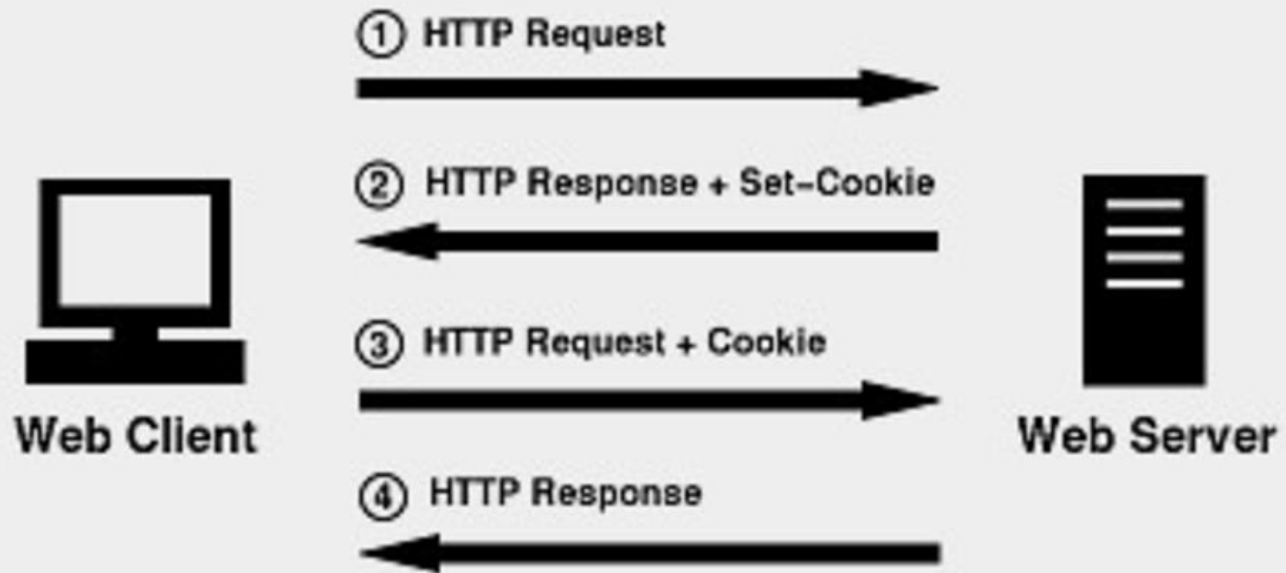


# JavaScript

Web storage – 김근형 강사

# Web Storage

## ○ Cookie



# Web Storage

## ○ Cookie

- 쿠키는 클라이언트의 정보를 여러 요청 또는 여러 세션동안 유지하기 위해 사용되는 기술
- 브라우저에 저장되는 작은 텍스트 파일로 저장되는 데이터이다.
- 웹 서버가 웹 페이지를 브라우저로 보내면 연결이 종료되고 서버는 사용자에게 대한 모든 것을 잊어 버리게 되며 이를 방지하기 위해 필요한 기술이다.
- 쿠키는 " 사용자에게 대한 정보를 기억하는 방법 " 문제를 해결하기 위해 고안되었다.
  - 사용자가 웹 페이지를 방문하면 특정 데이터가 쿠키에 저장 될 수 있다.
  - 다음에 사용자가 페이지를 방문 할 때 쿠키는 자신의 특정데이터를 "기억" 한다.

# Web Storage

- Cookie 접근 방법

- Javascript는 document.cookie 속성을 사용하여 쿠키를 만들고 읽고 삭제가 가능하다.

```
document.cookie
```

- \* 이 실행해서는 vscode의 Live Server가 반드시 실행 되어야 한다.

# Web Storage

- Cookie의 문제점

- 네트워크 부하 유발

- 일반적으로 클라이언트 요청의 크기는 매우 작다. 하지만, 매번 요청 시 클라이언트에 저장된 쿠키들은 서버로 전송된다. 클라이언트 브라우저에 저장된 쿠키가 많다면 이 자료들에 의한 네트워크 부하가 심각하게 발생한다.

- 보안상의 문제

- 쿠키는 네트워크를 타고 서버와 클라이언트를 왕래한다. 네트워크 공간은 공격자가 아주 쉽게 접근할 수 있는 공간이고 쿠키는 전송 과정에서 쉽게 탈취 및 조작될 수 있다. 이에 따라 암호화된 상태에서의 전송, HTTPS에서만 사용 등이 권장되며 점점 사용하기가 복잡해 지고 있다.

- 쿠키의 차단

- 보안상의 문제 때문에 쿠키를 싫어하는 클라이언트들이 많다. 따라서 모든 브라우저는 쿠키를 차단할 옵션을 제공하고 있고 쿠키가 차단되면 쿠키를 이용하는 모든 기능은 사용할 수 없다.

# Web Storage

- Cookie의 문제점

- 제한적인 용량

- 쿠키의 용량은 기껏해야 4kb 정도밖에 안된다. 이 용량은 일반적으로 저장하는 간단한 문자열 정보를 저장하기에는 충분하지만, 문서, 이메일, 등의 정보를 저장하기에는 턱없이 부족한 공간이다.

# Web Storage

- Web Storage
  - 웹 스토리지 API는 HTML5의 API 중 가장 널리 사용되는 기술
  - 웹 스토리지는 쿠키의 대체 기술이기 때문에 쿠키가 갖는 단점들이 모두 극복되었다.
  - 쿠키와 유사한 점은 다음과 같다
    - 이름과 값의 단순한 형태로 데이터를 저장한다.
    - 클라이언트의 공간에 데이터를 저장한다.

# Web Storage

- Web Storage

- 쿠키와의 차이점

- 웹 스토리지의 아이템은 서버로 전송되지 않는다. 물론 명시적으로 자료를 서버에 보낼 수도 있지만, 자료를 만들고 저장하는 과정에서 네트워크의 개입은 필요없다. 따라서 자료를 서버로 보낼 때 발생하는 네트워크 부하 문제가 존재하지 않는다.
    - 웹 스토리지의 아이템이 네트워크로 전송되지 않기 때문에 중간에 패킷을 가로채는 공격은 불가능 해졌지만 여전히 XSS 등의 공격 방식으로 클라이언트의 데이터는 공격자에 의해 탈취될 수 있다. 따라서 민감한 데이터들은 암호화된 상태에서 보관되어야 한다.
    - 보안상의 문제점들이 줄어들었기 때문에 사용을 차단해야 하는 이유가 줄었고, 어플리케이션은 프로그래머의 의도대로 잘 동작할 수 있을 것이다.
    - 웹 스토리지에 저장할 수 있는 데이터의 양도 획기적으로 늘어났다. 쿠키가 제공하는 4kb의 용량은 네트워크를 타고 전송되어야 한다는 부담 때문에 크게 잡을 수 없었다. 하지만, 웹 스토리지의 데이터는 전송되지 않기 때문에 용량에 대한 부담이 사라졌고 스토리지의 구현에 따라 다르지만 5mb 이상의 용량을 제공한다.



# Web Storage

## ○ Web Storage 종류

- 웹 스토리지 api 는 크게 로컬 스토리지(Local Storage)와 세션 스토리지(Session Storage)로 나뉜다.

구분	로컬 스토리지	세션 스토리지
특징	데이터는 사용자가 원하는 기간 동안 보존 데이터를 생성한 어플리케이션은 언제든지 사용 가능	세션 쿠키에 대한 대체로 특정 페이지의 세션 기간에만 데이터 사용 단일 윈도우나 탭에서만 접근 가능하며 해당 윈도우가 닫힐 때까지 유지
저장위치	크롬 : %userprofile%\Local Settings\Application Data\Google\Chrome\User Data\Default\Local Storage	브라우저의 메모리
용량	브라우저마다 다르지만 스펙 권장 최대량은 5MB	브라우저의 메모리가 허용하는 정도
위치	데이터 삭제, 브라우저 언인스톨, OS 재설치 등에서 삭제	브라우저 창이 열려있는 동안만 데이터 보관

# Web Storage

- Web Storage 객체의 속성과 함수

- 모든 스토리지 객체의 상위 객체인 Storage 객체의 속성과 함수는 다음과 같다.

속성 이름	설명
storage.length	읽기 전용의 값으로 Storage 객체에 저장된 데이터의 개수를 함수 형태로 반환한다.

함수 이름	설명
key(n)	파라미터로 정수 형태의 인덱스를 넘겨주면 그 인덱스로 연결된 키를 반환한다. 인덱스는 0부터 시작한다. 하지만 키의 순서는 브라우저마다 다르기 때문에 이 순서를 프로그래밍에 이용해서는 안된다.
getItem(keyName)	keyName을 넘겨주면 그 이름으로 등록된 아이템의 값인 keyValue를 반환한다. 만약 keyName으로 등록된 값이 없을 경우 null을 반환한다.
setItem(keyName, keyValue)	keyName과 keyValue의 쌍으로 아이템을 저장한다. 만일 keyName으로 등록된 아이템이 이미 있다면 기존 값을 keyValue로 업데이트 한다. 주의할 점은 이 함수는 스토리지의 공간이 다 찬 상태에서 호출할 경우 예외를 발생시킨다. 따라서 이 함수를 호출할 때는 try ~ catch 블록을 이용해서 예외 처리를 해야 한다.
removeItem(keyName)	지정된 keyName으로 등록된 아이템을 삭제한다.
clear()	스토리지에 저장된 모든 아이템을 삭제한다. 이 경우 특정 아이템만을 삭제하는 것이 아니라 스토리지 전체를 초기화 하는 것이므로 주의해야 한다.

# Web Storage

## ○ 스토리지 사용 가능 여부 확인

```
<script>
    function checkWebStorageAvailable() {
        try {
            if (localStorage) {
                localStorage.setItem("test", "test-value");
                localStorage.removeItem("test");
            } else {
                return "브라우저에서 localStorage 지원하지 않음";
            }
        } catch (exception) {
            return exception + " 발생으로 localStorage 사용 불가";
        }
    }

    var msg = checkWebStorageAvailable();
    if (msg) {
        document.write(msg);
    } else {
        document.write("localStorage를 사용하는 작업 진행");
    }
}</script>
```

# Web Storage

- local Storage & Session Storage 공통 속성/메서드 사용

Tom
null
foo
null

```
// localStorage를 이용한 데이터 저장
localStorage.setItem('myCat', 'Tom');
// 아이템 읽기
var cat = localStorage.getItem('myCat');
console.log(cat); // Tom
// localStorage 아이템 삭제
localStorage.removeItem('myCat');
var cat = localStorage.getItem('myCat');
console.log(cat); // null
// localStorage 아이템 전체 삭제
localStorage.clear();

// sessionStorage에 데이터 저장
sessionStorage.setItem('myDoc', 'foo');
// 아이템 읽기
var doc = sessionStorage.getItem('myDoc');
console.log(doc); // foo
// sessionStorage 아이템 삭제
sessionStorage.removeItem('myDoc');
var doc = sessionStorage.getItem('myDoc');
console.log(doc); // null
// sessionStorage 아이템 전체 삭제
sessionStorage.clear();
```

# Web Storage

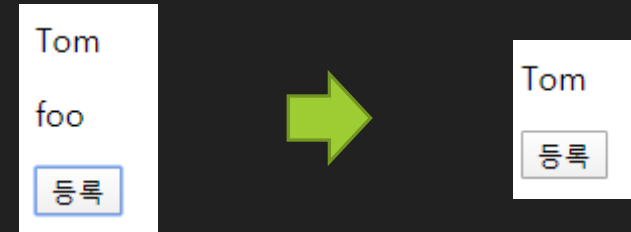
## ○ local Storage & Session Storage 차이점 확인

```
<p id="local"></p>
<p id="session"></p>
<button>등록</button>
<script type="text/javascript">
var btn = document.querySelector("button");

btn.addEventListener("click", () => {
  // localStorage를 이용한 데이터 저장
  localStorage.setItem('myCat', 'Tom');
  // sessionStorage에 데이터 저장
  sessionStorage.setItem('myDoc', 'foo');

  document.getElementById("local").innerHTML = localStorage.getItem('myCat');
  document.getElementById("session").innerHTML = sessionStorage.getItem('myDoc');
});

window.addEventListener('DOMContentLoaded', (event) => {
  if (localStorage.getItem('myCat')) {
    document.getElementById("local").innerHTML = localStorage.getItem('myCat');
  }
  if (sessionStorage.getItem('myDoc')) {
    document.getElementById("session").innerHTML = sessionStorage.getItem('myDoc');
  }
});
</script>
```



브라우저를 종료 후 다시 실행

# Web Storage

## ○ local Storage 활용

```
<fieldset>
  <legend>개인화 설정</legend>
  <label for="name">사용자명</label>
  <input type="text" id="name">
  <label for="color">선호색상</label>
  <input type="color" id="color">
  <label for="fontSize">폰트크기</label>
  <input type="number" id="fontSize">
  <button id="save">저장</button>
  <button id="remove">삭제</button>
</fieldset>
```

개인화 설정

사용자명  선호색상  폰트크기

개인화 설정

사용자명  선호색상  폰트크기

```
var storage = localStorage;
```

```
var body = document.querySelector("body");
var nameField = document.getElementById("name");
var colorField = document.getElementById("color");
var fontSizeField = document.getElementById("fontSize");
```

```
var defaultBackground = "#ffffff";
var defaultFontSize = 15;
```

```
document.getElementById("save").addEventListener("click", function() {
  storage.setItem("name", nameField.value);
  storage.setItem("fcolor", colorField.value);
  storage.setItem("fsize", fontSizeField.value);
  updateUserInfo(nameField.value, colorField.value, fontSizeField.value);
});
```

```
document.getElementById("remove").addEventListener("click", function() {
  storage.clear();
  updateUserInfo("", defaultBackground, defaultFontSize);
});
```

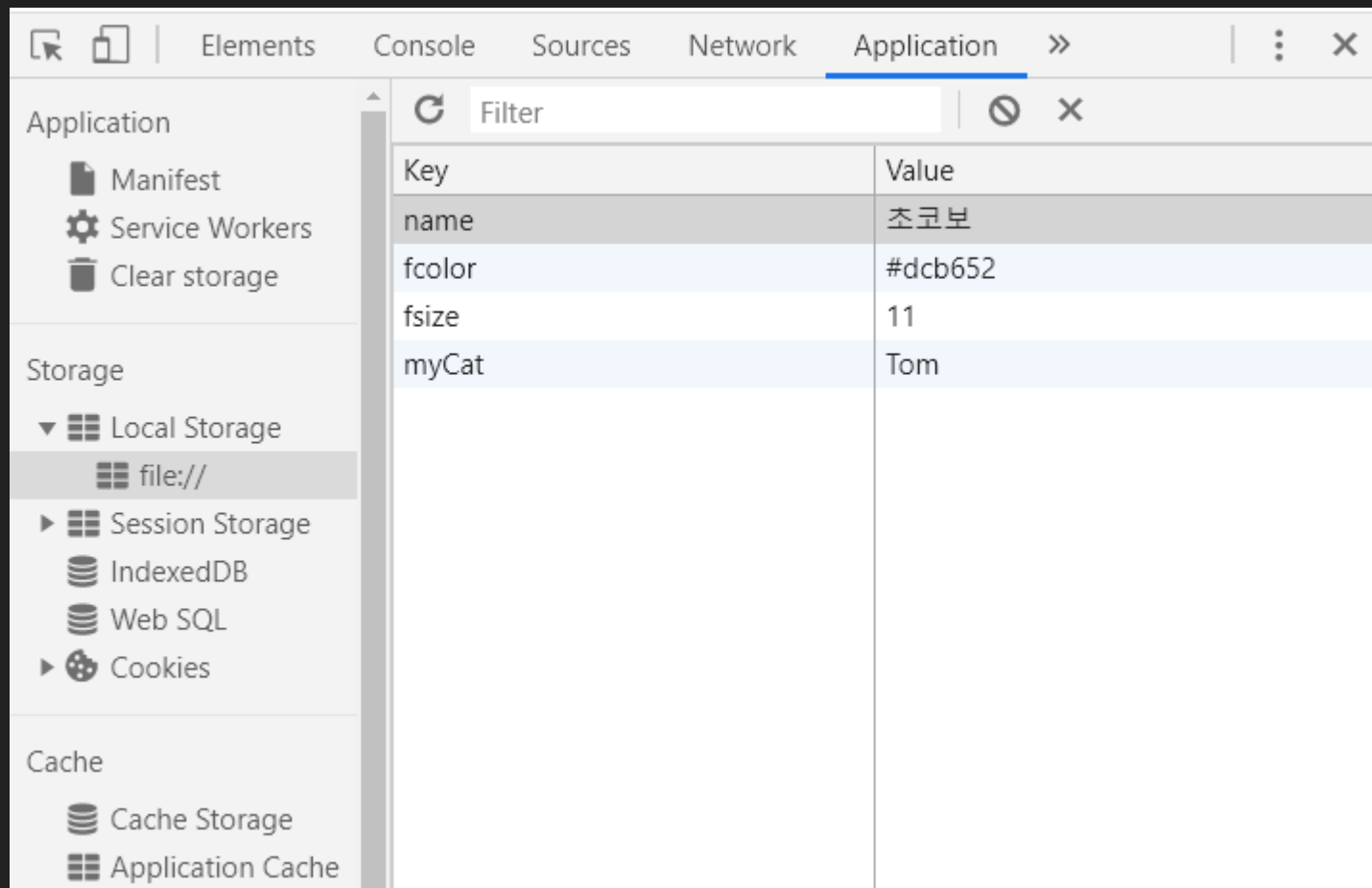
```
function updateUserInfo(id, color, size) {
  nameField.value = id;
  colorField.value = color;
  fontSizeField.value = size;
  body.style.background = color;
  body.style.fontSize = size + "px";
}
```

```
function init() {
  var name = storage.getItem("name");
  if (name) {
    updateUserInfo(name, storage.getItem("fcolor"), storage
      .getItem("fsize"));
  } else {
    updateUserInfo("", defaultBackground, defaultFontSize);
  }
}
```

```
init();
```

# Web Storage

## ○ local Storage 확인



The screenshot shows the 'Application' tab in a web browser's developer tools. The left sidebar is expanded to 'Storage' > 'Local Storage', showing a single entry for 'file://'. The main pane displays a table of stored data.

Key	Value
name	초코보
fcolor	#dcb652
fsize	11
myCat	Tom

# Web Storage

- local Storage JsonData 저장

- Json 데이터를 저장하기 위한 Json 객체의 함수는 다음과 같다

함수이름	설명
stringify(obj)	JSON의 정적 함수로 obj를 문자열 형태로 변환 후 반환한다.
parse(str)	JSON의 정적 함수로 str를 문자열 형태로 변환 후 반환한다.



# Web Storage

## ○ local Storage JsonData 저장 예제

```
<fieldset>
  <legend>로그인</legend>
  <label for="name">사용자명</label>
  <input type="text" id="name">
  <button id="login">로그인</button>
</fieldset>
<fieldset>
  <legend>개인화 설정</legend>
  <label for="color">선택색상</label>
  <input type="color" id="color">
  <label for="fontSize">폰트크기</label>
  <input type="number" id="fontSize">
  <button id="save">설정</button>
  <button id="remove">삭제</button>
</fieldset>
```

The screenshot shows a web application with two main sections. The top section, titled '로그인' (Login), contains a label '사용자명' (Username) followed by a text input field containing '초코보' (ChocoBo) and a '로그인' (Login) button. The bottom section, titled '개인화 설정' (Personalization Settings), contains a '선택색상' (Select Color) label with a color picker showing red, a '폰트크기' (Font Size) label with a numeric input field containing '11', and two buttons labeled '설정' (Settings) and '삭제' (Delete).

```
var storage = localStorage;

var body = document.querySelector("body");
var nameField = document.getElementById("name");
var colorField = document.getElementById("color");
var fontSizeField = document.getElementById("fontSize");

var defaultBackground = "#ffffff";
var defaultFontSize = 15;

document.getElementById("save").addEventListener("click", function() {
  var user = {
    name : nameField.value,
    color : colorField.value,
    size : fontSizeField.value
  };

  storage.setItem(nameField.value, JSON.stringify(user));
  updateUserInfo(nameField.value, colorField.value, fontSizeField.value);
});

document.getElementById("remove").addEventListener("click", function() {
  storage.removeItem(nameField.value);
  updateUserInfo("", defaultBackground, defaultFontSize);
});

document.getElementById("login").addEventListener("click", function() {
  var userStr = storage.getItem(nameField.value);
  if (userStr) {
    var userObj = JSON.parse(userStr);
    updateUserInfo(userObj.name, userObj.color, userObj.size);
  } else {
    updateUserInfo(nameField.value, defaultBackground, defaultFontSize);
  }
});

function updateUserInfo(id, color, size) {
  nameField.value = id;
  colorField.value = color;
  fontSizeField.value = size;
  body.style.background = color;
  body.style.fontSize = size + "px";
}
```

# Web Storage

## ○ StorageEvent

- StorageEvent 는 storage 이벤트가 발생했을 때 전달되는 파라미터이며 아래와 같은 함수를 가진다.

함수이름	설명
key	변경된 아이템의 키로 만약 clear() 함수의 호출에 의한 변경인 경우에는 null 이 반환된다.
newValue	키로 새롭게 저장된 아이템으로 만약 clear() 함수 호출이나 removeItem() 에 의한 변경은 null 을 반환한다.
oldValue	키로 원래 등록되어 있었던 아이템으로 만약 clear() 함수 호출이나 새롭게 아이템이 등록되는 경우에는 null 을 반환한다.
storageArea	영향을 받은 스토리지 객체가 반환된다.
url	변경이 발생한 url이 문자열로 반환된다.

# Web Storage

## ○ StorageEvent 예제

```
window.addEventListener("storage", function(e) {  
    if (e.key == nameField.value) {  
        var userObj = JSON.parse(e.newValue);  
        updateUserInfo(userObj.name, userObj.color, userObj.size);  
    }  
});
```

로그인

사용자명

개인화 설정

선호색상  폰트크기

로그인

사용자명

개인화 설정

선호색상  폰트크기

# Web Storage

## ○ 저장 용량의 한계

```
<script>
  var storage = localStorage;
  try {
    var index = 0;
    while (true) {
      storage.setItem(index++, index);
    }
  } catch (err) {
    alert("에러 발생: 스토리지에 데이터를 저장 할 수 없습니다. " + err.message);
  } finally {
    storage.clear();
  }
</script>
```

이 페이지 내용:

에러 발생: 스토리지에 데이터를 저장 할 수 없습니다. Failed to execute 'setItem' on 'Storage': Setting the value of '455416' exceeded the quota.

확인

# Web Storage

## ○ session 스토리지

```
var storage = sessionStorage;
```

로그인

사용자명

개인화 설정

선호색상  폰트크기

브라우저를 종료 후 다시 실행



로그인

사용자명

개인화 설정

선호색상  폰트크기

Application

- Manifest
- Service Workers
- Clear storage

Storage

- Local Storage
- Session Storage
  - file://
- IndexedDB
- Web SQL
- Cookies

Cache

- Cache Storage
- Application Cache

Background Services

- Background Fetch
- Background Sync

Key	Value
초코보	{"name": "초코보", "color": "#ff0080", "size": "12"}

▼ {name: "초코보", color: "#ff0080", size: "12"}

color: "#ff0080"

name: "초코보"

size: "12"

# Web Storage

## ○ session 스토리지 활용

상품 정보

상품  가격  수량

목록관리

구매희망상품 목록

저장 완료 : {"product":"양파","price":"5000","quantity":"1"}