lowdb\_json

더 간결하고 강력하지 않지만 텍스트 형식으로 저장되고, 번역없이 언어간 공통으로 사용가능한 데이터베이스. json언어를 사용한다.

설치

Npm install -s lowdb

const low = require('lowdb') //lowdb 모듈을 불러온다

const FileSync = require('lowdb/adapters/FileSync')//어떤방식으로 데이터를 저장할 것인가. 여기서는 파일동기형식

const adapter = new FileSync('db.json')//데이터를 db.json이라는 파일에 json 형식으로 저장하겠다.

const db = low(adapter)//lowdb에 파일 동기형식으로 데이터를 저장하겠다고 지정.

이후 실행시켜 보면 db.json은 비어있는 객체임을 확인할 수 있다.

Lowdb를 웹 브라우저에 설치하면 로컬스토리지라는 저장공간에 데이터베이스를 쉽게 사용할 수 있다.

설치

<script src="https://unpkg.com/lodash@4/lodash.min.js"></script><!--low db가 의존하는 파일-->

    <script src="https://unpkg.com/lowdb@0.17/dist/low.min.js"></script><!--low db-->

    <script src="https://unpkg.com/lowdb@0.17/dist/LocalStorage.min.js"></script><!--local storage 관련-->

    <script>

      var adapter = new LocalStorage('db')//로컬스토리지 db에 다가

      var db = low(adapter)//lowdb에 저장하겠다.

    </script>

해당 태그를 헤드안에 삽입

웹서버로 해당 html파일을 열면

개발자도구의 애플리케이션탭에 로컬스토리지에 db가 있는 것을 볼 수 있다.

두 가지 종류 중 어떤 방식이어도 상관 없다. 우선 nodejs기준으로 진행한다.

db.defaults({topic:[],auth:[]}).write();

//기본적으로 준비되어야 할 데이터를 초기화한다. 객체형식으로 json 파일에 추가된다.

실행 시 db.json파일에 객체 형식으로 두 프로퍼티가 배열로 추가된 것을 확인할 수 있다.

Web의 경우 script태그안에서 작성하면 동일하게 수행된다.

CRUD

모든 명령어는 끝에 .write()를 붙여야만 반영이 된다.

추가/create

db.get('author').push({

    id: 1, name: 'egoing', profile: 'developher'

}).write();

db.get(‘프로퍼티명’)가 해당 프로퍼티 라고 생각할 수 있으며 현 db에서 배열을 사용하므로 배열에 추가하는 명령어인 psuh를 사용한다. 내부적으로는 순서가 중요하지 않은 데이터이므로 객체를 사용하였다.

조회/read

Db.get(‘author’).value()는 해당 프로퍼티의 값 전체이다.

자바스크립트의 함수 중 하나인 .find를 사용한다. Sql의 select와 비슷하다.

Db.get(‘author’).find({id:1}).value() 이면 author에 아이디가 1인 값만을 가져온다.

Db.get(‘author’).find({id:1, name:’egoing’}).value() 이면 author에 아이디가 1이고 이름이 egoing인 값만 가져온다.

아마 find는 가장 먼저 나타나는 값을 알려주는 듯 하다. 여러 개를 받고싶다면 filter를 이용한다.

그 외에 여러가지 select처럼 다양한 검색 기능들이 있으므로 검색해서 알아보도록 한다

수정 /update

db.get('topic').find({id:2}).assign({title:'mysql & mariadb',description:'mysql & mariadb is ...'}).write()

find등을 통해 범위를 지정하고 assign을 통해 수정할 프로퍼티와 값을 지정한다. 한번에 여러 값을 수정 할 수도 있다.

삭제/delete

db.get('topic').remove({author:2}).write()

이런식으로 삭제가 가능하며 remove안의 값을 만족하는 여러 값이 한번에 삭제된다.

식별자

Mysql의 autoincres처럼 식별자가 자동으로 변경,추가 되는 기능이 없다. 따라서 외부 모듈로 비슷한 일을 해야 한다.

Short id 설치

Npm install -s shorted

const shortid = require('shortid')

모듈을 불러온다.

Shortid.generate() 는 랜덤한 값을 생성하므로 이를 id 값으로사용할 수 있다.(중복되지 않으므로)

다른 방법으로는 lowdash id 방법이 있다. 따로 다루지 않는다.