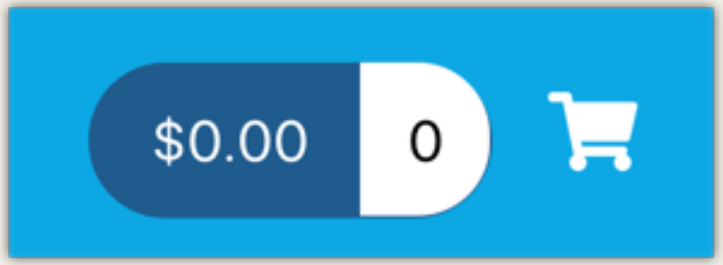
Handling Browser Refresh

We have a problem ...

- If we add products to the cart ...
  - Then refresh browser page ... we lose products in the cart ... yikes!!!



- The same problem happens when
  - We add products to the cart
  - Then login ... we lose products in the cart :-(

One Possible Solution

- Keep track of the cart products using client side browser web storage

- HTML5 introduced the **Web Storage API**
  - Store data in the browser using key / value pairs
  - Similar to cookies but provides a more intuitive API
  - Requires a modern web browser that supports HTML5

Based on a solution provided by Matt Seymour

Thanks Matt!!

Two Types of Web Storage

- **Session Storage**
  - Data is stored in web browser's memory

- **Local Storage**
  - Data is stored on client side computer

Session Storage

- Stores the data in the web browser's session (memory)
  - Data is never sent to the server (don't confuse with HttpSession)
- Each web browser tab is it's own "session"
  - Data is not shared between web browser tabs
- Once a web browser tab is closed then data is no longer available

Local Storage

- Stores the data locally on the client web browser computer
  - Data is never sent to the server

- Data is available to tabs of the same web browser for same origin
  - App must read data again ... normally with a browser refresh

- Data persists even if the web browser is closed
  - No expiration date on the data
  - Can clear data using JavaScript or clearing web browser cache

Data Scoping

Only pages from the same origin can access the data

- The data is scoped to a given "origin"
  - Origin is: **`protocol + hostname + port`**

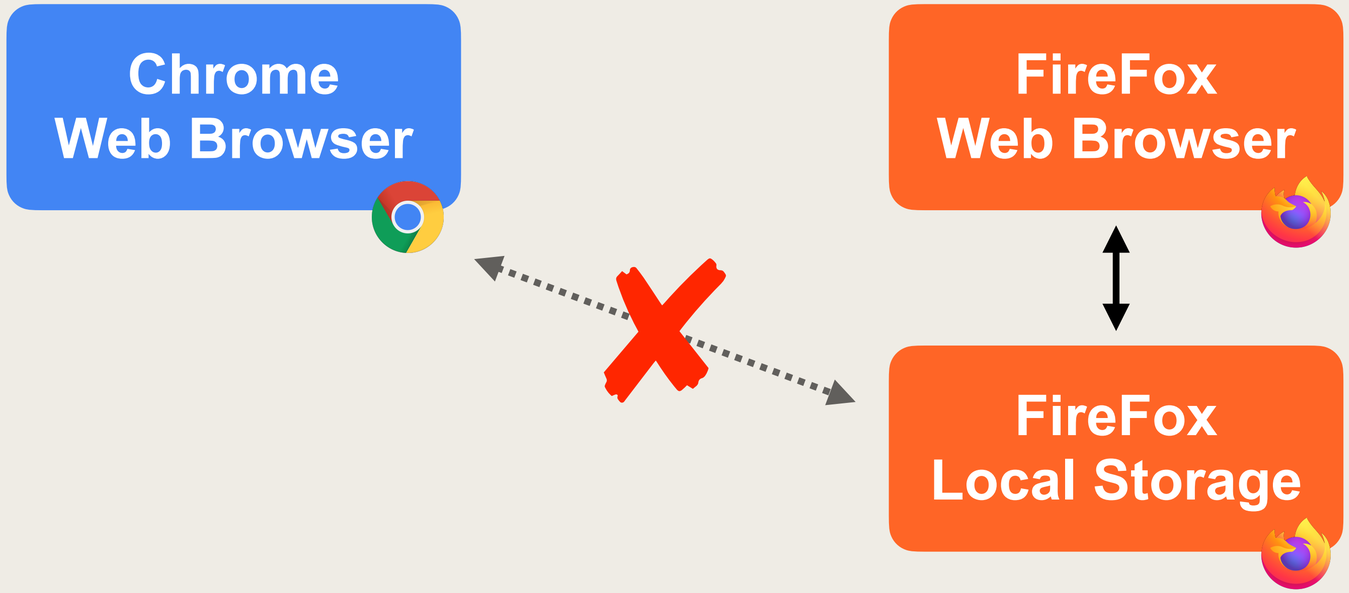
- Same origin `http://localhost:4200 == http://localhost:4200`

- Different origin `http://localhost:4200 != http://localhost:8080`

Different ports

Local Storage

- For Local Storage ... data is NOT shared between different web browsers
- For example ... Chrome can not access Local Storage of FireFox etc ...



Factors to consider

- Data in web storage is stored as plain text ... NOT encrypted
  - Do not use it to store sensitive info such as credit card numbers etc ...
  - Be aware that power users may tinker with files on their computer

- Your app should be resilient to still work if storage is not available
  - The user may clear their browser cache etc ...
  - Your app should use reasonable defaults

Web Storage API

- The API works the same for session storage or local storage
  - Based on concept of key and value
  - Key and value are always strings

Only pages from the same origin can access the data

- Remember data is scoped to the page origin: **`protocol + hostname + port`**
  - Store item using: `storage.setItem(key, value)`
  - Retrieve item using: `storage.getItem(key)`

Web Storage API

Only pages from the same origin can access the data

Method	Description
`setItem(key, value) : void`	Set the key with given value If already exists, will update the value
`getItem(key) : string`	Returns the key value If it doesn't exist, return null
`removeItem(key): void`	Remove item key If it doesn't exist, do nothing
`clear(): void`	Empties all keys out of storage

www.luv2code.com/web-storage-api

Development Process - Angular

1. Update CartService to read data from session storage

2. Add new method in CartService: persistCartItems()

3. Modify computeCartTotals() to call new method: persistCartItems()

Step 1: Update CartService to read data from session storage

File: cart.service.ts

```
export class CartService {

  cartItems: CartItem[] = [];
  ...

  storage : Storage = sessionStorage;

  constructor() {

    // read data from storage
    let data = JSON.parse(this.storage.getItem('cartItems'));

    if (data != null) {
      this.cartItems = data;

      // compute totals based on the data that is read from storage
      this.computeCartTotals();
    }

  }
```

Reference to web browser's session storage

Key

You can use any name just stay consistent

`JSON.parse(...)`
Reads JSON string and converts to object

Step 2: Add new method in CartService: persistCartItems()

File: cart.service.ts

```
export class CartService {

  cartItems: CartItem[] = [];
  ...

  storage : Storage = sessionStorage;

  persistCartItems() {
    this.storage.setItem('cartItems', JSON.stringify(this.cartItems));
  }
```

JSON.stringify(...)
Converts object to JSON string

Key

Value

Step 3: Modify computeCartTotals() to call new method: persistCartItems()

File: cart.service.ts

```
export class CartService {

  cartItems: CartItem[] = [];
  ...

  storage : Storage = sessionStorage;

  persistCartItems() {
    this.storage.setItem('cartItems', JSON.stringify(this.cartItems));
  }

  computeCartTotals() {
    ...
    ...

    // persist cart data
    this.persistCartItems();
  }
```