Music Search App Project

명세서

결과물 예시

Music Album Search App

사전 준비 사항

1. 스켈레톤 코드 준비
2. 스켈레톤 코드 구성 이해하기
    - `index.html`: 기능 구현에 필요한 일부 마크업이 이미 작성되어있습니다.
    - `style.css`: 모든 요소의 스타일링이 이미 작성되어 있습니다. 직접 수정하지 않습니다.
    - `loader.js`: 로더 구현에 사용되는 파일입니다. 직접 수정하지 않습니다.
    - `index.js`: 이번 실습에서 중점적으로 작성하는 부분입니다.
3. 음악 검색 API 신청하기
    - https://www.last.fm/api

필수 기능 구현 🎯

1. 검색창 (DOM & Event)

`index.js`에서 작성합니다.

1. 검색창에 해당하는 input 태그는 `search-box__input` 클래스를 가지고 있습니다. 선택자를 이용하여 해당 요소의 `value` 값을 가져온 뒤 `keyword`라는 이름의 변수에 저장합니다.
2. 검색 버튼에 해당하는 button 태그는 `search-box__button` 클래스를 가지고 있습니다. 선택자를 이용하여 해당 요소를 가져온 뒤 `searchBtn` 라는 이름의 변수에 저장하고, 클릭 이벤트를 부여합니다.
3. 2번에서 클릭 이벤트가 발동했을 때 실행할 콜백 함수인 `fetchAlbums` 를 정의합니다. `fetchAlbums` 함수는 아래의 조건을 만족해야 합니다.
    - page와 limit을 매개변수로 받으며 각각의 인자는 기본값으로 1과 10을 가집니다.
    - `alert()`를 이용하여 브라우저에 `확인!` 이라는 메세지를 출력합니다. 정상적으로 출력되는 것을 확인했으면 해당 코드를 지우고 다음 단계로 넘어갑니다.

2. API 요청 (Ajax)

`index.js`에서 작성합니다.

1. `fetchAlbums` 함수를 완성합니다. `fetchAlbums` 함수는 아래의 조건을 만족해야합니다.
    - 사용자가 입력한 검색어(keyword)를 포함하여 앨범 검색 요청을 보냅니다.
        - (참고 문서) https://www.last.fm/api/show/album.search
    - 앨범 검색을 위한 ajax 요청에는 `axios`를 활용합니다.

- 요청이 성공했을 경우 응답 결과에서 앨범 리스트만 뽑아서 `albums`라는 변수에 저장합니다.
- 요청이 실패했을 경우 `alert()`를 이용하여 `잠시 후 다시 시도해주세요` 라는 메세지를 출력합니다.

3. 검색 결과 (DOM & Event)

`index.js`에서 작성합니다.

1. `albums` 변수에 담긴 값을 순회하면서 `앨범 이미지의 주소`, `아티스트의 이름`, `앨범의 이름` 등 3가지 정보를 추출하여 아래와 동일한 구조로 요소들을 생성 해 줍니다. 이 때, 클래스명도 동일하게 부여합니다.

```
<div class="search-result__card">
  <img src="앨범_이미지_주소">
  <div class="search-result__text">
    <h2>아티스트의 이름</h2>
    <p>앨범의 이름</p>
  </div>
</div>
```

   - 예를 들어, `search-result__card` 라는 클래스를 가진 `div` 요소를 생성하고 그 안에 `앨범 이미지의 주소`를 `src` 속성으로 가진 `img` 태그를 만든다면 아래와 같이 작성 할 수 있습니다.

```
// 이미지 태그 만들기
const cardImg = document.createElement('img')
cardImg.src = album.image[1]['#text']

// div 태그 만들고 클래스 부여하기
const card = document.createElement('div')
card.classList.add('search-result__card')

// div 태그에 이미지 태그 추가하기
card.append(cardImg)
```

2. 이제 위에서 만든 요소를 화면에 보여줄 차례입니다. `search-result` 라는 클래스를 가진 요소를 선택자로 가져와서 위의 요소들을 `appendChild` 메서드를 활용하여 추가해줍니다.

심화 기능 구현

Loader (비동기 및 UX 대한 이해)

> utils/loader.js를 활용합니다.

1. 검색 버튼 클릭 후 API 요청 시 로더를 활성화합니다.
2. API 응답 성공 시 로더를 비활성화합니다.

Inifinite Scrolling (DOM 조작 심화)

> 📌 *Intersection Observer API를 사용합니다.

1. 브라우저의 스크롤을 최하단으로 내릴 때마다 추가 API 요청을 보냅니다.
2. API 응답 결과가 오기까지 화면에 로더를 활성화합니다.
3. API 응답 성공 시 로더를 비활성화합니다.
4. 추가 요청 응답 결과를 화면에 출력합니다.