Menerapkan localStorage pada Kalkulator

Sebelum kita menuliskan sintaks pada ***storage.js***, pastikan kita sudah menghubungkan berkas tersebut dengan ***index.html*** seperti berikut ini:

```
....
    <script src="assets/storage.js"></script>
    <script src="assets/kalkulator.js"></script>
</body>
```

Pastikan juga kita menghubungkannya sebelum ***kalkulator.js*** karena kita akan menggunakan ***storage.js*** pada berkas ***kalkulator.js***. Alhasil, berkas ***storage.js*** perlu dimuat terlebih dahulu.

Setelah kita menghubungkan berkas JavaScript dengan HTML, buka berkas ***storage.js***. Kemudian buat variabel `CACHE_KEY` dengan nilai **"calculation_history".**

```
const CACHE_KEY = "calculation_history";
```

Variabel ini digunakan sebagai *key* untuk mengakses dan menyimpan data pada `localStorage`.

Selanjutnya kita buat fungsi `checkForStorage()` dengan mengembalikan nilai boolean dari pengecekan fitur Storage pada browser.

```
function checkForStorage() {
 return typeof(Storage) !== "undefined"
}
```

Fungsi tersebut akan kita gunakan di dalam *if statement* setiap fungsi transaksi pada `localStorage`.

Lalu kita buat juga fungsi untuk menyimpan data riwayat kalkulasi pada `localStorage`. Fungsi tersebut memiliki satu argumen yang merupakan data dari hasil kalkulasi yang nantinya akan dimasukkan ke dalam `localStorage`.

```
    if (checkForStorage()) {
        let historyData = null;
        if (localStorage.getItem(CACHE_KEY) === null) {
            historyData = [];
        } else {
            historyData = JSON.parse(localStorage.getItem(CACHE_KEY));
        }

        historyData.unshift(data);

        if (historyData.length > 5) {
            historyData.pop();
        }

        localStorage.setItem(CACHE_KEY, JSON.stringify(historyData));
    }
```

Banyak istilah kode yang asing pada kode di atas. Mari kita lihat satu per satu.

Yang pertama fungsi `JSON.parse()` yang mana digunakan untuk mengubah nilai objek dalam bentuk string kembali pada bentuk objek JavaScript. Kemudian `JSON.stringify()` digunakan untuk mengubah objek JavaScript ke dalam bentuk String. Seperti yang kita tahu, bahwa `localStorage` hanya dapat menyimpan data primitif seperti string, sehingga kita perlu mengubah objek ke dalam bentuk string jika ingin menyimpan ke dalam `localStorage` .

JSON sendiri adalah singkatan dari *JavaScript Object Notation*. JSON merupakan format yang sering digunakan dalam pertukaran data. Saat ini JSON banyak diandalkan karena formatnya berbasis teks dan relatif mudah dibaca.

Lalu di sana juga ada fungsi `unshift()` , fungsi ini digunakan untuk menambahkan nilai baru pada array yang ditempatkan pada awal index. Fungsi ini juga mengembalikan nilai panjang array setelah ditambahkan dengan nilai baru.

Fungsi `pop()` di atas merupakan fungsi untuk menghapus nilai index terakhir pada array, sehingga ukuran array `historyData` tidak akan pernah lebih dari 5. Hal ini kita terapkan agar riwayat kalkulasi yang muncul adalah lima hasil kalkulasi terakhir oleh pengguna.

Dari sini kita bisa mengetahui bahwa data yang disimpan pada `localStorage` adalah array yang berisi beberapa objek hasil kalkulasi, kemudian array tersebut diubah menjadi string. Struktur data tersebut dalam bentuk string nampak seperti ini:

```
[
    {
        "firstNumber": "23",
        "secondNumber": "15",
        "operator": "-",
        "result": 8
    },
    {
        "firstNumber": "23",
        "secondNumber": "6",
        "operator": "-",
        "result": 17
    }
]
```

```
function showHistory() {
    if (checkForStorage()) {
        return JSON.parse(localStorage.getItem(CACHE_KEY)) || [];
    } else {
        return [];
    }
}
```

Fungsi ini mengembalikan nilai array dari `localStorage` jika sudah memiliki nilai sebelumnya melalui `JSON.parse()`. Namun jika `localStorage` masih kosong, fungsi ini akan mengembalikan nilai array kosong.

Lalu yang terakhir, kita tambahkan fungsi untuk merender data riwayat kalkulasi pada tabel HTML. Fungsi ini diberi nama dengan `renderHistory()`.

```
function renderHistory() {
    const historyData = showHistory();
    let historyList = document.querySelector("#historyList");


    // selalu hapus konten HTML pada elemen historyList agar tidak menampilkan data ganda
    historyList.innerHTML = "";


    for (let history of historyData) {
        let row = document.createElement('tr');
        row.innerHTML = "<td>" + history.firstNumber + "</td>";
        row.innerHTML += "<td>" + history.operator + "</td>";
        row.innerHTML += "<td>" + history.secondNumber + "</td>";
        row.innerHTML += "<td>" + history.result + "</td>";


        historyList.appendChild(row);
```

Pada akhir berkas ***storage.js***, panggil fungsi `renderHistory();` agar data history muncul ketika website pertama kali dijalankan.

```
renderHistory();
```

Sampai saat ini, struktur kode pada ***storage.js*** akan tampak seperti berikut:

```
2.
3.  function checkForStorage() {
4.     return typeof(Storage) !== "undefined";
5.  }
6.
7.  function putHistory(data) {
8.     if (checkForStorage()) {
9.         let historyData = null;
10.        if (localStorage.getItem(CACHE_KEY) === null) {
11.            historyData = [];
12.        } else {
13.            historyData = JSON.parse(localStorage.getItem(CACHE_KEY));
14.        }
15.
16.        historyData.unshift(data);
17.
18.        if (historyData.length > 5) {
```

Terakhir kita gunakan fungsi `putHistory()` yang sudah kita buat ketika kalkulator melakukan proses kalkulasi, tepatnya pada fungsi `performCalculation()` berkas ***kalkulator.js***.

Sebelum memanggil fungsi `putHistory()`, tentu kita harus menyiapkan data yang akan dikirimkan sebagai argumen fungsi tersebut. Pada `performCalculation()` kita buat variabel baru dengan nama `history` yang merupakan objek dari data `history` yang akan dikirimkan. Struktur objeknya tampak seperti berikut:

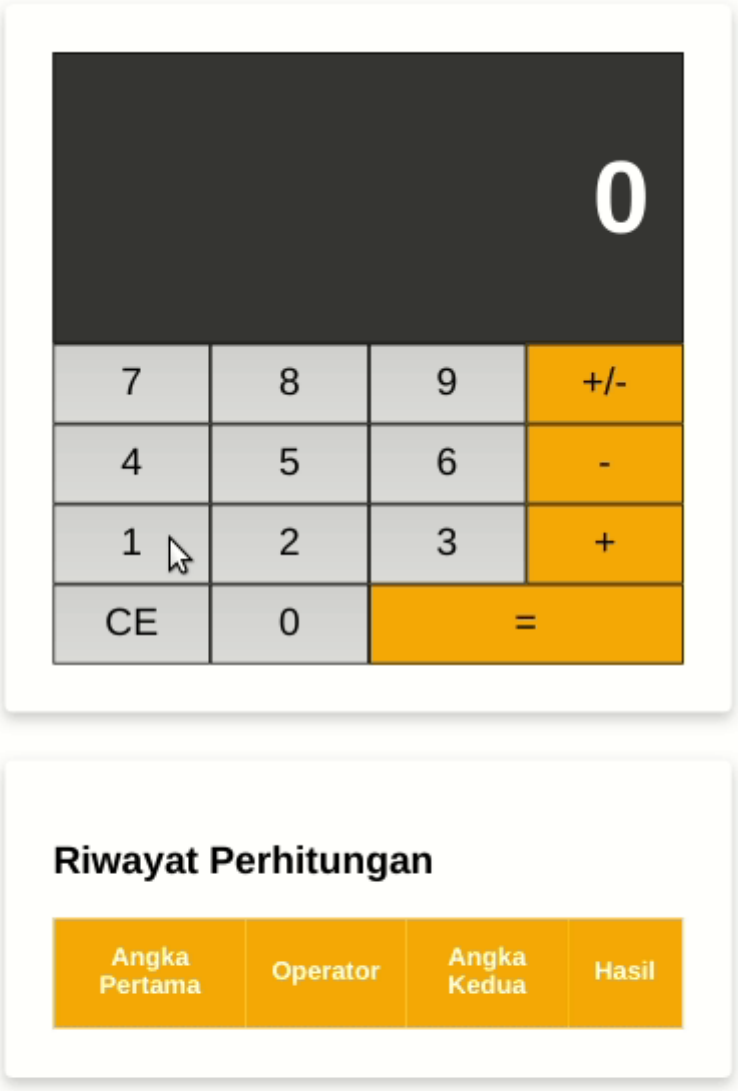
```
1.  const history = {
2.          firstNumber: calculator.firstNumber,
3.          secondNumber: calculator.displayNumber,
4.          operator: calculator.operator,
5.          result: result
6.  }
```

Setelah menyiapkan datanya, barulah kita bisa memanggil fungsi `putHistory()` dengan mengirimkan variabel history sebagai argumen fungsinya. Jangan lupa juga panggil fungsi `renderHistory()` agar riwayat kalkulasi langsung tampil pada tabel setelah kalkulasi dilakukan.

Sehingga sekarang struktur kode dari fungsi `performCalculation()` akan tampak seperti berikut:

```
2.      if (calculator.firstNumber == null || calculator.operator == null) {
3.          alert("Anda belum menetapkan operator");
4.          return;
5.      }
6.
7.      let result = 0;
8.      if (calculator.operator === "+") {
9.          result = parseInt(calculator.firstNumber) + parseInt(calculator.displayNumber);
10.     } else {
11.         result = parseInt(calculator.firstNumber) - parseInt(calculator.displayNumber)
12.     }
13.
14.     // objek yang akan dikirimkan sebagai argumen fungsi putHistory()
15.     const history = {
16.         firstNumber: calculator.firstNumber,
17.         secondNumber: calculator.displayNumber,
18.         operator: calculator.operator,
```

Pada tahap ini seharusnya kalkulator kita sudah dapat menampilkan riwayat kalkulasi yang dilakukan pengguna.



Selamat, Anda sudah berhasil membuat aplikasi kalkulator berbasis web dengan baik. Good Job!

Sebelumnya

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Decode Ideas

Discover Potential

> Tentang Kami

Reward

Showcase

FAQ

Penghargaan

image
click bila
belum muncul

DIBANTU