11) Vue SPA

1. 목표

- Vue.js 를 통한 Single Page Application 구축.

2. 준비 사항

0. 프로젝트 압축파일의 구성은 다음과 같습니다.

```
11_vue/
    README.pdf    >> 프로젝트 명세
    index.html    >> Boilerplate HTML
    db.json
    css/
        bootstrap.min.css  >> Bootstrap CSS
        main.css           >> Boilerplate CSS
     js/
        axios.js  >> Axios JS
        vue.js    >> Vue JS
        main.js   >> Boilerplate JS
```

    - 이번 프로젝트에는 js/css 파일을 포함한 상용구 코드(boilerplate)들이 작성되어 있습니다.
    - Boilerplate 를 참조하지 않고 비어있는 HTML 파일에서 프로젝트를 진행해도 무관합니다.
        - Boilerplate 코드 위에서 작성할 경우 이미 디자인 되어있는 pure HTML 템플릿에 Vue.js 를 이식하는 작업과 유사합니다.
        - 아무 markup 도 없는 HTML 위에서 작성할 경우, Front End 에 완전한 자율권을 가지고, 디자인 레이아웃부터 작성하는 작업입니다.
1. **(필수) Vue.js** (CDN || static js file || Node Package)
    - Vue.js 2.X
    - (boilerplate) `main.js` 에서 초기화된 Vue 인스턴스의 속성은 다음과 같습니다. 더 자세한 설명은 Vue 공식 API 에서 확인할 수 있습니다.
        1. `el` : Vue 인스턴스를 마운트(장착) 하고자 하는 DOM element 를 선택합니다.
        2. `data` : Vue 인스턴스가 생성될 때 초기화 되는 데이터 입니다.
        3. `methods` : Vue 인스턴스에 추가할 메서드들 입니다.
        4. `computed` Vue 인스턴스에 추가할 연산 후 캐싱(저장)되는 속성입니다. `computed` 의 value 는 함수이며, return 값을 갖습니다. 해당 함수 안에서 참조하는 `data` 의 변경을 감지하며, 이때 새로운 값을 캐싱합니다.
        5. `watch` : 특정 `data` 의 변화가 일어나면 자동으로 실행되는 콜백들을 정의하는 속성입니다.
        6. `created` : Vue 인스턴스가 생성되면 바로 실행되는 함수입니다. 아직 특정 DOM element 에 마운트되지 않았기 때문에 `$el` 로 접근은 불가능하지만, 함수를 통한 `data` 의 초기화를 하기 가

장 적절한 곳입니다. 해당 내용은 Vue instance Life Cycle 이라는 개념과 관련이 있습니다. 해당 개념은 이곳에서 자세히 볼 수 있습니다.

2. **(필수) axios** (CDN || static js file || Node Package)
    - axios 0.18.X
3. **(필수)** 영화 Data 제공 서버
    1. Django REST API server
        - 직접 구축한 Django REST API 서버를 사용합니다. 해당 서버에 `axios` 를 통해 요청을 보내고, JSON 응답을 받아 HTML 에 표시합니다.
    2. **json-server 0.14.X**
        - 프로젝트에 포함된 `db.json` 을 DB 로 사용하는 테스트용 RESTful API 서버를 구축합니다. 해당 서버에 `axios` 를 통해 요청을 보내고, JSON 응답을 받아 HTML 에 표시합니다.
        - node.js 환경을 설치합니다. download node (설치 과정에서 추가 설정이 필요하지 않습니다.)
        - 터미널에서 `$ npm install -g json-server` 를 명령어를 통해 `json-server` 를 설치합니다. `-g` 옵션은 Global 을 의미하기 때문에 명령어 실행에 있어 현재 디렉토리 위치(pwd)는 전혀 상관이 없습니다.
        - 프로젝트에서 제공한 `db.json` 파일이 있는 위치에서 `$ json-server –watch db.json` 명령어를 입력합니다.
        - 브라우저 혹은 Postman 을 통해 `http://localhost:3000` 에 접속합니다. 아래와 같은 형식으로 URL 접근이 가능합니다. 요청을 보내면 실제 `db.json` 파일을 read/write 합니다.

methods	url	description
`GET`	`/movies`	모든 영화정보를 가져온다.
`POST`	`/movies`	영화 레코드를 작성한다. id 는 request body 에 없어도 자동 할당(auto increment)된다.
`GET`	`/movies/<id>`	id 의 영화 정보를 가져온다.
`PUT` / `PATCH`	`/movies/<id>`	id 의 영화 정보를 수정한다.
`DELETE`	`/movies/<id>`	id 의 영화 정보를 삭제한다.
`GET`	`/genres/<id>/movies`	id 의 장르에 해당하는 모든 영화정보를 가져온다.
`GET`	`/movies/<id>/scores`	id 의 영화에 달린 모든 평점정보를 가져온다.
`GET`	`/movies/?q=<keyword>`	영화정보중 keyword 를 포함한 검색결과를 가져온다.
...	...	...

        - 가령, `POST : https://localhost:3000/posts` 요청을 보낸다면, request body 는 다음과 같이 구성합니다.

```
{
  "title": "New Movie",
  "audience": 1,
  "poster_url": "https://image.url",
  "description": "This is new Movie",
  "genreId": 1
}
```

요청을 보내면 실제 `db.json` 에 반영된 것을 확인할 수 있습니다.

- 보다 자세한 내용은 [json-server Github](#) 을 참조하세요.

4. **(선택)** Bootstrap4 (CDN || static css file || Node Package)
    - bootstrap 4.X
5. **(선택)** fontawesome (CDN || Node Package)
    - font-awesome 5.X

3. 요구 사항

1. **DB 스키마**
    - 기존 Django REST API 를 사용하시는 경우, 해당 서버 DB 스키마에 맞게 데이터를 주고 받습니다.
    - `json-server` 로 API 서버를 대체할 경우, 스키마는 아래와 같습니다.
        - Genres

필드명	자료형	설명
id	Integer	Primary Key
name	String	장르 구분

        - Movies

필드명	자료형	설명
id	Integer	Primary Key
tittle	String	영화명
audience	Integer	누적 관객수
poster_url	String	포스터 이미지 URL
description	Text	영화 소개
genreId	Integer	Genre 의 Primary Key

        - Scores

필드명	자료형	설명
id	Integer	Primary Key
content	String	한줄평(평가 내용)
value	Integer	평점
movieId	Integer	Movie의 Primary Key

2. **소스코드 구성**
    1. 브라우저 환경
        - **(필수)** 모든 마크업은 `index.html` 에 저장합니다.
        - **(필수)** 작성하는 모든 js/css 내용은 `main.js` 와 `main.css` 에 작성합니다.
    2. Node 환경
        - **(필수)** 설계한 프로젝트 구조에 맞게 작성합니다. (Vue CLI)
3. **데이터 요청**
    1. **(필수)** 적절한 API 서버에서 RESTful 하게 요청을 보내고 JSON 응답을 받아옵니다.
    2. **(필수)** API 에서 반드시 제공받아야 하는 정보는, 장르정보, 영화정보, 리뷰정보입니다. 추가로 다른 데이터를 제공받을 수 있습니다.
4. **페이지 레이아웃 설계**
    - 프로젝트 결과 SPA는 다음과 같은 UI 들을 제공합니다.
        - **(필수)** 영화 목록을 조회할 수 있는 UI
        - **(필수)** 영화 목록에서 특정 영화의 상세 정보를 조회할 수 있는 UI
        - **(필수)** 특정 영화에 리뷰와 평점을 작성할 수 있는 UI
        - **(필수)** 영화 상세 정보와 함께 리뷰와 평점을 확인할 수 있는 UI
        - **(필수)** 장르 별 영화 목록을 제공하는 버튼/링크 UI
        - (선택) 영화를 검색할 수 있는 검색창 UI
    - Boilerplate는 아래와 같은 UI 요소들로 구성되어 있습니다.
        - Nav Bar
            - Logo
            - Search Bar
        - Sidebar
            - Nav Bar
        - Section
            - Section (Detail)
                - Information
                - Input
            - Section (List)
                - Article
                    - Image
                    - Card
                - Article

        - Image
        - Card
      - Article..
    - Footer

5. **기능 설계**
    - 조회 (Read)
        1. **(필수)** 모든 영화정보를 조회합니다.
        2. **(필수)** 특정 영화의 다음과 같은 항목들을 조회합니다.
            1. 제목
            2. 평균 평점
            3. 누적 관객
            4. 포스터 이미지 url
            5. 설명
            6. 장르
        3. **(필수)** 특정 영화에 대한 리뷰 정보들을 조회합니다.
            1. 내용
            2. 평점
        4. (선택) 아무것도 조회하지 않을 때, 메인 헤더를 제공합니다. (Carousel, Jumbotron, Welcome Image..)
        5. (선택) 키워드로 영화를 검색합니다. 해당 키워드를 포함한 영화들의 목록을 조회합니다.
    - 생성 (Created)
        1. **(필수)** 특정 영화에 리뷰 데이터를 생성합니다.
        2. **(필수)** 영화에 대한 리뷰가 추가되면, 리뷰 데이터와 평균 평점이 갱신됩니다.

4. 제출 양식

- Node.js 환경: 프로젝트 빌드가 된 상태로 `node_modules/` 는 제외하고 압축하여 제출합니다.
- 브라우저 환경: 제공된 압축파일에서 `index.html`, `main.css`, `main.js` 를 제외한 기존 파일 구조를 유지한 채로 제출합니다. `index.html`, `main.css`, `main.js` 은 부분/완전 수정 가능합니다.

5. 결과 예시

결과물에 대해 `README.md` 파일에 활용/고민 하였던 내용을 정리해서 제출합니다. 아래는 브라우저 환경의 결과 제출 예시입니다.

```
11_vue/
    README.md       >> 프로젝트 결과정리
    README.pdf
    index.html      >> 작성한 HTML
    db.json         >> DB
    css/
        bootstrap.min.css
        main.css  >> 작성한 CSS
     js/
        axios.js
        vue.js
        main.js   >> 작성한 JS
```