MODUL 9

WEB STORAGE

A. TUJUAN PRAKTIKUM

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

B. ALOKASI WAKTU 2 x 50 menit

C. DASAR TEORI

1. Web Storage

Dalam pengembangan sistem berbasis website terdapat media penyimpanan (Storage) yang dapat digunakan untuk menyimpan data. Web Storage atau biasanya dapat diketahui sebagai DOM storage (Document Object Model Storage), adalah metode yang digunakan pada website untuk dapat menyimpan data pada sisi klien (client-side).

Web Storage sangat berguna untuk menyimpan preferensi pengguna, status aplikasi, data sementara, atau bahkan cache data yang mungkin diperlukan selama pengguna menjelajahi halaman web. Ini juga lebih aman daripada cookies karena data tidak secara otomatis dikirim ke server setiap kali permintaan HTTP dibuat.

Sebelum HTML5, data aplikasi harus disimpan dalam cookie, termasuk dalam setiap permintaan server. Web storage lebih aman, dan sejumlah besar data dapat disimpan secara lokal, tanpa memengaruhi kinerja situs web. Namun demikian, tidak semua browser mendukung fitur web storage ini.

Ada dua jenis Web Storage yang umum digunakan dalam HTML5:

a. **localStorage:** Method ini memungkinkan kita untuk menyimpan data dalam bentuk pasangan key-value (kunci-nilai). Data yang disimpan di localStorage akan tetap ada bahkan setelah browser ditutup. Data di localStorage tidak memiliki tanggal kadaluarsa dan harus dihapus secara manual oleh pengguna atau melalui kode JavaScript.
b. **sessionStorage:** Sama seperti localStorage, sessionStorage juga memungkinkan penyimpanan data dalam bentuk pasangan key-value. Namun, data yang disimpan dalam sessionStorage hanya akan ada selama sesi browser berlangsung. Ini berarti bahwa data akan hilang setelah browser ditutup.

2. Local Storage

Fungsi-fungsi yang digunakan untuk berinteraksi dengan localStorage dalam HTML5 adalah sebagai berikut:

- **setItem(key, value):** Fungsi ini digunakan untuk menyimpan data ke dalam localStorage. Anda harus memberikan sebuah kunci (key) dan nilai (value) yang ingin Anda simpan. Contoh penggunaan:

```
localStorage.setItem('username', 'john_doe');
```

- **getItem(key):** Fungsi ini digunakan untuk mengambil nilai dari localStorage berdasarkan kunci yang diberikan. Contoh penggunaan:

```
const username = localStorage.getItem('username');
```

- **removeItem(key):** Fungsi ini digunakan untuk menghapus data dari localStorage berdasarkan kunci yang diberikan. Contoh penggunaan:

```
localStorage.removeItem('username');
```

- **clear():** Fungsi ini digunakan untuk menghapus semua data yang ada di localStorage. Ini akan mengosongkan seluruh localStorage pada domain yang sama. Contoh penggunaan:

```
localStorage.clear();
```

- **length:** Properti ini mengembalikan jumlah pasangan key-value yang ada di localStorage. Contoh penggunaan:

```
const itemCount = localStorage.length;
```

Berikut contoh penggunaan dari beberapa method local storage sekaligus.

```
<script>
// Menyimpan data ke localStorage
localStorage.setItem('username', 'john_doe');
localStorage.setItem('theme', 'dark');

// Mengambil data dari localStorage
const username = localStorage.getItem('username');
const theme = localStorage.getItem('theme');

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

// Menghapus semua data dari localStorage
localStorage.clear();
</script>
```

Perlu diingat bahwa data yang disimpan dalam localStorage hanya dapat berupa tipe data string. Jika kita perlu menyimpan tipe data lain, seperti objek atau array, kita harus mengonversinya menjadi string terlebih dahulu (misalnya, dengan menggunakan `JSON.stringify`) saat menyimpan dan mengonversinya kembali menjadi tipe data aslinya (misalnya, dengan menggunakan `JSON.parse`) saat mengambilnya.

```
const userObject = { name: 'John', age: 30 };
localStorage.setItem('user', JSON.stringify(userObject));

const retrievedUser = JSON.parse(localStorage.getItem('user'));
```

Penting untuk diingat bahwa data di localStorage tetap ada meskipun pengguna menutup browser atau menghidupkan ulang komputer. Data hanya akan dihapus jika Anda secara

manual menghapusnya menggunakan fungsi `removeItem` atau `clear`, atau jika pengguna membersihkan data browsing pada browser.

3. **Session Storage**

   Session Storage dalam HTML5 adalah fitur yang memungkinkan Anda untuk menyimpan data di sisi klien (di browser) selama sesi browser berlangsung. Data yang disimpan dalam session storage akan tetap ada selama pengguna menjelajahi halaman-halaman dalam satu sesi browser, tetapi akan dihapus saat sesi browser ditutup.

   Fungsi-fungsi yang dapat digunakan untuk berinteraksi dengan session storage adalah:

   - **setItem(key, value):** Fungsi ini digunakan untuk menyimpan data di dalam session storage dengan format pasangan key-value. Anda memberikan sebuah kunci (key) dan nilai (value) yang ingin disimpan. Contoh penggunaan:

     ```
     sessionStorage.setItem('username', 'john_doe');
     ```

   - **getItem(key):** Fungsi ini digunakan untuk mengambil nilai dari session storage berdasarkan kunci yang diberikan. Contoh penggunaan:

     ```
     const username = sessionStorage.getItem('username');
     ```

   - **removeItem(key):** Fungsi ini digunakan untuk menghapus data dari session storage berdasarkan kunci yang diberikan. Contoh penggunaan:

     ```
     sessionStorage.removeItem('username');
     ```

   - **clear():** Fungsi ini digunakan untuk menghapus semua data yang ada dalam session storage untuk domain yang sama. Contoh penggunaan:

     ```
     sessionStorage.clear();
     ```

   Berikut contoh penggunaan beberapa method dari session storage sekaligus

```
<script>
// Menyimpan data ke sessionStorage
sessionStorage.setItem('language', 'english');
sessionStorage.setItem('loggedIn', 'true');

// Mengambil data dari sessionStorage
const language = sessionStorage.getItem('language');
const loggedIn = sessionStorage.getItem('loggedIn');

// Menghapus data dari sessionStorage
sessionStorage.removeItem('loggedIn');

// Menghapus semua data dari sessionStorage
sessionStorage.clear();

</script>
```

Dengan session storage, kita dapat menyimpan data sementara yang relevan selama interaksi pengguna dengan aplikasi tanpa perlu mempengaruhi penyimpanan jangka panjang. Contoh penggunaan umum termasuk penyimpanan status login pengguna, preferensi pengguna yang hanya relevan selama sesi, atau data keranjang belanja pada situs web e-commerce.

Perlu diperhatikan bahwa data dalam session storage terbatas pada satu sesi browser. Jika browser ditutup, data yang disimpan dalam session storage akan hilang. Juga, data dalam session storage hanya dapat diakses oleh halaman-halaman yang berasal dari domain yang sama seperti halaman yang menyimpan data tersebut.

4. **Perbedaan Local Storage dan Session Storage**

Cookies vs. Local Storage vs. Session Storage

	Cookies	Local Storage	Session Storage
Capacity	4kb	10mb	5mb
Browsers	HTML4 / HTML 5	HTML 5	HTML 5
Accessible from	Any window	Any window	Same tab
Expires	Manually set	Never	On tab close
Storage Location	Browser and server	Browser only	Browser only
Sent with requests	Yes	No	No

5. **Keunggulan dan Kelemahan menggunakan Web Storage**

Keunggulan:

1. **Kapasitas Lebih Besar:** Dibandingkan dengan cookies, Web Storage memiliki kapasitas yang lebih besar. Anda dapat menyimpan data hingga beberapa megabyte (tergantung pada browser).
2. **Efisiensi:** Data disimpan di sisi klien, sehingga mengurangi kebutuhan untuk terus-menerus mengirim data ke server. Ini membantu dalam mengurangi lalu lintas jaringan dan mengoptimalkan kinerja aplikasi.
3. **Tidak Termasuk dalam Permintaan HTTP:** Data di Web Storage tidak secara otomatis disertakan dalam setiap permintaan HTTP yang dikirim ke server. Ini mengurangi beban pada server, karena server tidak harus memproses data tersebut setiap kali permintaan dibuat.
4. **Penanganan Data yang Mudah:** Menggunakan metode seperti `setItem`, `getItem`, dan `removeItem`, Anda dapat dengan mudah menyimpan, mengambil, dan menghapus data dari Web Storage.
5. **Tersedia Selama Sesi:** Data yang disimpan dalam sessionStorage akan tetap ada selama sesi browser berlangsung, yang memungkinkan penyimpanan data sementara yang relevan selama interaksi pengguna dengan aplikasi.

Kelemahan:

1. **Penyimpanan Terbatas:** Meskipun memiliki kapasitas yang lebih besar daripada cookies, Web Storage masih memiliki batasan kapasitas (biasanya beberapa megabyte per domain), yang dapat menyebabkan masalah jika Anda perlu menyimpan data yang sangat besar.
2. **Tidak Cocok untuk Data Sensitif:** Data yang disimpan dalam Web Storage mudah diakses dan dapat dilihat oleh pengguna melalui konsol pengembang browser. Oleh karena itu, Web Storage tidak cocok untuk menyimpan data sensitif atau rahasia seperti kata sandi.

3. **Tidak Selalu Tersedia:** Beberapa pengaturan keamanan pada browser atau pengaturan pribadi pengguna dapat memblokir penggunaan Web Storage, yang dapat menghambat fungsionalitas aplikasi jika mengandalkan fitur ini.
4. **Tidak Mendukung Sinkronisasi Antar Perangkat:** Data dalam Web Storage tidak secara otomatis disinkronisasi antar perangkat pengguna. Jika pengguna mengakses aplikasi dari beberapa perangkat, data yang disimpan di satu perangkat tidak akan langsung tersedia di perangkat lain.
5. **Tidak Mendukung Struktur Data Kompleks:** Web Storage hanya dapat menyimpan data dalam bentuk pasangan key-value. Jika Anda perlu menyimpan struktur data yang lebih kompleks, seperti array atau objek bersarang, Anda perlu melakukan konversi manual ke dalam format yang sesuai.

D. LATIHAN

1. Latihan 1

a. Tulis skrip di bawah ini

```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(event)">Masukkan Nama</button>
</center>

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

    // Mengecek apakah sudah ada nama tersimpan di localStorage
    const namaTersimpan = localStorage.getItem('nama');
    if (namaTersimpan) {
        namaMember.innerHTML = 'Halo ' + namaTersimpan + '!';
    }

    function gantiNama(e) {
        let nama = prompt('Masukkan nama Anda:');
        if (nama && nama.trim() !== "") {
            localStorage.setItem('nama', nama);
            namaMember.innerHTML = 'Halo ' + nama + '!';
        } else {
            alert('Nama tidak boleh kosong!');
        }
        e.preventDefault();
    }
</script>
</body>
</html>
```

b. Jalankan, lalu masukkan isian sesuai perintah. Apa yang terjadi?

c. Tambahkan sebuah tombol yang dapat menghapus nama dari member!

d. Modifikasi javascript sehingga data nama member tetap akan ada selama belum dihapus!

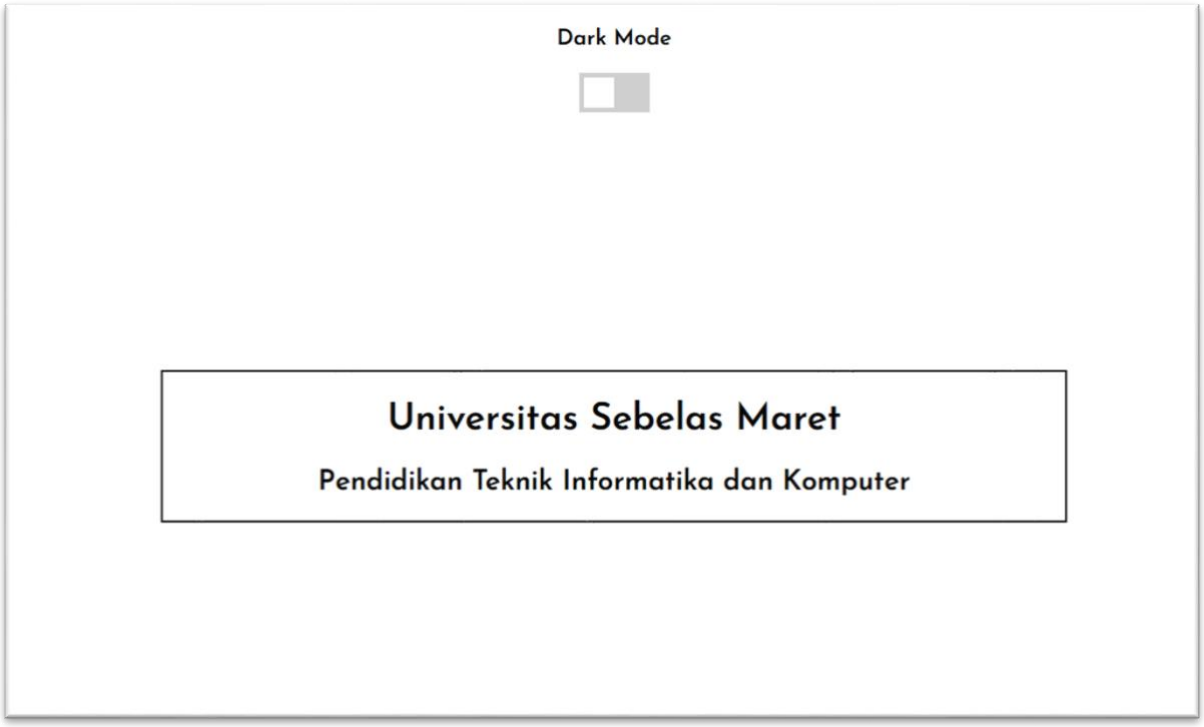
2. Latihan 3

a. Tulis skrip di bawah ini

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Web Storage</title>
    <style type="text/css">
        @import url('https://fonts.googleapis.com/css2?
            family=Josefin+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;1,100;1,200;1,300;1,400;1,500&
            family=Tilt+Prism&display=swap');

        body{
            font-family: 'Josefin Sans', sans-serif;
        }

        .label{
            display: block;
            text-align: center;
        }

        .tulisan{
            text-align: center;
            margin: auto;
            border: 2px solid;
            width: 50%;
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
        }

        .switch {
          position: relative;
          display: inline-block;
          width: 60px;
          height: 34px;
        }

        .switch input {
          opacity: 0;
          width: 0;
          height: 0;
        }

        .slider {
          position: absolute;
          cursor: pointer;
          top: 0;
          left: 0;
          right: 0;
          bottom: 0;
          background-color: #ccc;
          -webkit-transition: .4s;
          transition: .4s;
        }

        .slider:before {
          position: absolute;
          content: "";
          height: 26px;
          width: 26px;
          left: 4px;
          bottom: 4px;
          background-color: white;
          -webkit-transition: .4s;
          transition: .4s;
        }

```

```
66
67          input:checked + .slider {
68            background-color: #2196F3;
69          }
70
71          input:focus + .slider {
72            box-shadow: 0 0 1px #2196F3;
73          }
74
75          input:checked + .slider:before {
76            -webkit-transform: translateX(26px);
77            -ms-transform: translateX(26px);
78            transform: translateX(26px);
79          }
80
81
82      </style>
83  </head>
84  <body>
85      <div class="tulisan">
86          <h1>Universitas Sebelas Maret</h1>
87          <h2>Pendidikan Teknik Informatika dan Komputer</h2>
88      </div>
89          <div class="label">
90              <h3>Dark Mode</h3>
91              <label class="switch">
92                <input type="checkbox" onchange="ubahTema(event)">
93                <span class="slider"></span>
94              </label>
95          </div>
96  </body>
97  </html>
```

b. Jalankan, maka akan muncul tampilan seperti berikut



c. **Modifikasi file tersebut dengan menambahkan css dan javascript yang tepat dengan memanfaatkan local storage sehingga fungsi dark mode dapat berjalan ketika toggle dark mode dijalankan.**