

05

05-3. Storage

[데이터 저장이 필요한 이유]

(1) 우리가 만드는 앱은 html이 여러개일 가능성이 높다. 원래 html(윈도우)끼리의 데이터 교환은 불가능하다. 그래서 그 data를 저장할 수 있는 공유 저장소가 필요하다.

(2) 유저가 브라우저를 끄거나 컴퓨터를 껐다. 그러면 RAM 메모리 데이터가 다 날아간다. 그런데 어떤 데이터는 영속적으로 저장되어서 그 데이터를 다음에도 이용할 수 있게 해줘야 한다.

Web APIs

data 저장해서 다음에도 이용할 수 있게 하자.

Storage

- 데이터를 영속적으로 저장하거나 아니면 영속적으로 저장된 데이터를 가져와 애플리케이션에서 이용하는 것은 애플리케이션에서 가장 기초적면서 가장 빈번하게 작성되는 것 중 하나입니다.
- 백엔드 애플리케이션에서는 영속적인 데이터 저장을 위해 대부분 데이터베이스를 이용합니다.
- 오라클, SQL Server, MySQL, MariaDB 부터 클라우드에서 제공하는 데이터베이스까지 많은 데이터베이스들이 있으며 이를 이용해 데이터를 영속적으로 저장해 사용합니다.
- 브라우저에서 데이터를 영속적으로 저장할 수 있는 방법을 제공해야 하고 프런트 자바스크립트에서 Web API를 이용해 프로그램을 작성해 데이터를 영속화 시켜야 합니다.
- 프런트 웹 애플리케이션에서 데이터 영속적 저장을 위해 사용되는 Web API는 크게 스토리지와 IndexedDB가 있습니다.

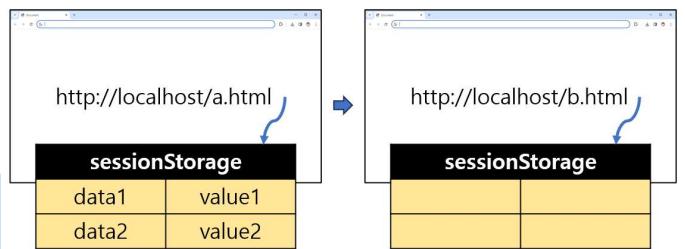
Storage

둘 다 많이 쓰임

- 스토리지(Storage) 저장되는 데이터의 이용 범위에 따라 두개로 구분되어 사용됩니다. 오리진이 다르다 = 앱이 다르다
}} 당연히 데이터 공유가 안 된다.
-
- 로컬 스토리지: 동일 오리진의 여러 세션에서 공유되는 스토리지 같은 오리진(도메인 + 포트) 내의 모든 페이지에서 데이터 공유 가능
- 세션 스토리지: 세션 단위로 이용되는 스토리지 같은 도메인 내의 같은 탭에서만 데이터를 공유 가능

sessionStorage

- 세션이란 웹 애플리케이션이 실행되고 있는 하나의 브라우저 창을 의미합니다.
- 하나의 브라우저 창에서 두개의 웹 애플리케이션이 실행되었다면 세션도 두개입니다.
- 또한 동일한 웹 애플리케이션이 두개의 브라우저 창에서 실행되고 있다면 세션도 두개입니다.
- 결국 세션 단위로 스토리지가 이용된다는 것은 하나의 창에 떠 있는 하나의 애플리케이션을 위한 스토리지라는 의미입니다.
- 그럼으로 세션이 다르면 스토리지 데이터가 공유되지 않으며 브라우저 창이 종료되면 데이터는 사라지게 됩니다.

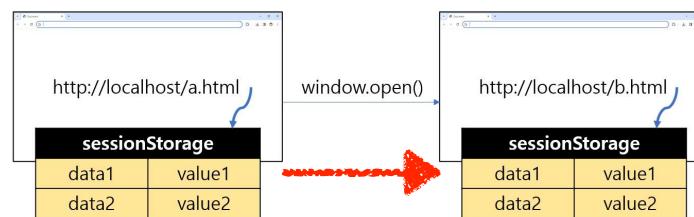


Storage

sessionStorage

예시) 로그인 정보 저장 기능

- 만약 `window.open()` 에 의해 새 창이 열린 경우라고 하면 새로운 창이 기존 창의 세션 스토리지가 **복제**됩니다.
- 하지만 이 경우도 브라우저 창이 두개가 됨으로 각각의 세션 스토리지는 다른 객체입니다.



세션은 다르지만 복제된다.
복제이기 때문에 열린 창에서
데이터를 바꿔도, 원래 창에서
데이터는 변경되지 않는다.

- 세션 스토리지는 브라우저가 종료되거나 컴퓨터가 꺼져도 유지되는 데이터를 저장하기에는 부적합합니다.
- 하나의 브라우저 창에서 실행되는 하나의 애플리케이션이 그 브라우저 창이 실행되는 동안 저장했다가 이용할 데이터를 저장하는 용도로 이용해야 합니다.

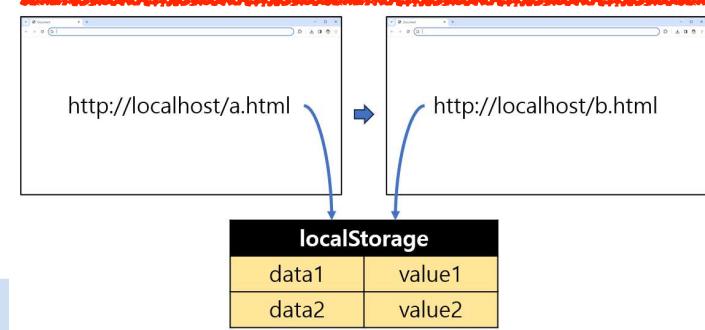
Storage

localStorage 예) 자동 로그인 기능

- 오리진이란 프로토콜+도메인+포트가 결합된 문자열을 의미합니다.



- 로컬 스토리지는 동일 오리진의 모든 세션에서 공유할 수 있는 데이터를 저장하기 위해서 사용됩니다.
- 로컬 스토리지에 저장된 데이터는 브라우저가 종료되어도 사라지지 않습니다.



Storage

스토리지 데이터 저장 및 획득

- Web API에서는 로컬 스토리지와 세션 스토리지를 위해 각각의 객체를 제공합니다.
 - 로컬 스토리지를 이용하고 싶으면 `localStorage` 객체를, 세션 스토리지를 이용하고 싶으면 `sessionStorage`라는 객체를 이용하면 됩니다.
 - 객체 명만 다르고 데이터를 저장하거나 획득하는 함수는 동일합니다.
-
- `setItem(key, value)` : 스토리지 데이터 저장 (**생성 및 업데이트**)
 - `getItem(key)` : 스토리지 데이터 획득
 - `removeItem(key)` : 스토리지 데이터 삭제
 - `clear()` : 스토리지 내의 모든 데이터 삭제

Storage

스토리지 데이터 저장 및 획득

객체명만 차이가 있지 localStorage도 똑같이 할 수 있다.

```
sessionStorage.setItem("data1", "홍길동")
sessionStorage.data2 = "김길동"
sessionStorage.setItem("data3", 10)
sessionStorage.setItem("data4", {
  name: "홍길동",
  age: 10
})
```



object이지만, 저장될 때는
문자열로 변형되어 저장된다.

{"
...
}"

```
let data1 = sessionStorage.getItem("data1")
let data2 = sessionStorage.data2
let data3 = sessionStorage.getItem("data3")
```

The screenshot shows the Chrome DevTools Application tab open. Under the Storage section, Local storage and Session storage are expanded for the origin `http://127.0.0.1:5500`. The Local storage table contains four items: `IsThisFirstTime_Log_From_L...` (Value: `true`), `data1` (Value: `홍길동`), `data2` (Value: `김길동`), and `data3` (Value: `10`). The Session storage table contains one item: `data4` (Value: `[object Object]`). A red box highlights the Local storage section.

Storage

스토리지 데이터 저장 및 획득

- 스토리지에 저장되는 키, 값은 모두 문자열 취급이 됩니다.
- 숫자, 객체등을 지정해도 에러가 발생하지는 않지만 숫자, 객체 타입이 유지되지 못하며 문자열로 저장되게 됩니다.
- 만약 객체를 저장하고 싶다면 객체를 문자열로 변형해서 저장해 주어야 합니다.

```
sessionStorage.setItem("data5", JSON.stringify({  
    name: "홍길동",  
    age: 10  
}))  
let data5 = JSON.parse(sessionStorage.getItem("data5"))
```

Storage

스토리지 데이터 삭제

- 데이터 삭제는 `removeItem()` 함수를 이용하거나 `clear()` 함수를 이용하면 됩니다.
- `removeItem()` 함수를 이용해 매개변수에 지정한 키에 해당되는 데이터만 삭제시킬 수 있으며 `clear()` 를 이용하면 모든 스토리지 데이터가 삭제됩니다.

```
sessionStorage.removeItem("data1")
//or
sessionStorage.clear()
```

Storage

모든 스토리지 데이터 획득하기

- 스토리지에 저장된 모든 데이터를 획득해야 하는 경우 스토리지의 length 프로퍼티를 이용해 반복문을 작성해 주면 됩니다.
- key() 함수의 매개변수에 인덱스 값을 지정해 해당 위치의 데이터 키 값을 얻을 수 있습니다.

```
for(let i =0; i < sessionStorage.length; i++){  
  let key = sessionStorage.key(i)  
  console.log(key, sessionStorage.getItem(key))  
}
```

Storage

객체의 key들을 배열로 반환하는 메서드

모든 스토리지 데이터 획득하기

- 스토리지의 length 를 이용해 반복문을 작성하지 않고 Object.keys() 를 이용해 모든 키를 배열로 획득할 수도 있습니다.

```
let keys = Object.keys(sessionStorage)
keys.forEach(key => {
    console.log(key, sessionStorage.getItem(key))
})
```

Storage

스토리지 이벤트

- 스토리지는 “storage”라는 이름의 이벤트를 제공합니다.
- 이 이벤트는 스토리지 데이터가 변경되는 순간 발생하는 이벤트입니다.
- 데이터 변경이란 새로운 스토리지 데이터가 추가되거나 기존 데이터가 수정되거나 혹은 삭제되는 상황을 의미합니다.

변경시킨 탭에서 이벤트를 걸면 안 된다는 뜻.

- 스토리지 데이터를 변경시킨 브라우저 창에서는 이벤트가 발생하지 않는다는 점입니다.
- 동일 오리진의 다른 브라우저 창에서 이벤트가 발생하여 어떤 창의 HTML 문서에서 스토리지 데이터를 변경시킨 것을 감지하기 위한 수단으로 이용됩니다.

localStorage에만 해당!

Storage

스토리지 이벤트

이 창에서 변경시켰으면
이 창에서는 이벤트가 안 먹음

동일 오리진, 다른 세션



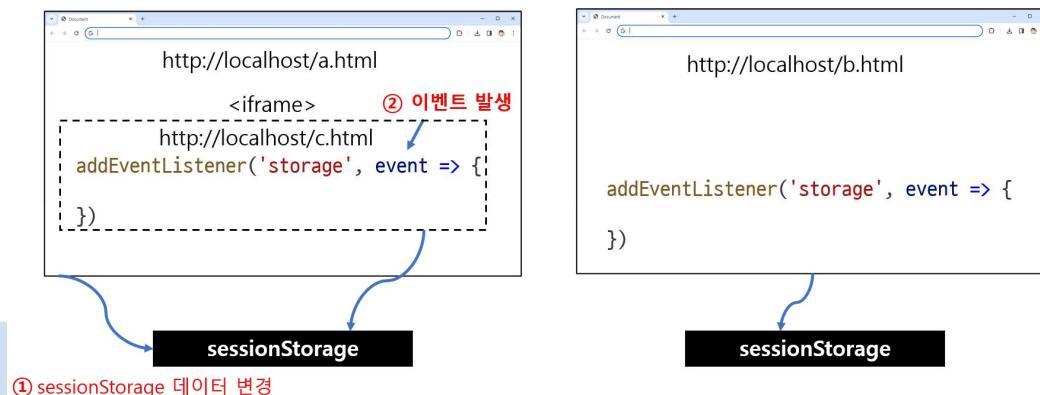
Storage

스토리지 이벤트

- 어느 브라우저 창에서 로컬 스토리지 데이터를 변경했을 때 이 로컬 스토리지를 참조하고 있는 다른 브라우저 창에서 변경 상황을 감지하기 위해서 이벤트가 발생하는 것입니다.
- 세션 스토리의 경우 동일 오리진의 브라우저 창이 여러 개 있다고 하더라도 브라우저 창별로 sessionStorage 객체가 따로 유지됨으로 다른 브라우저 창의 세션 스토리지 데이터 변경 이벤트를 감지할 수 없습니다.
- 세션 스토리지 데이터 변경 이벤트가 발생하는 경우는 하나의 창에 <iframe> 으로 두개의 HTML 이 실행된 경우입니다.

일반적으로 의미가 없는데
iframe을 쓸 때 의미가 있다.

iframe = 페이지의 특정 영역에
팝업처럼 뜨는 창. html 내에 또
다른 html이라고 생각하면 된
다.



Storage

스토리지 이벤트

- 스토리지 이벤트가 발생되게 되면 이벤트 콜백 함수의 매개변수로 StorageEvent 객체가 전달되는데 이 객체의 프로퍼티로 이벤트와 관련된 다양한 정보를 활용할 수 있습니다.
- key : 변경된 데이터 키
- newValue : 변경된 데이터 값
- oldValue : 변경되기 전의 데이터 값
- storageArea : 변경된 스토리지 객체, localStorage 혹은 sessionStorage
- url : 데이터 변경을 한 HTML 페이지의 URL



감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어

William Shakespeare