`localStorage`와 `sessionStorage`

로컬 스토리지와 세션 스토리지에 대한 정리

로컬 스토리지(`localStorage`)와 세션 스토리지(`sessionStorage`)는 모두 **웹 브라우저**에서 데이터를 저장하는 방식으로, **프론트엔드**에서 데이터를 관리하는 도구입니다. 이 두 가지는 웹 서버에서 관리하는 세션과는 전혀 다른 개념입니다.

차이점 요약:

- **로컬 스토리지 (`localStorage`)**: 데이터를 **영구적으로** 저장합니다. 사용자가 브라우저를 닫거나 컴퓨터를 꺼도 데이터는 사라지지 않으며, 사용자가 직접 삭제할 때까지 유지됩니다.
- **세션 스토리지 (`sessionStorage`)**: 데이터를 **일시적으로** 저장합니다. 브라우저의 **탭이나 창을 닫으면** 데이터가 사라집니다.

---

1. 로컬 스토리지 (`localStorage`)

- **데이터 유지 기간**: 사용자가 직접 삭제하지 않는 한, **영구적으로 저장**됩니다.
- **도메인 별로 저장**: 각 도메인마다 고유한 로컬 스토리지를 가지며, 서로 다른 도메인(예: `http://example.com`과 `http://sub.example.com`)은 다른 로컬 스토리지를 사용합니다.
- **프로토콜과 포트**: 도메인 외에도 프로토콜(`http`와 `https`)이나 포트가 다르면 각각 다른 로컬 스토리지를 사용합니다.

로컬 스토리지 예제 코드:

```
// 데이터 저장 (Create)
localStorage.setItem('username', 'Alice');

// 데이터 불러오기 (Read)
const username = localStorage.getItem('username');
```

```
console.log(username); // "Alice"

// 데이터 수정 (Update)
localStorage.setItem('username', 'Bob');
console.log(localStorage.getItem('username')); // "Bob"

// 데이터 삭제 (Delete)
localStorage.removeItem('username');

// 모든 데이터 삭제 (Clear)
localStorage.clear();
```

로컬 스토리지 특징:

- 각 도메인에 대해 고유한 저장소를 가짐.
- 프로토콜(`http`/`https`) 또는 포트 번호가 다르면 별도의 저장소로 간주.
- 수정은 별도의 메서드 없이 덮어쓰기 방식으로 구현.

---

2. 세션 스토리지 (`sessionStorage`)

- **데이터 유지 기간**: **브라우저의 탭이나 창이 열려 있는 동안**만 데이터가 유지됩니다. 창이나 탭을 닫으면 데이터가 사라집니다.
- **도메인 별로 저장**: 로컬 스토리지와 마찬가지로, 도메인마다 고유한 세션 스토리지를 가집니다.
- **브라우저 창/탭 별로 저장**: 각각의 탭이나 창은 서로 다른 세션 스토리지를 사용합니다. 동일한 도메인에서도 탭이 다르면 세션 스토리지를 공유하지 않습니다.

세션 스토리지 예제 코드:

```
// 데이터 저장 (Create)
sessionStorage.setItem('sessionID', '12345');

// 데이터 불러오기 (Read)
const sessionID = sessionStorage.getItem('sessionID');
console.log(sessionID); // "12345"

// 데이터 수정 (Update)
```

```
sessionStorage.setItem('sessionID', '67890');
console.log(sessionStorage.getItem('sessionID')); // "6789
0"

// 데이터 삭제 (Delete)
sessionStorage.removeItem('sessionID');

// 모든 데이터 삭제 (Clear)
sessionStorage.clear();
```

세션 스토리지 특징:

- 도메인, 프로토콜, 포트에 따라 고유한 세션 스토리지를 가짐.
- 각 브라우저 탭이나 창마다 독립된 세션 스토리지를 사용.

---

3. 로컬 스토리지와 세션 스토리지의 차이점 정리

특성	**로컬 스토리지 (`localStorage`)**	**세션 스토리지 (`sessionStorage`)**
데이터 유지 기간	영구적 (사용자가 직접 삭제할 때까지 유지)	탭/창을 닫으면 데이터가 사라짐
도메인 별 저장	도메인마다 고유한 저장 공간 제공	도메인마다 고유한 저장 공간 제공
프로토콜/포트에 따른 구분	프로토콜 및 포트가 다르면 다른 스토리지를 사용	프로토콜 및 포트가 다르면 다른 스토리지를 사용
브라우저 탭/창과의 관계	모든 탭/창에서 동일한 도메인 공유	각 탭/창마다 독립적인 저장소 제공

---

4. 로컬 스토리지와 세션 스토리지의 공통점

- 둘 다 **브라우저에서 데이터 저장**을 위해 사용.
- **키-값 쌍**으로 데이터를 저장하며, 문자열만 저장 가능.
- 데이터를 저장, 읽기, 삭제하는 방법이 유사 (`setItem`, `getItem`, `removeItem`, `clear`).
- 웹서버의 세션과는 전혀 다른 개념으로, **브라우저**(프론트엔드)에서 데이터를 관리.

---

5. CRUD 연산 요약

로컬 스토리지와 세션 스토리지는 모두 CRUD(Create, Read, Update, Delete) 연산을 지원하며, 이를 통해 데이터를 쉽게 다룰 수 있습니다.

Create (생성):

```
localStorage.setItem('key', 'value');
sessionStorage.setItem('key', 'value');
```

Read (읽기):

```
localStorage.getItem('key');
sessionStorage.getItem('key');
```

Update (수정):

```
localStorage.setItem('key', 'newValue'); // 덮어쓰기
sessionStorage.setItem('key', 'newValue');
```

Delete (삭제):

```
localStorage.removeItem('key');
sessionStorage.removeItem('key');

// 모든 데이터 삭제
localStorage.clear();
sessionStorage.clear();
```

동일한 도메인, 동일한 프로토콜

- `http://example.com/page1`
- `http://example.com/page2`

⇒ 같은 도메인으로 칩니다. ⇒ 같은 로컬스토리지외세션스토리지를 공유하겠지요?

```
javascript
코드 복사
// http://example.com/page1에서 실행
localStorage.setItem('sharedData', 'Data shared across page
s');
```

```
// http://example.com/page2에서 실행
const sharedData = localStorage.getItem('sharedData');
console.log(sharedData); // Data shared across pages
```

이로써 로컬 스토리지와 세션 스토리지의 기본 개념 및 활용 방법을 이해할 수 있습니다. **브라우저에서 영구적이거나 일시적인 데이터를 저장**할 때 유용하게 사용할 수 있는 방법입니다.