Recitation – Week 4

PRESENTED BY JEONGMIN LEE / SLIDE MADE BY JAIN PRANUT
Plan for Today

Storage Discussion

HTTP

Ungraded Quiz
Using the Web Storage API

The Web Storage API provides mechanisms by which browsers can securely store key/value pairs, in a much more intuitive fashion than using cookies.

Storage objects are simple key-value stores, similar to objects, but they stay intact through page loads. **The keys and the values are always strings.**

**Question:** What happens if the key is an integer?
Using the Web Storage API

The Web Storage API provides mechanisms by which browsers can securely store key/value pairs, in a much more intuitive fashion than using cookies.

Storage objects are simple key-value stores, similar to objects, but they stay intact through page loads. **The keys and the values are always strings.**

**Question: What happens if the key is an integer?**

**A: Integer key is automatically converted to string, like what objects do.**

You can access these values like an object, or with the Storage.getItem() and Storage.setItem() methods.
Types

`sessionStorage` maintains a separate storage area for each given origin that's available for the **duration of the page session** (as long as the browser is open, including page reloads and restores).

`localStorage` does the same thing, but **persists** even when the browser is closed and reopened.
Browser Support!

Newer versions of most browsers support Web Storage.

**Although!!!** You need to be sure.

**Question:** How would you check?
function storageAvailable(type) {
  try {
    var storage = window[type],
        x = '__storage_test__';
    storage.setItem(x, x);
    storage.removeItem(x);
    return true;
  }
  catch(e) {
    // Print Not supported!
    // Other reasons possible? YES!
    // Get error codes as e.code
    // Get error names as e.name
  }
}

window[type] ? What is this?
function storageAvailable(type) {
    try {
        var storage = window[type],
            x = '__storage_test__';
        storage.setItem(x, x);
        storage.removeItem(x);
        return true;
    }
    catch(e) {
        // Print Not supported!
        // Other reasons possible? YES!
        // Get error codes as e.code
        // Get error names as e.name
    }
}

window[type] ? What is this?
A: The type of storages; sessionStorage and localStorage
Getting values from storage

The `Storage.getItem()` method is used to get a data item from storage.

```javascript
function setStyles() {
  var currentColor = localStorage.getItem('bgcolor');
  var currentFont = localStorage.getItem('font');
  var currentImage = localStorage.getItem('image');

  document.getElementById('bgcolor').value = currentColor;
  document.getElementById('font').value = currentFont;
  document.getElementById('image').value = currentImage;

  htmlElem.style.backgroundColor = '#' + currentColor;
  pElem.style.fontFamily = currentFont;
  imgElem.setAttribute('src', currentImage);
}
```

SOURCE: MOZILLA DOCS
Setting values in storage

`Storage.setItem()` is used both to create new data items, and (if the data item already exists) update existing values. This takes two arguments — the key of the data item to create/modify, and the value to store in it.

```javascript
function populateStorage() {
  localStorage.setItem('bgcolor', document.getElementById('bgcolor').value);
  localStorage.setItem('font', document.getElementById('font').value);
  localStorage.setItem('image', document.getElementById('image').value);

  setStyles();
}
```
Deleting data records

Web Storage also provides a couple of simple methods to remove data.

`Storage.removeItem()` takes a single argument — the key of the data item you want to remove — and removes it from the storage object for that domain.

`Storage.clear()` takes no arguments, and simply empties the entire storage object for that domain.
A little bit of HTTP
Client and Server model

- **Client**, like your computer or mobile phone, makes requests to specific server.
Client and Server model

- **Client**, like your computer or mobile phone, makes requests to specific server.
- **Server** gets the request and processes it including retrieving data and sends it back to client.
Client and Server model

- **Client**, like your computer or mobile phone, makes requests to specific server.
- **Server** gets the request and processes it including retrieving data and sends it back to client.
- Client receives the data and present it to user through proper program. (i.e., Mail program, Web browser)
Client and Server model

- **Client**, like your computer or mobile phone, makes requests to specific server.
- **Server** gets the request and processes it including retrieving data and sends it back to client.
- Client receives the data and present it to user through proper program. (i.e., Mail program, Web browser)
- The communication is running over the **Internet** with protocol.
For specific service user needs, there exists several programs that handles communication with server.

- Surfing the web: Web browser (Chrome, Safari, IE)
- E-mail: Mail Client program (Outlook Express, Apple Mail)
- FTP: FTP Clients (Filezilla, Cyberduck)
- SSH: Putty
- Torrent: U-Torrent
Server side

- For specific service, different programs running on server to serve client’s requests.
  - Web: Flask/Django (python based), Apache, PHP, Ngix, etc..
  - Database: MySQL
  - FTP: Vsftpd

- These programs are always running on server. When clients’ request received, proper program processes user’s request.
HTTP Protocols

- **GET** - Fetch a document. Used only to read data and not change it.
HTTP Protocols

- **GET** - Fetch a document. Used only to read data and not change it.
- **POST** - Execute the document, using the data in body. Often used to create new resources.
HTTP Protocols

- **GET** - Fetch a document. Used only to read data and not change it.
- **POST** - Execute the document, using the data in body. Often used to create new resources.
- **HEAD** - Fetch just the header of the document
HTTP Protocols

- **GET** - Fetch a document. Used only to read data and not change it.
- **POST** - Execute the document, using the data in body. Often used to create new resources.
- **HEAD** - Fetch just the header of the document
- **PUT** - Store a new document on the server. Often used to update resources.
HTTP Protocols

- **GET** - Fetch a document. Used only to read data and not change it.
- **POST** - Execute the document, using the data in body. Often used to create new resources.
- **HEAD** - Fetch just the header of the document
- **PUT** - Store a new document on the server. Often used to update resources.
- **DELETE** - Remove a document from the server
Quiz – Ungraded

Try making use of space on the sheet itself for answers.