

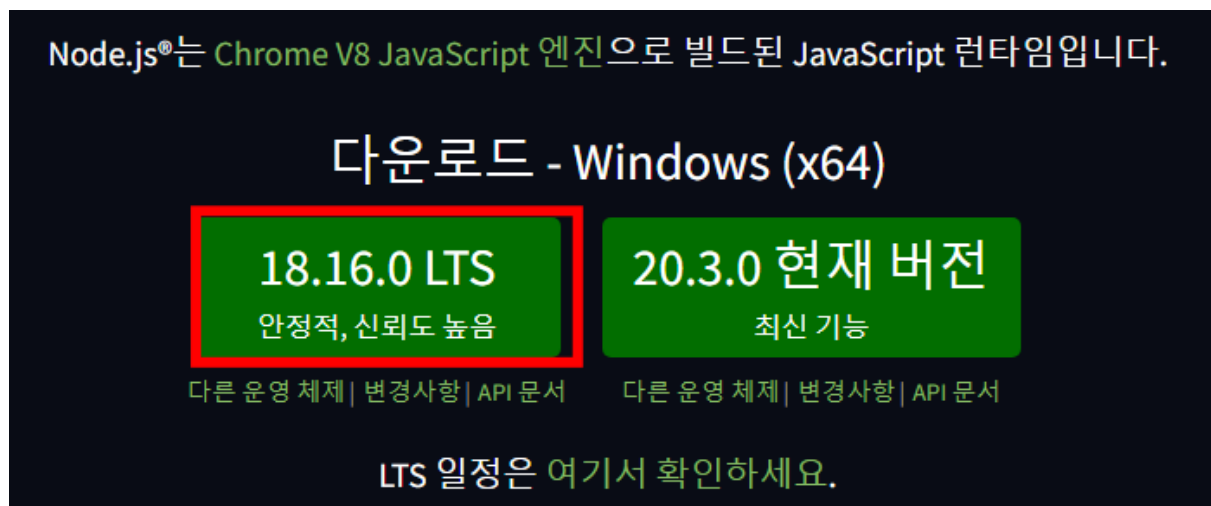
React 18 총정리 - 1 (초급)

- [1. 개발환경 세팅](#)
- [2. 서버와 클라이언트](#)
- [3. React 소개](#)
- [4. React 기본 문법](#)
 - [4-1. JSX](#)
 - [4-2. States, Curly Braces](#)
 - [4-3. Event Handler](#)
 - [4-4. Curly Braces in Attribute](#)
 - [4-5. <input> value to State](#)
 - [4-6. Conditional Rendering](#)
 - [4-7. Rendering Lists](#)
- [5. 응용문제](#)

1. 개발환경 세팅

Node.js 공식 홈페이지에 방문하자.

<https://nodejs.org/ko>



다음 화면에서 **LTS** 인스톨러를 다운로드해 설치한다.

설치 후, cmd 실행해 다음 두 가지 명령어를 입력해 버전 체크한다.

```
$ node --version
$ npm --version
```

cmd 실행 후, 원하는 디렉터리에 경로를 생성한다.

ex) 바탕화면

그리고, 다음 명령어를 입력해서 프로젝트를 생성한다.

여기선 `vite-project` 이라는 프로젝트를 생성하겠다.

```
$ npm create vite@latest
```

```
✓ Project name: ... vite-project
✓ Select a framework: » React
✓ Select a variant: » JavaScript + SWC
```

프로젝트 이름 `vite-project` 으로 지정

프레임워크는 React

variant 는 JavaScript + SWC

- SWC 는 빌드 속도를 빠르게 해주는 툴

```
Done. Now run:

cd vite-project
npm install
npm run dev
```

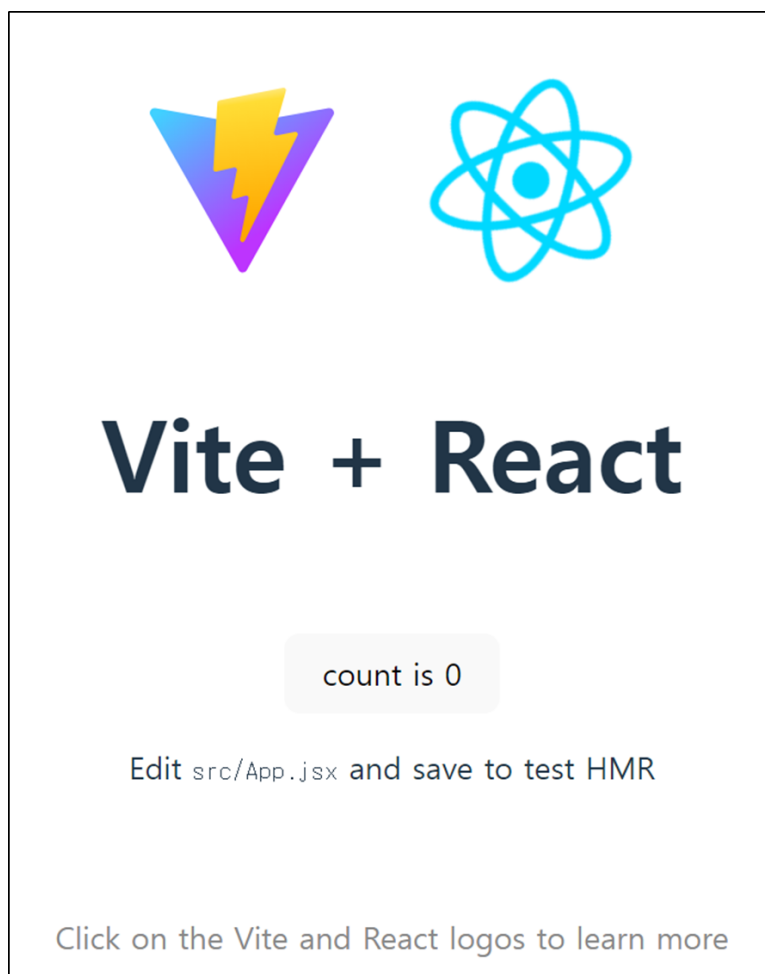
이후, 다음 명령어를 입력하자.

```
$ cd 디렉터리이름 # 해당 디렉터리로 진입
$ npm install # dependencies 설치 (추후 상세히 설명)
$ npm run dev # 테스트 서버 구동
```

```
VITE v4.3.9 ready in 327 ms

→ Local:  http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```


서버 구동 성공 메시지 확인 후, 크롬 브라우저에서 `localhost:5173` 으로 접속하자.



확인 후, `cmd` 를 종료한다.

크롬 확장 프로그램을 설치하자.

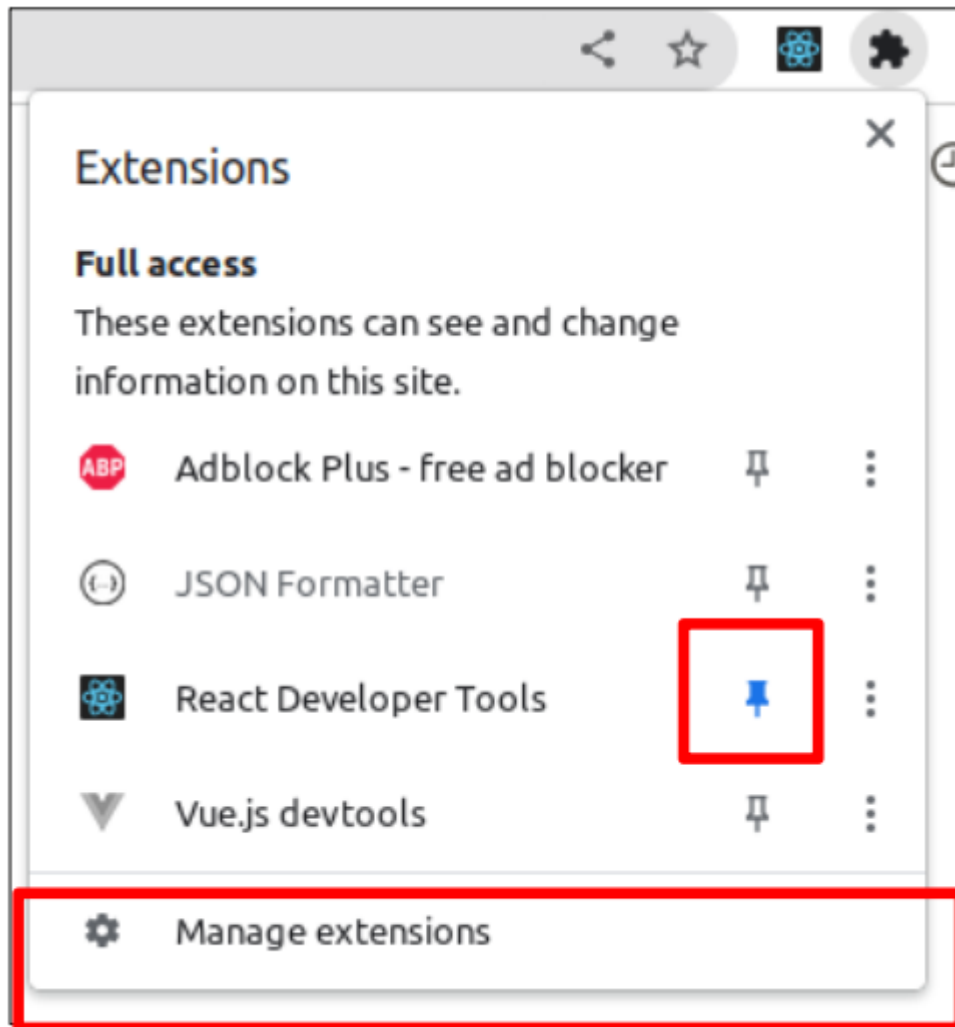
크롬 웹스토어 접속 후, React Developer Tools 를 설치한다.

 **React Developer Tools**

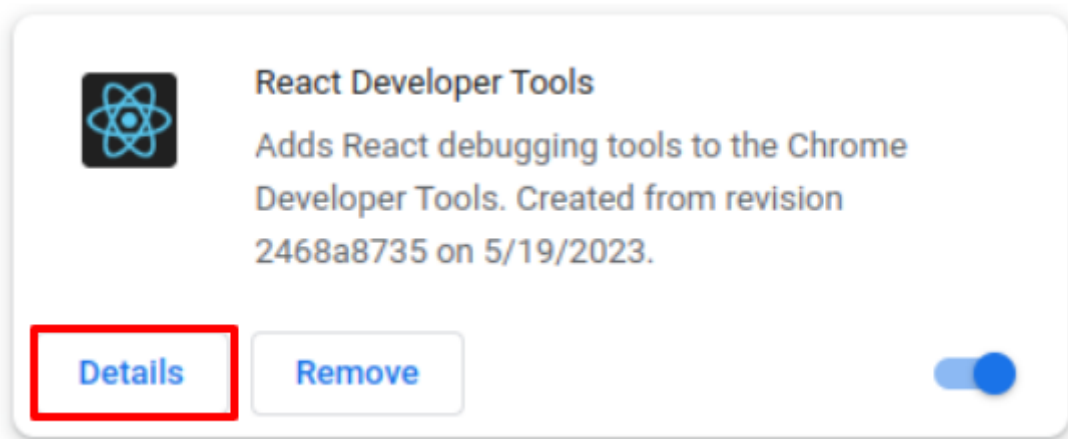
Chrome에서 삭제

 추천

★★★★★ 1,434 ⓘ | 개발자 도구 | 사용자 4,000,000+명



설치 후, 확장이 잘 보이도록 고정하고, 확장 프로그램 관리 버튼을 클릭한다.



세부정보 (Details) 클릭 후,



시크릿 모드에서 허용, 파일 URL에 대한 액세스 허용 두 가지를 허용해주고, 크롬창을 전부 종료한 후, 다시 켜주자.

이후, VSCode 로 프로젝트 디렉터리를 열어서 학습에 필요없는 파일을 정리하도록 하겠다.

1. `src/assets/` 에 있는거 모두 삭제
2. `src/App.css`, `src/index.css` 파일은 두되, 내용 모두 삭제

그리고, `App.jsx` 를 다음처럼 작성한다.

```
import './App.css';
import { useState } from 'react';

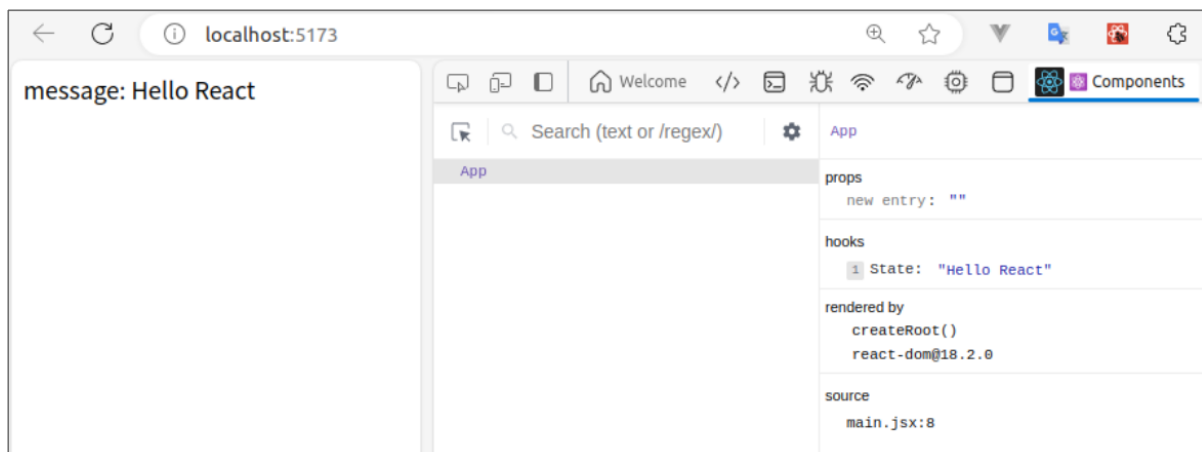
export default function App() {
  const [ message, setMessage ] = useState("Hello React");
  return (
    <>
      <div>message: { message }</div>
    </>
  );
}
```

```
);  
}
```

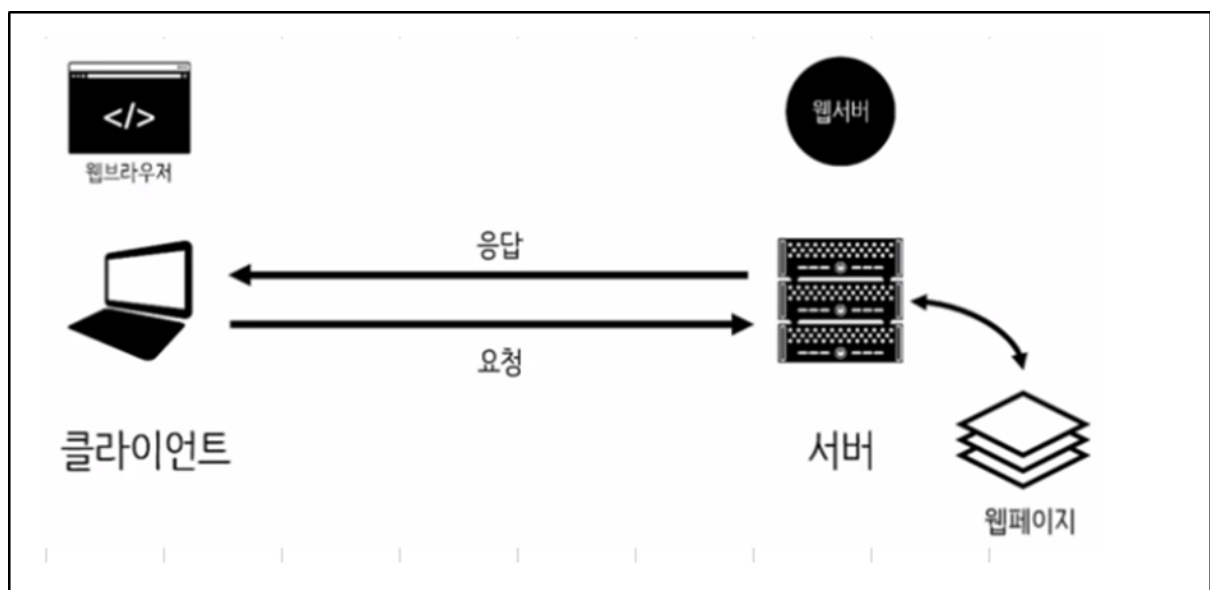
이후, 서버를 동작시키자. VSCode 에서 `Ctrl + `` 버튼을 눌러 터미널을 켜고, 다음 명령어를 입력한다.

```
$ npm run dev
```

그리고, 크롬 브라우저에서 `localhost:3000` 에 접속한 후, 개발자 도구도 잘 작동되는지 확인해보자.



2. 서버와 클라이언트



서버: 백엔드

- 사용자 눈에 보이지 않는 곳에서 작동하는 부분
- 클라이언트에게 정보나 서비스를 제공하는 시스템
ex) 웹서버, 데이터베이스 서버
- 클라이언트의 요청 (request) 을 받으면, 적절한 처리 후 클라이언트에게 응답 (response)
- 언어 및 기술: Java, Python, Node.js, Server, DB, API ...

클라이언트: 프론트엔드

- 웹에서는 클라이언트가 프론트엔드
- 사용자 눈에 보이고 상호작용하는 부분
- 서버를 통해 접속할 수 있는 애플리케이션이나 서비스
ex) 브라우저, 웹사이트
- 서버에게 요청 (request) 하면, 서버로부터 응답 (response) 을 받는다.
- 언어 및 기술: HTML, CSS, JavaScript, React.js, Vue.js ...

3. React 소개

React 공식문서의 링크는 다음과 같다.

<https://react.dev/>

여기선 React 를 다음과 같이 소개한다.



웹과 네이티브 유저 인터페이스를 위한 라이브러리

이건 홍보용 정의이며, 공식문서에서 소개하는 좀 더 정확한 정의는 다음과 같다.

A JavaScript library for rendering user interfaces (UI).

즉, UI 를 만들기위한 JavaScript 라이브러리다.

장점은 다음과 같다.

- HTML 과 JavaScript 를 결합한 JSX 라는 문법 사용
 - HTML, JavaScript 만으로 개발 할 때에 비해 훨씬 쉬움
- 화면에 변경사항이 생기면 새로고침 없이 즉각 변경
 - 앱처럼 부드러운 웹 개발 가능
- 협업의 편리함
 - 화면 각 부분을 컴포넌트 단위로 나눠서 개발
 - 프론트엔드와 백엔드가 서로 다른 프로젝트로 분리되어 통신
- 가장 중요한 장점: 대기업 Meta 가 개발 및 유지보수
 - 경쟁 프레임워크 Vue , Svelte 는 대기업이 아니라 자체 팀에서 개발 및 유지보수
 - 즉, 여러가지 이유로 하루 아침에 개발이 중단될 수 있음

단점은 다음과 같다.

- 최초 로딩속도가 느림
- SEO(검색엔진 최적화) 불편

위 두가지 단점을 극복하기 위해, 풀스택 프레임워크 Next.js 사용

수업은 다음과 같이 진행한다.

- 특정 시기에 이르기 전까진, `App.jsx` 만 변경하며 수업 진행
- 프로젝트 구조에 대한 상세 설명은 컴포넌트를 배울 때 진행

왜냐면, 처음부터 지나치게 이론만 설명할 시 React 의 재미를 느끼기 힘들기 때문이다.

4. React 기본 문법

4-1. JSX

React 는 JSX 라는, 단일 HTML 태그를 리턴하는 JavaScript 문법을 사용한다.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [ message, setMessage ] = useState("Hello React");
  return (
    <>
      <div>message: { message }</div>
    </>
  );
}
```

지켜야 할 몇 가지 원칙이 있으니 기억하자.

1. `export default` 를 `function` 키워드 앞에 붙인 특별한 함수를 컴포넌트라고 한다.
2. 컴포넌트 이름은 파일명과 일치하게 네이밍하고, 대문자로 시작하는 PascalCase 이다.
3. 리턴값은 단 하나의 HTML 태그이며, 관습적으로 `<></>` 로 작성한다. 이를 Fragment 라고 한다. 함수의 리턴값은 두 개가 될 수 없기에, 태그는 하나만 존재한다.
4. 모든 태그는 반드시 닫아야한다. `` 처럼, 닫지 않아도 되는 태그라하더라도 `` 처럼 명시적으로 닫아야한다.

5. 태그의 `class` attribute 는 JavaScript 의 `class` 키워드와의 혼동을 피하기 위해 `className` 으로 쓴다.
6. CSS 는 파일을 분리해 `import` 해서 사용한다. CSS 파일을 네이밍할땐, 담당하는 jsx 파일 이름과 일치시킨다.

4-2. States, Curly Braces

우선 다음 코드를 살펴보자.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [ message, setMessage ] = useState("Hello React");
  return (
    <>
      <div>message: { message }</div>
    </>
  );
}
```

`useState` 는 state (한국어로는 상태) 를 생성할 때 사용한다.

state 는 일반 변수와는 다르다.

- state 가 변하면, 화면에서 해당 부분을 다시 렌더링
 - 렌더링: 화면을 그린다는 뜻

`useState` 사용 시엔 다음을 알아야 한다.

- 사용하려면 react 에서 `import` 해야 함
- argument 로 들어가는 값은 `message` 변수의 기본값이 됨
- 하나의 배열을 리턴받는데, `state` , `setState` 이며, `set` 으로 시작하는 함수는 state 를 변경할 때 쓰인다.

Curly Braces 는 중괄호 `{ }` 를 의미하며, HTML 에서 변수를 사용하기 위해 존재한다.

앞으로 강의 진행 편의상, state 를 변수라고 부를 것이지만, 일반 변수와의 차이점은 반드시 알고 있도록 하자.

4-3. Event Handler

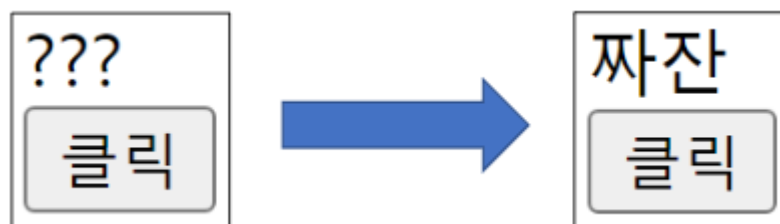
event handler: 이벤트 발생시 동작하는 함수를 말한다.

- event: 브라우저에서 일어나는 모든 종류의 사건
 - ex) click, change, keyup, scroll
- 이벤트 종류는 다음 링크를 참조
 - https://www.w3schools.com/jsref/dom_obj_event.asp

다음 코드를 확인해보자.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [ text, setText ] = useState("???");
  function handleClick() {
    setText("짜잔");
  }
  return (
    <>
      <div>{ text }</div>
      <button onClick={ handleClick }>클릭</button>
    </>
  );
}
```



클릭 이벤트일 경우, **on** 접두사를 앞에 붙여 **onClick** 이다.

클릭 이벤트가 발생했을 때, **handleClick** 이라는 함수를 실행시키며, 해당 함수에서 **setText** 를 사용해 **text** 변수의 값을 변경하는 것을 알 수 있다.

- 이벤트핸들러는 관습적으로 **handle** 접두사 + 이벤트명 으로 네이밍한다.

- 함수 안에 함수를 선언했기 때문에 이상하게 보일 수 있지만, 올바른 방식이다. 쓰고자 하는 함수는 `export default` 바깥에 선언하지 않는다.

흔히 쓰이는 기법으로, 화살표 함수를 사용해 함수 정의 없이 다음과 같이 사용할수도 있다.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [text, setText] = useState("???");
  return (
    <>
      <div>{text}</div>
      <button
        onClick={() => {
          setText("짜잔");
        }}
      >
        클릭
      </button>
    </>
  );
}
```

- 사실, 이벤트핸들러를 별도의 함수로 분리하는것은 권장되지 않는 방법이다. 만약 버튼이 두개라면 둘 다 `handleClick` 이라는 이름의 이벤트핸들러를 사용해야 되는데, 두 버튼의 용도가 완전히 다를 것이기 때문이다. 이 경우, `Button` 컴포넌트로 분리한 후, `props` 로 각각의 이벤트핸들러를 내려주는 기법을 사용한다.

순수 JavaScript 로 작성된 코드와 비교해보자.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div>???

```

```
</script>
</body>
</html>
```

과정은 다음과 같다.

- 두 개의 DOM 요소를 `querySelector()` 로 각각 가져옴
- `<button>` 에, `addEventListener()` 를 사용해 이벤트 등록
- `<h1>` 의 `textContent` 에 접근해 값 변경

생각나는 단점은 다음과 같다.

- 이벤트와 DOM 에 대한 전문성 필요
 - 이 경우, `textContent` 속성을 변경해야 한다는 것을 정확히 알고 있어야 함
 - 만약 다른 속성을 변경해야 한다면 추가적인 학습 필요
- React 코드에 비해 가독성 떨어짐
 - 화면에서 변경될 데이터, 클릭 시 작동할 이벤트 핸들러 한눈에 파악 어려움
 - 변수와 함수가 많아질수록 생산성 떨어짐
- 매번 DOM 제어가 이루어지기에 성능 하락
 - React 는 Virtual DOM 사용하고, 모든 DOM 제어 이후 단 한번만 렌더링 (추후 학습 예정)

도전: 로또 번호 생성



- 1 부터 45 사이의 번호를 랜덤으로 생성

구현을 위해 참고할 코드는 다음과 같다.

```
// 1 부터 45 사이 난수 생성
const num = Math.floor(Math.random() * 45) + 1;
```

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [lottoNum, setLottoNum] = useState(null);
  return (
    <>
      <h1>로또 번호 생성기</h1>
      <button
        onClick={() => {
          const num = Math.floor(Math.random() * 45) + 1;
          setLottoNum(num);
        }}
      >
        번호 생성
      </button>
      <div>{lottoNum}</div>
    </>
  );
}
```

4-4. Curly Braces in Attribute

이번엔 중괄호를 활용해, 태그의 attribute 를 변수화해보자.

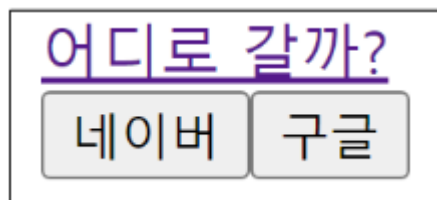
```
<a href="https://google.com">어디로 갈까?</a>
```

특정 버튼을 누르면 해당 링크는 네이버로 갈 수 있고, 다른 버튼을 누르면 구글로 갈 수 있도록 만드는 상황을 가정하자. 만약 `href` 가 정해져있다면 불가능한 일이지만, `href` 를 아래 코드처럼 변수화한다면 가능하다.

이번에도 중괄호를 사용한다.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [url, setUrl] = useState("");
  return (
    <>
      <a href={url}>어디로 갈까?</a>
      <div>
        <button
          onClick={() => {
            setUrl("https://naver.com");
          }}
        >
          네이버
        </button>
        <button
          onClick={() => {
            setUrl("https://google.com");
          }}
        >
          구글
        </button>
      </div>
    </>
  );
}
```



이번엔 색깔 변경을 연습해보자. `style` 역시 attribute 의 일종이므로, `style` 객체의 `color` property 에 접근해 값을 설정하면 된다.

- 만약 `background-color` 라면, camelCase 를 사용해 `backgroundColor` 로 접근한다.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [selectedColor, setSelectedColor] = useState("red");
  return (
    <>

```

```

    <h1 style={{ color: selectedColor }}>color</h1>
  </>
);
}

```



- 여기서 `{{ }}` 라고 두 개의 중괄호가 쓰였는데, 이는 별도의 문법이 아니다. React 문법인 Curly Braces `{ }` 를 사용 후, `style` 객체 `{ }` 에 접근한 것이다.

이 코드를 순수 JavaScript 로 변경하면 다음과 같다. 확인 후, React 코드와 비교해보았을 때 어떤 단점이 있는지 생각해보자.

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>color</h1>
    <script>
      const h1 = document.querySelector("h1");
      h1.style.color = input.value;
    </script>
  </body>
</html>

```

4-5. <input> value to State

이번엔 사용자 입력값을 변수로 받아볼 것이다. 다음 코드를 확인해보자.

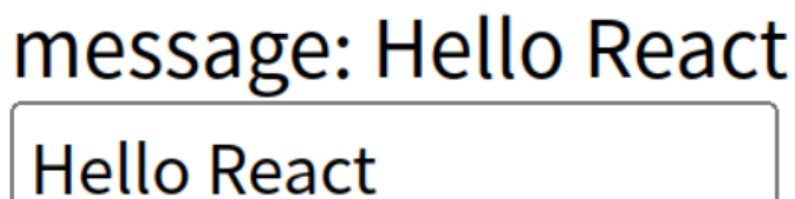
```

import './App.css';
import { useState } from 'react';

```



```
export default function App() {
  const [message, setMessage] = useState("");
  return (
    <>
      <div>message: {message}</div>
      <input
        value={message}
        onChange={(e) => setMessage(e.target.value)}
      />
    </>
  );
}
```



`<input>` 태그의 `value` attribute 를 `message` 변수와 연결시켰다. 이번에 등장할 이벤트는 바로 `change` 이며, `e` 라는 argument 하나를 받고, `e.target.value` 에 접근해 해당 값으로 `message` 변수를 갱신하며, 입력과 동시에 글자가 바뀐다.

- `e` 는 이벤트 객체를 말한다. 이벤트 발생 시점에, 해당 이벤트에 대한 모든 데이터가 포함되어있다. 궁금하면 `console.log(e)` 를 찍어보면서 어떤 데이터들이 있는지 확인해보자.
- `<input>` 태그는 HTML에선 닫아줄 필요가 없었지만, React에선 반드시 self closing 을 사용해 `<input />` 처럼 닫아줘야한다.

`<input>` 은 `text` 가 기본이지만, 자주 사용하는 `<input>` type 도 알아둘 필요가 있다.

- `text`
- `password`
- `checkbox`
- `radio`
- `number`
- `<input>` 은 아니지만, `<select>` 도 가능

도전1: 회원가입 양식

- 아이디, 비밀번호 입력창 오른쪽에 입력 값 즉시 출력
- 체크박스 오른쪽에 체크 여부 boolean 출력
 - `checkbox` 의 경우, `value` 가 아니라 `checked` 에 접근해야하며, `true` 는 HTML 에서 표시가 안 되므로 백틱 ``` 을 사용할 것.
- 라디오버튼 아래에 현재 값 출력
- `<select>` 아래에 현재 값 출력

testid

testpw

☐ false

☒ 빨강 ☐ 파랑

red

직업

student

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [userId, setUserId] = useState("");
```

```

const [userPw, setUserPw] = useState("");
const [isAgreed, setIsAgreed] = useState(false);
const [userColor, setUserColor] = useState("not selected");
const [userJob, setUserJob] = useState("student");
return (
  <>
    <div>
      <label>
        <input
          type="text"
          placeholder="아이디"
          value={userId}
          onChange={(e) => setUserId(e.target.value)}
        />
        {userId}
      </label>
    </div>
    <div>
      <label>
        <input
          type="password"
          placeholder="비밀번호"
          value={userPw}
          onChange={(e) => setUserPw(e.target.value)}
        />
        {userPw}
      </label>
    </div>
    <div>
      <label>
        <input
          type="checkbox"
          checked={isAgreed}
          onChange={(e) => setIsAgreed(e.target.checked)}
        />
        {`${isAgreed}`}
      </label>
    </div>
    <br />
    <label>
      <input
        type="radio"
        name="color"
        value="red"
        onChange={(e) => setUserColor(e.target.value)}
      />
      빨강
    </label>
    <label>
      <input
        type="radio"
        name="color"
        value="blue"
        onChange={(e) => setUserColor(e.target.value)}
      />
      파랑
    </label>
  </div>{userColor}</div>

```

```

    <br />
    <label>
      직업
      <select
        onChange={(e) => setUserJob(e.target.value)}
      >
        <option value="student">학생</option>
        <option value="teacher">교사</option>
      </select>
    </label>
    <div>{userJob}</div>
  </>
);
}

```

`<label>` 태그 사용 시, React에선 `<label>` 이 `<input>` 을 감싸는 게 일반적이다.

도전2: 입력값으로 색 변경

- 사용자 입력값을 받아 색 변경

color

color

정답코드

```
import "../App.css";
import { useState } from "react";

export default function App() {
  const [selectedColor, setSelectedColor] = useState("black");
  return (
    <>
      <h1 style={{ color: selectedColor }}>color</h1>
      <input
        type="text"
        value={selectedColor}
        onChange={(e) => setSelectedColor(e.target.value)}
      />
    </>
  );
}
```

4-6. Conditional Rendering

조건에 일치하면 태그를 보여준다. 총 세 가지의 방법을 배운다.

1. 조건이 복잡할 때 - `if` , `else if` , `else`
2. 조건이 간단할 때 - `?` `:`
3. 오로지 `true` 일 때만 보여줄 때 - `&&`

먼저, `if` , `else if` , `else` 를 사용하는 방법이다.

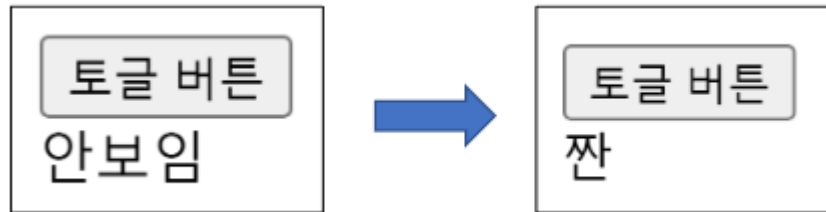
```
import "../App.css";
import { useState } from "react";

export default function App() {
  const [isActive, setIsActive] = useState(false);
  let content;
  if (isActive) {
    content = <div>짤</div>;
  } else {
    content = <div>안보임</div>;
  }
  return (
    <>
      <button
        onClick={() => setIsActive(!isActive)}
      >
        토글 버튼
      </button>
    </>
  );
}
```

```

    </button>
    {content}
  </>
);
}

```



누를때마다 토글되는것을 확인할 수 있다.

먼저, `content` 를 `let` 으로 선언한 것을 확인할 수 있다. `const` 가 아닌 이유는 당연하게도, `isActive` 에 따라 `content` 는 달라져야 하기 때문이다.

그리고 클릭 이벤트 발생 시, `isActive` 가 `true` 인 경우 `false` 로, `false` 인 경우 `true` 로 변경하여 서로 다른 화면을 보이게 만든다.

그러나, 조건 자체가 길지 않기에, 두 번째 방법인 `? :` 를 사용하기 적합하다. 한국어로 삼항연산자라고 하며, 영어로는 Conditional Operator 라고 한다.

코드를 개선해보자.

```

import './App.css';
import { useState } from "react";

export default function App() {
  const [isActive, setIsActive] = useState(false);
  return (
    <>
      <button
        onClick={() => setIsActive(!isActive)}
      >
        토글 버튼
      </button>
      {isActive ? <div>짤</div> : <div>안보임</div>}
    </>
  );
}

```

기존의 `content` 부분을 아예 빼버리고 `? :` 구문으로 변경했으며, 클릭 이벤트핸들러 부분도 마찬가지로 변경했다.

사용법은 다음과 같다.

조건 ? 참일때 : 거짓일때

세번째는 `&&` 사용이다. 만약 예제에서 `안보임` 일때의 태그는 존재하지 않고, `짤` 일때만 존재한다면, 즉 `false` 일때는 아예 보이지 않고 `true` 일 때만 보인다면 다음과 같이 작성할 수 있다.

```
import "./App.css";
import { useState } from "react";

export default function App() {
  const [isActive, setIsActive] = useState(false);
  return (
    <>
      <button
        onClick={() => setIsActive(!isActive)}
      >
        토글 버튼
      </button>
      {isActive && <div>짤</div>}
    </>
  );
}
```

`&&` 는 논리연산자이며, `isActive` 가 참이면 렌더링한다.

흔히 하는 실수 하나만 잡고 넘어가자. 장바구니 기능을 생각해보면, 장바구니에 물품이 하나도 없을때는 화면에 아무것도 안 보이게 하고싶을 수 있다.

```
import "./App.css";
import { useState } from "react";

export default function App() {
  const [orderCount, setOrderCount] = useState(0);
  return (
    <>
      {orderCount && <div>물건이 있습니다.</div>}
    </>
  );
}
```

이 경우, 아무것도 안 보일 것이라 기대하지만 화면에 숫자 `0` 이 표시된다. 즉, 좌항의 결과가 숫자가 되어서는 안되며, boolean 이 되어야 한다.

다음과 같이 수정할 경우, 정상 작동한다.

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [orderCount, setOrderCount] = useState(0);
  return (
    <>
      {orderCount > 0 && <div>물건이 있습니다.</div>}
    </>
  );
}
```

도전: A B C

A B C 입력

- A 입력 시, A 출력
- B 입력 시, B 출력
- C 입력 시, C 출력
- A B C 가 아닐 경우, "A B C 가 아닙니다" 출력

A
A

B
B

C
C

D
A B C 가 아닙니다

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [userInput, setUserInput] = useState("");
  let content;
  if (userInput === "A") {
    content = <div>A</div>;
  } else if (userInput === "B") {
    content = <div>B</div>;
  } else if (userInput === "C") {
    content = <div>C</div>;
  } else {
    content = <div>A B C 가 아닙니다</div>;
  }
  return (
    <>
      <input
        type="text"
        placeholder="A B C"
        value={userInput}
        onChange={(e) => setUserInput(e.target.value)}
      />
      {content}
    </>
  );
}
```

4-7. Rendering Lists

태그를 반복할 때는 `map()`, `filter()` 배열 메서드를 사용한다.

```
import './App.css';

export default function App() {
  const menus = [
    {
      id: 1,
      name: "짜장면",
      isRecommend: true,
    },
    {
      id: 2,
      name: "짬뽕",
      isRecommend: false,
    },
    {
      id: 3,
```

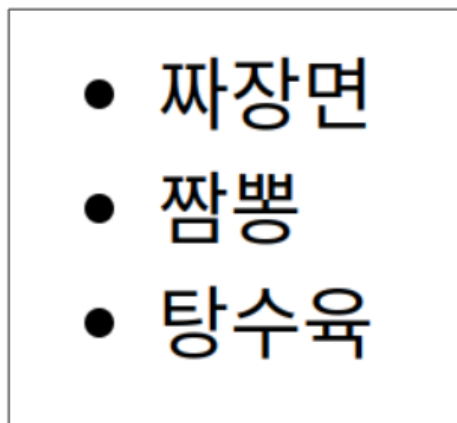
```

        name: "탕수육",
        isRecommend: true,
      },
    ];
    return (
      <>
        <ul>
          {menus.map((menu) => (
            <li key={menu.id}>{menu.name}</li>
          ))}
        </ul>
      </>
    );
  }
}

```

`useState` 를 사용하지 않았음에 유의하자. 변경이 일어나지 않는 변수는 state 로 지정해선 안된다.

`map()` 메서드를 사용해 `menus` 각각을 `menu` 로 받고, 이름을 출력하는것을 확인할 수 있다.



- `key` 는 반드시 지정해야 한다. 배열은 DB 에서 가져오는게 보통이며, 대부분 객체 배열로 이루어져있고, 반드시 `id` 가 존재하므로 `id` 를 `key` 로 설정해야 한다.

만약, `isRecommend` 가 `true` 인 데이터만 출력하고 싶다면 `filter()` 메서드를 사용하는것이 적절하다.

```

import './App.css';

export default function App() {
  const menus = [
    {
      id: 1,
      name: "짜장면",
      isRecommend: true,
    },
  ],

```

```

    {
      id: 2,
      name: "짬뽕",
      isRecommend: false,
    },
    {
      id: 3,
      name: "탕수육",
      isRecommend: true,
    },
  ];
  const recommendedMenus = menus.filter((menu) => menu.isRecommend);
  return (
    <>
      <ul>
        {recommendedMenus.map((menu) => (
          <li key={menu.id}>{menu.name}</li>
        ))}
      </ul>
    </>
  );
}

```

`filter()` 를 사용해 새로운 배열을 만든 후, `map()` 으로 렌더링한다.

5. 응용문제

문제1: BMI 계산기

- `<input type="number">`
- BMI = 체중 / (신장 * 신장)
- 다음 조건에 따른 결과를 출력
 - BMI < 18.5 : 저체중
 - 18.5 <= BMI < 23 : 정상
 - 23 <= BMI < 25 : 과체중
 - 25 <= BMI : 비만
 - NaN : 계산 불가

BMI 계산기

신장:

체중:

정상

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [height, setHeight] = useState(0);
  const [weight, setWeight] = useState(0);
  let bmiResult;
  const heightMeter = height / 100;
  const bmi = weight / (heightMeter * heightMeter);
  if (bmi < 18.5) bmiResult = <div>저체중</div>;
  else if (bmi < 23) bmiResult = <div>정상</div>;
  else if (bmi < 25) bmiResult = <div>과체중</div>;
  else if (bmi >= 25) bmiResult = <div>비만</div>;
  else bmiResult = <div>계산불가</div>;
  return (
    <>
      <h1>BMI 계산기</h1>
      <div>
        <label>
          신장:
          <input
            type="number"
            value={height}
            onChange={(e) => setHeight(e.target.value)}
          />
        </label>
      </div>
      <div>
        <label>
          체중:
          <input
            type="number"
            value={weight}
            onChange={(e) => setWeight(e.target.value)}
          />
        </label>
      </div>
      {bmiResult}
    </>
  );
}
```

```
);  
}
```

문제2: hideCompleted

제시된 코드 개선

```
import './App.css';  
import { useState } from 'react';  
  
export default function App() {  
  const [hideCompleted, setHideCompleted] = useState(false);  
  const todos = [  
    { id: 1, text: "HTML 배우기", done: true },  
    { id: 2, text: "JavaScript 배우기", done: true },  
    { id: 3, text: "React 배우기", done: false },  
  ];  
  
  return (  
    <>  
      <ul>  
        {todos.map((todo) => (  
          <li key={todo.id}>  
            <span>{todo.text}</span>  
          </li>  
        ))}  
      </ul>  
      <button onClick={() => setHideCompleted(!hideCompleted)}>토글</button>  
    </>  
  );  
}
```

버튼을 누르면, 두 가지 결과 토글

- `done: true` 만 출력
- 전체 출력

- HTML 배우기
- JavaScript 배우기
- Vue 배우기

토글

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [hideCompleted, setHideCompleted] = useState(false);
  const todos = [
    { id: 1, text: "HTML 배우기", done: true },
    { id: 2, text: "JavaScript 배우기", done: true },
    { id: 3, text: "React 배우기", done: false },
  ];

  const filteredTodos = hideCompleted
    ? todos.filter((todo) => !todo.done)
    : todos;

  return (
    <>
      <ul>
        {filteredTodos.map((todo) => (
          <li key={todo.id}>
            <span>{todo.text}</span>
          </li>
        ))}
      </ul>
      <button onClick={() => setHideCompleted(!hideCompleted)}>토글</button>
    </>
  );
}
```

문제3: 섭씨를 화씨로

화씨 = (섭씨 * 1.8) + 32

입력과 동시에 결과 확인

섭씨

화씨

212

정답코드

```
import './App.css';
import { useState } from 'react';

export default function App() {
  const [c, setC] = useState(0);

  return (
    <>
      <h1>섭씨</h1>
      <div>
        <input type="number" value={c} onChange={(e) => setC(e.target.value)} />
      </div>
      <h1>화씨</h1>
      <div>{c * 1.8 + 32}</div>
    </>
  );
}
```

문제4: !dlroW olleH

- Reverse Message 클릭 시
 - 문자를 역순으로 출력
 - split, reverse, join 메서드 활용
- Append "!" 클릭 시
 - 글자 맨 뒤에 ! 붙이기
 - 역순일 경우, 역순 글자 맨 뒤에 ! 붙이기



정답코드

```
import "../App.css";
import { useState } from "react";
```



```
export default function App() {
  const [message, setMessage] = useState("Hello World!");
  return (
    <>
      <h1>{message}</h1>
      <div>
        <button
          onClick={() => setMessage(message.split("").reverse().join(""))}
        >
          Reverse Message
        </button>
      </div>
      <div>
        <button onClick={() => setMessage(message + "!")}>Append "!"</button>
      </div>
    </>
  );
}
```