Storage 활용

1. 개요

이 매뉴얼은 JavaScript와 Bootstrap을 사용하여 웹 애플리케이션에서 로그인 세션을 유지하는 방법을 설명합니다. 사용자 인증을 통해 사용자 경험을 향상시키는 방법을 배웁니다.

2. 목표

- 로그인 세션의 개념 이해
- JavaScript와 Bootstrap 활용 능력 향상
- 사용자 인증 및 세션 유지 기능 구현
- 로컬 스토리지와 세션 스토리지의 이해

3. 필요한 기술

- **HTML**: 웹 페이지 구조를 작성합니다.
- **CSS (Bootstrap)**: 웹 페이지의 디자인을 개선합니다.
- **JavaScript(jQuery)**: 동적인 기능을 구현합니다.

4. 파일 구조

```
project/
├── index.html
├── best.html
├── mypage.html
├── login.html
├── signup.html
└── js /
     └── script.js
```

---

5. 각 파일 설명

index.html

- 목적: 홈페이지, 사용자 환영 메시지 표시.
- 주요 코드: 네비게이션 바와 사용자 이름 표시 로직 포함.

best.html

- 목적: 인기 콘텐츠 페이지.
- 주요 코드: 네비게이션 바와 사용자 이름 표시 로직 포함

mypage.html

- 목적: 로그인한 사용자의 정보 페이지.
- 주요 코드: 사용자 이름 표시 및 로그아웃 기능 포함.

login.html

- 목적: 사용자 로그인 페이지.
- 주요 코드: 사용자 이름 입력 및 로그인 버튼 기능 구현.

signup.html

- 목적: 사용자 회원가입 페이지.

- 주요 코드: 사용자 이름 입력 및 회원가입 버튼 기능 구현.

script.js

- 목적: 로그인, 로그아웃 및 사용자 세션 유지 로직 구현.
- 주요 기능: 로컬 스토리지에 사용자 이름 저장 세션 확인 및 상태에 따라 페이지 리다이렉션

`sessionStorage` vs `localStorage` 비교

`sessionStorage`와 `localStorage`는 모두 클라이언트 측 웹 스토리지 API에 속하는 저장소로, 웹 브라우저 내에서 데이터를 저장할 수 있도록 합니다. 두 저장소는 유사한 기능을 제공하지만, 데이터의 지속성 및 사용 목적에 차이가 있습니다. 아래에서는 `sessionStorage`와 `localStorage`의 주요 차이점을 비교하고, 각각의 사용 상황에 적합한 용도를 설명합니다.

1. 기본 개념

`localStorage`:

- 브라우저에 데이터를 **영구적으로 저장**합니다.
- 브라우저를 닫거나 컴퓨터를 재부팅해도 데이터는 계속 유지됩니다.
- 특정 도메인에 대해서 데이터를 저장하며, **세션 종료 후에도 유지**됩니다.

`sessionStorage`:

- 브라우저의 **세션**에 데이터를 저장합니다.
- **브라우저 탭이 닫히면 데이터가 사라짐**.
- 세션 동안만 데이터를 유지하며, **브라우저의 다른 탭 또는 창에서 접근할 수 없음**.

2. 주요 차이점

특성	`localStorage`	`sessionStorage`
데이터 지속성	브라우저가 종료되거나 컴퓨터를 재부팅해도 데이터가 유지됨.	브라우저 탭을 닫거나 새로 고침하면 데이터가 삭제됨.
유효 범위	동일한 도메인에서 여러 탭과 브라우저 세션 간 데이터 공유	동일한 탭 내에서만 데이터가 유효하며, 다른 탭에서 접근 불가
용량 제한	대체로 5MB 정도의 저장 용량이 주어짐.	대체로 5MB 정도의 저장 용량이 주어짐.
데이터 삭제	명시적으로 `localStorage.removeItem()`이나 `clear()`로 삭제해야 함.	브라우저 탭을 닫으면 자동으로 삭제됨.
데이터 접근 방법	`localStorage.getItem()`, `localStorage.setItem()` 등 사용	`sessionStorage.getItem()`, `sessionStorage.setItem()` 등 사용
보안	클라이언트 측에서 쉽게 접근 가능하므로 민감한 정보를 저장하면 안 됨.	`sessionStorage`도 동일하게 보안이 취약하므로 민감한 정보를 저장하면 안 됨.

3. 주요 특징

`localStorage`

- **영구적 저장**: 브라우저를 종료해도 저장된 데이터는 계속 남아 있으며, 페이지 새로 고침과 상관없이 데이터가 유지됩니다.
- **다중 탭 공유**: 동일 도메인에서 여러 탭을 열어도 데이터가 **공유**됩니다. 예를 들어, 여러 탭에서 `localStorage`에 저장된 사용자 정보를 읽거나 변경할 수 있습니다.
- **크로스 브라우저 지원**: 대부분의 최신 브라우저에서 지원되며, HTML5의 일부로 지원됩니다.

`sessionStorage`

- **세션 기반 저장**: 한 탭 내에서만 유효하고, 브라우저 탭이 닫히면 데이터가 삭제됩니다. 새로 고침이 되더라도 데이터는 유지됩니다.
- **다중 탭 비공유**: 다른 탭에서는 데이터를 공유하지 않기 때문에, 각 탭에서 독립적으로 데이터를 처리할 수 있습니다.
- **세션 데이터**: 사용자가 웹사이트에서 세션 동안만 유지될 정보를 저장할 때 유용합니다.

4. 사용 사례

`localStorage` 사용 사례:

1. **사용자 설정 저장**: 사용자의 테마 설정(예: 다크 모드/라이트 모드)이나, 선호하는 언어 설정을 로컬에 저장하여 사용자가 다시 방문했을 때 같은 설정을 유지할 수 있습니다.
2. **로그인 상태 유지**: 사용자가 로그인 상태를 `localStorage`에 저장하여, 페이지를 새로 고침하거나 브라우저를 닫고 다시 열었을 때 로그인 상태를 유지할 수 있습니다.
3. **장바구니**: 온라인 쇼핑몰에서 장바구니에 담긴 아이템을 `localStorage`에 저장하여, 사용자가 브라우저를 닫았다가 다시 열어도 장바구니가 유지될 수 있도록 합니다.

`sessionStorage` 사용 사례:

1. **한 페이지 내에서만 유효한 데이터 저장**: 페이지 내에서만 사용하는 임시 데이터를 저장하는 데 유용합니다. 예를 들어, 폼 데이터 검증이나 페이지 간 전환 데이터를 세션 동안만 저장할 수 있습니다.
2. **단기적인 정보 저장**: 사용자가 로그인한 후 세션 동안만 유효한 정보를 저장할 때 사용할 수 있습니다. 예를 들어, 세션 기반 인증 정보를 저장하는 데 유용합니다.
3. **탭 간의 독립적인 데이터 저장**: 여러 탭을 열었을 때 각 탭에 대해 독립적인 상태를 유지하고 싶을 때 유용합니다. 예를 들어, 각 탭에서 다른 사용자 데이터를 다룰 때 사용할 수 있습니다.

5. 보안 고려사항

- **민감한 데이터 저장 금지**: `localStorage`와 `sessionStorage`는 클라이언트 측에서 접근할 수 있는 저장소입니다. 따라서 비밀번호, 결제 정보와 같은 민감한 정보는 절대로 저장해서는 안 됩니다. 이 데이터를 저장하는 대신, **서버 세션**을 사용하여 보안을 강화하는 것이 좋습니다.
- **클라이언트 측 노출 위험**: 브라우저의 개발자 도구를 통해 `localStorage`와 `sessionStorage`에 저장된 데이터에 쉽게 접근할 수 있기 때문에, 민감한 정보는 항상 **암호화 후 저장**하는 방법을 고려해야 합니다.

6. API 사용법

`localStorage` API 사용법

```
// 값 저장
localStorage.setItem('username', 'JohnDoe');

// 값 가져오기
let username = localStorage.getItem('username');
console.log(username);  // "JohnDoe"

// 값 삭제
localStorage.removeItem('username');

// 모든 값 삭제
localStorage.clear();
```