

# 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

```

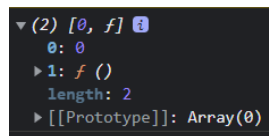
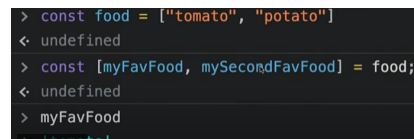
let counter=0;
function countUp() {}

-----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

auto re-loading이 됨 (새로고침 안해도)

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

기본 파일들이 많이 달려있음

- 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) =>
  item.toUpperCase())
동영상 더보기
< ▶ (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