



# **ReactJS로 영화 웹 서비스 만들기**

**이정민**



## How does it work?

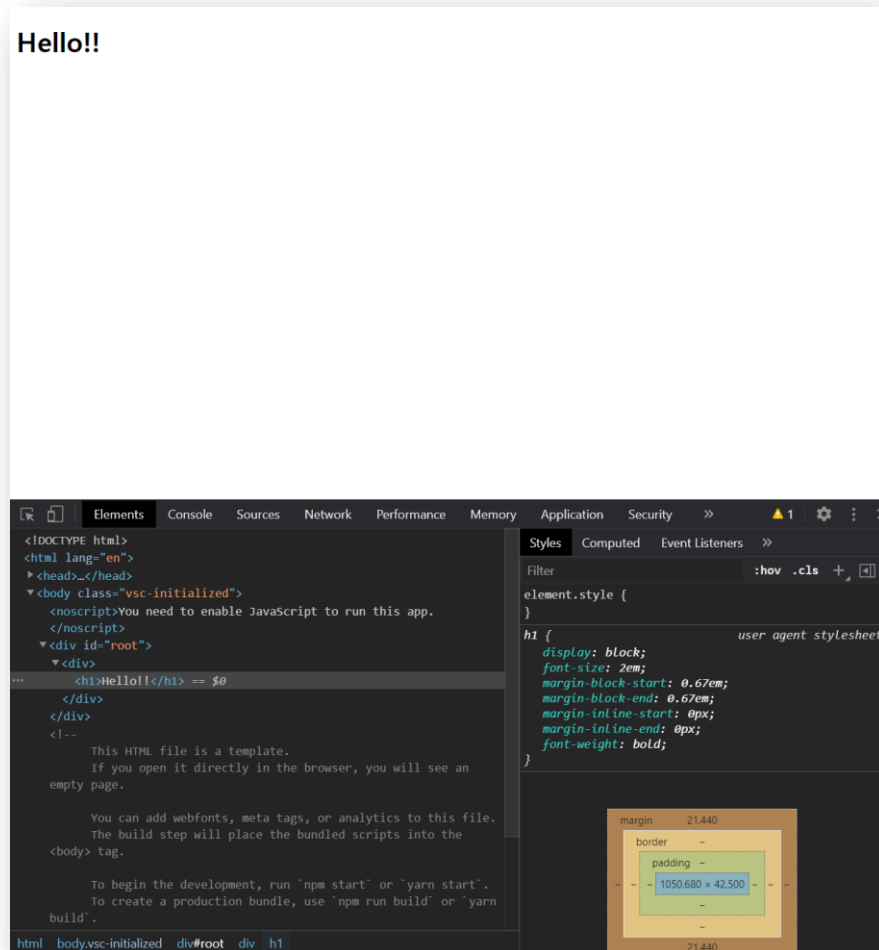
```
JS index.js  X  JS App.js
src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import App from './App';
4
5
6  ReactDOM.render(() => {
7    <React.StrictMode>
8    <App />
9  </React.StrictMode>,
10  document.getElementById('root')
11  );
12
```

```
JS index.js  JS App.js  X
src > JS App.js > App
1  import React from 'react';
2
3  function App() {
4    return <div><h1>Hello!!</h1></div>;
5  }
6
7  export default App;
8
```

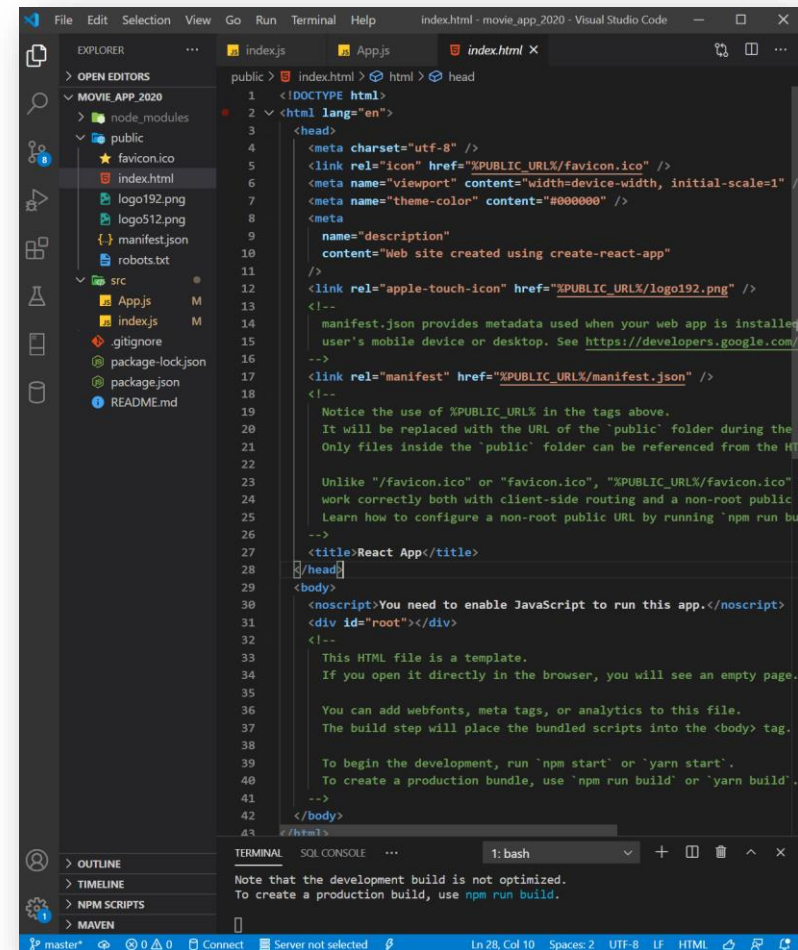
- React의 동작원리는 react에다 적은 모든 요소를 생성시켜서 그것들을 HTML에다가 밀어 넣어줌
- App에 있는 component들을 ElementById내부에 넣어주는것
- 소스코드에 처음부터 HTML을 넣지 않기 때문에 빠르다
- Virtual dom을 통해 이를 수행함.



# How does it work?



<크롬 페이지>



<실제 index.html 파일>

# React Component



```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
)  
;  
  
function App() {  
  return <div><h1>Hello!!</h1></div>  
}
```

<App />을 Component라고 부르며 컴포넌트는 HTML을 반환하는 함수이다.

```
import React from 'react';
```

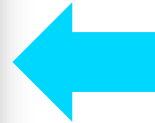
Jsx가 component를 인식하기 위해 사용하는 import문  
Jsx란 자바스크립트 안의 html

# React Component

```
src >  App.js >  default
1  import React from 'react';
2  import Potato from './potato'
3
4  function App() {
5    return <div>
6      <h1>Hello!!</h1>
7      <Potato />
8    </div>
9  }
10
11  export default App;
12
```

Potato.js 라는 파일을 만들어 App에다가 Component  
형식으로 추가를 해주어서 다음과 같이 사용할 수 있음

```
1  import React from 'react';
2
3  function Potato() {
4    return <h3>I love potato</h3>
5  }
6
7  export default Potato;
```



```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
);
```

다음과 같이 REACT는 한 번에 <App />라는 하나의 component만 rendering 할 수 있음!



# Reusable Component

```
src > Js App.js > App
1  import React from 'react';
2
3
4  function Food({fav}) {
5    return <h1>I like {fav}</h1>
6  }
7
8  function App() {
9    return <div>
10      <h1>Hello!!</h1>
11      <Food fav = "kimchi"/>
12      <Food fav = "ramen"/>
13      <Food fav = "chicken"/>
14      <Food fav = "pizza"/>
15    </div>
16  }
17
18  export default App;
19
```

기본 HTML에서 <div class = "이름"></div> 이렇게 하는 것과  
같이 React에서도 Component에 특정 값을 주게 되면 이것이  
Props가 되고 위의 함수의 인자로써 전달받게 할 수 있음

Jsx+props로 모두 재사용하게 만들 수 있음

```
function Food(props) {
  return <h1>I like {props.fav}</h1>
}
```

```
function Food({fav}) {
  return <h1>I like {fav}</h1>
}
```

인자(props)를 전달 받을 때 두 가지 방법으로 모두 전달 받을 수 있음



# Dynamic Component

```
const foodILike = [
  {
    name: "Kimchi",
    image: "https://cdn.imweb.me/thumbnail/20200415/6",
  },
  {
    name: "Ramen",
    image: "https://post-phinf.pstatic.net/MjAxODAyMD",
  },
  {
    name: "Chicken",
    image: "https://pelicana.co.kr/resources/1",
  },
  {
    name: "Pizza",
    image: "https://src.hidoc.co.kr/image/lib",
  }
];
```

```
function App() {  
  return <div>  
    {foodILike.map(dish => <Food name = {dish.name} picture = {dish.image}/>)}  
  </div>  
}
```

```
function Food({name, picture}) {  
  return <div>  
    <h2>I like {name}</h2>  
    <img src={picture} />  
  </div>  
}
```

기존의 코드로는 웹사이트에서 온 데이터 즉 api를 통해 받은 데이터를 처리할 수 없기 때문에 다음과 같은 예시를 만들어서 확인함  
foodLike이 동적으로 온 데이터라고 하였을 때 저 json데이터를 화면에 출력하기 위해 map이라는 js함수를 이용하여 하나하나씩 받아서 화면에다가 출력해주는 방법임.

위에서 화살표 함수를 이용하여 dish가 foodLike의 객체이기 때문에 dish.name, dish.image로 데이터들을 props로 만들어주고 food에다가 name과 picture를 이용해 데이터를 전달해줌.



I like Kimchi



I like Ramen



I like Chicken



I like Pizza



# React Map recap

✖ ▶ Warning: Each child in a list should have a unique "key" prop.

index.js:1

Check the render method of `App`. See <https://fb.me/react-warning-keys> for more information.  
in Food (at App.js:34)  
in App (at src/index.js:8)  
in StrictMode (at src/index.js:7)

foodLike라는 객체를 배열로 만들었기 때문에 이를 분별할 수 있는 고유한 key가 있어야 하므로 다음과 같이 id의 키 값을 넣어줘 오류를 없앴

```
const foodLike = [  
  {  
    id : 1,  
    name: "Kimchi",  
    image:  
      "https://cdn.imweb.me/thumbnaill/20200415/6b6e035658bac.png"  
  },  
]
```

# React Map recap

Compiled with warnings.

`./src/App.js`

Line 7:3: img elements must have an alt prop, either with meaningful text, or an empty string for decorative images [jsx-a11y/alt-text](#)

Search for the [keywords](#) to learn more about each warning.  
To ignore, add `// eslint-disable-next-line` to the line before.

다음의 오류는 alt라는 값을 사용해주시 않아서 그런데 이는 시각장애인들을 위한 알림이며 이를 위한 코드를 추가해주면 warnings이 없어진다.

```
function Food({name, picture}) {  
  return <div>  
    <h2>I like {name}</h2>  
    <img src={picture} alt={name} />  
  </div>  
}
```



# React Protection with Proptypes

Props안에 있는 prop들의 type을 검증하기 위해 사용하는 것

```
dlwoa@DESKTOP-J840PQF MINGW64 ~/Desktop/movie_app_2020 (master)
$ npm i prop-types
npm WARN tsutils@3.17.1 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\jest-haste-map\node_modules\
```

이를 사용하기 위해 npm | prop-types로 설치를 우선 해주어야함!

```
const foodLike = [
  {
    id : 1,
    name: "Kimchi",
    image:
      "https://cdn.imweb.me/thumbnail/20200415/6b6e035658bac.png",
    rating: 5
  }
]
```

```
function Food({name, picture, rating}) {
  return <div>
    <h2>I like {name}</h2>
    <h4>{rating}/5.0</h4>
  </div>
}
```

```
function App() {
  return <div>
    {foodLike.map(dish =>
      <Food name = {dish.name} picture = {dish.image} rating = {dish.rating}/>)}
  </div>
}
```

그 후에 rating을 추가해주고,



# React Protection with Proptypes

```
✖ Warning: Failed prop type: Invalid prop `rating` of type `number` supplied to `Food`, expected `string`.   index.js:1  
    in Food (at App.js:50)  
    in App (at src/index.js:8)  
    in StrictMode (at src/index.js:7)
```

```
Food.propTypes = {  
  name: propTypes.string.isRequired,  
  picture : propTypes.string.isRequired,  
  rating : propTypes.string.isRequired  
}
```

propTypes을 검사해주는 코드를 넣고 실행을 시키면 위와같이 rating은 number로 표시되는데 기대하는 것은 string이므로 에러가 발생!

isRequired가 있으면 요구되는 값이 무조건 있어야하고 이러한 propTypes를 이용해서 많은 것들을 체크해볼 수 있다!



# React Class components and State

```
class App extends React.Component {  
  render() {  
    return <h1>Im a class component</h1>  
  }  
}
```

Class component는 function component와 달리 state를 표현할 수 있음

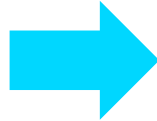
하지만 function이 아니기 때문에 return을 할 수 없고 대신에 render라는 method를 이용!

```
class App extends React.Component {  
  state = {  
    count: 0  
  };  
  add = () => {  
    console.log("add");  
  };  
  minus = () => {  
    console.log("minus");  
  };  
  render() {  
    return (  
      <div>  
        <h1>The number is: {this.state.count}</h1>  
        <button onClick={this.add}>Add</button>  
        <button onClick={this.minus}>Minus</button>  
      </div>  
    );  
  }  
}
```

State는 유동적인 데이터이며 변화를 주고 싶은 데이터를 state로 정하고 state 자체는 객체라고 생각하면 된다. 또한 이 값을 사용하려면 this.state.count와 같이 this를 붙여서 사용해야한다!

# React All you need to know about State

```
add = () => {  
  this.state.count = 1;  
};  
minus = () => {  
  this.state.count = -1;  
};
```



Do not mutate state directly. Use setState()  
Do not mutate state directly. Use setState()

다음과 같이 state count를 변경하려고 하면 경고가 뜨며 state는 객체이기때문에 setState라는 method를 사용하여 새로운 state를 받아야 한다.

setState를 호출하면 React는 state를 refresh하고 render()를 다시 재호출하므로써 변한 값을 출력하게 된다.

```
add = () => {  
  this.setState({count : this.state.count + 1 });  
};  
minus = () => {  
  this.setState({count : this.state.count - 1 });  
};
```

# React All you need to know about State

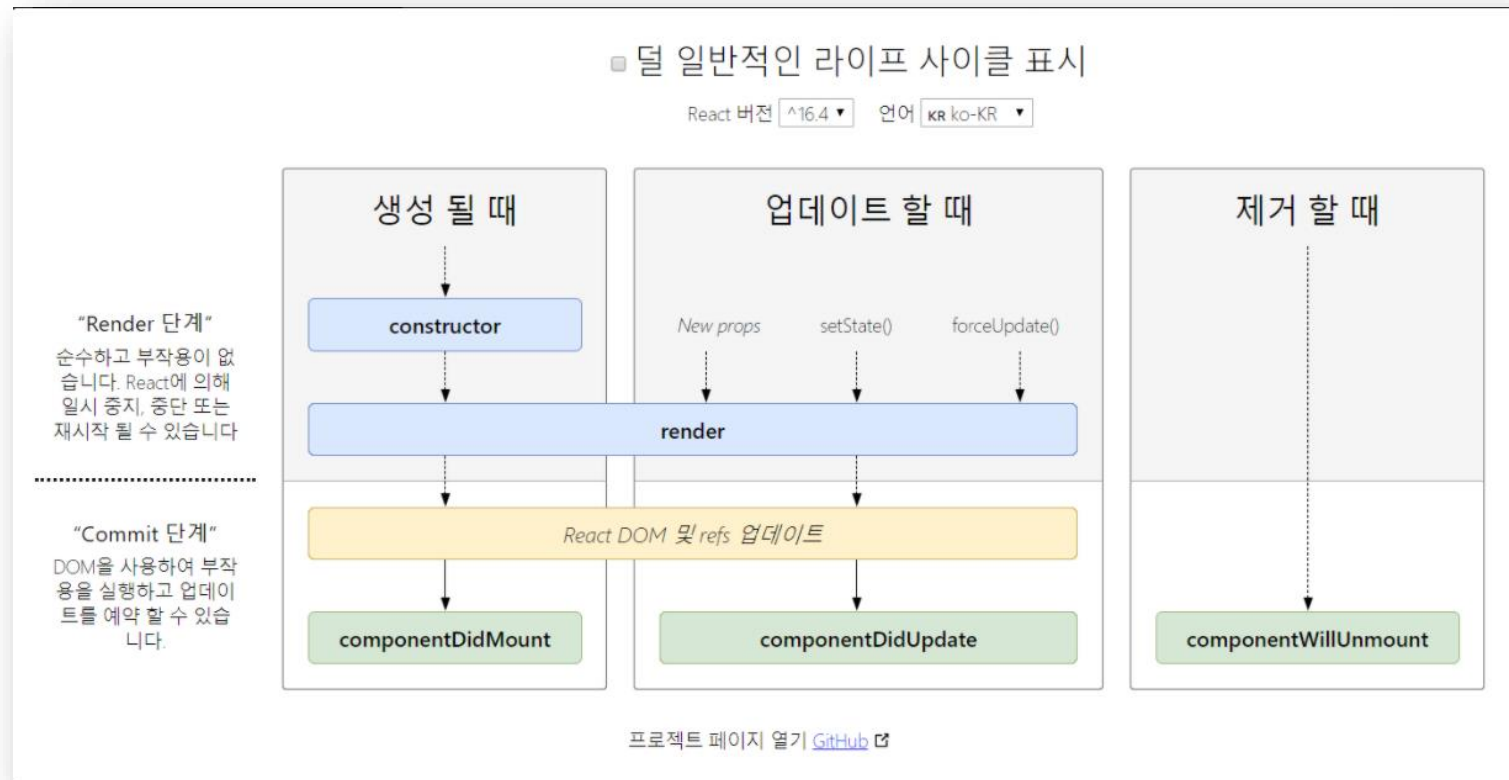
```
add = () => {  
  this.setState({count : this.state.count + 1 });  
};  
minus = () => {  
  this.setState({count : this.state.count - 1 });  
};
```

이렇게 작성을 하게되면 this.state.count에서 외부의 state에 의존하는 것이므로 이런 방법을 추천하지 않는다 대신에 current function을 사용해서 코드를 작성하는 것을 추천!



```
add = () => {  
  this.setState(current => ({count : current.count + 1 }));  
};  
minus = () => {  
  this.setState(current => ({count : current.count - 1 }));  
};
```





React에서 component를 생성하고 없애는 cycle



# Component Life Cycle

## 마운트

아래 메서드들은 컴포넌트의 인스턴스가 생성되어 DOM 상에 삽입될 때에 순서대로 호출됩니다.

- `constructor()`
- `static getDerivedStateFromProps()`
- `render()`
- `componentDidMount()`

## 주의

아래 메서드는 기존에 사용되었지만 이제는 사용하지 않습니다.

- `UNSAFE_componentWillMount()`

```
class App extends React.Component {  
  constructor(props) {  
    super(props);  
    console.log("constructor");  
  }  
  state = {  
    count: 0  
  };  
  add = () => {  
    this.setState(current => ({count : current.count + 1 }));  
  };  
  minus = () => {  
    this.setState(current => ({count : current.count - 1 }));  
  };  
  render() {  
    console.log("render");  
  }  
}
```

Mount -> component를 생성하는 행위  
constructor -> render -> componentDidMount



# Component Life Cycle

## 업데이트

props 또는 state가 변경되면 갱신이 발생합니다. 아래 메서드들은 컴포넌트가 다시 렌더링될 때 순서대로 호출됩니다.

- `static getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`
- `componentDidUpdate()`

### 주의

아래 메서드는 기존에 사용되었지만 이제는 사용하면 안 됩니다.

- `UNSAFE_componentWillUpdate()`
- `UNSAFE_componentWillReceiveProps()`

# The number is: 1

AddMinus

```
render
```

```
render
```

```
didupdate!
```

Update는 setState를 호출해서 state가 변경될 때마다 발생하는 것!

Unmount도 있는데 이는 다른페이지로 넘어갈때, 컴포넌트를 삭제하거나 교체될 때 호출함.



# React Planning the move Component

```
class App extends React.Component {  
  state = {  
    isLoading: true  
  };  
  componentDidMount(){  
    setTimeout(() => {  
      this.setState({isLoading: false}):  
    }, 6000);  
  }  
  render() {  
    const {isLoading} = this.state;  
    return (  
      <div>  
        {isLoading ? "Loading..." : "I'm ready"}  
      </div>  
    );  
  }  
}
```

삼항 연산자 부분에 원래는 `this.state.isLoading`을 적어주어야 하지만 `render` 바로 아래 부분에 `const {isLoading}`을 정의하여 `this.state`를 받아서 변수 이름만 적어주면 됨!

`componentDidMount`에서 `setState`를 사용하여 동적으로 몇초 뒤에 `isLoading`의 값을 바꾸어 프로그램을 제어할 수 있음

# Fetching Movies from API

```
import React from 'react';
import axios from "axios";




class App extends React.Component {
  state = {
    isLoading: true,
    movies = []
  };
  getMoives = async () => {
    const movies = await axios("https://yts-proxy.now.sh/list_movies.json")
  }
  componentDidMount(){
    getMovies();
  }
}
```

Fecth대신 axios를 사용하여 movies list api를 가져오고  
비동기 처리를 위해 async, await를 사용함

```
getMovies = async () => {  
  const {data: {data: {movies}}} = await axios.get("https://yts-proxy.now.sh/list_movies.json?sort_by=rating");  
  this.setState({movies, isLoading: false})  
};
```

Movie 객체 안의 data.data.movies를 다음과 같이  
{data: {data~{movies} 만 가져오는 것으로 표현할 수 있다.

그 뒤로 setState를 사용해 movies:[]를 api를 이용해 가져온 데이터인 movies로 state를 바꾸어주고  
loading의 상태를 false로 변화시켜준다

```
src >  Movie.js >  Movie >  constructor
1  import React from "react";
2  import PropTypes from "prop-types";
3
4  function Movie({id, year, title, summary, poster}) {
5    return <h4>{title}</h4>
6  }
7
8  Movie.propTypes = {
9    id: PropTypes.number.isRequired,
10   year: PropTypes.number.isRequired,
11   title: PropTypes.string.isRequired,
12   summary: PropTypes.string.isRequired,
13   poster: PropTypes.string.isRequired
14 };
15
16 export default Movie;
```

Movie.js 를 만들어 prop검사를 하여 파일을 가져오기

```
render() {  
  const {isLoading, movies} = this.state;  
  return (  
    <div>  
      {isLoading ? "Loading..." : movies.map(movie => {  
        return (  
          <Movie  
            key = {movie.id}  
            id = {movie.id}  
            year = {movie.year}  
            title = {movie.title}  
            summary = {movie.summary}  
            poster = {movie.medium_cover_image} />);  
        })}  
    </div>  
  ),  
}
```

Movies도 const로 추가를 해주고 movies의 객체를 map을 이용해 movie의 prop을 각각정의하고 return해주기



```
render() {
  const {isLoading, movies} = this.state;
  return (
    <section class = "container">
      {isLoading ? (
        <div class = "loader">
          <span class = "loader_text">Loading...</span>
        </div>
      ) : (
        <div class = "movies">
          {movies.map(movie => {
            return(
              <Movie
                key = {movie.id}
                id = {movie.id}
                year = {movie.year}
                title = {movie.title}
                summary = {movie.summary}
                poster = {movie.medium_cover_image} />
            )
          })}
        </div>
      )}
    </section>
  )
}
```

```
function Movie({id, year, title, summary, poster}) {
  return (
    <div class = "movie">
      <img src = {poster} alt={title} title = {title} />
      <div class = "movie__data">
        <h3 class = "movie__title">{title}</h3>
        <h5 class = "movie__year">{year}</h5>
        <p class = "movie__summary">{summary}</p>
      </div>
    </div>
  );
}
```

무비 앱을 좀 더 예쁘게 만들기 위해 class들을 추가



## Adding Genres

```
<section className = "container">
```

React에서 class와 html의 class가 헷갈릴 수 있기 때문에 class 대신 className을 사용

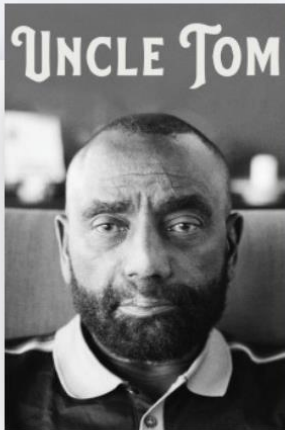
```
<ul className = "genres">  
  {genres.map((genre,index) => (  
    <li key = {index} className="genres__genre">{genre}</li>  
  ))}  
</ul>
```

Genre와 index라는 key값을 가져와서 새로운 genre라는 항목을 추가함

## Cutting the summary

```
{summary.slice(0,180)}...
```

영화 소개에 대한 내용을 표시해주는 부분을 180자로 한정해서 보여줌



### Uncle Tom

2020

Documentary

In a collection of intimate interviews with some of America's most provocative black conservative thinkers, Uncle Tom takes a unique look at being black in America. Featuring media...



### Natsamrat

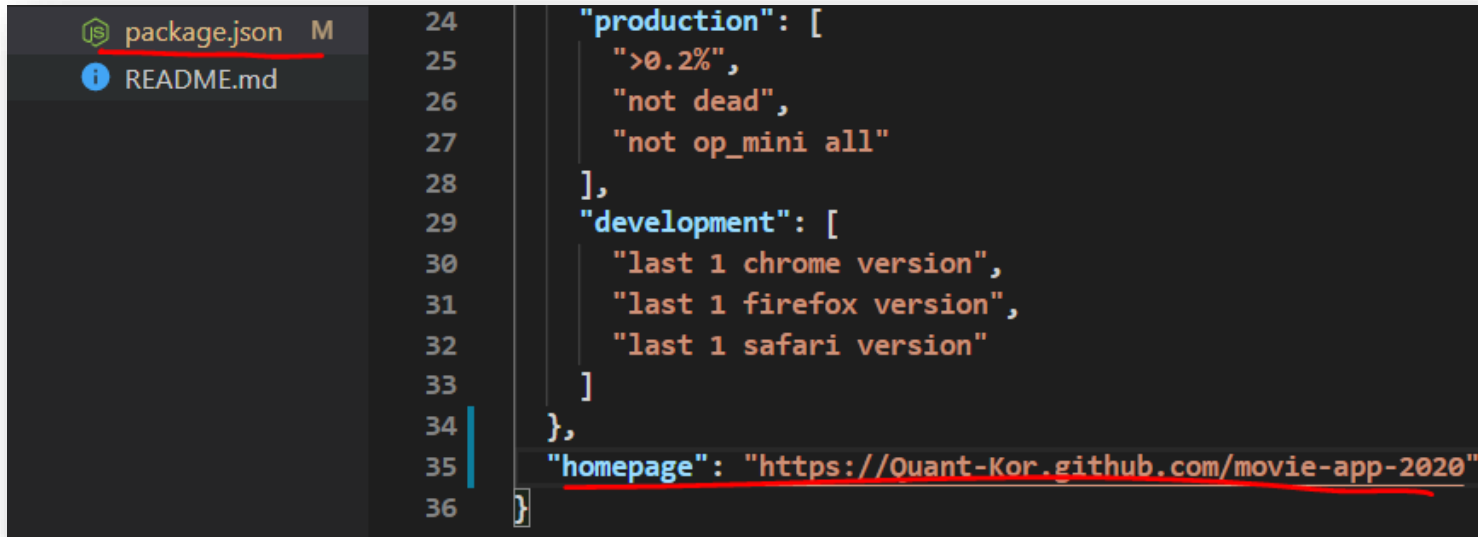
2016

Action Drama Family

The film is a tragedy about a veteran theatre actor named Ganpat "Appa" Belwalkar (Nana Patekar) who has been the best of his lot during his heyday, garnering fame and fortune acti...

```
dlwoa@DESKTOP-J840PQF MINGW64 ~/Desktop/movie_app_2020 (master)
$ npm i gh-pages
[.....] / rollbackFailedOptional: verb npm-session ba83bde2c38a90ee
```

깃허브 페이지에 deploy를 하기 위해 다음과 같은 모듈 설치

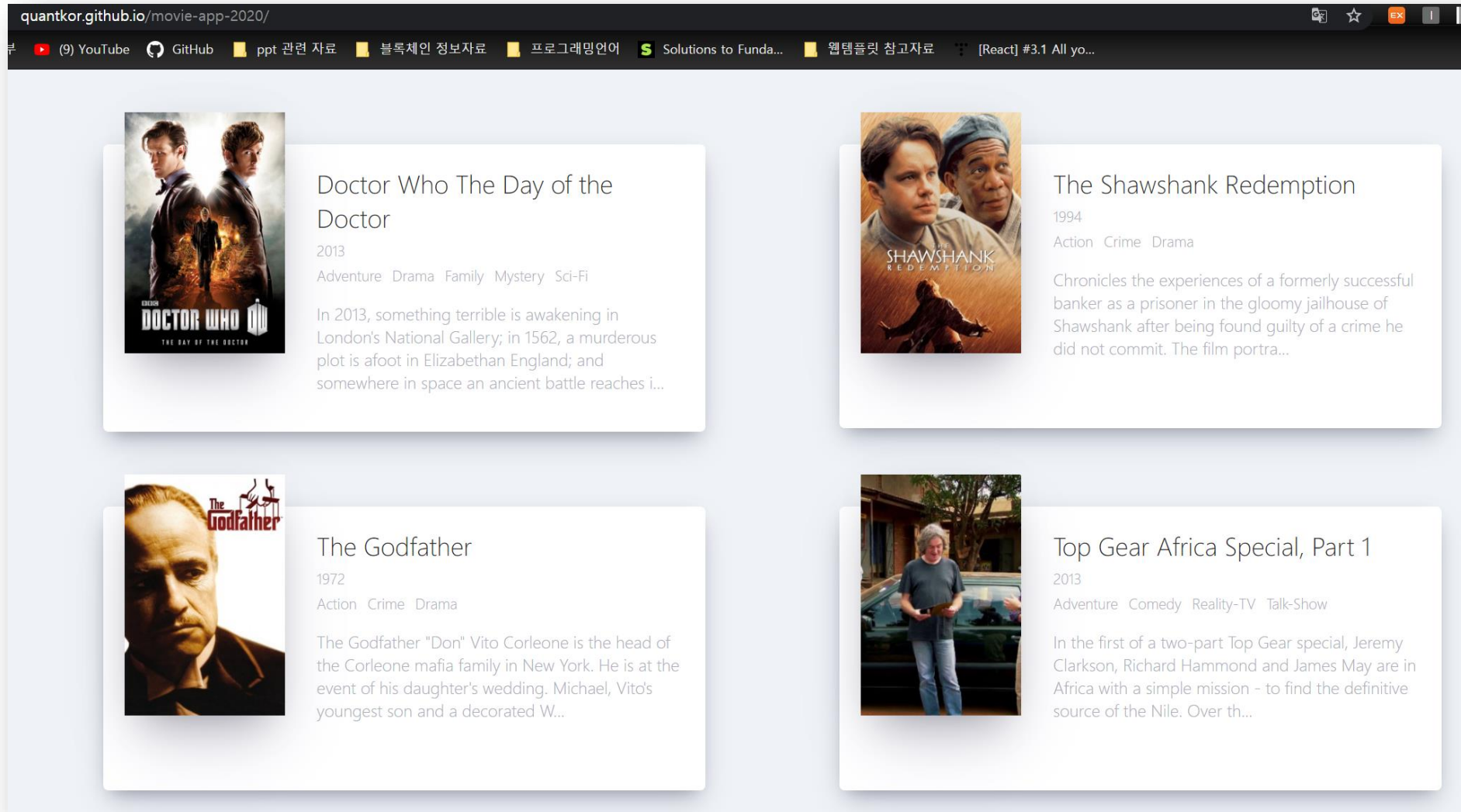


```
24   "production": [
25     ">0.2%",
26     "not dead",
27     "not op_mini all"
28   ],
29   "development": [
30     "last 1 chrome version",
31     "last 1 firefox version",
32     "last 1 safari version"
33   ]
34 },
35   "homepage": "https://Ouant-Kor.github.com/movie-app-2020"
36 }
```

Package.json에 들어가서 다음과 같이  
Homepage설정을 해주어야 함  
`https://{username}.github.io/작업공간` 이름  
주의 해야할점: 모두 소문자이어야만 함!!

```
"scripts": {  
  "start": "react-scripts start",  
  "build": "react-scripts build",  
  "deploy" : "gh -pages -d build",  
  "predeploy" : "npm run build"  
},
```

gh-pages 를 호출하기 위해 script부분에 deploy와 predeploy를 설정해 주어야함  
그 후 npm run deploy를 하여서 build가 만들어지고 페이지가 깃허브에 올라가짐

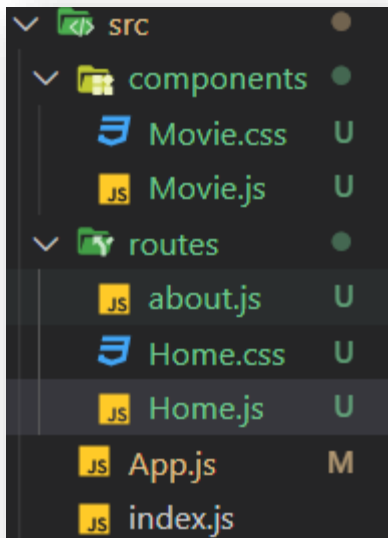




# Getting Ready for the Router

```
dlwoa@DESKTOP-J840PQF MINGW64 ~/Desktop/movie_app_2020 (master)
$ npm install react-router-dom
```

Router설정을 위해 다음의 모듈을 설치해 주어야함



폴더 구조를 다음과 같이 바꾸어준다음에 App.js에 있던 파일을 home.js에 옮김

# Building the Router

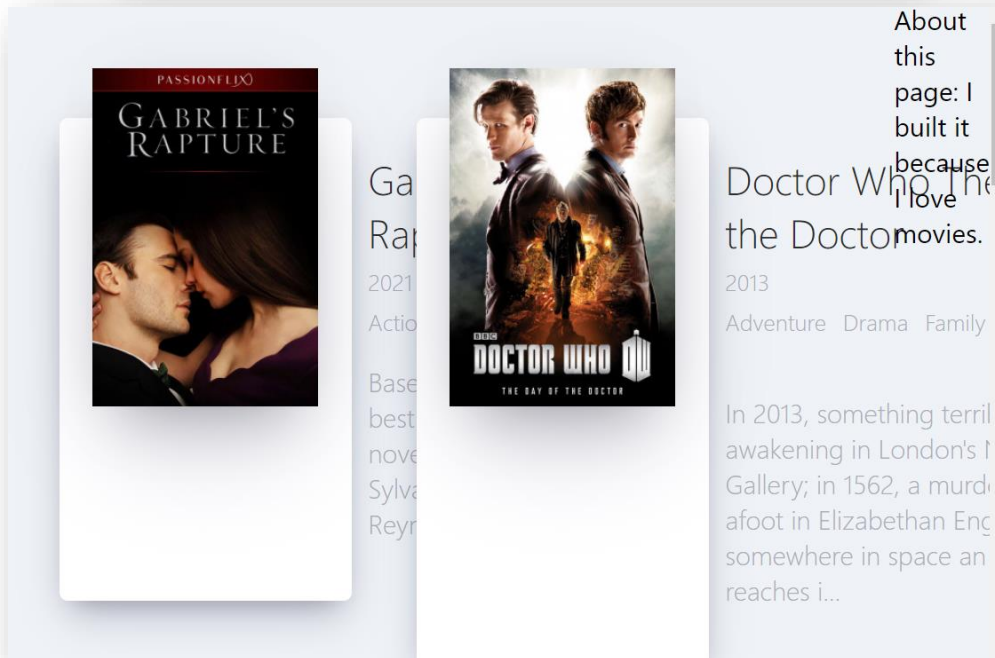
```
import Home from "../routes/Home"
import About from "../routes/About"

function App () {
  return <HashRouter>
    <Route path = "/" component = {Home} />
    <Route path = "/about" component = {About} />
  </HashRouter>
}
```

Route안에는 중요한 props가 들어가는데 path로 들어가는 url주소이고 component를 통해 어떠한 파일을 보여줄 지 정하는 것. 하지만 이렇게 만들게 되면 아래 사진과 같이 파일이 겹쳐서 보임

그 이유가 path가 현재 "/"와 "/about"인데 "/about"에서 "/"또한 component 인식이 되기 때문에 2개가 동시에 렌더링이 되기때문

→ Exact = {true}를 사용하는 이유







## Building the Navigation

```
function Navigation(){  
  return (  
    <div>  
      <a href= "/">Home</a>  
      <a href= "/about">About</a>  
    </div>  
  )  
}
```

HomeAbout

다음과 같이 navigation을 만들어 버리게 되면 html이기 때문에 그냥 페이지를 새로고침하는 꼴이 되어 react형식으로 바꾸어 주어야함

# Building the Navigation

```
function Navigation(){  
  return (  
    <div>  
      <Link to= "/">Home</Link>  
      <Link to= "/about">About</Link>  
    </div>  
  )  
}
```

Link를 사용하고 있는 경우에는 무조건 Router안에다가  
넣어야함

```
return <HashRouter>  
  <Navigation />  
  <Route path = "/" exact={true} component = {Home} />  
  <Route path = "/about" component = {About} />  
</HashRouter>
```

# React Sharing Props Between Routes

```
function About(props) {  
  console.log(props);  
  return (  
    <div className="about__container">  
      <span>  
        "Freedom is the freedom to say that two plus two make four. If that is  
        granted, all else follows."  
      </span>  
      <span>- George Orwell, 1984</span>  
    </div>  
  );  
}
```

About에 props를 전달하여 chrome에서 console로 찍어 보면 다음과 같은 객체들을 전달하는 것을 볼 수 있다. 이것을 이용해서 영화의 설명을 눌렀을 때, 자세한 설명을 볼 수 있도록 만들 것임.

```
{history: {...}, location: {...}, match: {...}, staticContext: undefined} i  
  ▶ history: {length: 6, action: "POP", location: {...}  
  ▶ location: {pathname: "/about", search: "", hash...  
  ▶ match: {path: "/about", url: "/about", isExact:...  
    staticContext: undefined  
  ▶ __proto__: Object
```

# React Sharing Props Between Routes

```
function App () {  
  return <HashRouter>  
    <Navigation />  
    <Route path = "/" exact={true} component = {Home} />  
    <Route path = "/about" component = {About} />  
    <Route path = "/movie-detail" component = {Detail} />  
  </HashRouter>  
}
```

App 부분에 Detail로 라우팅 할 수 있는 코드를 만들어 주고  
Detail.js에 다가 다음과 같이 적어줌

```
src > routes > Detail.js > default  
1  import React from "react";  
2  
3  function Detail(props){  
4    console.log(props);  
5    return <span>hello</span>;  
6  }  
7  
8  export default Detail;
```

# React Sharing Props Between Routes

```
function Movie({id, year, title, summary, poster, genres}) {  
  return (  
    <Link to = {{  
      pathname: "/movie-detail",  
      state: {  
        year,  
        title,  
        summary,  
        poster,  
        genres  
      }  
    }}>  
  )  
}
```

Movie의 모든 요소에 Link를 걸어  
정보를 라우터로 보냄.

Path를 설정한 다음 Detail.js로 가게 만들어 연결시켜줌

```
{  
  history: {length: 14, action: "PUSH", location: {...}, createHref: f,...  
  location:  
    hash: ""  
    pathname: "/movie-detail"  
    search: ""  
    state:  
      genres: (2) ["Action", "Romance"]  
      poster: "https://yts.mx/assets/images/movies/gabriels_rapture_2...  
      summary: "Based on the best selling novel from by Sylvain Reyna...  
      title: "Gabriel's Rapture"  
      year: 2021  
      __proto__: Object  
    __proto__: Object  
  match: {path: "/movie-detail", url: "/movie-detail", isExact: true,...  
    staticContext: undefined  
  __proto__: Object
```

```
▼ location:
  hash: ""
  pathname: "/movie-detail"
  search: ""
▼ state:
  ► genres: (3) ["Action", "Crime", "Drama"]
  poster: "https://yts.lt/assets/images/movies/The_Shawshank_Redemption_1994/med
  summary: "Chronicles the experiences of a formerly successful banker as a pris
  title: "The Shawshank Redemption"
  year: 1994
```

Detail을 클릭해서 들어갔을 때와  
바로 url에 movie-detail로 들어갔을 때의 차이  
State가 정의되지 않음

```
{pathname: "/movie-detail", search: "", hash: "", state: undefined} ⓘ Detail.js
  hash: ""
  pathname: "/movie-detail"
  search: ""
  state: undefined
  proto : Object
```

```
import React from "react";

class Detail extends React.Component {
  componentDidMount() {
    const {location, history} = this.props;
    if(location.state === undefined) {
      history.push("/");
    }
  }
  render() {
    const {location} = this.props;
    return <span>{location.state.title}</span>
  }
}

export default Detail;
```

따라서 location.state가 정의되어 있지 않은 경우  
즉, url에 디렉트로 들어온 경우 홈화면으로 돌려보내줌

그러나 여기서 문제점은 이렇게 만들었을 경우 새로고침하는  
할 때 render가 먼저  
실행되고 나서 componentDidMount가 실행되기 때문에  
Location이 undefined되어 있어서 에러가 발생

# React Redirecting

```
}  
render() {  
  const {location} = this.props;  
  if(location.state) {  
    return <span>{location.state.title}</span>  
  } else {  
    return null;  
  }  
}
```

따라서 다음과 같이 control 해줌