LAPORAN PRAKTIKUM

"**Web Storage**"

Dalam rangka memenuhi tugas mata kuliah Praktikum Desain Website

Dosen pengampu: Vearen Dika Sofirudin, S.Pd., M.Ed.

Disusun oleh:

Nama : Rifkhi Ardi Mustaqim

NIM : K3524033

PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS SEBELAS MARET

SURAKARTA

2025

A. Tujuan Praktikum

Tujuan dari praktikum ini adalah untuk:

1. Mahasiswa mengenal mekanisme penyimpanan data pada HTML.
2. Mahasiswa dapat menggunakan local storage HTML.
3. Mahasiswa dapat menggunakan session storage HTML.

B. Ringkasan Materi

Web Storage, atau DOM Storage, adalah sebuah metode yang disediakan oleh HTML5 bagi website untuk menyimpan data langsung di sisi klien (browser pengguna). Mekanisme ini sangat berguna untuk menyimpan preferensi pengguna, status aplikasi, atau data sementara tanpa harus bergantung pada cookies. Keunggulan utamanya adalah kapasitas penyimpanan yang lebih besar dari cookies (biasanya 5-10MB) dan data tersebut tidak dikirim secara otomatis ke server pada setiap permintaan HTTP, sehingga lebih aman dan efisien. Terdapat dua jenis Web Storage:

1. localStorage: Menyimpan data dalam bentuk pasangan key-value (kunci-nilai) tanpa batas waktu kedaluwarsa. Data ini akan tetap tersimpan bahkan setelah browser ditutup dan dibuka kembali, menjadikannya ideal untuk preferensi jangka panjang.
2. sessionStorage: Juga menyimpan data sebagai key-value, namun data ini bersifat sementara. Data akan hilang secara otomatis ketika sesi browser (atau tab) ditutup.

Kedua mekanisme storage ini menggunakan fungsi JavaScript yang serupa untuk berinteraksi dengan data:

1. setItem(key, value): Untuk menyimpan atau memperbarui data.
2. getItem(key): Untuk mengambil data berdasarkan kuncinya.
3. removeItem(key): Untuk menghapus data tertentu.
4. clear(): Untuk menghapus semua data yang ada di storage tersebut.

Penting untuk diingat bahwa Web Storage hanya dapat menyimpan data dalam format string. Jika kita perlu menyimpan data kompleks seperti object atau array, kita harus mengubahnya menjadi string (misalnya menggunakan JSON.stringify()) sebelum menyimpan, dan mengubahnya kembali saat mengambil (menggunakan JSON.parse()).

C. Langkah Kerja

Praktikum ini terdiri dari dua latihan utama yang berfokus pada implementasi localStorage.

1. Latihan 1

Latihan ini berfokus pada kemampuan localStorage untuk menyimpan data *string* secara persisten.

Kode Awal hanya mampu menyimpan (setItem) dan memuat (getItem) nama saat halaman dimuat ulang. Namun, kode ini belum memenuhi persyaratan kontrol data penuh karena tidak ada mekanisme untuk menghapus nama yang tersimpan.

- Kode awal

latihan1.html

TUgas 8 Web Storage > latihan1.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Web Storage</title>
</head>
<body>
    <center>
        <h1>Selamat Datang, Member!</h1>
        <button onclick="gantiNama()">Masukkan Nama</button>
    </center>

    <script type="text/javascript">
        const namaMember = document.querySelector('h1');

        const namaTersimpan = localStorage.getItem('nama');

        if (namaTersimpan) {
            namaMember.innerHTML = 'Halo, ' + namaTersimpan + '!';
        }

        function gantiNama() {
            let nama = prompt('Masukkan nama Anda:');

            if (nama && nama.trim() !== "") {
                localStorage.setItem('nama', nama);
                namaMember.innerHTML = 'Halo, ' + nama + '!';
            } else {
                alert('Nama tidak boleh kosong!');
            }
        }
    </script>
</body>
</html>
```

- Hasil Awal:

Saat dijalankan, halaman menampilkan sapaan "Selamat Datang, Member!" dan sebuah tombol. Saat tombol di klik, muncul prompt untuk memasukkan nama. Setelah nama dimasukkan, sapaan berubah dan nama tersebut tersimpan. Jika halaman di-refresh, sapaan "Halo [NamaTersimpan]!" akan langsung muncul karena data diambil dari localStorage. Namun, belum ada cara untuk menghapus nama ini.

- Langkah Perbaikan Sesuai instruksi (poin c), perlu ditambahkan tombol untuk menghapus nama.
    1) Menambahkan elemen <button> baru di HTML untuk fungsi hapus.
    2) Membuat fungsi JavaScript baru, misalnya hapusNama().
    3) Di dalam hapusNama(), kita menggunakan localStorage.removeItem('nama') untuk menghapus data yang tersimpan.
    4) Setelah dihapus, kita mengembalikan teks <h1> ke sapaan default ("Selamat Datang, Member!").
- Kode Revisi

Kode Revisi menambahkan fungsi hapusNama() yang menggunakan localStorage.removeItem('nama'). Dengan modifikasi ini, data nama pengguna tidak hanya persisten, tetapi juga dapat dikelola dan dihapus secara manual, yang diperlukan untuk memenuhi instruksi praktikum. Implementasi ini secara efektif menunjukkan perbedaan antara localStorage dan *cookies*, di mana data tidak kedaluwarsa dan harus dihapus secara eksplisit

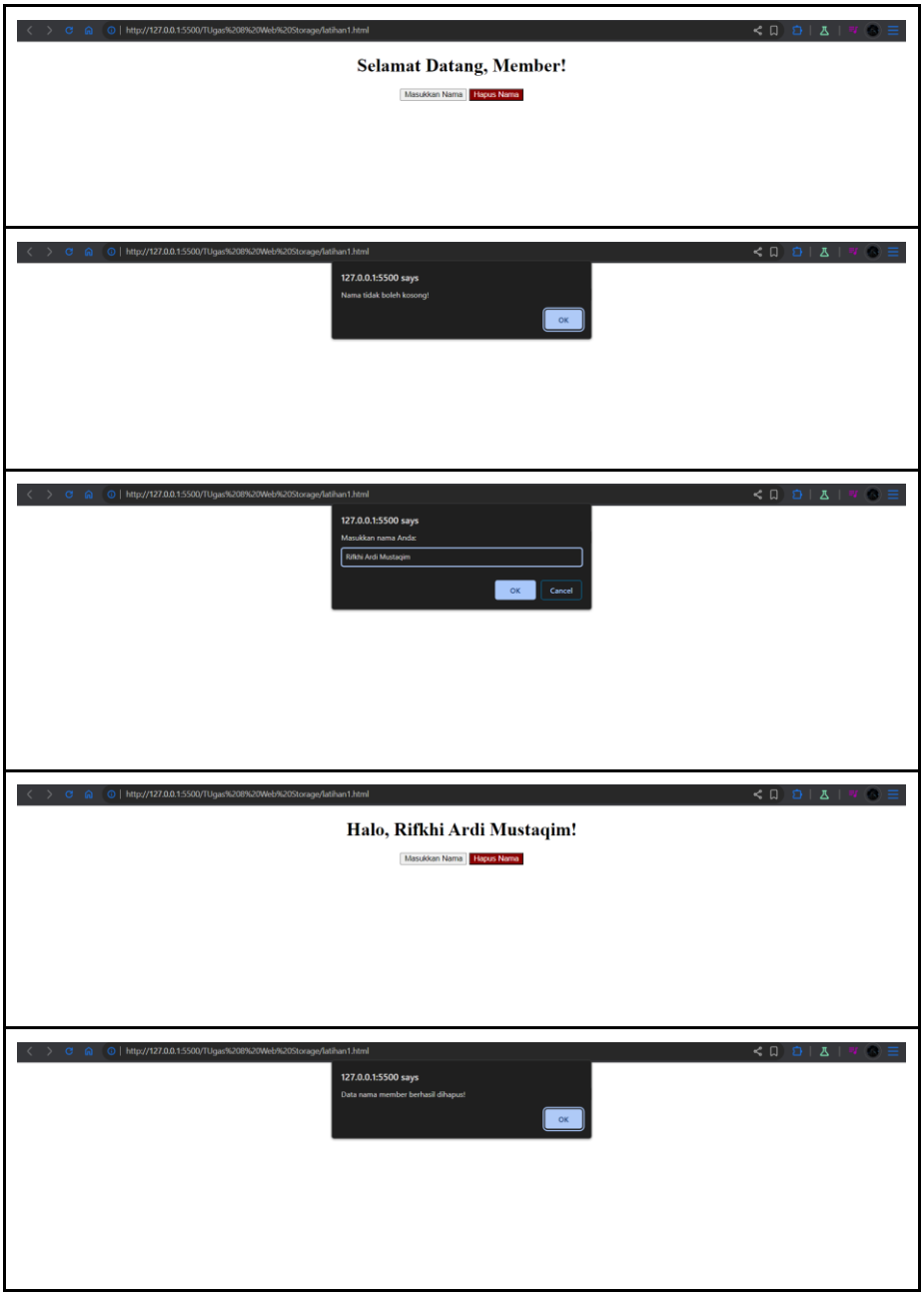
latihan1.html

TUgas 8 Web Storage > latihan1.html > html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Web Storage</title>
</head>
<body>
    <center>
        <h1 id="greeting">Selamat Datang, Member!</h1>
        <button onclick="gantiNama()">Masukkan Nama</button>

        <button onclick="hapusNama()" style="background-color: darkred; color: white;">Hapus Nama</button>
    </center>

    <script type="text/javascript">
        const textElement = document.getElementById('greeting');

        const storedName = localStorage.getItem('nama');

        if (storedName) {
            textElement.innerHTML = 'Halo, ' + storedName + '!';
        }

        function gantiNama() {
            let nama = prompt('Masukkan nama Anda:');

            if (nama && nama.trim() !== "") {
                localStorage.setItem('nama', nama);
                textElement.innerHTML = 'Halo, ' + nama + '!';
            } else {
                alert('Nama tidak boleh kosong!');
            }
        }

        function hapusNama() {
            localStorage.removeItem('nama');
            textElement.innerHTML = 'Selamat Datang, Member!';
            alert('Data nama member berhasil dihapus!');
        }
    </script>
</body>
</html>
```

- Hasil Akhir:

  Halaman kini memiliki dua tombol. Tombol "Masukkan Nama" berfungsi seperti semula. Tombol "Hapus Nama" akan menghapus nama dari localStorage dan mengembalikan sapaan ke "Selamat Datang, Member!". Jika halaman di-refresh setelah nama dihapus, sapaan akan tetap default.

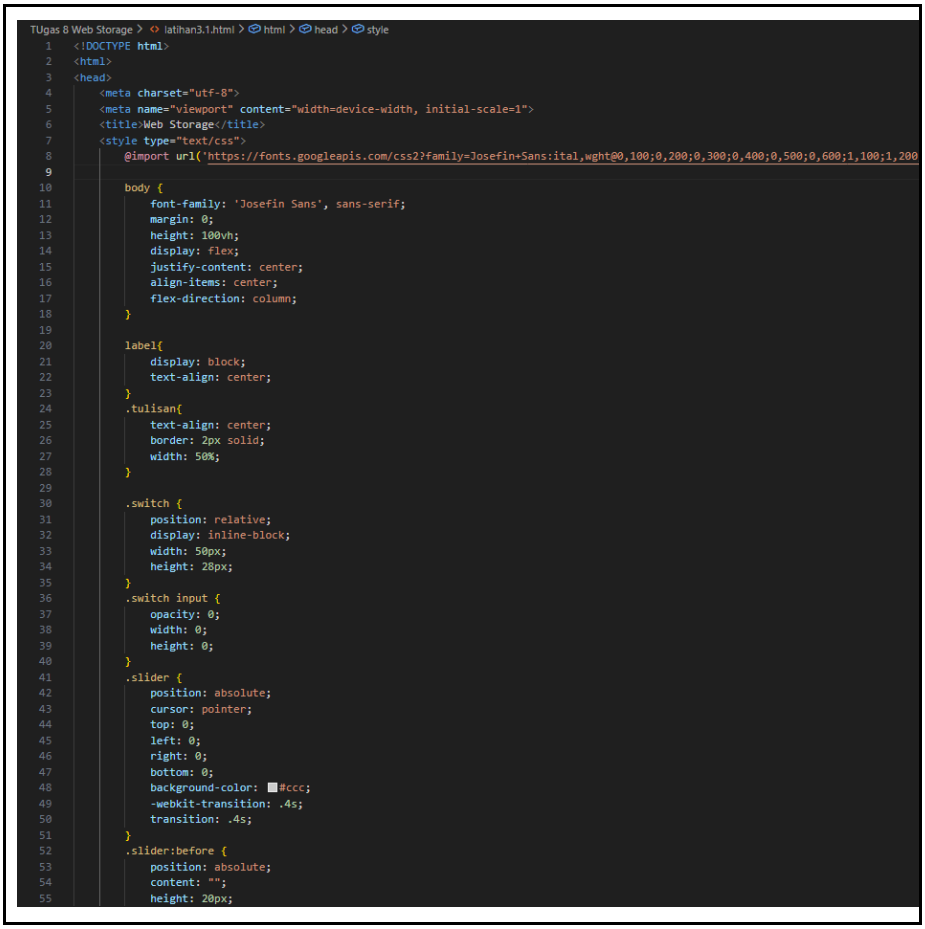


2. Latihan 3

Tujuan dari latihan ini adalah menggunakan localStorage untuk menyimpan *state* preferensi tema (Dark/Light) pengguna, memastikan tema tetap sama setelah *refresh*.

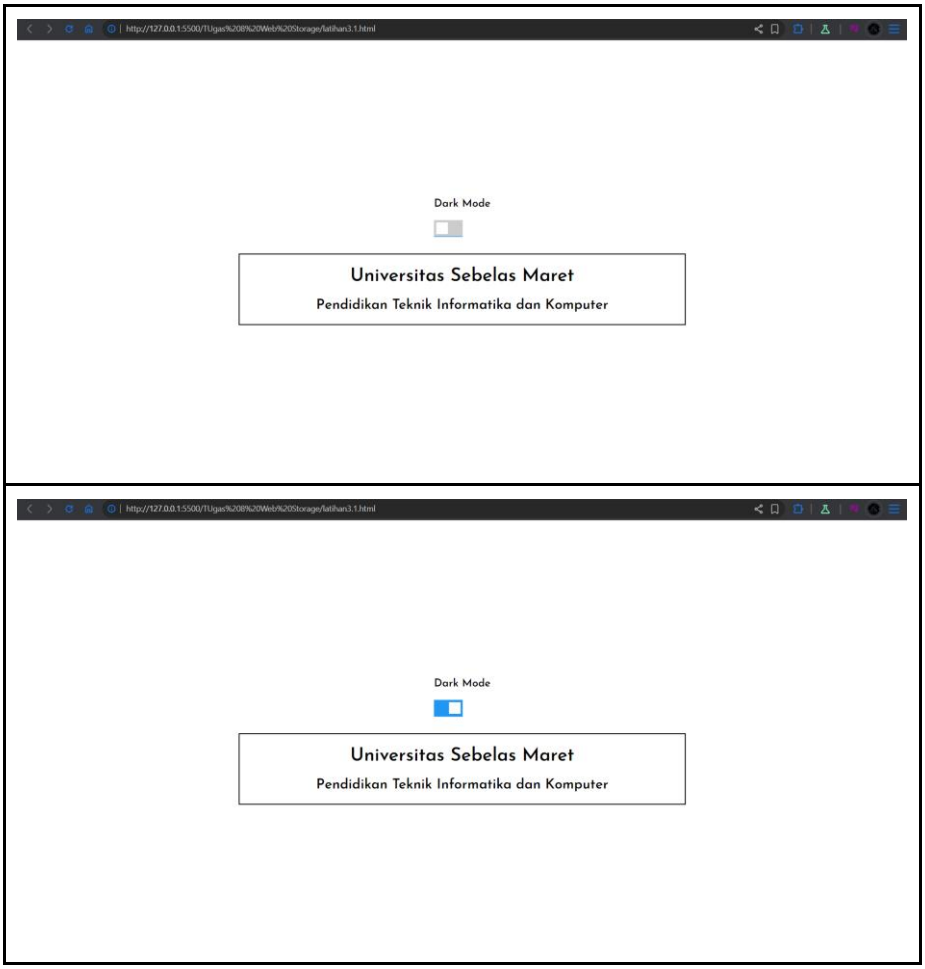
Kode Awal hanya menampilkan *toggle switch* yang dapat digeser dan berubah warna menjadi biru saat aktif. Namun, secara fungsional, tidak ada perubahan visual pada tema halaman karena tidak memiliki CSS .dark-mode. Lebih penting, kode ini tidak memiliki logika JavaScript untuk menyimpan status *toggle* di *storage*, sehingga tema selalu kembali ke *light* saat halaman di-*refresh*.

- Kode awal



```
            width: 20px;
            left: 4px;
            bottom: 4px;
            background-color: white;
            -webkit-transition: .4s;
            transition: .4s;
        }
        input:checked + .slider {
            background-color: #2196F3;
        }
        input:focus + .slider {
            box-shadow: 0 1px #2196F3;
        }
        input:checked + .slider:before {
            -webkit-transform: translateX(22px);
            -ms-transform: translateX(22px);
            transform: translateX(22px);
        }
    </style>
</head>
<body>
    <div class="label" style="margin-bottom: 30px;">
        <h3>Dark Mode</h3>
        <label class="switch">
            <input type="checkbox" onchange="ubahTema(event)">
            <span class="slider"></span>
        </label>
    </div>

    <div class="tulisan">
        <h1>Universitas Sebelas Maret</h1>
        <h2>Pendidikan Teknik Informatika dan Komputer</h2>
    </div>
    </body>
</html>
```

- Hasil Awal:



Halaman menampilkan teks di tengah dan sebuah toggle switch. Switch dapat di klik (berubah warna dan bergeser), namun tidak ada perubahan

visual pada tema halaman (latar belakang dan teks). Jika halaman di-refresh, switch kembali ke posisi off.

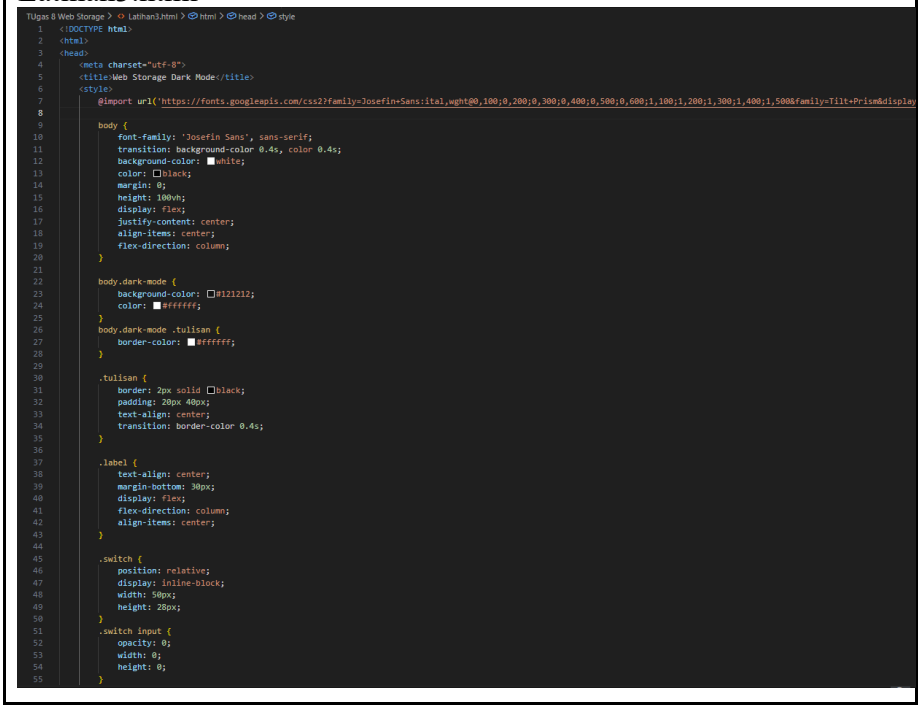
Kode Revisi mengatasi masalah ini dengan langkah-langkah terpisah:

- CSS: Menambahkan kelas .dark-mode pada body untuk membalik *background* dan *text color*. Properti border-color pada .tulisan juga diubah menjadi putih untuk kontras.
- JavaScript: Menggunakan fungsi ubahTema() yang dipanggil saat *toggle* berubah. Fungsi ini menggunakan localStorage.setItem('theme', 'dark/light') untuk menyimpan status. Pengecekan status saat halaman dimuat memastikan tema terakhir yang dipilih (persisten) diterapkan kembali.

Modifikasi ini berhasil membuktikan bahwa localStorage ideal untuk menyimpan preferensi UI jangka panjang.

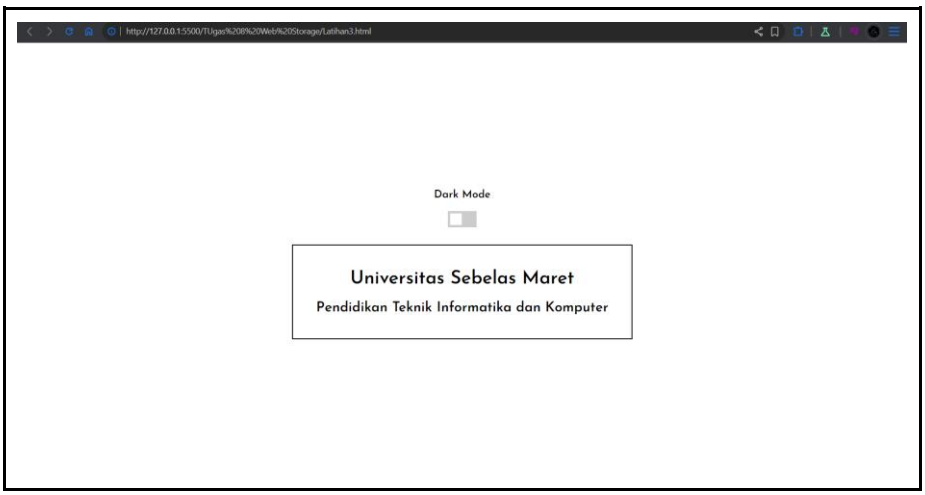
- Kode Revisi

Latihan3.html



```
        .slider {
            position: absolute;
            cursor: pointer;
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
            background-color: #ccc;
            transition: .4s;
        }
        .slider:before {
            position: absolute;
            content: "";
            height: 20px;
            width: 20px;
            left: 4px;
            bottom: 4px;
            background-color: white;
            transition: .4s;
        }
        input:checked + .slider {
            background-color: #2196F3;
        }
        input:checked + .slider:before {
            transform: translateX(22px);
        }
    </style>
</head>
<body>
    <div class="label">
        <h3>Dark Mode</h3>
        <label class="switch">
            <input type="checkbox" id="darkModeToggle" onchange="ubahTema()">
            <span class="slider"></span>
        </label>
    </div>

    <div class="tulisan">
        <h1>Universitas Sebelas Maret</h1>
        <h2>Pendidikan Teknik Informatika dan Komputer</h2>
    </div>

    <script>
        const toggle = document.getElementById('darkModeToggle');
        const body = document.body;

        const currentTheme = localStorage.getItem('theme');

        if (currentTheme === 'dark') {
            body.classList.add('dark-mode');
            toggle.checked = true;
        } else if (currentTheme === 'light') {
            body.classList.remove('dark-mode');
            toggle.checked = false;
        }

        function ubahTema() {
            if (toggle.checked) {
                body.classList.add('dark-mode');
                localStorage.setItem('theme', 'dark');
            } else {
                body.classList.remove('dark-mode');
                localStorage.setItem('theme', 'light');
            }
        }
    </script>
</body>
</html>
```

- Hasil Akhir:



Fungsionalitas dark mode sekarang berjalan sempurna. Saat toggle di klik, tema halaman berubah antara terang dan gelap. Preferensi ini disimpan di localStorage, sehingga jika halaman di-refresh atau ditutup dan dibuka kembali, tema yang terakhir dipilih akan otomatis diterapkan.

D. Kesimpulan

Dari praktikum ini, dapat disimpulkan bahwa:

Praktikum Web Storage berhasil menguji kemampuan mahasiswa dalam mengelola data di sisi klien.

1. Mekanisme Storage: localStorage digunakan secara efektif untuk menyimpan data yang persisten dan tidak memiliki tanggal kedaluwarsa. Ini berbeda dengan sessionStorage yang datanya hilang saat sesi *browser* ditutup.
2. Kontrol Data: Melalui Latihan 1, mahasiswa mengimplementasikan kontrol data penuh (setItem, getItem, removeItem).
3. Aplikasi UI: Melalui Latihan 3, Web Storage terbukti penting untuk personalisasi UI, di mana *state* tema disimpan antar-sesi, memastikan pengalaman pengguna yang konsisten saat menjelajahi web.
4. Efisiensi: Penggunaan Web Storage meningkatkan efisiensi karena data tidak secara otomatis dikirim ke *server* pada setiap permintaan HTTP.