webOS

Storage

webOS OSE의 Storage

블루투스 프로파일의 종류와 사용하는 방법

webOS

Storage

Prerequisite

- 파일 다운로드
  - 이미지
  - 샘플 앱

- 실습 코드(Github Gist)
  - DB8: bit.ly/db8_gist
  - External Storage: bit.ly/externalStorage_gist
  - Local Storage: bit.ly/localstorage_gist

webOS

Storage

목표

- webOS OSE에서 파일을 생성, 읽기, 삭제하는 방법을 익히고, 다양한 파일 시스템 작업을 수행

webOS 데이터 저장 방법

- 내부 저장 장치 (라즈베리 파이의 경우 주로 MicroSD 카드)
- Database (DB8)
- 외부 저장 장치
- USB storage
- Network storage (Samba, UPnPMediaServer)
- Web Storage

webOS

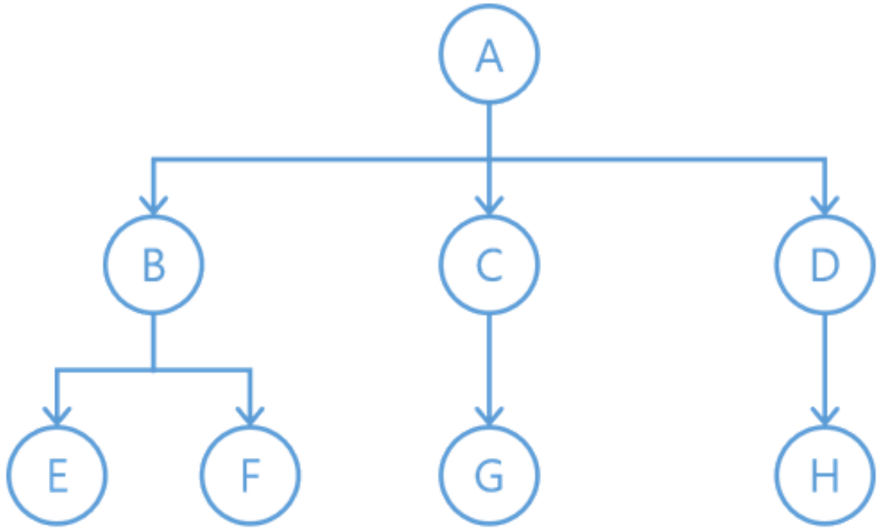
Database

Database

- Database (DB)는 데이터의 모음 혹은 집합을 일컫는 말

DBMS 종류

- Hierarchical (계층형)
  - 하나의 부모와 다수의 자식의 관계를 가짐 (Tree 구조)
  - 1:[illegible]

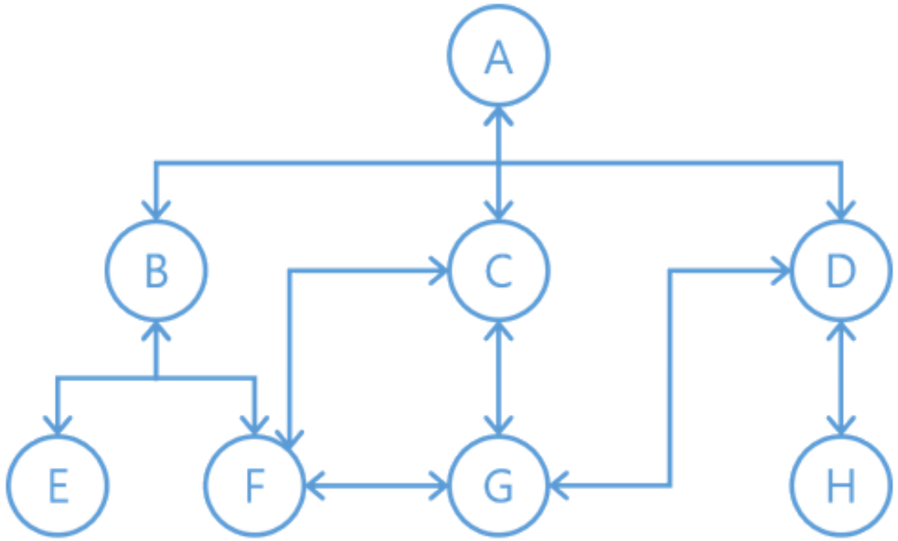


webOS

Database

DBMS 종류

- Network (망형)
  - 1:1, 1:M, M:N 관계를 허용하여 유연한 구조의 표현이 가능
  - 구조 변경 시 참조하는 모든 관계를 고려해야 하므로 수정이 어려움

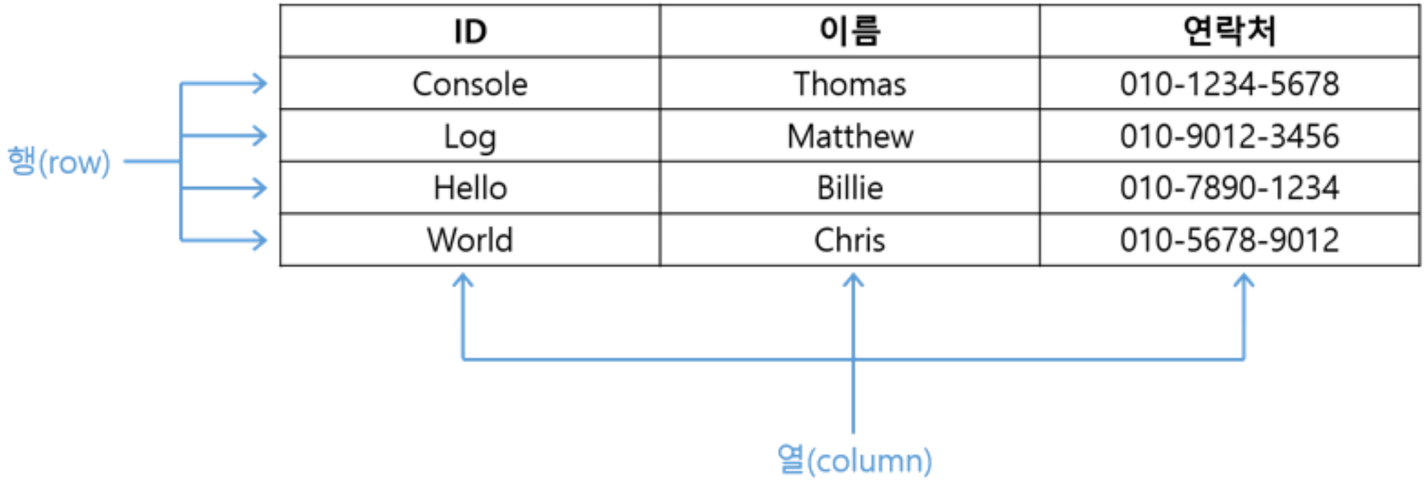


webOS

Database

DBMS 종류

- Relation (관계형)
  - 모든 데이터가 행과 열로 구분
  - 높은 성능으로 널리 사용됨
  - 대량의 데이터를 다루는데 비효율적



webOS

Database

DBMS 종류

- NoSQL
  - 대용량 데이터 처리에 특화된 목적을 위해 비관계형 데이터 저장소에, 비구조적인 데이터를 저장하기 위한 모델의 통칭
  - 특징
    - 관계형 모델을 사용하지 않으며 테이블 간의 조인 기능이 없음
    - 직접 프로그래밍 하는 등의 비SQL 인터페이스를 통한 데이터 접근
    - 확장성, 가용성, 높은 성능

webOS

Database

DB8

- webOS를 위해 만들어진 embedded JSON 기반 DBMS (NoSQL)
- 비정형 데이터베이스를 back-end로 사용할 수 있도록 지원하며 현재 LevelDB를 사용
- 특징
  - NoSQL 지원에 따른 빠른 속도, 기본적인 CRUD (Create, Read, Update, Delete)만을 지원
  - 모든 데이터는 JSON 오브젝트 포맷으로 저장
  - 클라우드 서비스와의 쉬운 동기화
  - 접근 권한 제어(Access Permission Control) 지원
  - 변경에 대한 notification 가능
  - 데이터 스토리지에 대한 백업 제공
  - JSON Schema Standard를 채용하여 데이터 오브젝트의 validation 가능
  - 한 번에 최대 500개까지 조회할 수 있는 페이징 제공

webOS

Database

주요 DB API 설명

API	설명
putkind	• 데이터베이스에 Kind를 등록 • 앱이나 서비스가 객체 속성의 하위 집합이 업데이트될 때만 알림을 받으려면 개정 세트를 사용
putPermissions	• 다른 앱이나 서비스가 앱에 저장된 DB8 데이터에 액세스할 수 있게 함
merge	• 기존 객체의 속성을 업데이트
put	• 특정 Kind의 JSON 데이터 객체를 데이터베이스에 저장
get	• ID로 JSON 데이터 객체를 검색
del	• 데이터베이스에서 JSON 데이터 개체를 삭제
find	• 쿼리 매개변수에 지정된 쿼리와 일치하는 개체 집합을 반환
search	• 전체 텍스트를 검색하는데 사용됨 • '?(물음표)' 연산자 지원
watch	• 쿼리 결과를 변경하는 데이터베이스 업데이트를 감시
load	• 덤프된 JSON 파일에서 데이터베이스를 복원

webOS

Database

DB8 사용 방법

- DB API (com.webos.service.db)를 사용

webOS

Database

Kind

- DB8에 저장할 데이터의 형식을 결정하는 template(스키마, 메타데이터)
- 스키마
  - DB8에 저장될 데이터가 유효한지 검증하기 위한 서식
- 메타데이터
  - ID, owner, 상속 관계, 검색 규칙 등등을 정의한 것

속성	필수 여부	설명
id	필수	Kind 의 고유한 ID 입니다. ID 끝에 ':숫자'의 형태로 버전도 같이 명시합니다.
owner	필수	Kind 에 대한 관리 권한을 갖는 앱의 ID 입니다. 여기에 명시된 앱들만 해당 kind 를 수정할 수 있는 권한을 갖습니다.
schema	옵션	새로 입력될 데이터가 유효한 값들로 이루어져 있는지 검사할 때 사용합니다. JSON 스키마에 대한 자세한 내용은 다음 링크를 참고하시기 바랍니다. • JSON 스키마 공식 사이트 문서 • TCP school 설명 문서
indexes	옵션	DB 검색 시 데이터에서 어떤 값을 기준으로 검색할지 결정합니다. 위 예제에서는 'studentId'를 검색에 사용합니다.

webOS

Database 실습

실습 – luna-send 명령어를 이용한 DB8 이용 (25분)

- 시나리오
  - 개인 정보를 관리하는 실습 진행
  - DB 사용 절차를 터미널에서 luna-send 명령어를 통해 실행
- 실습 목표
  - webOS의 DB8를 이용해 데이터베이스 생성, 데이터 삽입, 조회, 수정 및 삭제 등 기본 작업을 학습합니다.

- 참고 자료
  - https://www.webososse.org/docs/reference/ls2-api/com-webos-service-db/
  - **bit.ly/db8_gist**

webOS

Database 실습

데이터 형태 결정

- 데이터 형태
  - 학생연락처

예시 연락처

```
{
  "studentId" : 2022017823 ,
  "name" : { "firstName" : "Peter" , "lastName" : "Parker" } ,
  "email" : "hello-webos@webososse.org" ,
  "phoneNumber" : "01012345678"
}
```

각 속성의 type을 정리한 연락처

```
{
  "studentId" : number ,
  "name" : { "firstName" : string , "lastName" : string } ,
  "email" : string ,
  "phoneNumber" : string
}
```

webOS

Database 실습

Kind 편집

- 데이터 형태를 기반으로 Kind를 편집

Kind 예시

```
{
  "id" : "com.webos.service.test:1" ,
  "owner" : "com.webos.service.test" ,
  "indexes" : [
    {
      "name" : "idForSearch" ,
      "props" : [
        { "name" : "studentId" }
      ]
    }
  ],
  "schema" : {
    "title" : "contact-validation" ,
    "description" : "This schema checks the validation of JSON data." ,
    "properties" : {
      "studentId" : { "type" : "number" } ,
      "name" : {
        "type" : "object" ,
        "properties" : {
          "firstName" : { "type" : "string" } ,
          "lastName" : { "type" : "string" }
        }
      } ,
      "email" : { "type" : "string" } ,
      "phoneNumber" : { "type" : "string" }
    }
  }
}
```

webOS

Database 실습

Kind 편집

- 새로운 데이터베이스 종류를 정의하고 생성
- **명령어** (https://www.webososse.org/docs/reference/ls2-api/com-webos-service-db/#putkind)
  - Kind가 준비되면 putKind 메서드를 사용하여 생성 (파라미터는 앞 페이지 Kind 이용)

```sh
luna-send -n 1 –f –a com.webos.service.test luna://com.webos.service.db/putKind '{  }'
```

  - Permission 설정
    - 정의한 Kind 대한 권한은 설정

```sh
luna-send -n 1 -f -a com.webos.service.test luna://com.webos.service.db/putPermissions '{
    "permissions":[
      {
        "operations":{
          "read":"allow",
          "create":"allow",
          "update":"allow",
          "delete":"allow"
        },
        "object":"com.webos.service.test:1",
        "type":"db.kind",
        "caller":"com.webos.service.testapp"
      }
    ]
  }'
```

- **결과 확인**: 데이터베이스 종류가 정상적으로 생성되었는지 성공 메시지를 확인합니다.

webOS

Database 실습

데이터 편집 (삽입, put)

- 데이터베이스에 새로운 데이터를 삽입
- **명령어**
  - 데이터 삽입

```sh
luna-send -n 1 -f -a com.webos.service.test luna://com.webos.service.db/put '{
  "objects":[
        {
          "_kind":"com.webos.service.test:1",
          "studentId": 2022017823,
          "name": {"firstName":"Peter", "lastName":"Parker"},
          "email": "hello-webos@webososse.org",
          "phonenumber": "0100000000"
        }
      ]
}'
```

- **결과 확인**: 데이터가 성공적으로 삽입되었는지 확인합니다.

webOS

Database 실습

데이터 편집 (조회, find)

- 데이터베이스에서 연락처 데이터를 조회
- **명령어**
  - 데이터 조회

```sh
luna-send -n 1 -f -a com.webos.service.test luna://com.webos.service.db/find '{
  "query":{
    "from":"com.webos.service.test:1",
  }
}'
```

- **결과 확인**: 데이터베이스에서 삽입한 연락처 데이터가 출력되는지 확인합니다.

webOS

Database 실습

데이터 편집 (수정, merge)

- 데이터베이스에 저장된 연락처 데이터를 수정
- **명령어**
  - 데이터 수정
    - 참고: '<object_id>'는 조회한 데이터의 '_id' 값을 사용

```sh
luna-send -n 1 -f -a com.webos.service.test luna://com.webos.service.db/merge '{
    "objects":[
        {
            "_id":"<object_id>",
            "studentId":2023017823,
            "email":"hello-webos@gmail.com"
        }
    ]
}'
```

- **결과 확인**: 데이터가 성공적으로 수정되었는지 확인합니다.

webOS

Database 실습

데이터 편집 (삭제, del)

- 데이터베이스에서 연락처 데이터를 삭제
- **명령어**
  - 데이터 삭제
    - 참고: '<object_id>'는 조회한 데이터의 '_id' 값을 사용

```sh
luna-send -n 1 -f luna://com.webos.service.db/del '{"ids" :["<object_id>"]}'
```

- **결과 확인**: 데이터가 성공적으로 삭제되었는지 확인합니다.

webOS

외부저장장치

외부 저장 장치

- webOS에서 사용할 수 있는 외부 저장 장치
  - USB storage
  - Network storage (Samba, UPnPMediaServer)

외부 저장 장치 사용 방법

- Storageaccess API (com.webos.service.storageaccess)를 사용

webOS

외부저장장치

주요 외부 저장 장치 API 설명

- 저장 장치의 종류와 관계없이 파일을 제어하는 메서드들은 동일

API	설명
listStorageProviders	• 데이터베이스에 Kind를 등록 • 앱이나 서비스가 객체 속성의 하위 집합이 업데이트될 때만 알림을 받으려면 개정 세트를 사용
device/copy	• 다른 앱이나 서비스가 앱에 저장된 DB8 데이터에 액세스할 수 있게 함
device/eject	• 기존 객체의 속성을 업데이트
device/move	• 특정 Kind의 JSON 데이터 객체를 데이터베이스에 저장
device/remove	• ID로 JSON 데이터 객체를 검색
device/rename	• 데이터베이스에서 JSON 데이터 개체를 삭제
device/list	• 쿼리 매개변수에 지정된 쿼리와 일치하는 개체 집합을 반환

webOS

외부 저장 장치 실습

실습 – luna-send 명령어를 활용한 외부 저장장치 이용

- 시나리오
  - 외부 저장 장치 사용 절차를 터미널에서 luna-send 명령어를 통해 실행
- 실습 목표
  - 저장장치를 검색하고, 특정 위치의 파일을 조작하기 위한 기본 작업을 학습합니다.
- 참고 자료
  - com.webos.service.storageaccess API
  - **bit.ly/externalStorage_gist**

webOS

외부 저장 장치 실습

장치 검색

- 현재 시스템에 연결된 저장 장치를 검색
- **명령어**
  - 장치 검색

```sh
luna-send -n 1 -f luna://com.webos.service.storageaccess/listStorageProviders '{
  "subscribe": true
}'
```

- **결과 확인**: 연결된 저장 장치 목록이 출력되는지 확인합니다

webOS

외부 저장 장치 실습

파일 제어(복사, copy)

- 현재 시스템에 연결된 저장 장치의 파일을
- **명령어**
  - 장치 검색

```sh
luna-send -n 1 -f luna://com.webos.service.storageaccess/device/copy '{
  "srcStorageType":“internal",
  "srcDriveId":"INTERNAL_STORAGE",
  "destStorageType":“internal",
  "destDriveId":"INTERNAL_STORAGE",
  "srcPath":"/tmp/test1.png",
  "destPath":"/tmp/dest"
}'
```

- **결과 확인**: 연결된 저장 장치 목록이 출력되는지 확인합니다

webOS

외부 저장 장치 실습

파일 제어(이동, move)

- 현재 시스템에 연결된 저장 장치를 검색
- **명령어**
  - 장치 검색

```sh
luna-send -n 1 -f luna://com.webos.service.storageaccess/device/move '{
  "srcStorageType": "internal",
  "srcDriveId": "INTERNAL_STORAGE",
  "destStorageType": "internal",
  "destDriveId": "INTERNAL_STORAGE",
  "srcPath": "/tmp/test2.png",
  "destPath": "/tmp/dest",
  "overwrite": true,
  "subscribe": true
}'
```

- **결과 확인**: 연결된 저장 장치 목록이 출력되는지 확인합니다

webOS

외부 저장 장치 실습

파일 제어(삭제, remove)

- 외부 저장 장치에서 파일을 삭제
- **명령어**
  - 파일 삭제

```sh
luna-send -n 1 -f luna://com.webos.service.storageaccess/device/remove '{
  "storageType": "internal",
  "driveId": "INTERNAL_STORAGE",
  "path": "/tmp/dest/test1.png"
}'
```

- **결과 확인**: 파일이 성공적으로 삭제되었는지 확인합니다.

webOS

Web Storage

Web Storage (HTML5 표준)

- 서버가 아닌 클라이언트에 데이터를 저장할 수 있도록 지원하는 HTML5의 기능

- 특징
  - key-value pair로 값을 저장
  - 요청마다 서버로 데이터를 전송하지 않고 local PC에 저장하여 사용하므로 웹사이트 성능에 영향이 없음
- 종류
  - 로컬 스토리지(Local Storage)
  - 세션 스토리지 (Session Storage)

- 지원 메서드
  - setItem(key, value)
  - getItem(key)
  - removeItem(key)
  - clear()

webOS

Web Storage

Local Storage

- 브라우저 내에 영구적으로 데이터를 저장
- 브라우저를 종료하거나 컴퓨터를 재 시작해도 데이터 유지

Session Storage

- 저장된 데이터는 브라우저 세션이 유지되는 동안에만 유효
- 브라우저를 닫거나 탭을 닫으면 해당 세션이 종료되며 세션 스토리지에 저장된 데이터도 삭제

	LocalStorage	SessionStorage
데이터 유지 측면	브라우저 종료시 데이터 보관, 재접속시 사용 가능	브라우저 종료시 데이터 삭제, 재접속시 사용 붇가
데이터 범위 측면	도메인만 같으면 전역적으로 공유 가능	같은 브라우저라도 탭이 다르면 공유 붇가

webOS

Web Storage

Local Storage

- window.localStorage에 위치
- key, value 저장소이기 때문에 key, value를 순서대로 저장

API	설명
setItem	• 키 이름과 값이 전달되면 해당 키를 지정된 Storage 개체에 추가하거나 해당 키 값이 이미 있는 경우 업데이트
getItem	• 키 이름이 전달되면 해당 키의 값을 반환하거나, 해당 Storage 객체에 키가 없으면 null을 반환
removeItem	• 키 이름이 전달되면 지정된 Storage 개체에서 해당 키가 있는 경우 해당 키를 제거
clear	• 득정 Storage 객체에 저장된 모든 키를 삭제

webOS

Web Storage
샘플 앱

샘플 – Local Storage 앱

- 시나리오
  - Local Storage를 이용하여 webOS 웹 앱 구현
- 목표
  - Local Storage를 사용하는 샘플 앱을 통하여 webOS에서 어떻게 웹 스토리지를 구현하지는지 학습합니다.
- 참고 자료
  - Local Storage API 및 사용 예제
  - **bit.ly/localStorage_gist**

webOS

Web Storage 샘플 앱

appinfo.json

```
1.  {
2.    "id": "com.sample.webstorage",
3.    "version": "1.0.0",
4.    "vendor": "SW Documentation Part",
5.    "type": "web",
6.    "main": "index.html",
7.    "title": "Web Storage Sample",
8.    "icon": "icon.png",
9.    "requiredPermissions": [
10.   ]
11. }
```

- Line (2) : 앱의 ID입니다. 추후 앱을 호출할 일이 있을 때 고유한 식별자로써 사용됩니다.
- Line (5) : 앱의 종류입니다. 본 샘플 앱은 웹 앱이므로 web으로 설정하였습니다.
- Line (6) : 앱의 실행 시 시작 지점입니다. 웹 앱 실행 시 시작 지점으로 지정된 파일의 내용을 렌더링하여 화면에 보여줍니다.

webOS

Web Storage 샘플 앱

main.js

- 앱의 실질적인 동작을 구현하는 파일

```
1.  // setItem method 호출을 위한 event listener
2.  const setitem_button = document.getElementById("setitem-btn");
3.  setitem_button.addEventListener('click', function (e) {
4.    console.log("setItem button is clicked. Save key-value pair!");
5.
6.    // 입력한 key-value 값을 받아옴
7.    let setitem_key = document.getElementById("setitem-key-input").value;
8.    let setitem_value = document.getElementById("setitem-value-input").value;
9.
10.   if ((!setitem_key) || (!setitem_value)) { // key나 value 중 하나라도 입력되지 않은 경우
11.     document.getElementById("console-window").innerHTML = "Please enter a key-value pair.";
12.     console.log("setItem() failed!");
13.   }
14.   else if (localStorage.getItem(setitem_key)) { // 기존에 local storage 안에 존재하는 key인 경우
15.     console.log("key: " + setitem_key + ", value: " + setitem_value);
16.     console.log("key: " + setitem_key + " already exists. Overwrite this value.");
17.     document.getElementById("console-window").innerHTML = "key: " + setitem_key + " already
      exists. Overwrite this value.";
18.
```

webOS

Web Storage
샘플 앱

main.js

- 앱의 실질적인 동작을 구현하는 파일

```
19.    // local storage에 key-value 저장 (기존 값 덮어쓰기)
20.    localStorage.setItem(setitem_key, setitem_value);
21.
22.    console.log("setItem() succeeded!");
23.   }
24.   else { // 신규로 입력된 key인 경우
25.    console.log("key: " + setitem_key + ", value: " + setitem_value);
26.    document.getElementById("console-window").innerHTML = "Save (key: " + setitem_key + ",
      value: " + setitem_value + ") in local storage.";
27.
28.    // local storage에 key-value 저장
29.    localStorage.setItem(setitem_key, setitem_value);
30.
31.    console.log("setItem() succeeded!");
32.   }
33. });
```

webOS

Web Storage
샘플 앱

main.js

- 앱의 실질적인 동작을 구현하는 파일

```
19.    // local storage에 key-value 저장 (기존 값 덮어쓰기)
20.    localStorage.setItem(setitem_key, setitem_value);
21.
22.    console.log("setItem() succeeded!");
23.   }
24.   else { // 신규로 입력된 key인 경우
25.    console.log("key: " + setitem_key + ", value: " + setitem_value);
26.    document.getElementById("console-window").innerHTML = "Save (key: " + setitem_key + ",
      value: " + setitem_value + ") in local storage.";
27.
28.    // local storage에 key-value 저장
29.    localStorage.setItem(setitem_key, setitem_value);
30.
31.    console.log("setItem() succeeded!");
32.   }
33. });
```

Question