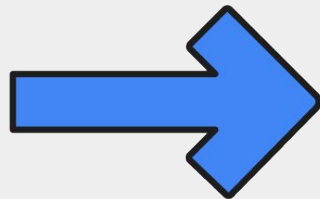




Google Developer Group
Editable University Name

Web & React

GDG on INU Front Core 최수환



웹 개발의 개념과 역사

팀 버너스 리와 월드 와이드 웹의 시작

HTML, CSS, JavaScript의 탄생 & **AJAX** 등장

반응형 웹, **SPA** 등 다양한 형태

웹 개발의 개념과 역사

팀 버너스 리와 월드 와이드 웹의 시작

HTML, CSS, JavaScript의 탄생 & **AJAX** 등장

반응형 웹, **SPA** 등 다양한 형태



웹 개발의 개념과 역사

팀 버너스 리와 월드 와이드 웹의 시작

HTML, CSS, JavaScript의 탄생 & **AJAX** 등장

반응형 웹, **SPA** 등 다양한 형태



웹 개발의 개념과 역사

팀 버너스 리와 월드 와이드 웹의 시작

HTML, CSS, JavaScript의 탄생 & **AJAX** 등장

반응형 웹, **SPA** 등 다양한 형태

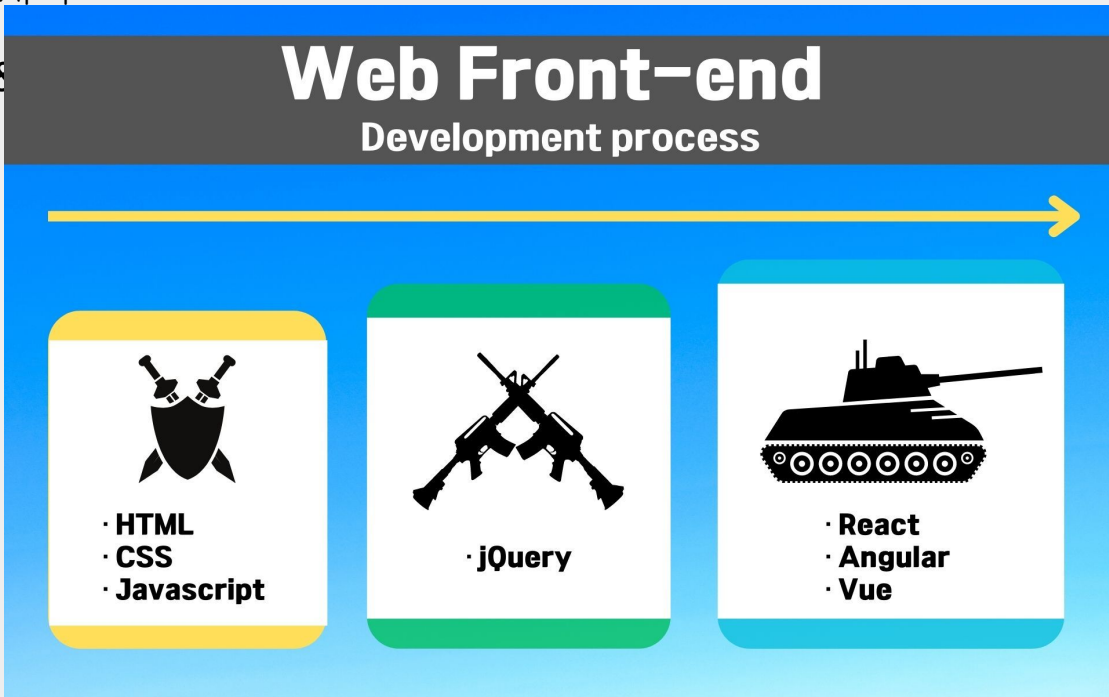


웹 개발의 개념과 역사

팀 버너스 리와 월드 와이드 웹의 시작

HTML, CSS, JavaScript의 탄생 &

반응형 웹, **SPA** 등 다양한 형태



웹의 기술적 원리

서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript, Dom Tree

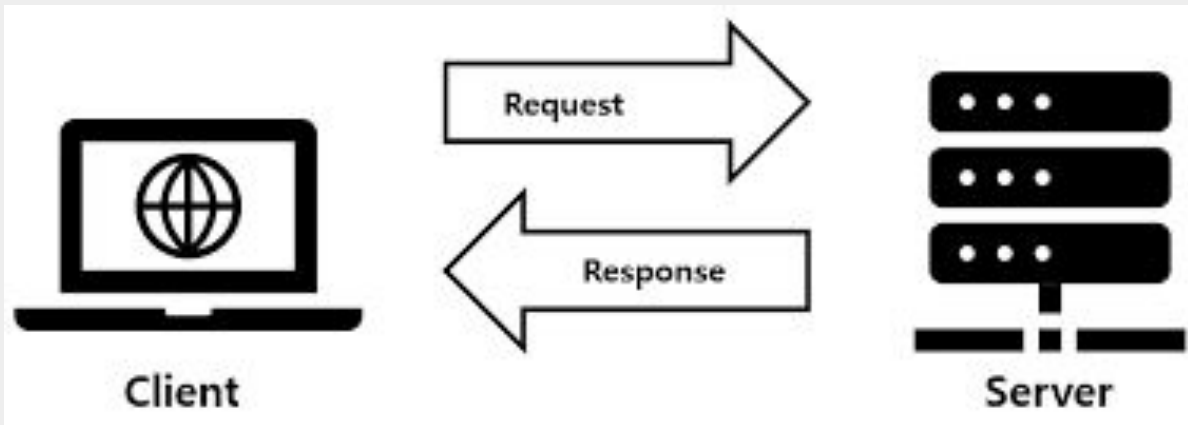
HTTP/HTTPS

웹의 기술적 원리

서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript, Dom Tree

HTTP/HTTPS



웹의 기술적 원리

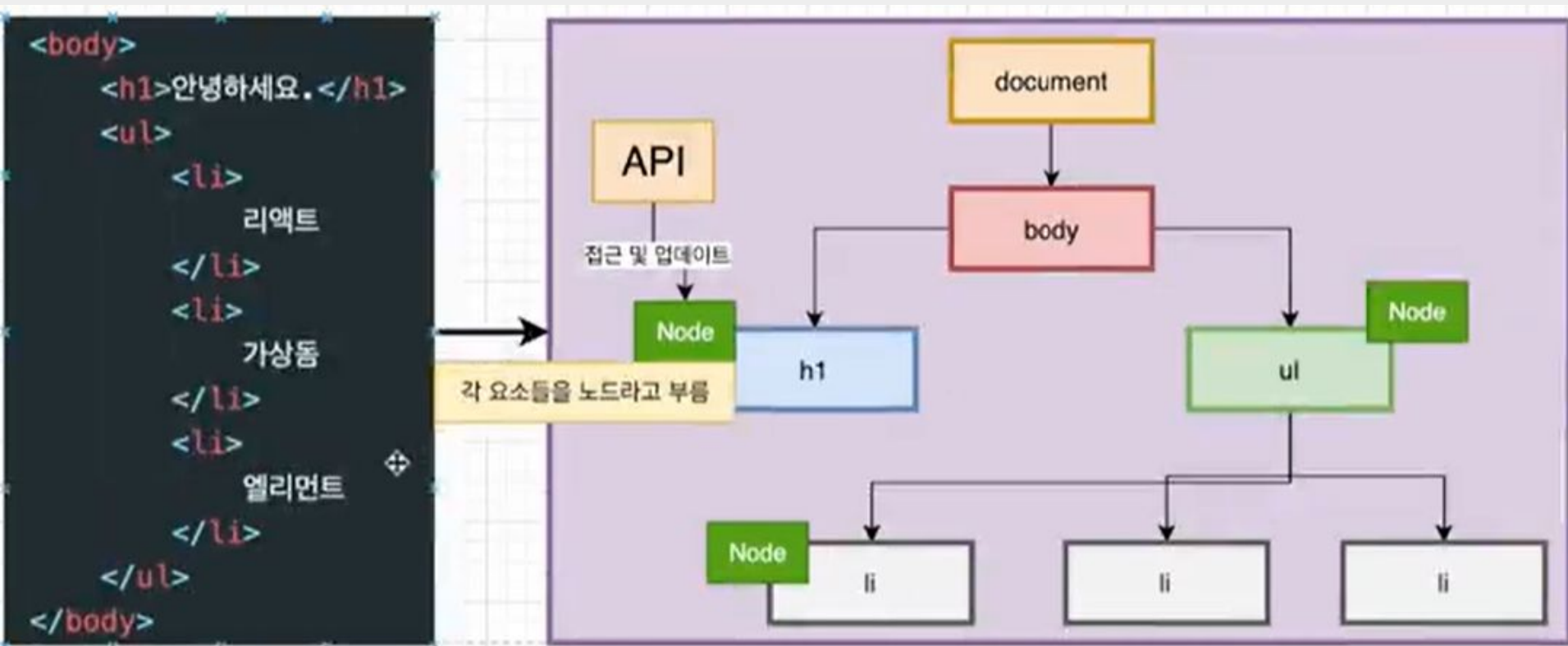
서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript, Dom Tree

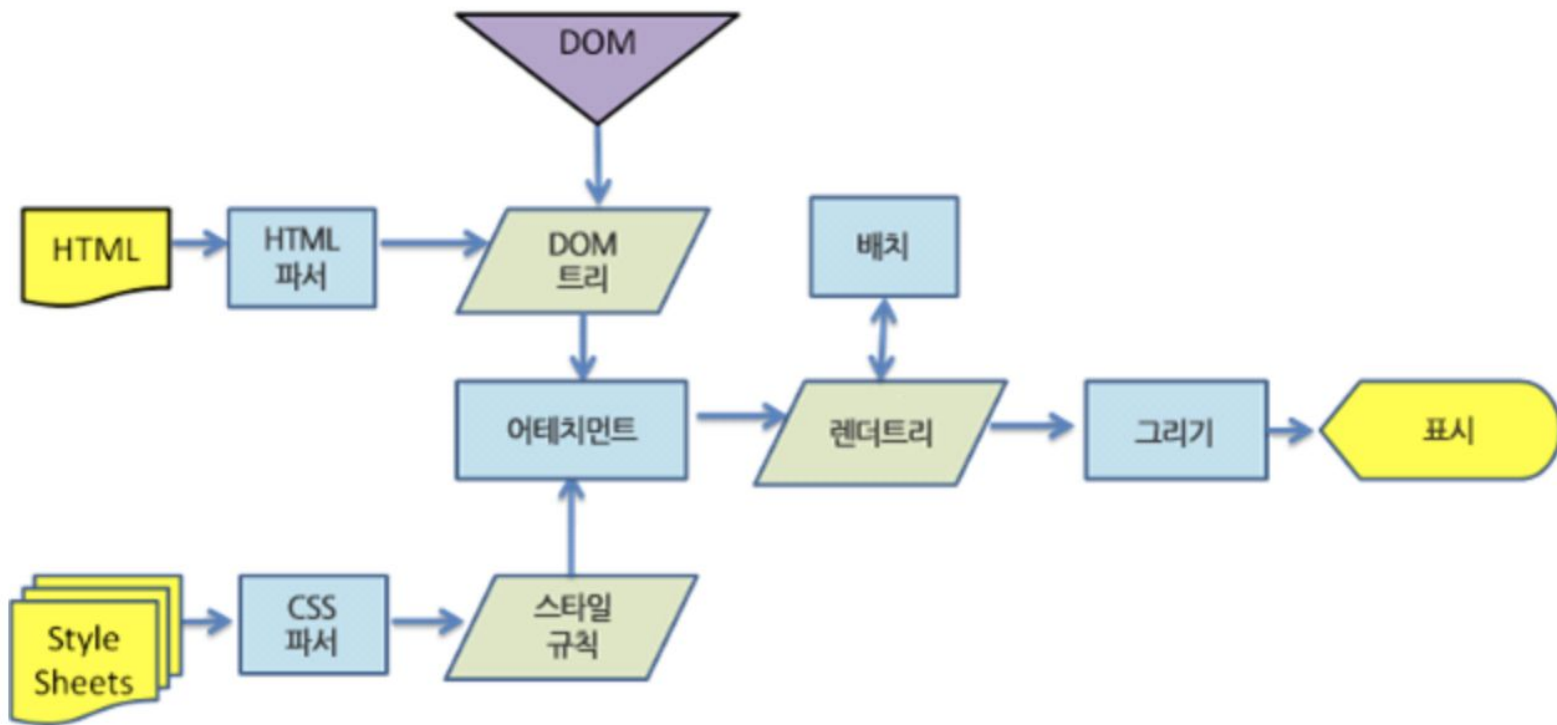
HTTP/HTTPS

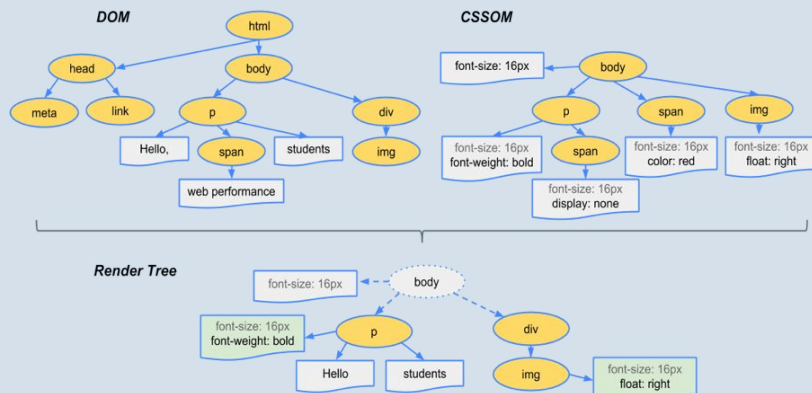
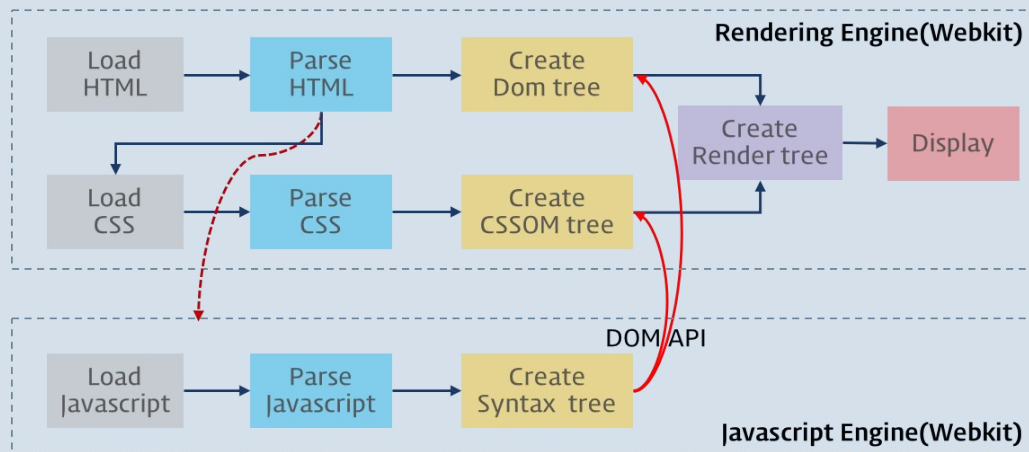
웹의 기술적 원리

Dom Tree

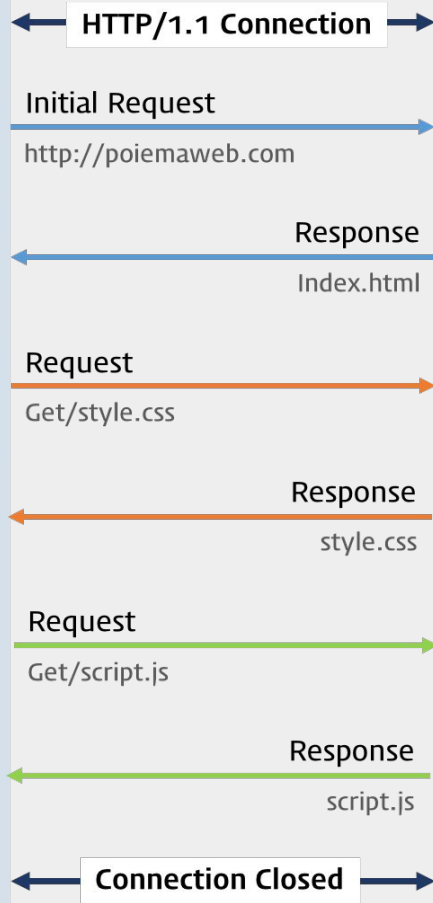


웹의 기술적 원리





Client



node.js
express



© poiemaweb.com

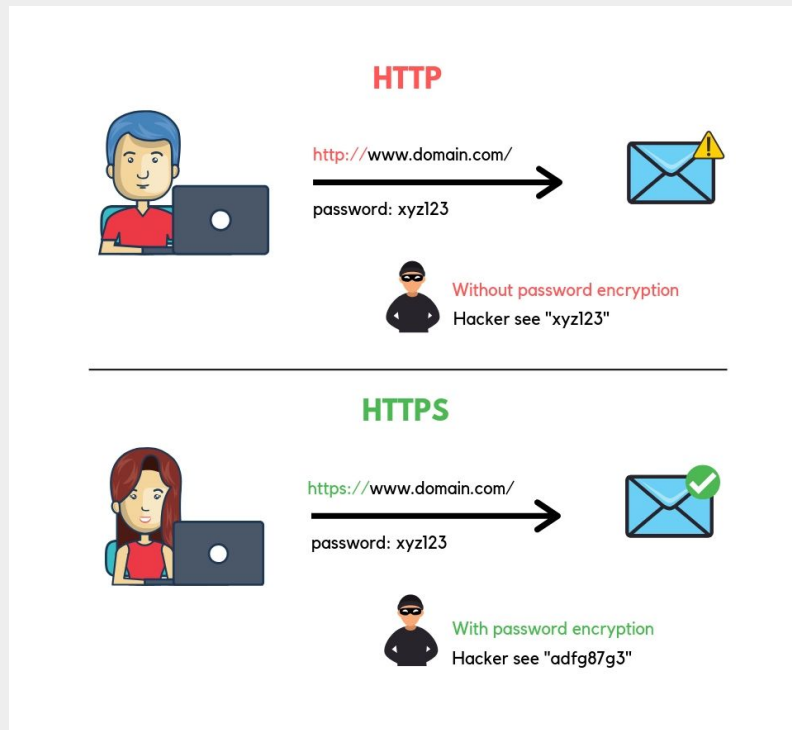
Server

웹의 기술적 원리

서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript

HTTP/HTTPS

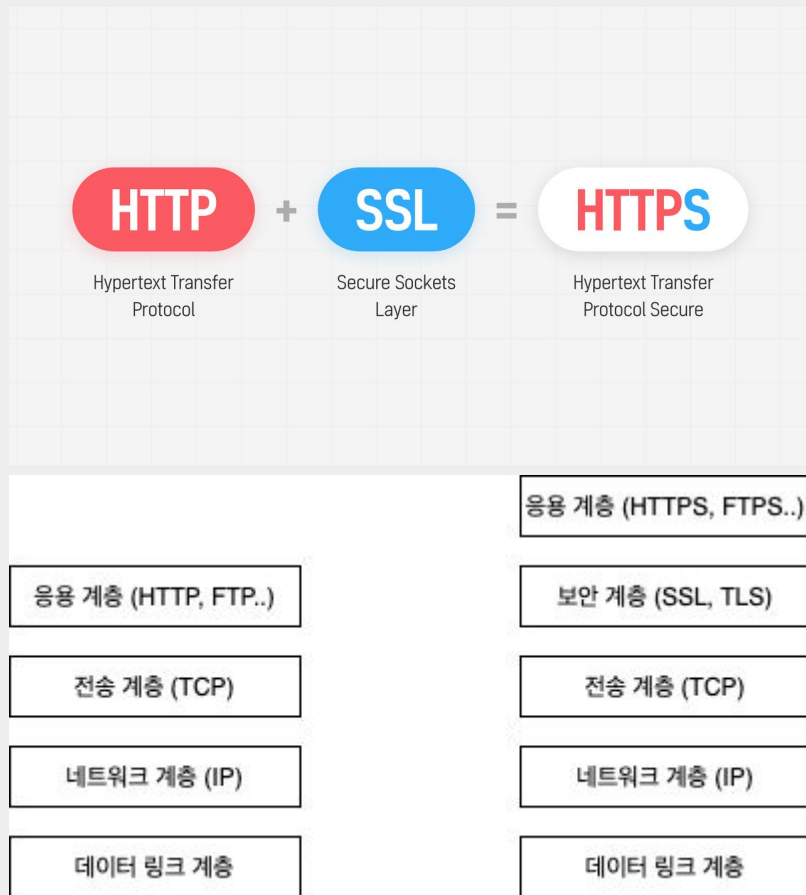


웹의 기술적 원리

서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript

HTTP/HTTPS



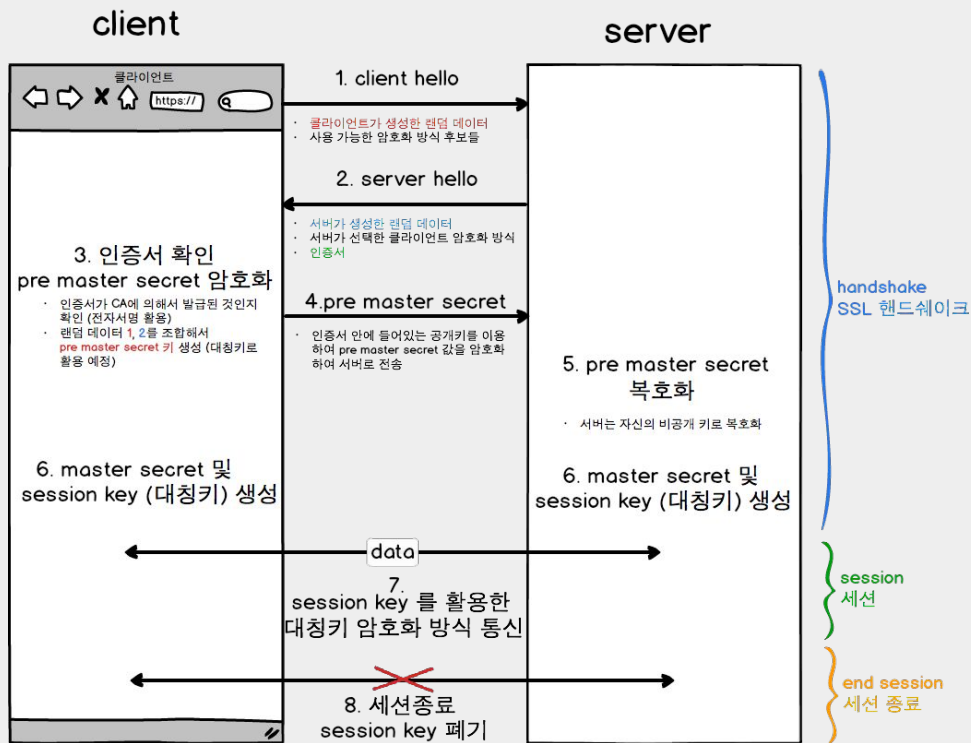
웹의 기술적 원리

SSL 통신과정

서버 - 클라이언트, 요청과 응답의 기본 구조

HTML, CSS, JavaScript

HTTP/HTTPS



리액트(React)의 소개

페이스북 개발 - 사용자 인터페이스(UI) 구축에 최적화

컴포넌트 기반 아키텍처 제공

복잡한 웹 애플리케이션 구조 단순화

빠르고 효율적인 개발 도구

리액트(React)의 소개

페이스북 개발 - 사용자 인터페이스(UI) 구축에 최적화

컴포넌트 기반 아키텍처 제공

복잡한 웹 애플리케이션 구조 단순화

빠르고 효율적인 개발 도구



리액트(React)의 소개

페이스북 개발 - 사용자 인터페이스(UI) 구축에 최적화

컴포넌트 기반 아키텍처 제공

복잡한 웹 애플리케이션 구조 단순화

빠르고 효율적인 개발 도구

리액트(React)의 소개

페이스

컴포넌

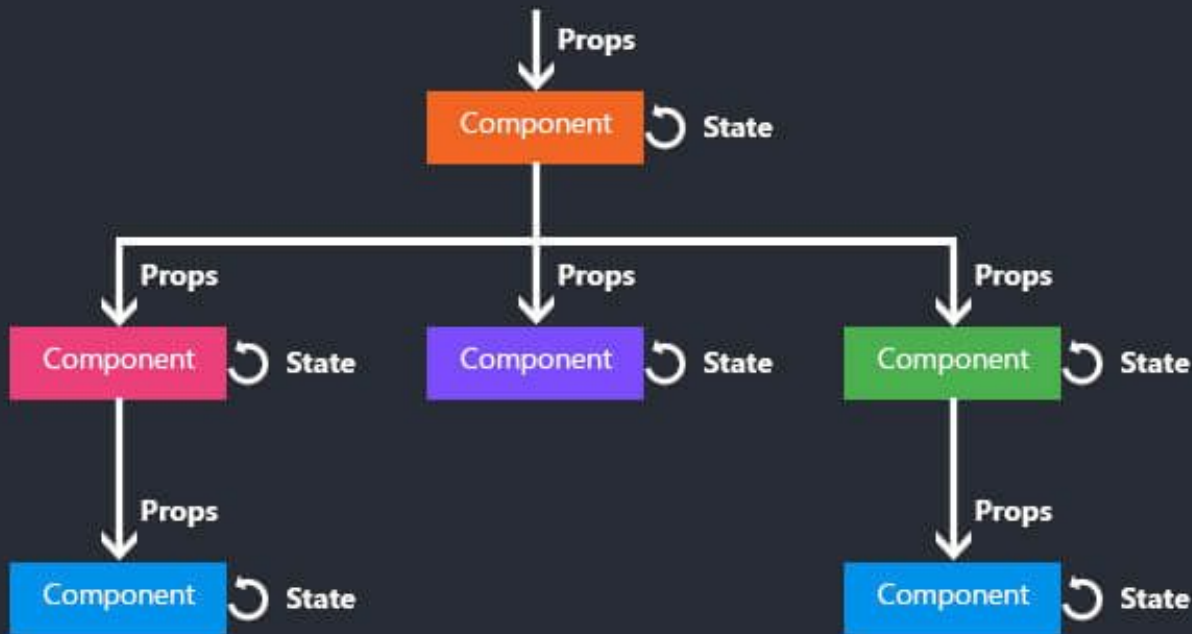
복잡한

빠르고



리액트(React)의 소개

컴포넌트 기반 아키텍처 제공, prop, state



ReactJS
Component State

리액트(React)의 소개

컴포넌트 기반 아키텍처 제공, prop, state

```
const Button = ({ children, onClick, color = 'primary' }) => {  
  return (  
    <button  
      className={`button ${color}`}  
      onClick={onClick}  
    >  
      {children}  
    </button>  
  );  
}  
  
export default Button;
```

리액트의 기술적 원리

컴포넌트 기반 UI 구성 – 재사용성과 유지보수 강화

Virtual DOM: 효율적 업데이트 및 성능 최적화

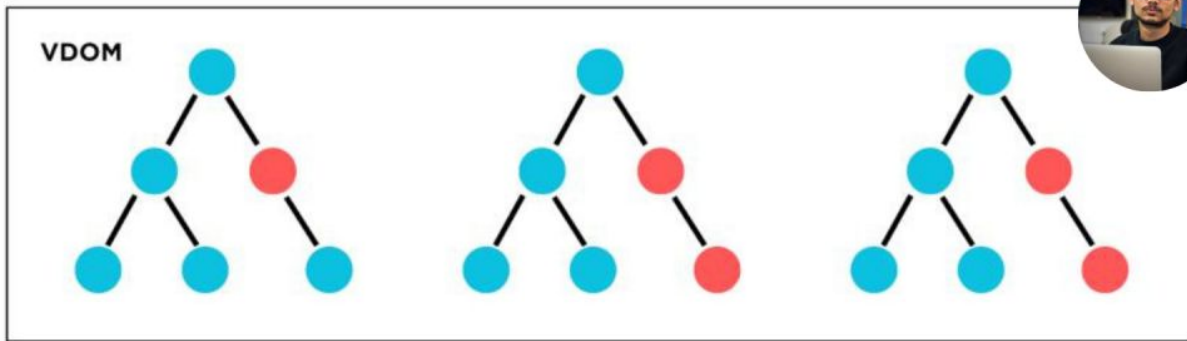
SPA vs MPA, SSR vs CSR

단방향 데이터 흐름: 예측 가능한 상태 관리

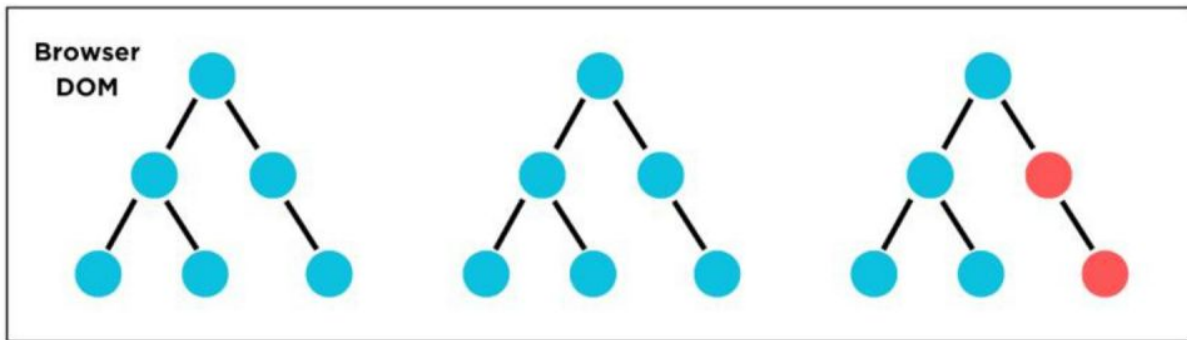
리액트의 기술적 원리

Virtual DOM

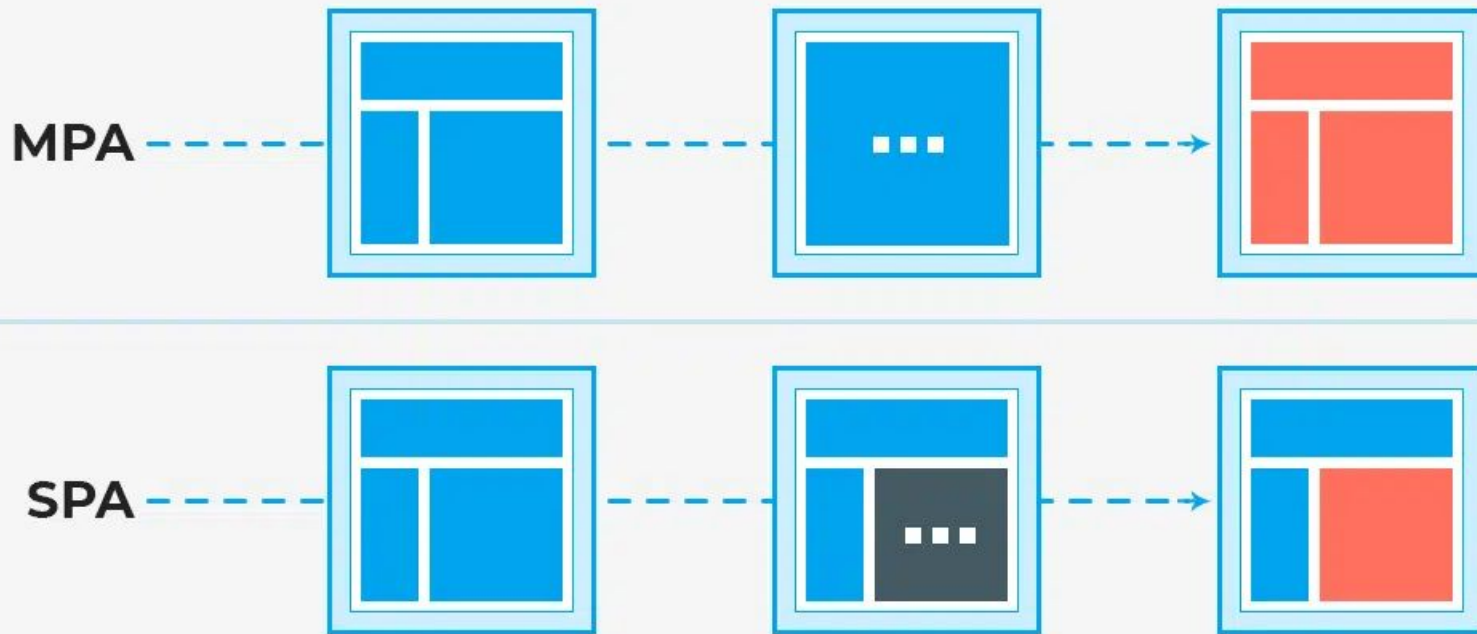
**VIRTUAL
DOM**



DOM



MPA vs SPA

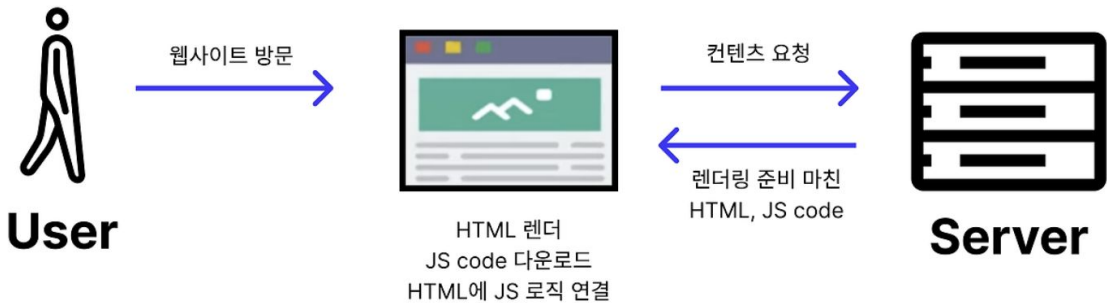


✓ CSR의 동작 과정



CSR 동작 과정


✓ SSR의 동작 과정




SSR 동작 과정


말 그대로 서버쪽에서 렌더링 준비를 끝마친 상태로 클라이언트에 전달하는 방식이다.



1. 
User Requests a Website


2. 
Server creates "Ready to Render" HTML files

3. 
The Browser can quickly render the HTML but the site isn't interactive

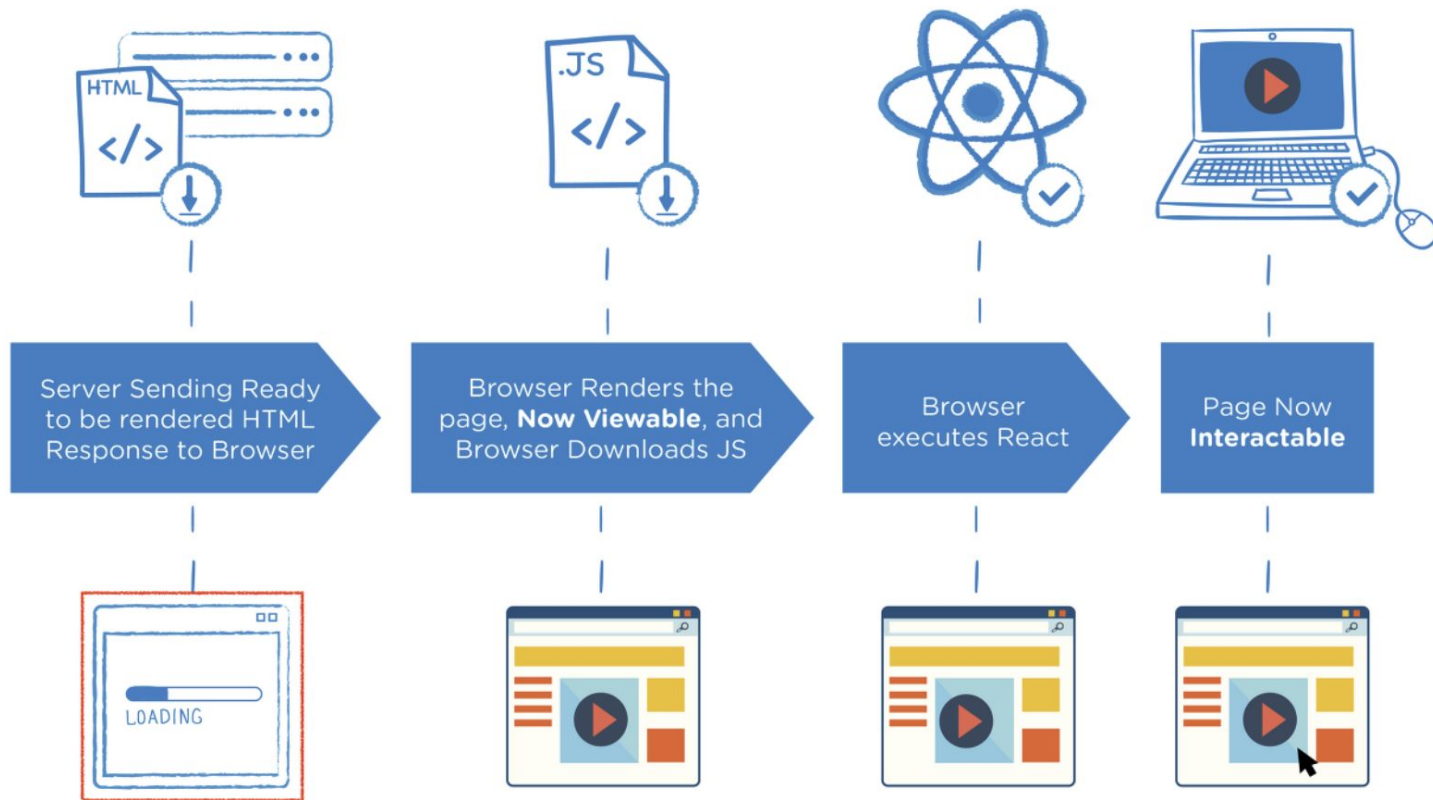
4. 
The Browser downloads the Javascript

5. 
The user can view content and the interactions can be recorded

6. 
The Browser Executes the JS Framework

7. 
The recorded interactions can be executed and the page is now interactive

SSR



리액트의 주요 기능 및 장점

JSX: 자바스크립트 내 XML 문법으로 직관적 UI 작성

Lifecycle Methods & Hooks: 유연한 컴포넌트 관리

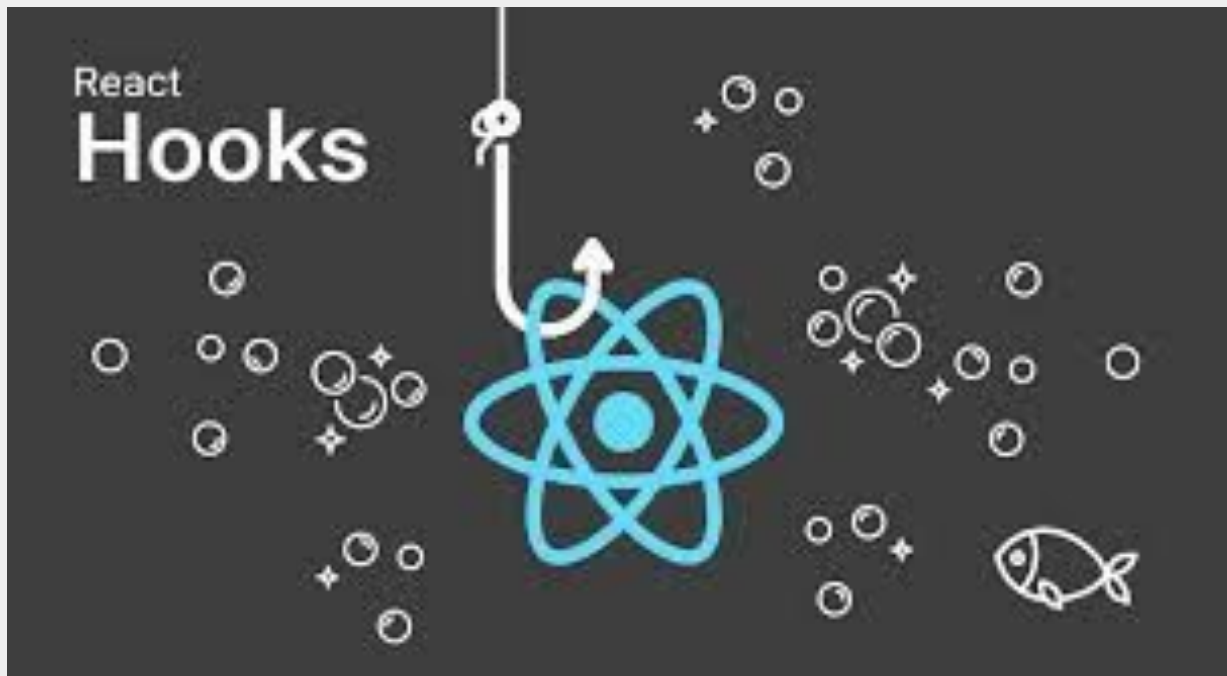
재사용성, 빠른 렌더링, 활발한 커뮤니티, 유연한 통합

리액트의 주요 기능 및 장점

JSX

```
const App = () => {  
  return (  
    <div className="App">  
      <Header />  
      <divider className="topDivider"></divider>  
      <main>  
        <Sidebar />  
        <Features />  
      </main>  
    </div>  
  );  
};
```

리액트의 주요 기능 및 장점



웹 개발과 리액트의 실제 활용 사례

전통적 웹 개발: 정적 웹사이트 & 동적 웹 애플리케이션

리액트: SPA 구축 및 빠른 UI 구현 (예: 페이스북, 인스타그램)

React Native: 모바일 앱 개발

기업/스타트업: 빠른 프로토타입 제작 및 확장성 확보

리액트 예제

Vite 활용 : CSR, 빠르고 간편함


```
npm create vite@latest react-memo-app -- --template react
cd react-memo-app
npm install
npm run dev
```

react-memo-app/

├─ src/

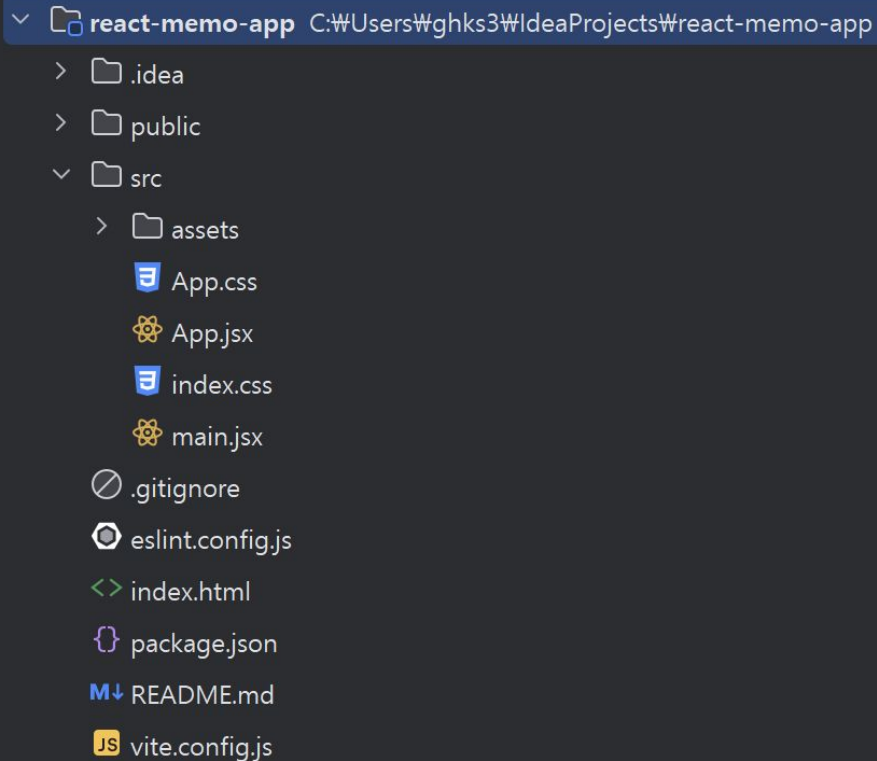
| └─ components/

| └─ MemoForm.jsx

| └─ MemoList.jsx

| └─ App.jsx

| └─ main.jsx



M↓ README.md



MemoList.jsx



App.jsx



main.jsx ×



MemoForm.jsx

```
1  > import ...
5
6  createRoot(document.getElementById(elementId: 'root')).render(
7    <StrictMode>
8    |   <App />
9    </StrictMode>,
10 )
11 |
```

```
1 import { useState } from 'react';
2
3 export default function MemoForm({ onAdd }) {  사용 위치 없음
4   const [text : string , setText] = useState( initialState: '' );
5
6   const handleSubmit = (e) : void => {  사용 위치 표시
7     e.preventDefault();
8     if (!text.trim()) return;
9     onAdd(text);
10    setText( value: '' );
11  };
12
13  return (
14    <form onSubmit={handleSubmit}>
15      <input
16        type="text"
17        value={text}
18        onChange={(e : ChangeEvent<HTMLInputElement> ) : void => setText(e.target.value)}
19        placeholder="메모를 입력하세요"
20      />
21      <button type="submit">추가</button>
22    </form>
23  );
24 }
25 |
```



```
1 export default function MemoList({ memos, onDelete }) { 사용 위치 없음
2   return (
3     <ul>
4       {memos.map((memo, index) => (
5         <li key={index}>
6           {memo}
7           <button onClick={() => onDelete(index)}>삭제</button>
8         </li>
9       ))}
10    </ul>
11  );
12 }
```

```

1 import { useState } from 'react';
2 import MemoForm from './components/MemoForm';
3 import MemoList from './components/MemoList';
4
5 function App() { 사용 위치 표시
6   const [memos : any[], setMemos] = useState( initialState: []);
7
8   const addMemo = (text) : void => { 사용 위치 표시
9     setMemos( value: [...memos, text]);
10  };
11
12  const deleteMemo = (index) : void => { 사용 위치 표시
13    setMemos(memos.filter((_, i : number ) : boolean => i !== index));
14  };
15
16  return (
17    <div>
18      <h1> 📝 메모장</h1>
19      <MemoForm onAdd={addMemo} />
20      <MemoList memos={memos} onDelete={deleteMemo} />
21    </div>
22  );
23 }
24
25 export default App; 사용 위치 표시

```



메모장

메모를 입력하세요

추가

- react 공부하기 삭제
- GDG 발표 준비하기 삭제



Node.js

→ JavaScript로 백엔드 서버를 만들 수 있게 해주는 런타임 환경

→ 예: 브라우저 없이도 JS 코드 실행 가능, 서버 만들기 가능

Express.js

→ Node.js 위에서 동작하는 웹 프레임워크

→ 라우팅, 미들웨어, 요청/응답 처리 등을 간단하고 효율적으로 구현할 수 있게 도와줌

```
import express from 'express';
import cors from 'cors';
import bodyParser from 'body-parser';
import fs from 'fs/promises'; // ☒ import
```

```
const app : Express = express();
const PORT : number = 5000;
const DB_FILE : string = './notes.json';
```

```
app.use(cors());
app.use(bodyParser.json());
```

```
app.get('/', async (req : Request<P, ResBody, ReqBody, Req  
  res.send( body: "server is running");  
});
```

```
✓ app.listen(PORT, hostname: () : void => {  
  console.log(`Server is running on http://localhost:${PORT}`);  
});
```



localhost:50

server is running

```
// Read notes
✓ app.get('/notes', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj>
✓   try {
      const data : string = await fs.readFile(DB_FILE, options: 'utf8');
      res.json(JSON.parse(text: data || '[]'));
    } catch (err) {
      console.error('Error reading notes:', err);
      res.status(code: 500).send(body: 'Error reading notes');
    }
  });

// Save notes
✓ app.post(path: '/notes', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<
✓   try {
      const notes : ReqBody = req.body;
      await fs.writeFile(DB_FILE, JSON.stringify(notes, replacer: null, space: 2));
      res.sendStatus(code: 200);
    } catch (err) {
      console.error('Error saving notes:', err);
      res.status(code: 500).send(body: 'Error saving notes');
    }
  });
```



```
33 import axios from 'axios';
34
35 ∨ function App() { 사용 위치 표시
36     const [notes : string[], setNotes] = useState( initialState: ['']);
37
38     ∨ useEffect( effect: () : void => {
39         ∨ axios.get('http://localhost:5000/notes').then(res => {
40             setNotes(res.data);
41         });
42     }, deps: []);
43
44     ∨ const handleChange = (e, idx) : void => { 사용 위치 표시
45         const newNotes : string[] = [...notes];
46         newNotes[idx] = e.target.value;
47         setNotes(newNotes);
48     };
49
50     const handleAddNote = () : void => setNotes( value: [...notes, '']); 사용 위치 표시
51
52     ∨ const handleSave = () : void => { 사용 위치 표시
53         axios.post('http://localhost:5000/notes', notes).then(() : void => alert('Saved!'));
54     };
55
```

```
return (  
  <div style={{ padding: '2rem' }}>  
    <h1>📝 My Notepad</h1>  
    {notes.map((note : string , idx : number ) => (  
      <textarea  
        key={idx}  
        value={note}  
        onChange={e : ChangeEvent<HTMLTextAreaElement> => handleChange(e, idx)}  
        rows={5}  
        cols={50}  
        style={{ display: 'block', marginBottom: '1rem' }}  
      />  
    ))}  
    <button onClick={handleAddNote}>Add Note</button>  
    <button onClick={handleSave} style={{ marginLeft: '1rem' }}>Save</button>  
  </div>  
);  
}
```

export default App; 사용 위치 표시



My Notepad

메모장 1

메모장 2

메모장 3

메모장 4

Add Note

Save

note.json

```
1  [  
2    "메모장 1 ",  
3    "메모장 2 ",  
4    "메모장 3",  
5    "메모장 4"  
6  ]
```

“



”