

클론코딩 리액트 영화웹사이트

리액트 js = 페이스북에서 만든 프레임워크, 그전엔 angular

▼ 배우는건 좋은 선택인 이유

- 누가 이 기술을 쓰는지 - 큰 기업들이 웹사이트를 이거로 씀
- 이들은 안정적이고 얼마 뒤 버릴 언어를 택하지 않음
- 페이스북이 이걸로 작업하고 개선 위해 투자 중임, 금방 폐기 안하겠지
- 큰 커뮤니티 가짐- js와 상당히 가까움 - 대부분이 일반 js와 비슷함
커뮤니티 = lib, 가이드, pkg, 가르쳐줄 사람, 직업, 채용 등 생태계
- react native는 ios, android 어플을 react js 코드로 만들 수 있게 해줌
- 심지어 vr까지 확장 중
- ui를 인터랙티브 하게 만들어줌
- 리액트 쓰기 전 html에 script src에 react와 react dom 임포트 해야
dom은 react ele를 html에 두는 역할
- 리액트는 html 코드를 직접 작성 x - 자바스크립트로만 작성
리액트가 createElement를 html로 번역함 - 이것이 파워

•

```
const span = React.createElement("span", {id:"spna1"}, "hello") //1파라미터는 만드려는 태그명, 2파라미터는 넣고 싶은 프로퍼티, 3파라미터는 내용
const root = document.getElementById("root")
ReactDOM.render(span, root) //사용자에게 보여주는 역할 - html root에 span을 넣겠다
```

2파라미터에 넣을 수 있는 프로퍼티는

id, class, style도 되지만 이벤트리스너 가능

•

```
const button = React.createElement("button", {
  id: "title",
  onClick: () => console.log("i'm clicked")
}, "click me")
```

이는 버튼 getID, 이벤트리스너, 실행시킬 함수를 하나로 함축함

이벤트는 앞에 on을 붙인 on+event는 이벤트 리스너인 것을 가리킴

- JSX
js를 확장한 문법, html과 비슷한 문법 - createElement를 대신한 편한 방법

```

const title() {
  return (
    <h3 id="title" onMouseEnter={() => console.log("mouse enter")}>
      Hello, title
    </h3>
  );
}
=====동일(아래와 함수 써주는 차이)

const title = () => (
  <h3 id="title" onMouseEnter={() => console.log("mouse enter")}>
    Hello, title
  </h3>
)
=====동일 기능(아래는 함수화 안하긴했음)

const h3 = React.createElement("h3", {
  onMouseEnter: () => console.log("i'm dd")
}, "Hello title")

```

- 바벨은 jsx로 적은 코드를 createElement로, 브라우저가 이해할 수 있게 바꿔주는 것

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

```
<script type="text/babel"> ~~~
```

이런식으로 짜면 head에 createEle 형식의 코드가 적혀있음

•

```

//두개의 컴포넌트를 div 안에 넣어주는 것
const Container = ( //여기서 title button은 처음이 대문자여야 여기 있는 것임을 뜻함
  <div>
    <Title />
    <Button />
  </div>
)
=====동일

const Container = React.createElement("div", null, [title, button])

```

이 함수화 된 컴포넌트 재사용 가능

- 바벨, 리액트 임포트 해주는 코드가 선행되어야 코드 실행이 됨
- onclick 이 아닌 onClick!!!
- 바닐라js와 달리, 버튼, 토글 클릭, span이 다 업데이트되는 것이 아닌 리액트에서는 counter 숫자 하나만 업데이트하고 있는 것을 볼 수 있음 = 강점
코드로 보면 container를 업데이트하고 있는 거지만, 새 컨테이너 컴포넌트를 생성하는 것이 아닌 바뀐부분만 새로 생성하게 됨
- 렌더는 사용자에게 바뀐 데이터를 보여주기 위해 쓰는 함수

```

const root = document.getElementById("root")

let counter = 0;
function counterUp() {

```

```

    counter=counter+1;
    console.log(counter);

    ReactDOM.render(<Container />, root);
  }

  const Container = () => ( //이것도 호출했을 때만 동작하는 함수임
    <div>
      <h3>Total clicks: {counter}</h3>
      <button onClick={counterUp}>click me</button>
    </div>
  );

  ReactDOM.render(<Container />, root);

```

리렌더링 할 필요없는 코드 보여주기 전의 코드

```

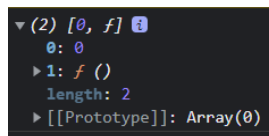
let counter=0;
function countUp() {}

-----동일

const data = React.useState(0);

console.log(data); //1파 = 데이터, 2파 = 데이터 바꿀 수 있는 함수

```




•

```

function App() { //이것도 호출했을 때만 동작하는 함수임
  const [counter, modifier] = React.useState(0); //데이터(0초기값), 데이터 체인저
  const onClick = () => {
    modifier(counter+1); //이제껏 직접 리렌더링 코드 썼지만 이 함수로 counter 업뎃, 아래 html도 업뎃됨
  }
  return (
    <div>
      <h3>Total clicks: {counter}</h3>
      <button onClick={onClick}>click me</button>
    </div>
  );
};

```

→ 데이터가 바뀔때마다 컴포넌트를 리렌더링 후 ui를 리프레쉬 해줌

모든 요소를 새로운 값을 가지고 컴포넌트를 재생성함

state가 바뀌면 리렌더링이 전체에서 일어남

카운터만 오직 바뀐건 안바뀜

• state를 변경하는 방법

1. modifier 함수로 직접 변경 ——— modifier(counter+1)
2. 함수로 전달 ————— modifier((current) => current + 1)

이 cur이 현재 값이라는 걸 보장함 ⇒ 업데이트 되는 값

현 state 바탕으로 다음 state를 계산하려 할 때 (어디 다른데서 업데이트해줄 수도 있으니)

- <label>은 <input> 옆에 써있고 컨트롤? 가능한 태그
- jsx에서 html 형식에서는 for → htmlFor, class → className으로 써야

html 태그와 헷갈리지 않도록

•

```
function App() { //이것도 호출했을 때만 동작하는 함수임
  const [minutes, setMinutes] = React.useState(0)

  const onChange = (event) => {
    setMinutes(event.target.value) //minute 업데이트
  }
  return (
    <div>
      <h1 className="h1"> Super converter </h1>
      <label htmlFor="minutes">Minutes</label> //input 옆에
      <input
        value={minutes} //ui에 보여주는 것
        id="minutes"
        placeholder="Minutes"
        type="number"
        onChange={onChange}
      /> //값 변경을 리스닝해서 위 함수 호출
      //이게 없으면 input에서 뭘 쓰려해도 쓰진게 안보임
      <label htmlFor="hours">Hours</label>
      <input placeholder="Hours" type="number"/>
    </div>
  );
};
```

<input value={minutes}>

유용하다 - input이 다른 곳에서 업데이트 될때도 반영 가능

- 자동 정렬

ctrl + k + f

- 라벨은 input의 이름을 적는 태그입니다. htmlFor에 input의 아이디나 네임을 적어 인풋과 연결합니다.

- prop = 부모 컴포로부터 자식 컴포에 데이터를 보낼 수 있게 해주는 법

- 컴포넌트는 단지 함수 function App() {} 이다, 어떤 jsx를 반환하는

함수형 컴포넌트

그리고 return 이후를 jsx의 내부라고 부른다

- 컴포넌트의 jsx 내부 style을 복붙하는건 귀찮으니 그 속성을 가지는 컴포넌트를 만들어 버리면 어떨까

- 리액트 memo

부모 컴포넌트a가 state가 변하면 전체가 render 되므로 컴포b도 render 되는데 그렇게 할 필요 없으니 리액트에게 알려주는 것

각 컴포의 prop이 변경되는지 여부를 이용

- propTypes은 너가 어떤 타입의 prop을 받고 있고 받아야 하는지를 체크
타입 써주고 그거랑 다른 값 넣어주면 작동은 하지만 콘솔에서 에러메시지

- create react app

node 설치 → cmd에 npx → npx create-react-app 앱이름 → code 앱을

auto re-loading이 됨 (새로고침 안해도)

index.html, js 다 되어있음, 우리 코드는 src에 넣되 이 index에 다 들어갈것임

기본 파일들이 많이 딸려있음

- npm start

이걸 실행하려는 곳으로 cd로 옮겨가서 실행해줘야

PS C:\Users\dlrhd\.vscode\Agile_Frontend\수하> cd ./my-react-app2

PS C:\Users\dlrhd\.vscode\Agile_Frontend\수하\my-react-app2> npm start

- 글로벌 css style - 모든 버튼에 같은 style을 적용할게 아니라면

css 모듈!

```
import styles from "./Button.module.css";

function Button({text}) {
  return (
    <button className={styles.btn}>{text}</button>
  );
}

----- Button.module.css
.btn {
  background-color: tomato;
  color: white;
}
//이 btn 이름을 다른 파일에서도 사용 가능
//css 파일 이름은 상관 없음 .~.module.css 만 지킨다면
```

이런 식으로 하면 같은 btn class 네임이라도 다른 모듈 파일을 사용하는 다른 js 파일이라면 사용 가능 - class 이름을 랜덤으로 지정해줌, 각기 다른 클래스 네임을 기억해도 되지 않는다는 장점

- 특정 코드들이 첫번째 comp render에서만 실행되게 할지
만약 api 데이터 받아오는 거면 state 변할때마다 받아올수 없지

- useEffect(특정시에만 돌리고싶은 함수, [keyword])

state가 변해도, 어떤일이 있어도 한번만 ([] 비었을때)

시작할때 + keyword 변수가 변할때만 코드를 실행할 것이라고 알려줌

```
useEffect(()=>{ //언제 코드가 실행될지 정함 + 특정 조건도 걸었음 + keyword 변할 때라는 조건
  if(keyword!==""&&keyword.length>4) //이 조건은 처음 실행할 땐 무조건 실행 안되도록 하는 것도 있음
    console.log("keyword move");
}, [keyword])
```

[keyword, counter] 면 둘 중 any 면 동작

□ 면 그냥 시작할 때 한번만 실행

-

```
import Button from "../Button"; //button.js에서 버튼 컴포 가져옴
import styles from "../App.module.css";
import {useState, useEffect} from "react";

function Hello() {
  return <h2>Hello</h2>;
}

function App() {
  const [counter, setCounter] = useState(0);
  const [keyword, setKeyword] = useState("");
  const [value, setValue] = useState(true);
  const onClick = () => setCounter((prev) => prev+1)
  const onChange = (event) => setKeyword(event.target.value)
  const onClick1 = () => setValue((value) => !value)

  console.log("all the time")

  useEffect(()=>{ //언제 코드가 실행될지 정함 + 특정 조건도 걸었음 + keyword 변할 때라는 조건
    if(keyword!=="&&keyword.length>4)
      console.log("keyword move");
  }, [keyword])

  useEffect(()=>{
    console.log("counter move");
  }, [counter])

  return (
    <div>
      <input onChange={onChange} value={keyword} type="text" placeholder="search here" />
      <h1> {counter} </h1>
      <button onClick={onClick}>Click me</button>
      {value ? <Hello /> : null}
      <button onClick={onClick1}>{value ? "hide" : "show"}</button>
    </div>
  );
}

export default App;
```

이걸 보니 return (div 태그 하나에 다 있어야 하나봄)

- cleanup function

컴포넌트가 destroy 될 때 뭔가 할 수 있도록 해주는

```
import {useState, useEffect} from "react";

function Hello() {
  function byeFn() {
    console.log("bye")
  }
  function hiFn() {
    console.log("hi")
    return byeFn;
  }
  useEffect(hiFn, [])
  return <h2>Hello</h2>;
}

function App() {
  const [value, setValue] = useState(true);
  const onClick1 = () => setValue((value) => !value)

  return (
    <div>
      {value ? <Hello /> : null}
      <button onClick={onClick1}>{value ? "hide" : "show"}</button>
    </div>
  );
}
```

```
}
export default App;
```

- 리액트에선 js 처럼 todos.push() 나 todo="" 처럼 state를 직접 바꾸지않고 setTodo("") 등 함수로 변화시킴
- 기존 어레이에 엘리먼트 추가하기

```
> const food = [1,2,3,4]
< undefined
> food
< ▶ (4) [1, 2, 3, 4]
> [6, ...food]
< (5) [6, 1, 2, 3, 4]
```

- jsx (return 에서의 html) 에서의 자바스크립트는 무조건 중괄호를 넣어줘
- [1,2,3].map(함수)
3번 함수 실행되고 그 return 값은 새로운 array에 들어가게 됨

```
> ['there', 'are', 'you', 'are', 'how', 'hello!'].map(() => ":)")
< ▶ (6) [':)', ':)', ':)', ':)', ':)', ':)']
> ['there', 'are', 'you', 'are', 'how', 'hello!'].map((item) =>
동영상 더보기
  item.toUpperCase())
< ▶ (6) ['THERE', 'ARE', 'YOU', 'ARE', 'HOW', 'HELLO!']
```

원본 array를 가져와 변형

- <hr> 태그는 콘텐츠 내용에서 주제가 바뀔 때 사용할 수 있는 수평 가로선을 정의
- 같은 compo list를 render 하려면 key prop을 넣어줘야한다?
map(item, index) 에서 각 li 에 key={index} 넣어주면 됨
-

`fetch()` 함수는 첫번째 인자로 URL, 두번째 인자로 옵션 객체를 받고, Promise 타입의 객체를 반환합니다. 반환된 객체는, API 호출이 성공했을 경우에는 응답(response) 객체를 resolve하고, 실패했을 경우에는 예외(error) 객체를 reject합니다.

```
fetch(url, options)
  .then((response) => console.log("response:", response))
  .catch((error) => console.log("error:", error));
```

- coin tracker

```
import { useEffect, useState } from "react"

function App () {
  const [loading, setLoading] = useState(true)
  const [coins, setCoins] = useState([])
```

```

useEffect(() => { //api 한번만 불러오면됨
  fetch("https://api.coinpaprika.com/v1/tickers") //원격 api 호출하는 방법
  .then((response) => response.json()) //리턴 받은 것 json 변환
  .then((json) => { //그 json을 coin으로 변환
    setCoins(json)
    setLoading(false) //api 받기를 끝내면 loading은 false가 됨
  })
}, [])

return <div>
  <h1> The coins! {loading ? "" : `${coins.length}`}</h1> {/*로딩이면 x 아니면 코인 개수 보여줌 */}

  {loading ? (
    <strong> Loading...</strong>
  ) : ( //로딩 끝나면 로딩메시지 없애고
    <select>
      {coins.map((coin) => ( //coins 어레이 각 리스트 하나씩 뽑아옴
        <option>
          {coin.name} ({coin.symbol}): ${coin.quotes.USD.price} USD
        </option>
      ))}
    </select>
  )}
</div>
}

export default App;

```

- usd → coin

```

import { useEffect, useState } from "react"

function App () {
  const [loading, setLoading] = useState(true)
  const [coins, setCoins] = useState([])
  const [usd, setUsd] = useState()
  const onChange = (event) => setUsd(event.target.value); //인풋
  const onSubmit = (event) => { //인풋 제출
    event.preventDefault();
    if(usd==="")
      {return;}
    setLoading(false)
    setUsd("");
  }

  useEffect(() => { //api 한번만 불러오면됨
    fetch("https://api.coinpaprika.com/v1/tickers") //원격 api 호출하는 방법
    .then((response) => response.json()) //리턴 받은 것 json 변환
    .then((json) => {
      setCoins(json); //그 json을 coin으로 변환
    });
  }, [usd])

  return (
    <div>
      <h1> The coins!</h1>

      <form onSubmit={onSubmit}> {/*변환해보려는 usd값 입력받음*/}
        <input onChange={onChange} type="number" value={usd} placeholder="type usd"></input>
      </form>

      {loading ? (
        <strong> Loading...</strong>
      ) : ( //변환된 값 출력
        <select>
          {coins.map((coin, index) => ( //coins 어레이 각 리스트 하나씩 뽑아옴
            <option key={index}>
              {coin.name} ({coin.symbol}): ${usd/(coin.quotes.USD.price)} USD
            </option>
          ))}
        </select>
      )}
    </div>
  )
}

```



```
export default App;
```

- 콘솔 2번 찍히는 이유

App.js를 호출하는 index.js에서 strict mode를 쓰기 때문에 정상적인 상황임

- 영화1

```
import { useEffect, useState } from "react"

function App () {
  const [loading, setLoading] = useState(true)
  const [movie, setMovie] = useState([])
  useEffect(() => {
    fetch(`https://yts.mx/api/v2/list_movies.json?minimum_rating=8.8&sort_by=year`)
      .then((response) => response.json()) //json이 반환받는 object
      .then((json) => {
        setMovie(json.data.movies); //log 찍어보면 ob 안 data 안 movies에 영화 어레이 있음
        setLoading(false); //영화 로딩 끝났으니 이 때 한번 false로 만들기
      })
  }, [])

  return (
    <div>
      {loading ? <h1>Wait please</h1> :
      <div>
        {movie.map((movie) => ( //html은 {}로 감싸지 않는다 / 어레이를 각각 출력하려면 map!
          <div key={movie.id}>
            <img src={movie.medium_cover_image}/>
            <h1>{movie.title}</h1>
            <p>{movie.summary}</p>

            <ul>
              {movie.genres.map(genre => (
                <li key={genre}>{genre}</li>
              ))}
            </ul>

          </div> //map을 통해 변환되는 부분에 key를 주어야 함

        ))}
      </div>
    </div>
  )
}
```

```
export default App;
```

원본.map(아무변수 => (해줄 html { ~ }))
이 html 태그에서 key 줌

여기서 movie.map은
movie는 [어레이] 형태임, 그리고 하나의 movie는 객체 형식으로 됨
어레이를 계속 돌면서 하나의 영화마다 아래 html로 객체 형식의 그 영화 정보를 선별해서 출력

```
useEffect(() => {
  fetch(`https://yts.mx/api/v2/list_movies.json?minimum_rating=8.8&sort_by=year`)
    .then((response) => response.json())
    .then((json) => {
      setMovie(json.data.movies);
      setLoading(false);
    })
}, [])
```

-----동일, 아래것이 promise 변형 형태

```
const getMovie = async () => {
  const response = await fetch (링크)
  const json = await response.json()
  setMovie(json.data.movies);
  setLoading(false);
}

useEffect(() => {
  getMovie();
})
```

- Movie.js 로 뻥뻥 - prop으로 App에서 작동

```
import PropTypes from "prop-types"

function Movie({medium_cover_image, title, summary, genres}) {
  return(
    <div>
      <img src={medium_cover_image} alt="title"/>
      <h1>{title}</h1>
      <p>{summary}</p>
      <ul>
        {genres.map((genre) => (
          <li key={genre}>{genre}</li>
        ))}
      </ul>
    </div>
  )
}

//각 prop이 어떤 타입이어야 하는지 정함
Movie.propTypes = {
  medium_cover_image: PropTypes.string.isRequired,
  title: PropTypes.string.isRequired,
  summary: PropTypes.string.isRequired,
  genres: PropTypes.arrayOf(PropTypes.string).isRequired,
}

//우선 각 무비에 해당하는 html을 app.js에서 가져와서 만들, 코드 정리라고 함
//key 필요없으니 지우고 movie.가 정의되지 않았으니 지우고 title 등 정의 안되었기 때문에 app에서 파라미터로(prop으로) 받아와야함
//Movie compo는 이 property를 다 부모 compo에서 받아옴
export default Movie;
```

- route = 스크린, 페이지
하나의 라우트는 home, 홈페이지 등
페이지마다 이동을 위해서 하는 행위임 = fragment라 생각
- App을 비우고 route 폴더와 component 폴더 만들
router는 url을 보고있는 compo
compo 폴더엔 Movie.js
routes 폴더엔 Detail.js 과 Home.js(기존 하던 App을 옮긴것)
App은 따로 있고 여기서 여러 페이지를 띄울거같음
- router
browserRouter는 localhost:3000 같이 생김 - 대부분 이용
hash는 #이 붙음

- switch

router-dom으로 오면서 routes로 대체됨

한번에 하나만 렌더되도록 하는 것임

App.js

```
import {
  BrowserRouter as Router,
  Switch,
  Route,
} from "react-router-dom" //이들은 페이지의 url에 따라 다른 페이지를 보여줌
import Home from "../routes/Home"
import Detail from "../routes/Detail";

function App () {
  return <Router>
    <Switch>
      <Route path="/movie">
        <Detail />
      </Route>
      <Route path="/"> {/*홈으로 간다는 뜻 - home route를 렌더링 - home이 가장 아래 있어야하는듯*/}
        <Home />
      </Route>
    </Switch>
  </Router>
}

export default App;
```

detail.js

```
function Detail () {
  return <h1> Detail~ </h1>
}

export default Detail
```

home.js

```
import { useEffect, useState } from "react"
import Movie from "../components/Movie"

function Home () {
  const [loading, setLoading] = useState(true)
  const [movie, setMovie] = useState([])
  useEffect(() => {
    fetch(`https://yts.mx/api/v2/list_movies.json?minimum_rating=8.8&sort_by=year`)
      .then((response) => response.json()) //json이 반환받는 object
      .then((json) => {
        setMovie(json.data.movies); //log 찍어보면 ob 안 data 안 movies에 영화 어레이 있음
        setLoading(false); //영화 로딩 끝났으니 이 때 한번 false로 만들기
      })
  }, [])

  return (
    <div>
      {loading ? <h1>Wait please</h1> :
      <div>
        {movie.map((movie) => (
          <Movie
            key={movie.id} //추가 중요

```

```

        medium_cover_image={movie.medium_cover_image}
        title={movie.title}
        summary={movie.summary}
        genres={movie.genres}
      />

    )}
  </div>
}

</div>
)
}

export default Home;

```

movie.js

```

import PropTypes from "prop-types"

function Movie({medium_cover_image, title, summary, genres}) {
  return(
    <div>
      <img src={medium_cover_image} alt="title"/>
      <h1>{title}</h1>
      <p>{summary}</p>
      <ul>
        {genres.map((genre) => (
          <li key={genre}>{genre}</li>
        ))}
      </ul>

    </div>

  )
}

//각 prop이 어떤 타입이어야 하는지 정함
Movie.propTypes = {
  medium_cover_image: PropTypes.string.isRequired,
  title: PropTypes.string.isRequired,
  summary: PropTypes.string.isRequired,
  genres: PropTypes.arrayOf(PropTypes.string).isRequired,
}

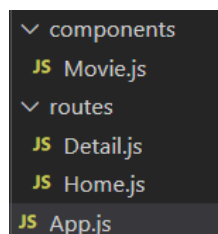
//우선 각 무비에 해당하는 html을 app.js에서 가져와서 만들, 코드 정리라고 함
//key 필요없으니 지우고 movie.가 정의되지 않았으니 지우고 title 등 정의 안되었기 때문에 app에서 파라미터로(prop으로) 받아와야함
//Movie compo는 이 property를 다 부모 compo에서 받아옴
export default Movie;

```

원래 모든걸 App에 때려박았지만

app - detail, home으로 라우트를 나눴고

home에서 표시해줘야 할 각각의 영화 정보들을 movie에서 표시해줌



여기서 영화제목 태그에 a href 해서 링크 /movie 하면 되지않나?

→ 전체 페이지가 실행되는 비효율이 생겨버림

이걸 위해 존재하는 link 라는 컴포넌트 - 브라우저 새로고침 없이도 다른 페이지로 이동시켜줌

```
<a href="/movie">{title}</a>
<Link to="/movie">{title}</Link>
```

- path="/movie/:id"
/movie/(어느숫자)

- 변경되는 부분 url 감지
react-router에서 제공하는 함수 useParams

- 왜 home route가 맨 routes중에서 맨 아래 작성되어야 작동이 되는지 궁금하네요

route는 path의 앞부터 일치하는 부분이 있으면 그걸로 작동합니다

예를들어 "/movie" 라면 맨 앞쪽의 "/"가 있기 때문에 "/movie"로 못 넘어가요. 하지만 만약 순서에 상관없이 route를 작성하고싶다면 `Route exact path="/" 이렇게 exact를 써서 완전일치를 시켜주시면 됩니다.

- invalid hook call
route dom 버전 안맞든지, 이상한데다 useParams 썼던지, react 중복됐던지
이유가 있었는데 router-dom을 cra 안에 설치 안했던 게 문제였음

- 최종 코드
detail.js

```
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom"

function Detail () {
  const [detail, setDetail] = useState([]);
  const [loading, setLoading] = useState(true);

  const {id} = useParams(); //현재 url의 변경된 부분만 catch, 아래 링크에서 덧붙여 쓰임

  const getMovie = async() => { //링크에서 json 가져오는 부분
    const json = await (
      await fetch(`https://yts.mx/api/v2/movie_details.json?movie_id=${id}`)
    ).json();
    console.log(json.data.movie);
    setDetail(json.data.movie); // 위 링크에서 영화의 디테일 정보를 어레이에 저장
    setLoading(false); //링크 받아왔으니 로딩창 끄
  };

  useEffect(() => {
    getMovie();
  }, []) //링크 한번만 불러와

  return ( //리턴 내에선 무조건 태그로 감싸는 습관!!
    <div>
      {loading ? <h1> loading... </h1> :
        <div>
          <img src = {detail.background_image_original} />
          <p>{detail.title}</p>
          <ul>
            {detail.genres.map((g) => (
              <li key={g}>
                {g}
              </li>
            ))}
          </ul>
        </div>
      }
    </div>
  )
}
```

```

    </div>

    )
}

export default Detail

```

movie

```

import PropTypes from "prop-types"
import {Link} from "react-router-dom"

function Movie({id, medium_cover_image, title, summary, genres}) { //prop은 obj
  return(
    <div>
      <img src={medium_cover_image} alt="title"/>
      <h1>
        { /*<a href="/movie">{title}</a>*/ }
        <Link to={`/movie/${id}`}>{title}</Link>
      </h1>
      <p>{summary}</p>
      <ul>
        {genres.map((genre) => (
          <li key={genre}>{genre}</li>
        ))}
      </ul>

    </div>

  )
}

//각 prop이 어떤 타입이어야 하는지 정함
Movie.propTypes = {
  id:PropTypes.number.isRequired,
  medium_cover_image: PropTypes.string.isRequired,
  title: PropTypes.string.isRequired,
  summary: PropTypes.string.isRequired,
  genres: PropTypes.arrayOf(PropTypes.string).isRequired,
}

//우선 각 무비에 해당하는 html을 app.js에서 가져와서 만들, 코드 정리라고 함
//key 필요없으니 지우고 movie.가 정의되지 않았으니 지우고 title 등 정의 안되었기 때문에 app에서 파라미터로(prop으로) 받아와야함
//Movie compo는 이 property를 다 부모 compo에서 받아옴
export default Movie;

```

home

```

import { useEffect, useState } from "react"
import Movie from "../components/Movie"

function Home () {
  const [loading, setLoading] = useState(true)
  const [movie, setMovie] = useState([])
  useEffect(() => {
    fetch(`https://yts.mx/api/v2/list_movies.json?minimum_rating=8.8&sort_by=year`)
    .then((response) => response.json()) //json이 반환받는 object
    .then((json) => {
      setMovie(json.data.movies); //log 찍어보면 ob 안 data 안 movies에 영화 어레이 있음
      setLoading(false); //영화 로딩 끝났으니 이 때 한번 false로 만들기
    })
  }, [])

  return (
    <div>
      {loading ? <h1>Wait please</h1> :
      <div>
        {movie.map((movie) => (
          <Movie
            key={movie.id} //추가 중요
            id={movie.id} //링크로 넘길때 사용
            medium_cover_image={movie.medium_cover_image}
            title={movie.title}
            summary={movie.summary}
            genres={movie.genres}

```

```

        />

        )))
      </div>
    }
  </div>
)
}

export default Home;

```

app

```

import {
  BrowserRouter as Router,
  Switch,
  Route,
} from "react-router-dom"
import Home from "../routes/Home"
import Detail from "../routes/Detail";

function App () {
  return <Router>
    <Switch>
      <Route path="/movie/:id">
        <Detail />
      </Route>
      <Route path="/"> { /*홈으로 간다는 뜻 - home route를 렌더링 - home이 가장 아래 있어야하는듯*/ }
        <Home />
      </Route>
    </Switch>
  </Router>
}

export default App;

```

- gh pages

깃헙 페이지에 결과물 업로드 하게 해주는 패키지

npm i gh-pages → 패키지 설치

npm run build → 코드 압축, 최적화

pac.json 에서 devel 밑에 }, 찍고 추가

```
"homepage" : "http://assistant8.github.io/Agile_Frontend_Web"
```

json에서 scripts에 추가

```
"deploy": "gh-pages -d build",
"predeploy": "npm run build"
```

npm run deploy

- git 복구

git reflog

`git reset HEAD ^^` (커밋 2번 삭제)