

COMP4021
Internet Computing

Storing Things on Browsers

Gibson Lam

Storing Things on Browsers

- You have learned to use cookies to store things inside browsers
- In this presentation, we will look at a more ‘proper’ way to store things on browsers
- It uses the objects called `localStorage` and `sessionStorage`

Using Cookies

- Cookies are small pieces of data that you store on browsers
- Problems:
 - Small: for each domain, you can store a total of only 4096 bytes of data and only 1024 bytes for each cookie
 - The browser must send the cookies to the server every time (even if they are not used)
 - The server can do everything: it can create your cookies and even delete your cookies



Other Ways to Store Things

- The browsers provide two other ways for you to store things
 - Using local and session storages
 - They are simple method for storing key/value pairs on browsers
 - We will talk about them in this presentation
 - Using IndexedDB
 - This is a sophisticated database API which allows you to create an in-browser database
 - Since we are not a database course, we will not further explore the use of IndexedDB

Local and Session Storage

- To access the two storages, you use:
 - `window.localStorage`, and
 - `window.sessionStorage`
(You can omit `window.` if you want to)
- Like cookies:
 - The local and session storages store key and value pairs
 - They store values for each domain (including the protocol) separately
 - They can only store text (JSON is fine)

Local/Session Storages and Cookies

- Unlike cookies,
 - Local and session storages can store larger data, up to 5MB depending on the browsers' implementation
 - They are accessible by JavaScript only so the server cannot automatically get their values

Local Vs Session Storage

- Local storage stores things permanently, i.e. it does not set expiry date
 - An example use is to store the username of a website sign-in page
- Session storage stores things in a tab session only, i.e. all data are destroyed when the browser/tab is closed
 - An example use is to store the status of the GUI within a session of a web page

Setting and Getting Items

- To put things to the storages, you do this:

```
localStorage.addItem("name", "Paul");
```

- To get things back, you do this:

```
name = localStorage.getItem("name");
```

- Although the above example uses `localStorage`, they work the same way for `sessionStorage`

Deleting Items

- You can delete individual items:

```
localStorage.removeItem("name");
```

- Or delete the entire storage:

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

- Again, `sessionStorage` uses the same functions for deleting things

Example 1: Sign-In Page

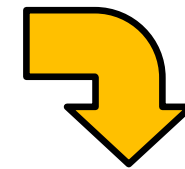
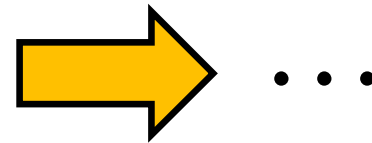
- You can make a 'Remember Me' box in a sign-in page as shown below
- The username can be stored in the local storage every time the user clicks sign-in

Username:

Password:

☒ Remember Me

Click Sign-In



*Visit the
page some
time later*

*Automatically load
the username*

Username:

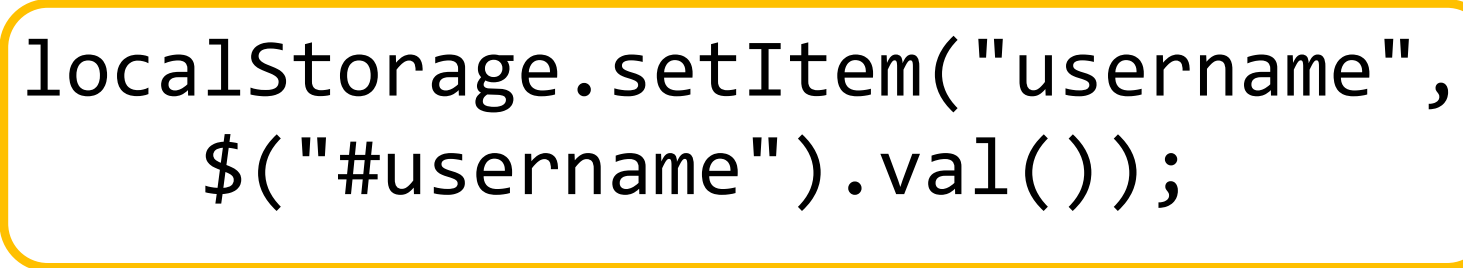
Password:

☐ Remember Me

Remembering the Username

- When the Sign-In button is clicked, this code stores the username:

```
$("#button").on("click", () => {  
    if ($("#save").prop("checked")) {  
        localStorage.setItem("username",  
            $("#username").val());  
    }  
    ...  
});
```



*Save the username
when the checkbox
is selected*

Retrieving the Username

- Then, when the page is loaded, the local storage is checked every time:

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

```
if (username) {  
    $("#username").val(username);  
}
```

*If a previous username exists,
it will be put in the input box*

Example 2: Sign-In Status

- You can use session storage to remember the sign-in status of a user
- If the user has signed in already, the sign-in form is then not shown

Username:

Password:

Click Sign-In



Oh, you can see the secret content!

Reload the page after sign in



No sign-in form is shown




Oh, you can see the secret content!

Remembering the Status

- The session storage stores the status after signing in:

```
$( "button" ).on( "click", () => {  
    sessionStorage.setItem( "signin-status",  
                             "true" );  
    ...  
}
```



Set the session storage data

- This example has a 'dummy' sign-in page so you can sign-in using any information

Checking the Status

- Then, the status is checked by this code to see if the user has signed-in already:

```
const status =  
    sessionStorage.getItem("signin-status");  
if (status == "true") {  
    ...Close the sign-in form and show page content...  
}
```

The user has signed in already

- However, you need to remember this works in the same session only