storage)



Penjelasan HTML, JS, CSS

HTML

- `<!DOCTYPE html>`:
Deklarasi tipe dokumen untuk menginformasikan browser bahwa dokumen ini menggunakan HTML5.
- `<html lang="id">`:
Elemen root HTML dengan atribut `lang="id"` yang menunjukkan bahwa bahasa yang digunakan dalam dokumen ini adalah Bahasa Indonesia.
- `<meta charset="UTF-8" />`:
Mengatur karakter encoding ke UTF-8 untuk mendukung berbagai karakter, termasuk aksara khusus.
- `<meta http-equiv="X-UA-Compatible" content="IE=edge" />`:
Menginstruksikan browser untuk menggunakan mode render yang paling kompatibel (seperti Edge untuk Internet Explorer).
- `<meta name="viewport" content="width=device-width, initial-scale=1.0" />`:
Mengatur tampilan agar responsif dengan menyesuaikan lebar halaman sesuai dengan layar perangkat.
- `<title>Aplikasi Buku Favorit</title>`:
Menetapkan judul halaman yang ditampilkan di tab browser.
- `<link rel="stylesheet" href="style.css" />`:
Menyertakan file CSS eksternal (`style.css`) yang digunakan untuk menentukan tampilan halaman web.
- `<div id="session-container">`:
Container yang menampung input nama pengguna.

- `<input type="text" id="namaPengguna" placeholder="Masukkan Nama Anda" />`:
Input field untuk memasukkan nama pengguna dengan placeholder "Masukkan Nama Anda".
- `<button id="btnSimpanNama">Simpan Nama</button>`:
Tombol untuk menyimpan nama pengguna yang dimasukkan.
- `<h3 id="salamPengguna"></h3>`:
Elemen heading `<h3>` yang digunakan untuk menampilkan salam setelah nama pengguna disimpan (akan diisi melalui JavaScript).
- `<form id="form-tambah-buku">`:
Formulir untuk menambahkan buku baru.
- `<input type="text" id="judul" placeholder="Judul Buku" required />`:
Input untuk judul buku.
- `<input type="text" id="penulis" placeholder="Penulis" required />`:
Input untuk penulis buku.
- `<input type="number" id="tahun" placeholder="Tahun Terbit" required />`:
Input untuk tahun terbit buku.
- `<button type="submit">Tambah Buku</button>`:
Tombol untuk mengirimkan data dan menambahkan buku ke daftar.
- `<h2>Daftar Buku</h2>`:
Sub-judul yang menampilkan teks "Daftar Buku".
- `<div id="daftar-buku"></div>`:
Elemen `<div>` kosong yang nantinya akan diisi dengan daftar buku yang ditambahkan oleh pengguna menggunakan JavaScript.

- `<h2>Buku Favorit</h2>`:
Sub-judul yang menampilkan teks "Buku Favorit".
- `<div id="buku-favorit"></div>`:
Elemen `<div>` yang akan menampilkan buku favorit pengguna setelah mereka menandainya dengan JavaScript.
- `<script src="script.js"></script>`:
Menyertakan file JavaScript eksternal (`script.js`) yang akan menangani logika interaktif, seperti menyimpan nama pengguna, menambahkan buku ke daftar, dan menandai buku sebagai favorit.

JAVASCRIPT

```
1 // Definisi Class Buku
2 class Buku {
3     constructor(judul, penulis, tahun) {
4         this.judul = judul;
5         this.penulis = penulis;
6         this.tahun = tahun;
7     }
8
9     tampilkanInfo() {
10         return `${this.judul} oleh ${this.penulis} (${this.tahun})`;
11     }
12 }
```

Class Buku digunakan untuk merepresentasikan buku.

- **Constructor** menerima tiga parameter (**judul**, **penulis**, dan **tahun**) untuk membuat objek buku.
- **tampilkanInfo()** adalah metode yang mengembalikan informasi buku dalam format string: "**judul** oleh **penulis** (**tahun**)".

```
// Array untuk menyimpan daftar buku dan buku favorit
let daftarBuku = [];
let bukuFavorit = [];
```

daftarBuku adalah array yang digunakan untuk menyimpan daftar semua buku yang ditambahkan oleh pengguna.

bukuFavorit adalah array yang menyimpan buku yang ditambahkan sebagai buku favorit oleh pengguna.

```
// Mengambil elemen DOM
const formTambahBuku = document.getElementById('form-tambah-buku');
const divDaftarBuku = document.getElementById('daftar-buku');
const divBukuFavorit = document.getElementById('buku-favorit');
const btnSimpanNama = document.getElementById('btnSimpanNama');
const salamPengguna = document.getElementById('salamPengguna');
```

Mengambil elemen DOM yang akan digunakan untuk interaksi. Setiap elemen HTML diambil berdasarkan **id** masing-masing:

- **formTambahBuku** untuk formulir tambah buku.
- **divDaftarBuku** untuk menampilkan daftar buku.
- **divBukuFavorit** untuk menampilkan buku favorit.
- **btnSimpanNama** untuk tombol penyimpanan nama pengguna.
- **salamPengguna** untuk menampilkan salam kepada pengguna.

Event Listener untuk disambungkan dengan Form Tambah Buku yang ada di HTML

```
// Event Listener untuk Form Tambah Buku
formTambahBuku.addEventListener('submit', function (e) {
  e.preventDefault();
  tambahBuku();
});
```

Event listener ini akan memanggil fungsi `tambahBuku()` ketika pengguna mengisi formulir tambah buku dan menekan tombol submit.

`e.preventDefault()` digunakan untuk mencegah perilaku default form (misalnya halaman di-refresh) saat form disubmit.

```
// Fungsi untuk menambahkan buku ke daftar
function tambahBuku() {
  const judul = document.getElementById('judul').value;
  const penulis = document.getElementById('penulis').value;
  const tahun = document.getElementById('tahun').value;

  // Validasi input
  if (judul === '' || penulis === '' || tahun === '') {
    alert('Semua kolom harus diisi!');
    return;
  }

  const bukuBaru = new Buku(judul, penulis, tahun);
  daftarBuku.push(bukuBaru);
  simpanDaftarBuku(); // Simpan daftar buku setelah menambahkan buku baru
  tampilkanDaftarBuku();
  formTambahBuku.reset();
}
```

Fungsi ini mengambil nilai input dari form (judul, penulis, tahun), memvalidasi apakah input kosong, dan jika valid, membuat instance baru dari class `Buku` dan menambahkannya ke array `daftarBuku`.

Setelah buku ditambahkan, data buku disimpan ke **Local Storage** menggunakan `simpanDaftarBuku()`, dan daftar buku diperbarui pada halaman dengan `tampilkanDaftarBuku()`. Lalu, form akan di-reset

```
// Fungsi untuk menyimpan daftar buku ke Local Storage
function simpanDaftarBuku() {
  localStorage.setItem('daftarBuku', JSON.stringify(daftarBuku));
}
```

Fungsi ini menyimpan array `daftarBuku` ke **Local Storage** dalam format JSON. Ini memungkinkan daftar buku untuk disimpan bahkan setelah halaman di-refresh.

```
// Fungsi untuk menampilkan daftar buku
function tampilkanDaftarBuku() {
  divDaftarBuku.innerHTML = '';

  daftarBuku.forEach((buku, index) => {
    const divBuku = document.createElement('div');
    divBuku.classList.add('buku');
    divBuku.innerHTML = `
      <p>${buku.tampilkanInfo()}</p>
      <button onclick="tambahKeFavorit(${index})">Tambah ke Favorit</button>
    `;
    divDaftarBuku.appendChild(divBuku);
  });
}
```

Fungsi ini memperbarui elemen `divDaftarBuku` dengan daftar buku yang ada dalam array `daftarBuku`. Setiap buku ditampilkan dalam elemen div yang berisi informasi buku dan tombol "Tambah ke Favorit" untuk menambahkan buku ke daftar favorit.

```
// Fungsi untuk menampilkan daftar buku
function tampilkanDaftarBuku() {
    divDaftarBuku.innerHTML = '';

    daftarBuku.forEach((buku, index) => {
        const divBuku = document.createElement('div');
        divBuku.classList.add('buku');
        divBuku.innerHTML = `
            <p>${buku.tampilkanInfo()}</p>
            <button onclick="tambahKeFavorit(${index})">Tambah ke Favorit</button>
        `;
        divDaftarBuku.appendChild(divBuku);
    });
}
```

Fungsi ini memperbarui elemen `divDaftarBuku` dengan daftar buku yang ada dalam array `daftarBuku`. Setiap buku ditampilkan dalam elemen div yang berisi informasi buku dan tombol "Tambah ke Favorit" untuk menambahkan buku ke daftar favorit.

```
// Fungsi untuk menambahkan buku ke favorit
function tambahKeFavorit(index) {
    const buku = daftarBuku[index];

    // Cek apakah buku sudah ada di favorit
    const sudahAda = bukuFavorit.some(favBuku => {
        return favBuku.judul === buku.judul &&
            favBuku.penulis === buku.penulis &&
            favBuku.tahun === buku.tahun;
    });

    if (sudahAda) {
        alert('Buku ini sudah ada di daftar favorit!');
        return;
    }

    bukuFavorit.push(buku);
    simpanBukuFavorit();
    tampilkanBukuFavorit();
}
```

Fungsi ini memperbarui Fungsi ini menambahkan buku dari daftar umum ke daftar favorit, namun

pertama-tama mengecek apakah buku tersebut sudah ada di favorit. Jika sudah, pengguna diberi notifikasi. Jika belum, buku ditambahkan ke `bukuFavorit`, dan data disimpan ke **Local Storage** dengan `simpanBukuFavorit()` serta memperbarui tampilan favorit dengan `tampilkanBukuFavorit()`.

```
// Fungsi untuk menyimpan buku favorit ke Local Storage
function simpanBukuFavorit() {
  localStorage.setItem('bukuFavorit', JSON.stringify(bukuFavorit));
}
```

Fungsi ini menyimpan array `bukuFavorit` ke **Local Storage** dalam format JSON

```
// Fungsi untuk menampilkan buku favorit
function tampilkanBukuFavorit() {
  divBukuFavorit.innerHTML = '';

  bukuFavorit.forEach((buku, index) => {
    const divBuku = document.createElement('div');
    divBuku.classList.add('buku');
    divBuku.innerHTML = `
      <p>${buku.tampilkanInfo()}</p>
      <button onclick="hapusDariFavorit(${index})">Hapus</button>
    `;
    divBukuFavorit.appendChild(divBuku);
  });
}
```

Fungsi ini menampilkan buku yang telah ditandai sebagai favorit pada elemen `divBukuFavorit`. Untuk setiap buku favorit, juga ditambahkan tombol "Hapus" untuk menghapus buku dari favorit.

```
// Fungsi untuk menghapus buku dari favorit
function hapusDariFavorit(index) {
    bukuFavorit.splice(index, 1);
    simpanBukuFavorit();
    tampilkanBukuFavorit();
}
```

Fungsi ini menghapus buku dari daftar favorit berdasarkan indeksinya, lalu memperbarui data di **Local Storage** dan memperbarui tampilan buku favorit.

```
// Event Listener untuk tombol simpan nama pengguna
btnSimpanNama.addEventListener('click', function () {
    const nama = document.getElementById('namaPengguna').value;
    if (nama === '') {
        alert('Masukkan nama Anda!');
        return;
    }
    sessionStorage.setItem('namaPengguna', nama);
    tampilkanNamaPengguna();
    document.getElementById('namaPengguna').value = '';
});
```

Event listener ini menyimpan nama pengguna ke **Session Storage** saat tombol "Simpan Nama" ditekan, dan memanggil `tampilkanNamaPengguna()` untuk menampilkan salam dengan nama pengguna

```
// Fungsi untuk menampilkan nama pengguna
function tampilkanNamaPengguna() {
    const nama = sessionStorage.getItem('namaPengguna');

    if (nama) {
        salamPengguna.textContent = `Selamat datang, ${nama}!`;
    } else {
        salamPengguna.textContent = '';
    }
}
```

Fungsi ini mengambil nama pengguna dari **Session Storage** dan menampilkannya sebagai salam di halaman

```

// Memuat data saat halaman dimuat
window.onload = function () {
    // Memuat daftar buku dari Local Storage
    if (localStorage.getItem('daftarBuku')) {
        const storedBooks = JSON.parse(localStorage.getItem('daftarBuku'));
        // Rekonstruksi objek buku menjadi instance dari kelas Buku
        daftarBuku = storedBooks.map(book => {
            return new Buku(book.judul, book.penulis, book.tahun);
        });
        tampilkanDaftarBuku();
    }

    // Memuat buku favorit dari Local Storage
    if (localStorage.getItem('bukuFavorit')) {
        const storedFavorites = JSON.parse(localStorage.getItem('bukuFavorit'));
        // Rekonstruksi objek buku menjadi instance dari kelas Buku
        bukuFavorit = storedFavorites.map(book => {
            return new Buku(book.judul, book.penulis, book.tahun);
        });
        tampilkanBukuFavorit();
    }

    // Menampilkan nama pengguna dari Session Storage
    tampilkanNamaPengguna();
};

```

Fungsi ini dijalankan saat halaman dimuat untuk mengambil data buku dan buku favorit dari **Local Storage**, serta menampilkan nama pengguna dari **Session Storage**.

CSS

```

/* Menggunakan Tailwind secara umum */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f7fafc; /* Set background-color lebih terang */
}

```

font-family: Mengatur font default halaman menggunakan 'Arial' sebagai font utama dan **sans-serif** sebagai alternatif.

background-color: Mengatur warna latar belakang halaman menjadi warna abu terang

(#f7fafc).

```
/* Styling untuk header aplikasi */
h1 {
  font-size: 2rem; /* 3xl dari Tailwind */
  font-weight: 600; /* Font-semibold */
  text-align: center;
  padding-top: 2rem;
  padding-bottom: 2rem;
  color: #2d3748; /* Text-gray-800 */
}
```

font-size: Mengatur ukuran font header menjadi **2rem** (mirip ukuran **3xl** di Tailwind).

font-weight: Mengatur ketebalan teks menjadi **600** (sebanding dengan **font-semibold** di Tailwind).

text-align: Mengatur teks agar berada di tengah.

padding-top & **padding-bottom**: Memberi ruang atas dan bawah sebesar **2rem**.

color: Mengatur warna teks menjadi abu gelap (#2d3748).

```
/* Styling untuk Input Nama Pengguna */
#session-container {
  max-width: 32rem; /* max-w-lg dari Tailwind */
  margin: 0 auto; /* Centering */
  background-color: #ffffff;
  padding: 1.5rem;
  border-radius: 0.75rem; /* rounded-lg */
  box-shadow: 0 4px 6px #0000001a; /* shadow-md */
  margin-bottom: 2rem;
}
```

max-width: Membatasi lebar kontainer hingga **32rem** (seperti **max-w-lg** di Tailwind).

margin: 0 auto: Membuat kontainer berada di tengah.

background-color: Mengatur warna latar belakang kontainer menjadi putih.

padding: Memberikan jarak dalam sebesar **1.5rem**.

border-radius: Membuat sudut-sudut kontainer membulat (0.75rem, seperti **rounded-lg** di Tailwind).

box-shadow: Memberikan bayangan ringan (efek **shadow-md** di Tailwind).

margin-bottom: Memberikan jarak bawah sebesar **2rem** untuk pemisahan elemen berikutnya.

```
#session-container input {
  width: 100%;
  padding: 1rem;
  border: 1px solid #e2e8f0; /* border-gray-300 */
  border-radius: 0.375rem; /* rounded-md */
  margin-bottom: 1rem;
  outline: none;
}
```

width: Memastikan input memenuhi lebar kontainer.

padding: Memberikan ruang dalam sebesar **1rem**.

border: Mengatur warna border dengan abu terang (**#e2e8f0**).

border-radius: Membuat sudut input sedikit membulat (**0.375rem** atau **rounded-md** di Tailwind).

margin-bottom: Memberi jarak bawah antara input lain sebesar **1rem**.

outline: none: Menghilangkan outline default saat input difokuskan

```
#session-container button {
  width: 100%;
  padding: 1rem;
  background-color: #4299e1; /* bg-blue-500 */
  color: #ffffff;
  border: none;
  border-radius: 0.375rem; /* rounded-md */
  cursor: pointer;
  transition: background-color 0.2s;
}
```

width: Mengatur tombol memenuhi lebar kontainer.

padding: Memberikan ruang dalam **1rem**.

background-color: Memberikan warna biru terang (**#4299e1**).


color: Mengatur warna teks tombol menjadi putih.

border: Menghapus border default.



border-radius: Membuat sudut membulat (**rounded-md**).

cursor: Menjadikan ikon mouse berubah saat tombol diarahkan (**pointer**).

transition: Memberikan efek transisi halus pada perubahan warna saat tombol diarahkan.

```
#session-container button:hover {  
|   background-color:  #3182ce; /* hover:bg-blue-600 */  
}
```

`background-color`: Saat tombol di-hover, warnanya berubah menjadi biru yang lebih gelap (#3182ce).

```
/* Form Tambah Buku */  
#form-tambah-buku {  
|   max-width: 32rem; /* max-w-lg */  
|   margin: 0 auto;  
|   background-color:  #ffffff;  
|   padding: 1.5rem;  
|   border-radius: 0.75rem;  
|   box-shadow: 0 4px 6px  rgba(0, 0, 0, 0.1); /* shadow-md */  
|   margin-bottom: 2rem;  
}
```

```

#form-tambah-buku input {
  width: 100%;
  padding: 1rem;
  border: 1px solid #e2e8f0; /* border-gray-300 */
  border-radius: 0.375rem; /* rounded-md */
  margin-bottom: 1rem;
  outline: none;
}

#form-tambah-buku button {
  width: 100%;
  padding: 1rem;
  background-color: #4299e1; /* bg-blue-500 */
  color: #ffffff;
  border: none;
  border-radius: 0.375rem; /* rounded-md */
  cursor: pointer;
  transition: background-color 0.2s;
}

#form-tambah-buku button:hover {
  background-color: #3182ce; /* hover:bg-blue-600 */
}

```

Struktur CSS untuk `#form-tambah-buku` sama dengan `#session-container`. Digunakan untuk memberikan tampilan yang konsisten antara form input pengguna dan form tambah buku.

```

/* Styling Daftar Buku dan Buku Favorit */
h2 {
  font-size: 1.5rem; /* text-2xl */
  font-weight: 600; /* font-semibold */
  text-align: center;
  margin-bottom: 1.5rem;
}

```

`font-size`: Mengatur ukuran font menjadi `1.5rem` (sama dengan `text-2xl` di Tailwind).
`font-weight`: Membuat teks lebih tebal (`font-semibold`).
`text-align`: Mengatur teks di tengah.
`margin-bottom`: Memberi jarak bawah sebesar `1.5rem`.

```
#daftar-buku, #buku-favorit {  
  max-width: 32rem;  
  margin: 0 auto;  
  background-color: #ffffff;  
  padding: 1.5rem;  
  border-radius: 0.75rem; /* rounded-lg */  
  box-shadow: 0 4px 6px #0000001a; /* shadow-md */  
  margin-bottom: 2rem;  
}
```

`max-width`, `margin`, `background-color`, `padding`, `border-radius`, `box-shadow`, dan `margin-bottom` memiliki fungsi yang sama dengan `#session-container`.

`#daftar-buku` dan `#buku-favorit` diatur sedemikian rupa agar memiliki gaya yang konsisten dengan elemen form sebelumnya, dengan batas lebar maksimal `32rem`, warna latar putih, padding, efek bayangan, dan sudut membulat.

Link Github:

<https://github.com/AndiOdeLarios/tugas-desain-web-7-/blob/main/README.md>