**Session Storage - Menyimpan Data**

Berbeda dengan *local storage*, walaupun masuk ke dalam *web storage*, data yang tersimpan pada *session storage* **akan hilang** ketika *session* dari sebuah laman berakhir.

*Session storage* mempunyai beberapa karakteristik, yaitu:

1. Data yang disimpan pada *session storage* akan terus tersimpan selama *browser* terbuka dan tidak hilang jika laman di-*reload*.
2. Membuka banyak *tab/window* dengan URL yang sama, akan menciptakan *session storage* yang berbeda di masing-masing *tab/window*.
3. Menutup *tab/window* akan mengakhiri *session* dan menghapus data yang tersimpan di *session storage* pada *tab/window* tersebut.
4. Data yang tersimpan dalam *session storage* harus berbentuk *string*.
5. Hanya dapat menyimpan data sebanyak 5MB.

Pada umumnya kita memanfaatkan *session storage* untuk menyimpan data keranjang belanja (***cart***) pada sebuah situs *e-commerce*. Kita tentu pernah memasukkan beberapa barang ke keranjang belanja pada sebuah situs dalam keadaan belum *login*, namun setelah kita tutup *window* / *tab* nya, keranjang belanja yang tadi telah kita isi menjadi kosong lagi.

Sama dengan *local storage*, sintaks untuk menyimpan data pada *session storage* adalah sebagai berikut:

// menambah session storage

sessionStorage.setItem('key', value);

Untuk memahami penggunaan *session storage* lebih detail, pada topik ini kita akan membuat sebuah fitur penyimpanan keranjang belanja.

**Langkah 1 - buat *file* index.html**

Isi *file* index.html dengan kode berikut:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<style>

body {

background: white;

color: #323232;

margin: 0;

height: 100vh;

display: flex;

align-items: center;

justify-content: center;

}

.list-wrapper {

width: 50%; display: flex; flex-direction: column

}

.list {

display: flex; flex-direction: row; justify-content: space-between; margin-bottom: 10px

}

</style>

</head>

<body>

<div class="list-wrapper">

<div class="list">

<div style="flex: 1">Susu - Rp. 18.000,-</div>

<button onclick="addToCart('susu', 1)">Add to cart</button>

</div>

<div class="list">

<div style="flex: 1">Telor - Rp. 28.000,-</div>

<button onclick="addToCart('telor', 1)">Add to cart</button>

</div>

<div class="list">

<div style="flex: 1">Madu - Rp. 48.000,-</div>

<button onclick="addToCart('madu', 1)">Add to cart</button>

</div>

<div class="list">

<div style="flex: 1">Jahe - Rp. 8.000,-</div>

<button onclick="addToCart('jahe', 1)">Add to cart</button>

</div>

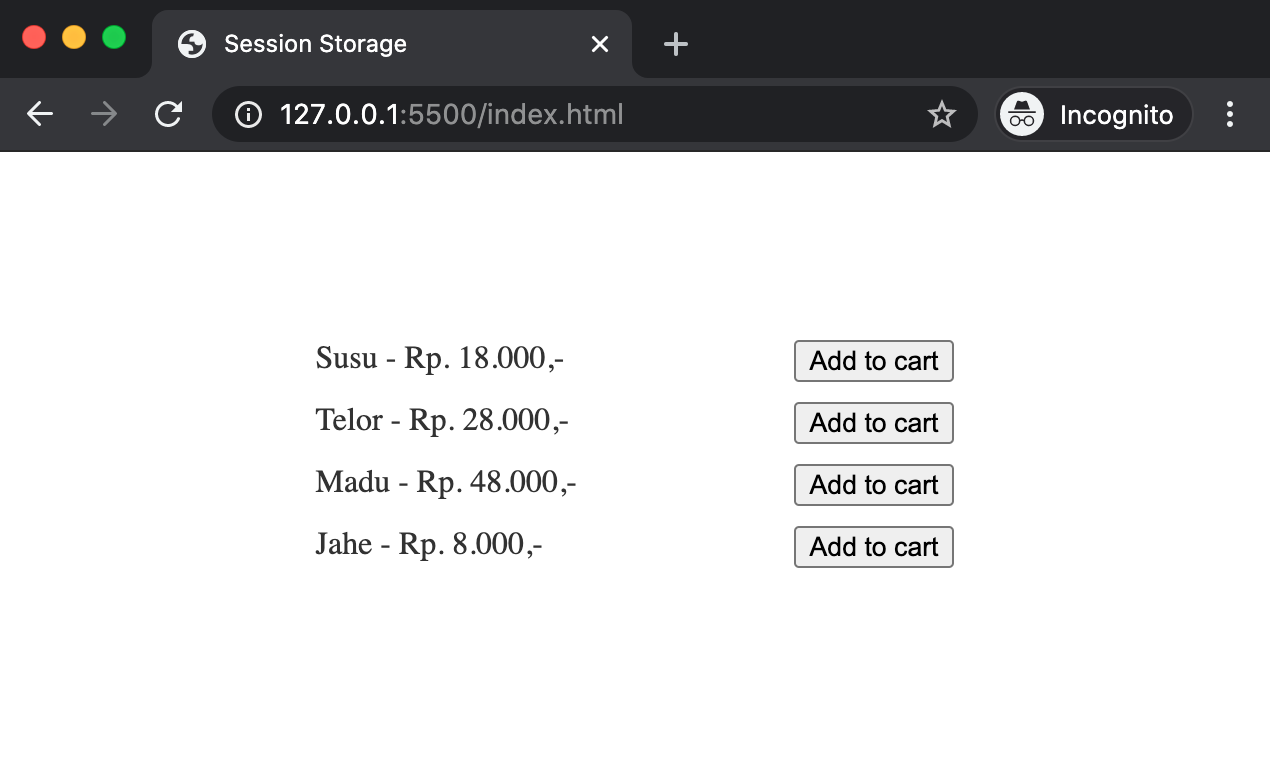
</div>

</body>

<script src="index.js"></script>

</html>

Saat kita jalankan kode di atas, kita akan mendapatkan tampilan sebagai berikut:



**Langkah 2 - buat *file* index.js**

Isi *file* index.js dengan kode berikut:

let cartList = [];

function addToCart(name, qty){

cartList.push({

name,

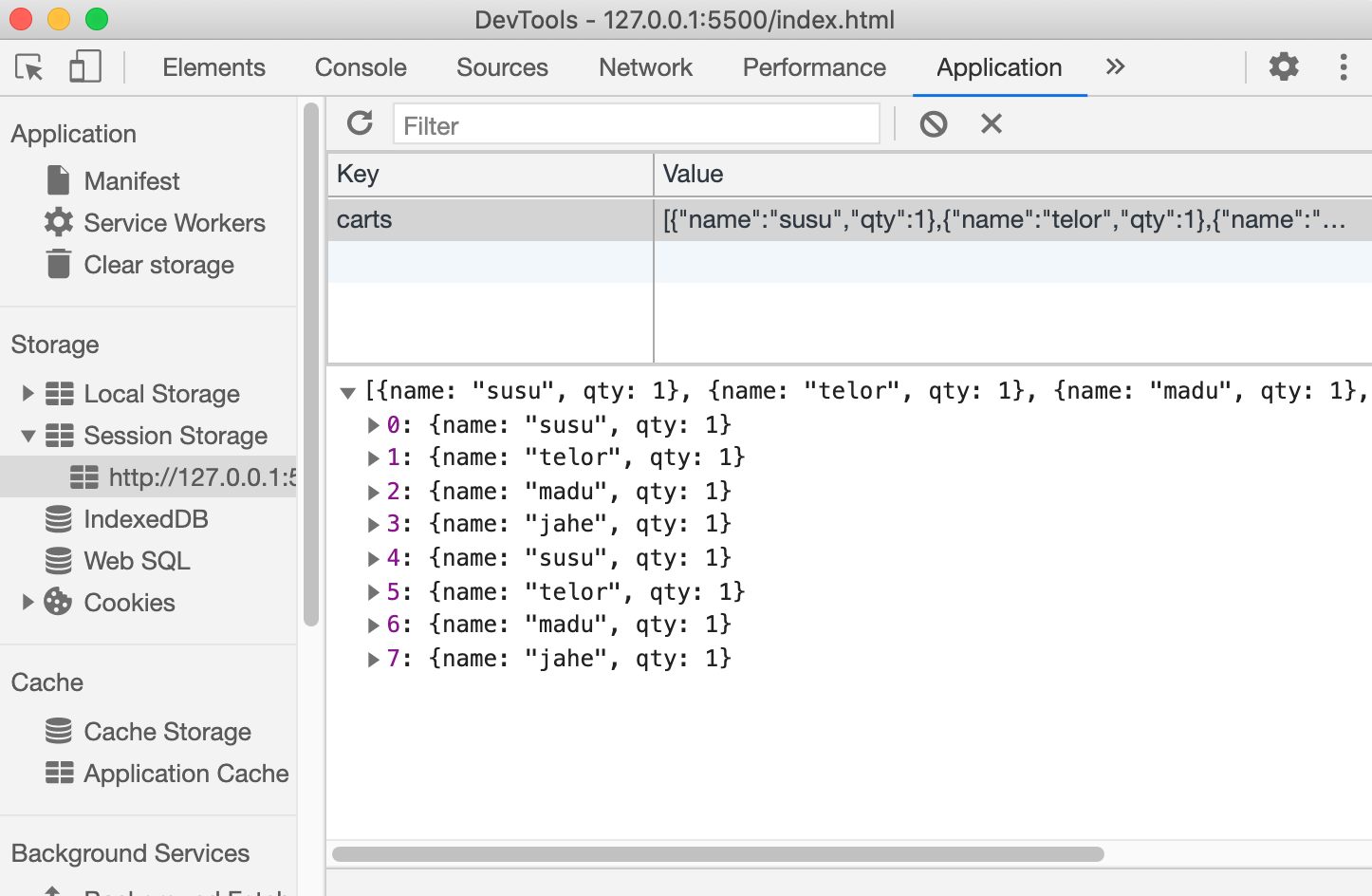
qty

});

sessionStorage.setItem("carts", JSON.stringify(cartList)); // array object diubah menjadi string

}

Saat kita jalankan kode di atas, lalu masuk ke *Developer Tools* di browser dengan cara klik kanan > *inspect element* > *application* > *session storage*, maka kita akan mendapatkan hasil sebagai berikut:



Namun, pada fungsi addToCart terdapat masalah dimana saat kita mengklik kembali tombol Add to cart pada masing-masing *item*, maka akan terjadi duplikasi data seperti gambar di atas. Maka untuk mengatasi masalah tersebut, kita dapat memodifikasi fungsi addToCart di langkah selanjutnya.

**Langkah 3 - modifikasi *function* addToCart**

Lakukan modifikasi pada *function* addToCart seperti berikut:

const cartList = [];

function addToCart(name, qty){

const indexItem = cartList.findIndex(data => data.name === name) // memeriksa apakah item name sudah ada atau belum pada cartList

if(indexItem > -1) {

cartList[indexItem].qty +=1 // jika sudah ada, qty+1 pada data di index ke indexItem; ingat kembali materi array object

} else { // jika belum ada, push data baru ke dalam cartList

cartList.push({

name,

qty

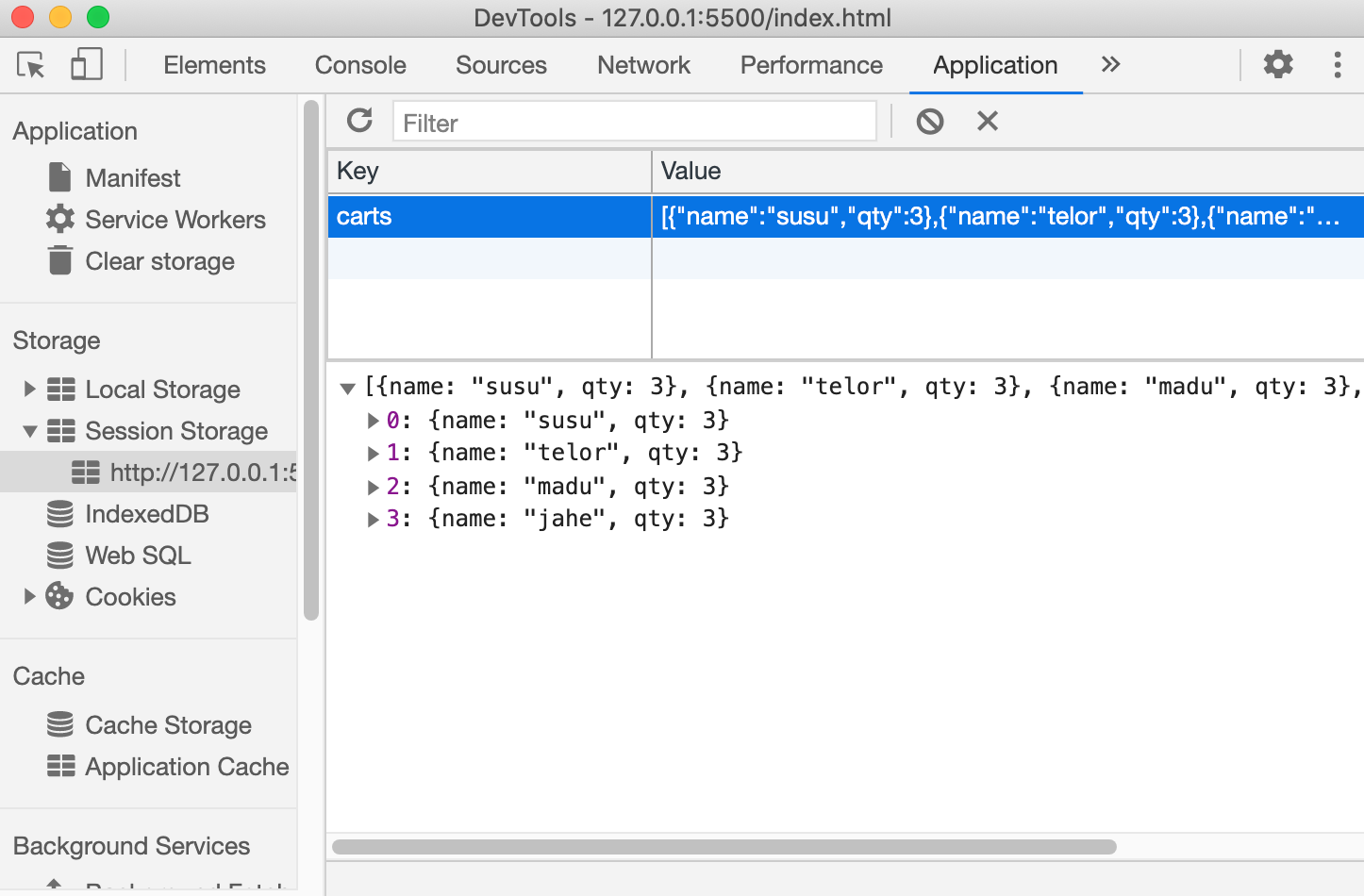
});

}

sessionStorage.setItem("carts", JSON.stringify(cartList)); // set session storage

}

Pada kode di atas, kita menggunakan JSON.stringify() untuk mengubah *array of object* menjadi *string* saat disimpan ke dalam *session storage*. Hasil dari kode di atas akan melakukan *update* qty menjadi seperti ini:



Pada topik berikutnya kita akan melanjutkan fitur hapus data di keranjang dan menghapus data dari *session storage*.

# Session Storage - Mengambil Data

Setelah kita mempelajari cara menyimpan data di session storage, pada topik ini kita akan belajar bagaimana kita dapat mengambil dan mengolah data dari session storage.

Sama seperti local storage, cara mendapatkan data dari session storage juga menggunakan getItem(), seperti berikut ini:

// mendapatkan session storage

sessionStorage.getItem('key');

Untuk lebih jelasnya mengenai penggunaan getItem(), mari kita lanjutkan aplikasi yang telah kita buat pada topik sebelumnya.

## Langkah 1 - Membuat menu View Cart

Kita akan menambahkan baris kode berikut pada file index.html dari topik sebelumnya:

<div class="list-wrapper">

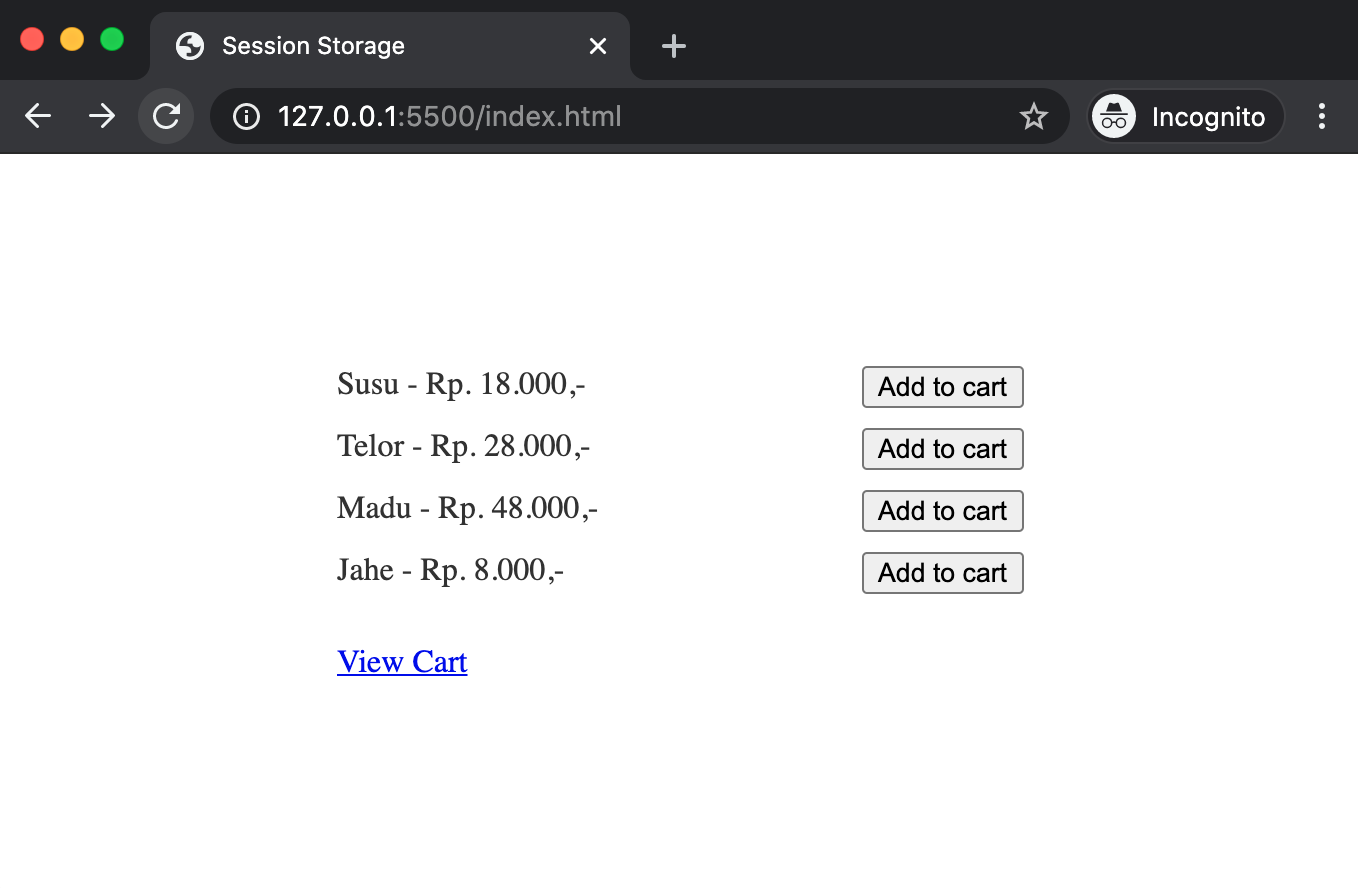
...

...

<div style="margin-top: 15px;"><a href="cart.html">View Cart</a></div>

</div>

Misalnya kita buat trigger View Cart menggunakan tag <a> di baris paling bawah dalam list-wrapper. Sehingga hasilnya menjadi seperti ini:



Jika kita perhatikan kode di atas, tag <a> ini jika diklik akan menuju file cart.html. Maka, langkah berikutnya kita harus membuat file cart.html.

## Langkah 2 - Membuat file cart.html

Tambahkan kode berikut ini ke dalam file cart.html yang telah kita buat:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<style>

body {

background: white;

margin: 0;

height: 100vh;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

}

#list-cart {

display: flex;

flex-direction: column;

}

.row {

display: flex;

flex-direction: row;

justify-content: space-between;

margin-bottom: 10px;

border-bottom: 1px solid black;

}

</style>

</head>

<body>

<div style="width: 50%">

<div class="row">

<div>Name</div>

<div>Quantity</div>

</div>

<div id="list-cart"></div>

</div>

<script></script>

</body>

</html>

Dari kode di atas, dapat kita lihat bahwa cart.html berisi tag <div> dengan id list-cart tidak memiliki isi konten. Lalu tag <script> juga tidak lagi membaca index.js.

## Langkah 3 - Tambahkan kode pada tag script

Tambahkan kode berikut ke dalam tag script pada file cart.html:

var cartList = JSON.parse(sessionStorage.getItem('carts')); // mengambil data dari session storage, di parsing kembali dari string menjadi array object

// merender name dan qty dari cart

var list = '';

for (var i = 0; i < cartList.length; i++) {

list += `<div class="row">

<div id="name">${cartList[i].name}</div>

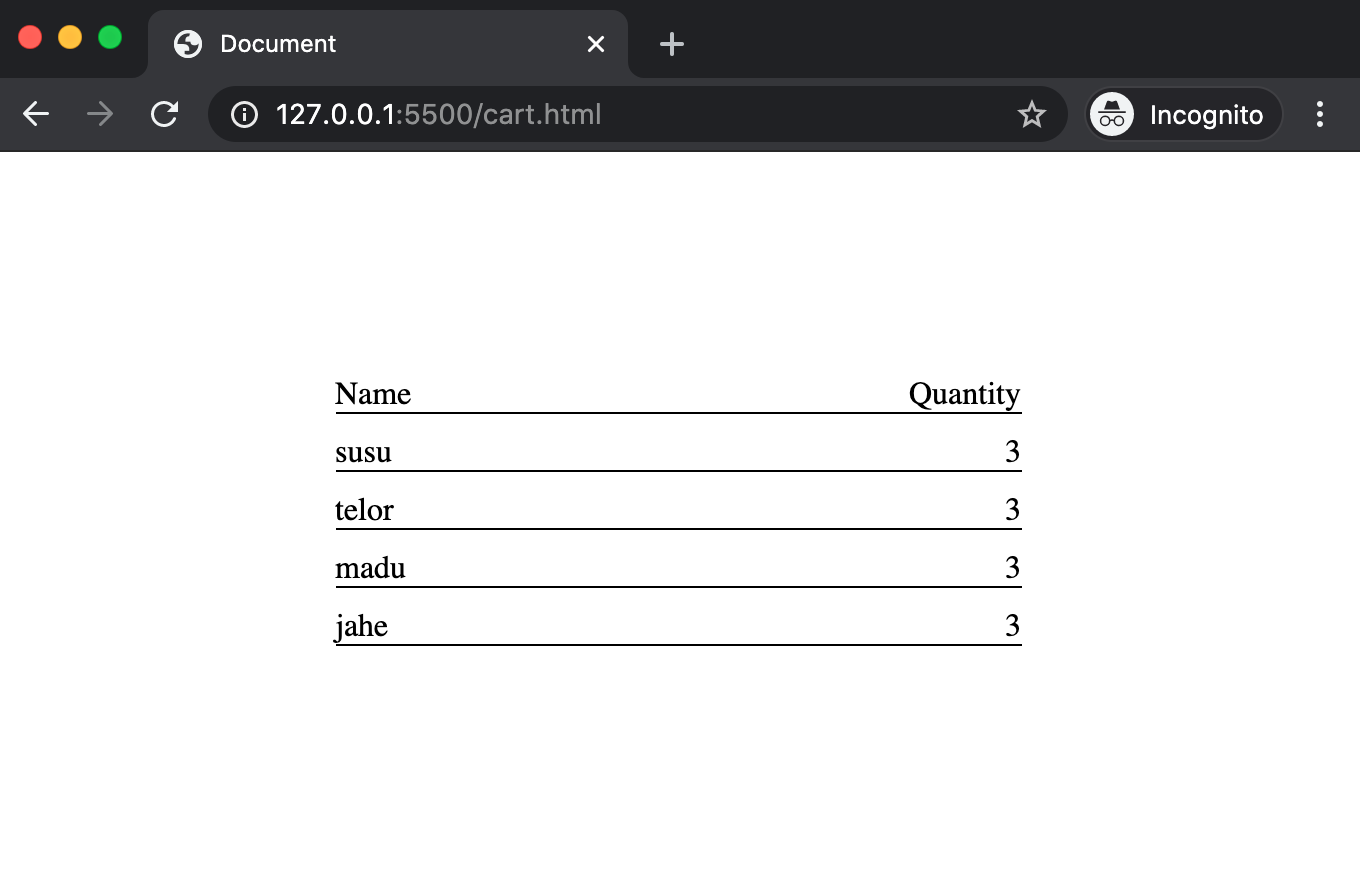
<div id="qty">${cartList[i].qty}</div>

</div>`

}

document.getElementById('list-cart').innerHTML = list;

Jika file cart.html dijalankan, maka hasilnya akan menjadi seperti ini:



Dapat kita lihat bahwa dari kode di atas, kita mendapatkan apa yang ada di keranjang belanja dengan cara:

var cartList = JSON.parse(sessionStorage.getItem('carts'));

Perlu diketahui, JSON.parse() berfungsi mengembalikan tipe data string yang kita dapat dari session storage kembali menjadi JSON. Lalu untuk merender data dari session storage, kita menggunakan innerHTML, dimana innerHTML menerima string berisikan tag HTML.

**Session Storage - Menghapus Data**

Setelah kita belajar cara menyimpan data ke dalam *session storage* pada topik sebelumnya, pada topik ini kita akan mempelajari cara menghapus data dari *session storage*.

Syntax untuk menghapus data dari *session storage* ada 2, yaitu:

// menghapus session storage satu persatu berdasarkan key

sessionStorage.removeItem('key');

// menghapus seluruh session storage sekaligus

sessionStorage.clear();

Agar lebih paham mengenai implementasinya, mari kita lanjutkan pembuatan aplikasi pada topik sebelumnya. Jika pada topik sebelumnya kita telah menambahkan fungsi addToCart(), maka pada topik ini, kita akan membuat fungsi removeFromCart().

**Langkah 1 - Tambahkan button Remove from cart**

Kita akan menambahkan beberapa baris kode pada file index.html sebelumnya untuk membuat button Remove from Cart pada setiap produk:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Session Storage</title>

<style>

body {

background: white;

color: #323232;

margin: 0;

height: 100vh;

display: flex;

align-items: center;

justify-content: center;

}

.list-wrapper {

width: 50%; display: flex; flex-direction: column

}

.list {

display: flex; flex-direction: row; justify-content: space-between; margin-bottom: 10px

}

</style>

</head>

<body>

<div class="list-wrapper">

<div class="list">

<div style="flex: 1">Susu - Rp. 18.000,-</div>

<button onclick="addToCart('susu', 1)">Add to cart</button>

<button onclick="removeFromCart('susu')">Remove from cart</button>

</div>

<div class="list">

<div style="flex: 1">Telor - Rp. 28.000,-</div>

<button onclick="addToCart('telor', 1)">Add to cart</button>

<button onclick="removeFromCart('telor')">Remove from cart</button>

</div>

<div class="list">

<div style="flex: 1">Madu - Rp. 48.000,-</div>

<button onclick="addToCart('madu', 1)">Add to cart</button>

<button onclick="removeFromCart('madu')">Remove from cart</button>

</div>

<div class="list">

<div style="flex: 1">Jahe - Rp. 8.000,-</div>

<button onclick="addToCart('jahe', 1)">Add to cart</button>

<button onclick="removeFromCart('jahe')">Remove from cart</button>

</div>

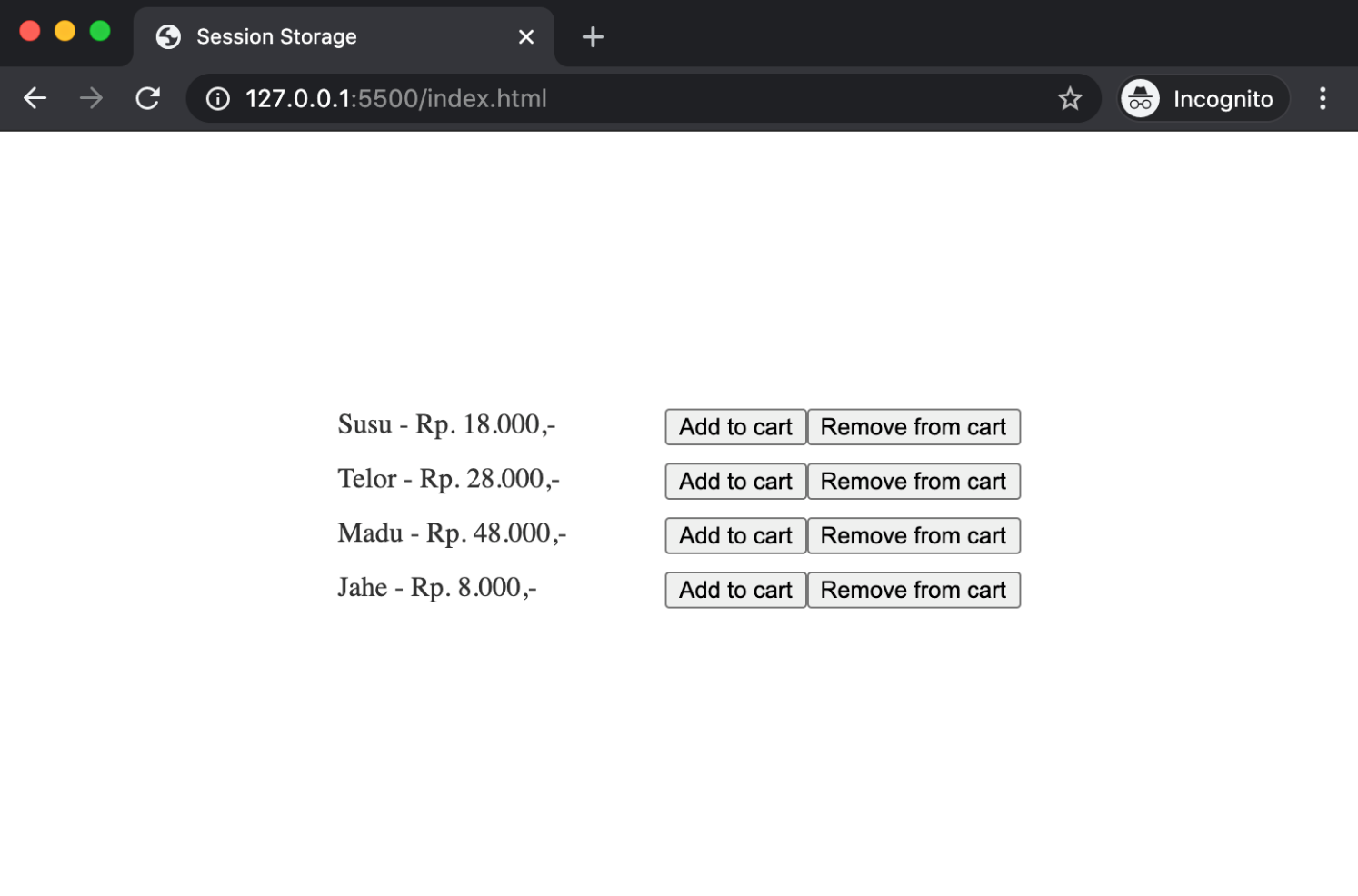
</div>

</body>

<script src="index.js"></script>

</html>

Jika kode diatas dijalankan maka akan didapatkan hasilnya seperti berikut ini:



**Langkah 2 - Buat *function* removeFromCart()**

Selanjutnya kita akan membuat *function* bernama removeFromCart() pada *file* index.js dari topik sebelumnya:

function removeFromCart(name) {

const indexItem = cartList.findIndex(data => data.name === name)

if(indexItem > -1) {

if(cartList[indexItem].qty > 1) { // Jika qty barang lebih dari 1

cartList[indexItem].qty -=1 // maka qty barang dikurangi 1

} else { // tapi jika qty barang = 1

cartList.splice(indexItem, 1) // maka hapus barang dari cart

}

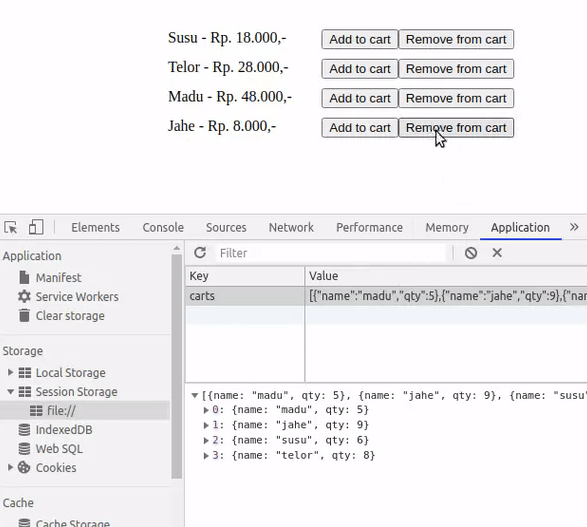
}

sessionStorage.setItem("carts", JSON.stringify(cartList)); // memperbaharui data keranjang

};

dari kode di atas, dapat kita lihat bahwa di fungsi ini kita **hanya** memanipulasi kuantitas (qty) barang yang ada di keranjang belanja. Kita terus memperbaharui data di *session storage* menggunakan sessionStorage.setItem() terhadap jumlah barang yang kita masukan ke keranjang maupun yang kita hapus dari keranjang.

Sehingga kita mendapatkan hasil seperti ini:



Lalu kapan kita menggunakan sessionStorage.removeItem() ? Kita dapat mengimplementasikan sessionStorage.removeItem() saat kita hendak mengosongkan seluruh barang yang ada pada keranjang belanja kita. Mari kita lanjutkan kode di atas.

**Langkah 3 - Buatlah button Empty Cart**

Buatlah sebuah button bernama Empty cart pada *file* index.html sebelumnya seperti berikut ini:

<div class="list-wrapper">

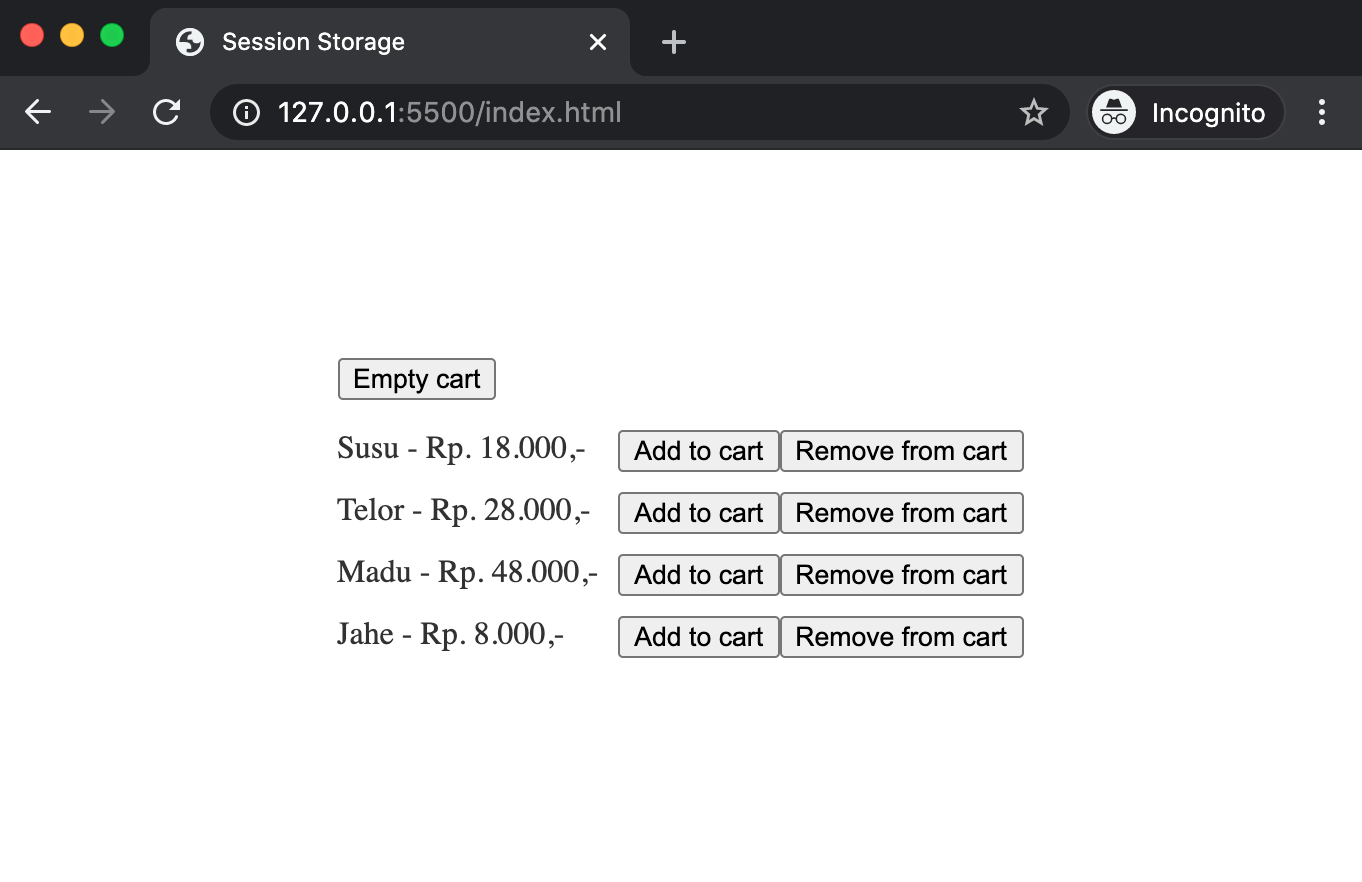
<div style="margin-bottom: 15px;"><button onclick="emptyCart()">Empty cart</button></div>

...

...

</div>

Letakkan button Empty cart tepat dibawah class="list-wrapper". Sehingga tampilannya akan menjadi seperti ini:



**Langkah 4 - Buat *function* emptyCart()**

Terakhir, buatlah sebuah *function* bernama emptyCart() pada file index.js sebelumnya:

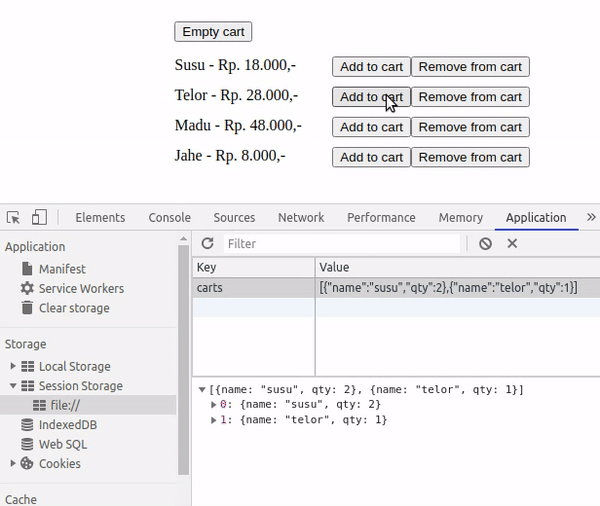
function emptyCart() {

sessionStorage.removeItem('carts'); // menghapus session storage 'carts'

cartList = []; // menjadikan cartList sebagai array kosong kembali

}

Ketika kode di atas dijalankan, maka hasilnya akan menjadi seperti ini:



Dari gambar di atas dapat kita lihat bahwa data carts pada *session storage* hilang seluruhnya setelah mengklik button Empty cart.

Sekarang kita telah mengetahui pengertian dan penerapan *local storage* dan *session storage*, bagaimana kita menentukan kapan untuk memakai antara keduanya? Jawabannya, tergantung bagaimana kebutuhan kita. Apakah data yang disimpan dalam *web storage* hendak disimpan dalam waktu lama atau langsung dihilangkan saat *user* menutup *browser*.

📝**Catatan**:

Penggunaan *web storage* ditentukan berdasarkan kebutuhan produk dan pengguna dengan mempertimbangkan karakteristik masing-masing pada *cookie*, *local storage* dan *session storage*