🌤️ OpenWeather 날씨 앱 - 완벽 시작 가이드

이 문서는 프로그래밍을 처음 시작하는 분들도 쉽게 따라할 수 있도록 모든 과정을 아주 자세하게 설명합니다.

📝 목차

---

1. 프로젝트 소개

🎯 이 프로젝트는 무엇인가요?

OpenWeather 날씨 앱은 **React**라는 현대적인 웹 프레임워크로 만들어진 **실시간 날씨 조회 웹 애플리케이션**입니다.

📱 주요 기능

| 기능 | 설명 | 예시 |

|------|------|------|

| 🔍 **도시 검색** | 전 세계 모든 도시의 날씨 조회 | Seoul, Tokyo, Paris |

| 🌡️ **온도 정보** | 현재 온도 및 체감 온도 표시 | 15°C (체감 13°C) |

| 💧 **습도** | 현재 습도 퍼센트 | 60% |

| 💨 **풍속** | 바람의 속도 | 3.5 m/s |

| 🎯 **기압** | 대기압 정보 | 1013 hPa |

| 🌤️ **날씨 아이콘** | 시각적인 날씨 상태 표시 | 🌞🌧️🌦️ |

| 📱 **반응형** | 모바일, 태블릿, PC 모두 지원 | 모든 화면 크기 |

🎓 학습 목표

이 프로젝트를 통해 배울 수 있는 것들:

- ✅ React 기본 개념 (컴포넌트, State, Props)
- ✅ React Hooks (useState, useEffect)
- ✅ 외부 API 연동 방법
- ✅ 비동기 프로그래밍 (async/await)
- ✅ HTTP 요청 처리 (axios)
- ✅ 에러 처리 및 로딩 상태 관리

- ✅ 현대적인 CSS 스타일링
- ✅ 반응형 웹 디자인

---

2. 사전 준비사항 - Node.js 설치

🤔 Node.js가 뭔가요?

Node.js는 JavaScript를 컴퓨터에서 실행할 수 있게 해주는 프로그램입니다.

- 웹 브라우저 밖에서도 JavaScript를 실행 가능
- React 앱을 개발하고 실행하는데 필수
- npm(Node Package Manager)이 함께 설치됨

📥 Node.js 설치 단계별 가이드

Windows 사용자

1단계: 다운로드

1. https://nodejs.org 접속
2. 왼쪽의 **LTS** 버전 클릭 (안정적인 버전)
   - 예: 20.10.0 LTS (추천)
3. 다운로드된 `.msi` 파일 실행

2단계: 설치

1. 설치 마법사 실행

2. "Next" 클릭

3. 라이선스 동의 체크 → "Next"

4. 설치 경로 확인 (기본값 사용 권장) → "Next"

5. "Automatically install necessary tools" 체크 → "Next"

6. "Install" 클릭

7. 설치 완료 후 "Finish"

3단계: 설치 확인

1. **Windows 키 + R** 누르기

2. `cmd` 입력 후 Enter (명령 프롬프트 실행)

3. 다음 명령어 입력:

   ```bash

   node --version

   ```

   출력 예시: `v20.10.0`

4. npm 버전도 확인:

   ```bash

   npm --version

   ```

   출력 예시: `10.2.3`

✅ 버전 번호가 표시되면 설치 성공!

macOS 사용자

방법 1: 공식 설치 프로그램

1. https://nodejs.org 접속

2. LTS 버전 다운로드 (.pkg 파일)

3. 다운로드된 파일 더블클릭

4. 설치 마법사 따라가기

5. 터미널에서 `node --version` 확인

방법 2: Homebrew 사용 (추천)

```bash

*# Homebrew가 없다면 먼저 설치*

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

*# Node.js 설치*

brew install node

*# 설치 확인*

node --version

npm --version

```

Linux 사용자 (Ubuntu/Debian)

```bash

*# NodeSource 저장소 추가*

curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
```

 # Node.js 설치

sudo apt-get install -y nodejs

 # 설치 확인

node --version

npm --version

```

### 🔗 npm이란?

**npm (Node Package Manager)**는 JavaScript 패키지 관리자입니다.

#### npm의 역할

- 📦 다른 개발자들이 만든 코드(패키지)를 쉽게 설치
- 🔄 패키지 버전 관리
- 🛠️ 프로젝트 의존성 자동 관리

#### 비유로 이해하기

npm은 앱스토어와 비슷합니다:

- **앱스토어**: 스마트폰 앱을 검색하고 설치
- **npm**: JavaScript 패키지를 검색하고 설치

예시:

```bash
*# React를 설치하고 싶다면*

npm install react

*# axios를 설치하고 싶다면*

npm install axios

```

---

3. 패키지 의존성 이해하기

📦 이 프로젝트가 사용하는 패키지들

주요 의존성 (dependencies)

이것들은 **앱이 실제로 동작하는데 필요한** 패키지들입니다.

1. React (^19.2.0)

무엇인가요?

- 사용자 인터페이스(UI)를 만들기 위한 JavaScript 라이브러리
- Facebook(Meta)에서 개발
- 컴포넌트 기반으로 UI를 구성

왜 필요한가요?

- 복잡한 UI를 쉽게 만들 수 있음
- 데이터가 변경되면 자동으로 화면 업데이트
- 재사용 가능한 컴포넌트로 코드 관리 용이
```

실제 사용 예시:

```javascript
function App() {
  return <h1>안녕하세요!</h1>
}
```

2. React-DOM (^19.2.0)

무엇인가요?

- React 컴포넌트를 실제 웹 페이지(DOM)에 렌더링하는 라이브러리

왜 필요한가요?

- React 코드를 브라우저가 이해할 수 있는 HTML로 변환
- React와 항상 함께 사용됨

실제 사용 예시:

```javascript
import ReactDOM from 'react-dom/client'
ReactDOM.createRoot(document.getElementById('root')).render(<App />)
```

3. Axios (^1.13.2)

무엇인가요?

- HTTP 요청을 쉽게 보낼 수 있게 해주는 라이브러리

- Promise 기반의 HTTP 클라이언트

왜 필요한가요?

- OpenWeatherMap API에서 날씨 데이터를 가져오기 위해

- fetch API보다 사용하기 편리

- 자동으로 JSON 변환, 에러 처리 등 지원

실제 사용 예시:

```javascript
// API에서 날씨 데이터 가져오기
const response = await axios.get('https://api.openweathermap.org/...')
console.log(response.data) // 날씨 정보
```

4. @vitejs/plugin-react (^5.1.0)

무엇인가요?

- Vite에서 React를 사용할 수 있게 해주는 플러그인

왜 필요한가요?

- JSX 문법을 JavaScript로 변환

- Fast Refresh 기능 제공 (코드 수정 시 즉시 반영)

개발 의존성 (devDependencies)

이것들은 **개발할 때만 필요한** 패키지들입니다. 실제 배포된 앱에는 포함되지 않습니다.

1. Vite (^7.1.7)

무엇인가요?

- 차세대 프론트엔드 빌드 도구

- 프랑스어로 "빠른"을 의미

왜 필요한가요?

- 개발 서버를 빠르게 실행

- 코드를 번들링하고 최적화

- Hot Module Replacement (HMR) 지원

Vite vs 기존 도구 (webpack) 비교:

| 특징 | Vite | Webpack |

|------|------|---------|

| 시작 속도 | ⚡ 매우 빠름 (수백ms) | 🐢 느림 (수초~수십초) |

| HMR 속도 | ⚡ 즉시 | ⏳ 느림 |

| 설정 | 😊 간단 | 😰 복잡 |

2. TypeScript (~5.9.3)

무엇인가요?

- JavaScript에 타입 시스템을 추가한 언어

왜 포함되어 있나요?

- Vite가 기본적으로 TypeScript를 지원

- 이 프로젝트는 JavaScript를 사용하지만, 필요시 TypeScript로 전환 가능

📊 버전 번호 이해하기

패키지 버전 표기법: `^19.2.0`

```
^19.2.0
| | |
| | └── 패치 버전 (Patch) - 버그 수정
| └──────── 마이너 버전 (Minor) - 새로운 기능 추가 (호환 가능)
└────────── 메이저 버전 (Major) - 큰 변경 (호환 불가능)
```

기호 의미:

- `^19.2.0`: 19.x.x 범위에서 최신 버전 사용 (19.2.0 ~ 19.9.9)

- `~19.2.0`: 19.2.x 범위에서만 업데이트 (19.2.0 ~ 19.2.9)

- `19.2.0`: 정확히 이 버전만 사용

---

4. 프로젝트 설치 및 실행

🚀 단계별 설치 가이드

0단계: 터미널/명령 프롬프트 열기

Windows:

1. **방법 1**: Windows 키 + R → `cmd` 입력 → Enter
2. **방법 2**: 시작 메뉴 → "명령 프롬프트" 검색
3. **방법 3** (추천): Windows Terminal 사용
4. **방법 4**: VS Code에서 `` Ctrl + ` `` (내장 터미널)

macOS:

1. **방법 1**: Command + Space → "Terminal" 입력
2. **방법 2**: VS Code에서 `` Ctrl + ` ``

Linux:

1. **Ctrl + Alt + T**
2. 또는 VS Code 내장 터미널

1단계: 프로젝트 폴더로 이동

현재 위치 확인:

```bash
# Windows
cd

# macOS/Linux
pwd
```

프로젝트 폴더로 이동:

```bash
# 예시: D 드라이브의 junsuk/react/openweather1 폴더로 이동
cd D:₩junsuk₩react₩openweather1
```

팁:
- Windows: `₩` 사용
- macOS/Linux: `/` 사용
- VS Code에서 프로젝트 폴더를 열면 터미널이 자동으로 해당 위치에서 시작

올바른 위치인지 확인:

```bash
# Windows
dir

# macOS/Linux
ls
```

```

다음 파일들이 보여야 합니다:

- `package.json`
- `index.html`
- `src` 폴더
- `vite.config.js`

---

#### 2단계: 패키지 설치 (`npm install`)

```bash
npm install
```

또는 짧게:

```bash
npm i
```

이 명령어가 하는 일:

1. **package.json 읽기**
   - 프로젝트에 필요한 패키지 목록 확인
```

2. **패키지 다운로드**

   - npm 레지스트리에서 패키지 다운로드

   - 인터넷 연결 필요

3. **node_modules 폴더 생성**

   - 모든 패키지가 이 폴더에 설치됨

   - 용량: 보통 100-300MB

4. **package-lock.json 생성**

   - 정확한 버전 정보 기록

   - 팀원들과 동일한 버전 사용 보장

설치 과정 화면 예시:

```
npm install

...

added 234 packages, and audited 235 packages in 15s

89 packages are looking for funding

  run `npm fund` for details

found 0 vulnerabilities
```

설치 시간:

- 인터넷 속도에 따라 10초 ~ 2분

- 처음 설치: 느림

- 이미 캐시가 있으면: 빠름

설치 후 확인:

```bash

*# Windows*

dir node_modules

*# macOS/Linux*

ls node_modules

```

`node_modules` 폴더에 수많은 하위 폴더가 생성되었으면 성공!

3단계: 개발 서버 실행 (`npm run dev`)

```bash

npm run dev

```

이 명령어가 하는 일:

1. **Vite 개발 서버 시작**

   - 로컬 웹 서버 실행

- 포트: 보통 5173

2. **프로젝트 빌드**
   - JSX → JavaScript 변환
   - CSS 처리
   - 최적화

3. **Hot Module Replacement (HMR) 활성화**
   - 파일 수정 시 자동으로 브라우저 업데이트
   - 페이지 새로고침 없이 변경사항 즉시 반영

실행 화면 예시:

```
  VITE v7.1.7  ready in 423 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: http://192.168.0.10:5173/
  ➜  press h + enter to show help
```

중요한 정보:

- **Local URL**: `http://localhost:5173/`
  - 자신의 컴퓨터에서만 접속 가능
  - 브라우저에서 이 주소로 접속

- **Network URL**: `http://192.168.0.10:5173/`

- 같은 네트워크의 다른 기기에서 접속 가능
- 예: 스마트폰으로 테스트

서버 중지 방법:

- **Ctrl + C** 누르기
- 터미널 창 닫기

4단계: 브라우저에서 접속

1. **웹 브라우저 열기** (Chrome, Edge, Firefox 등)

2. **주소창에 입력:**

   ```
   http://localhost:5173
   ```

3. **앱 화면 확인**
   - 보라색 그라데이션 배경
   - "🌤️ 날씨 앱" 제목
   - 검색 입력창
   - 서울 날씨 정보 (초기 로드)

✅ 화면이 정상적으로 보이면 설치 완료!

🎉 설치 완료 체크리스트

- [ ] Node.js 설치 확인 (`node --version`)
- [ ] npm 설치 확인 (`npm --version`)
- [ ] 프로젝트 폴더로 이동
- [ ] `npm install` 실행 완료
- [ ] `node_modules` 폴더 생성 확인
- [ ] `npm run dev` 실행
- [ ] 브라우저에서 앱 확인

---

5. Axios 설치 및 사용법

📡 Axios란?

Axios는 Promise 기반의 HTTP 클라이언트 라이브러리입니다.

- 브라우저와 Node.js 환경 모두에서 동작
- 외부 API로부터 데이터를 가져오거나 전송할 때 사용
- 이 프로젝트에서는 OpenWeatherMap API와 통신하는데 사용

📥 Axios 설치 방법

이 프로젝트에는 이미 Axios가 설치되어 있지만, 새로운 프로젝트에서 사용하려면 다음과 같이 설치합니다.

방법 1: npm으로 설치 (일반적)

```bash
npm install axios
```

방법 2: yarn으로 설치

```bash
yarn add axios
```

설치 확인

package.json 파일 확인:

```json
{
  "dependencies": {
    "axios": "^1.13.2"
  }
}
```

또는 터미널에서:

```bash
npm list axios
```

출력 예시:

```
openweather1@0.0.0
└── axios@1.13.2
```

🔨 Axios 기본 사용법

1. Import하기

```javascript
import axios from 'axios'
```

2. GET 요청 (데이터 가져오기)

기본 형태:

```javascript
```

```
axios.get('URL주소')
  .then(response => {
    console.log(response.data)
  })
  .catch(error => {
    console.error(error)
  })
```

async/await 형태 (추천):

```javascript
async function getData() {
  try {
    const response = await axios.get('https://api.example.com/data')
    console.log(response.data)
  } catch (error) {
    console.error(error)
  }
}
```

쿼리 파라미터 포함:

```javascript
const response = await axios.get('https://api.example.com/data', {
  params: {
    id: 123,
```

```
    name: 'John'
  }
})
// 실제 요청 URL: https://api.example.com/data?id=123&name=John
```

3. POST 요청 (데이터 전송)

```javascript
const response = await axios.post('https://api.example.com/users', {
  name: 'John Doe',
  email: 'john@example.com'
})
```

4. 헤더 설정

```javascript
const response = await axios.get('https://api.example.com/data', {
  headers: {
    'Authorization': 'Bearer YOUR_TOKEN',
    'Content-Type': 'application/json'
  }
})
```

💡 이 프로젝트에서 Axios 사용 예시

파일: `src/App.jsx`

```javascript
import axios from 'axios'

const API_KEY = 'cd63accc133fc76e1f94a3f270442688'

const fetchWeather = async (cityName) => {
  try {
    // OpenWeatherMap API 호출
    const response = await axios.get(
      `https://api.openweathermap.org/data/2.5/weather`,
      {
        params: {
          q: cityName,              // 도시 이름
          appid: API_KEY,           // API 키
          units: 'metric',          // 섭씨 온도
          lang: 'kr'                // 한국어
        }
      }
    )
```

```
      // 성공 시 데이터 사용
      console.log(response.data)
      setWeather(response.data)

    } catch (error) {
      // 에러 처리
      console.error('API 호출 실패:', error)
      setError('날씨 정보를 가져올 수 없습니다.')
    }
}
```

response 객체 구조:

```javascript
{
  data: { ... },          // 실제 응답 데이터
  status: 200,            // HTTP 상태 코드
  statusText: 'OK',       // 상태 텍스트
  headers: { ... },       // 응답 헤더
  config: { ... }         // 요청 설정
}
```

VS Axios vs Fetch 비교

JavaScript에는 HTTP 요청을 보내는 두 가지 주요 방법이 있습니다: **Axios**와 **Fetch API**

비교표

| 특징 | Axios | Fetch API |

|---|---|---|

| **설치 필요** | ✅ 필요 (`npm install axios`) | ❌ 불필요 (브라우저 내장) |

| **JSON 자동 변환** | ✅ 자동 (`response.data`) | ❌ 수동 (`response.json()` 호출) |

| **에러 처리** | ✅ HTTP 에러도 자동 catch | ❌ 네트워크 에러만 catch |

| **요청 취소** | ✅ 간단 | ⚠️ AbortController 필요 |

| **타임아웃 설정** | ✅ 쉬움 | ❌ 복잡 |

| **진행률 표시** | ✅ 지원 | ❌ 미지원 |

| **구형 브라우저** | ✅ 지원 | ❌ 폴리필 필요 |

| **문법** | 😊 간결 | 😐 다소 복잡 |

---

1. JSON 자동 변환

Axios:

```javascript
const response = await axios.get('https://api.example.com/users')
console.log(response.data)  // 자동으로 JSON 파싱 완료!
```

Fetch:

```javascript
const response = await fetch('https://api.example.com/users')
const data = await response.json()  // .json() 메서드 호출 필요
console.log(data)
```

---

2. 에러 처리

Axios:

```javascript
try {
  const response = await axios.get('https://api.example.com/users')
  console.log(response.data)
} catch (error) {
  // 404, 500 등 HTTP 에러도 여기서 catch됨
  if (error.response) {
    console.log('HTTP 에러:', error.response.status)
  } else {
    console.log('네트워크 에러')
  }
}
```

Fetch:

```javascript
try {
  const response = await fetch('https://api.example.com/users')

  // HTTP 에러는 자동으로 catch되지 않음! 수동 확인 필요
  if (!response.ok) {
    throw new Error(`HTTP 에러: ${response.status}`)
  }

  const data = await response.json()
  console.log(data)
} catch (error) {
  // 네트워크 에러만 자동으로 catch됨
  console.log('에러:', error)
}
```

---

3. 요청 취소

Axios:

```javascript
const controller = new AbortController()
```

```
axios.get('https://api.example.com/data', {
  signal: controller.signal
})

// 요청 취소
controller.abort()
```

Fetch:
```javascript
const controller = new AbortController()

fetch('https://api.example.com/data', {
  signal: controller.signal
})

// 요청 취소
controller.abort()
```
둘 다 AbortController 사용하지만, Axios는 더 많은 옵션 제공

---

4. 타임아웃 설정

Axios:

```javascript
// 5초 타임아웃 설정 - 매우 간단!
axios.get('https://api.example.com/data', {
  timeout: 5000  // 5000ms = 5초
})
```

Fetch:
```javascript
// 복잡한 방법으로 구현 필요
const controller = new AbortController()
const timeout = setTimeout(() => controller.abort(), 5000)

fetch('https://api.example.com/data', {
  signal: controller.signal
})
  .then(response => {
    clearTimeout(timeout)
    return response.json()
  })
```

---

5. 베이스 URL 설정

Axios:

```javascript
// 한 번만 설정하면 모든 요청에 적용
const api = axios.create({
  baseURL: 'https://api.example.com',
  timeout: 5000,
  headers: { 'Authorization': 'Bearer token' }
})

// 사용
api.get('/users')        // https://api.example.com/users
api.get('/products')     // https://api.example.com/products
```

Fetch:

```javascript
// 매번 전체 URL 입력 필요
fetch('https://api.example.com/users')
fetch('https://api.example.com/products')
```

---

6. 인터셉터 (Interceptor)

Axios:

```javascript
// 모든 요청 전에 자동 실행
axios.interceptors.request.use(config => {
  // 예: 토큰 자동 추가
  config.headers.Authorization = `Bearer ${getToken()}`
  return config
})

// 모든 응답 후에 자동 실행
axios.interceptors.response.use(
  response => response,
  error => {
    if (error.response.status === 401) {
      // 자동으로 로그인 페이지로 이동
      redirectToLogin()
    }
    return Promise.reject(error)
  }
)
```

Fetch:
```javascript
// 인터셉터 기능 없음 - 직접 구현 필요
```

📊 실전 코드 비교

같은 기능을 Axios와 Fetch로 구현:

Axios 버전

```javascript
import axios from 'axios'

async function getWeather(city) {
  try {
    const response = await axios.get(
      'https://api.openweathermap.org/data/2.5/weather',
      {
        params: {
          q: city,
          appid: API_KEY,
          units: 'metric'
        },
        timeout: 5000
      }
    )

    console.log(response.data)
```

```
    return response.data

  } catch (error) {
    if (error.response) {
      console.error('HTTP 에러:', error.response.status)
    } else if (error.request) {
      console.error('네트워크 에러')
    } else {
      console.error('요청 설정 에러:', error.message)
    }
    throw error
  }
}
```

Fetch 버전

```javascript
async function getWeather(city) {
  // 타임아웃 설정
  const controller = new AbortController()
  const timeout = setTimeout(() => controller.abort(), 5000)

  try {
    // URL 수동 생성
    const url = new URL('https://api.openweathermap.org/data/2.5/weather')
```

```
    url.searchParams.append('q', city)
    url.searchParams.append('appid', API_KEY)
    url.searchParams.append('units', 'metric')

    const response = await fetch(url, {
      signal: controller.signal
    })

    clearTimeout(timeout)

    // HTTP 에러 수동 체크
    if (!response.ok) {
      throw new Error(`HTTP 에러: ${response.status}`)
    }

    // JSON 수동 파싱
    const data = await response.json()
    console.log(data)
    return data

  } catch (error) {
    if (error.name === 'AbortError') {
      console.error('타임아웃')
    } else {
      console.error('에러:', error.message)
    }
```

```
    throw error
  }
}
```

🤔 어떤 것을 사용해야 할까?

Axios를 사용하는 것이 좋은 경우:

✅ **대부분의 실무 프로젝트**
- 코드가 더 깔끔하고 읽기 쉬움
- 에러 처리가 편리
- 추가 기능이 많음

✅ **복잡한 API 통신**
- 여러 API 엔드포인트 사용
- 인터셉터 필요
- 타임아웃, 재시도 등 필요

✅ **팀 프로젝트**
- 일관된 코드 스타일 유지
- 유지보수 용이

Fetch를 사용하는 것이 좋은 경우:

✅ **간단한 프로젝트**

- 몇 번의 API 호출만 필요

- 추가 라이브러리 설치 원하지 않음

✅ **번들 크기 최소화**

- 모바일 최적화 중요

- 패키지 의존성 최소화

✅ **최신 브라우저만 지원**

- 구형 브라우저 지원 불필요

💡 이 프로젝트에서 Axios를 선택한 이유

1. **학습 용이성** 😊

   - 초보자가 이해하기 쉬운 문법

   - 에러 처리가 직관적

2. **실무 활용도** 💼

   - 대부분의 회사에서 Axios 사용

   - 취업/실무에 도움

3. **기능 풍부** 🎁

   - 타임아웃, 인터셉터 등

- 확장성 좋음

4. **코드 가독성** 📖
   - 깔끔한 코드
   - 유지보수 편리

📎 더 알아보기

Axios 공식 문서:

- [Axios GitHub](https://github.com/axios/axios)
- [Axios 문서](https://axios-http.com)

Fetch API 문서:

- [MDN Fetch API](https://developer.mozilla.org/ko/docs/Web/API/Fetch_API)

---

6. API 키 발급 완벽 가이드

🔑 OpenWeatherMap API 키란?

API 키는 API 사용을 위한 인증 코드입니다.

비유:

- API 키 = 도서관 회원증

- API 키 없이 API 사용 = 회원증 없이 책 대출 시도 → ❌ 거부

현재 상황:

- 프로젝트에 테스트용 API 키 포함

- 학습용으로는 사용 가능

- 개인 프로젝트는 자신의 키 발급 권장

📋 API 키 발급 단계별 가이드

1단계: OpenWeatherMap 회원가입

1. **웹사이트 접속**

   - https://openweathermap.org

2. **Sign In 클릭**

   - 우측 상단의 "Sign In" 버튼

3. **Create an Account 클릭**

   - "Don't have an account?" 아래의 링크

4. **회원 정보 입력**

   - Username: 사용자명 (영문)

   - Email: 이메일 주소

- Password: 비밀번호 (8자 이상)
- Confirm Password: 비밀번호 재입력
- ☑️ I am 16 years old and over: 체크
- ☑️ I agree with Privacy Policy...: 체크
- reCAPTCHA: "나는 로봇이 아닙니다" 체크

5. **Create Account 버튼 클릭**

6. **이메일 인증**
   - 입력한 이메일로 인증 메일 도착
   - "Verify your email" 버튼 클릭
   - 또는 링크 복사 후 브라우저에 붙여넣기

7. **인증 완료**
   - 자동으로 로그인됨

2단계: API 키 확인

1. **로그인 후 자동 생성**
   - 회원가입과 동시에 Default API 키 자동 생성

2. **API Keys 페이지 접속**
   - 방법 1: 우측 상단 사용자명 클릭 → "My API keys"
   - 방법 2:

https://home.openweathermap.org/api_keys

3. **API 키 확인**

   ```
   Key
   ──────────────────────────────────────────────────
   abc123def456789...

   Status: Active
   ```

4. **API 키 복사**
   - 32자리 영문+숫자 조합
   - 복사 버튼 클릭 또는 직접 선택 후 복사

3단계: API 키 활성화 대기

중요! 새로 발급받은 API 키는 즉시 사용 불가

활성화 시간:
- 보통: 10분 ~ 2시간
- 최대: 24시간

활성화 전 사용 시 오류:

```json
{
  "cod": 401,
  "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more info."
}
```

활성화 확인 방법:

브라우저 주소창에 입력:

```
https://api.openweathermap.org/data/2.5/weather?q=Seoul&appid=YOUR_API_KEY
```

활성화 완료 응답:

```json
{
  "coord": {"lon": 126.9778, "lat": 37.5683},
  "weather": [...],
  "main": {...},
  ...
}
```

4단계: 프로젝트에 API 키 적용

1. **코드 에디터에서 파일 열기**

   - `src/App.jsx` 파일 열기

2. **5번째 줄 찾기**

   ```javascript
   const API_KEY = 'cd63accc133fc76e1f94a3f270442688'
   ```

3. **API 키 교체**

   ```javascript
   const API_KEY = '여기에_복사한_API_키_붙여넣기'
   ```

   예시:

   ```javascript
   const API_KEY = 'abc123def456ghi789jkl012mno345pq'
   ```

4. **파일 저장**

   - Ctrl + S (Windows/Linux)

   - Cmd + S (macOS)

5. **개발 서버 재시작 (필요 시)**

   - 보통 자동으로 반영됨

   - 안 되면 Ctrl+C 후 `npm run dev` 재실행

6. **테스트**

   - 브라우저에서 도시 검색

   - 정상 작동하면 성공!

🔒 API 키 보안

주의사항

❌ 하지 말아야 할 것:

- GitHub 등 공개 저장소에 API 키 노출

- 다른 사람과 API 키 공유

- 프론트엔드에 민감한 서버 키 노출

✅ 해야 할 것:

- 환경 변수(.env) 파일 사용

- .gitignore에 .env 추가

- 프로덕션에서는 서버 측에서 API 호출 권장

환경 변수로 API 키 관리 (고급)

1. **프로젝트 루트에 `.env` 파일 생성**

   ```

   VITE_API_KEY=your_api_key_here
   ```

```

2. **.gitignore에 추가**

```
.env
```

3. **App.jsx에서 사용**

```javascript
const API_KEY = import.meta.env.VITE_API_KEY
```

4. **개발 서버 재시작**

- .env 파일 변경 시 항상 재시작 필요

📊 API 사용량 제한

무료 플랜 (Free tier):

- ✅ 분당 60회 호출
- ✅ 일일 1,000,000회 호출
- ✅ 현재 날씨 조회
- ❌ 과거 데이터
- ❌ 예보 데이터 (일부 제한)
```

개인 프로젝트에는 충분!

사용량 확인:

1. OpenWeatherMap 로그인

2. "Statistics" 메뉴

3. 일일/시간별 호출 수 확인

---

7. 앱 사용 방법

🎮 기본 사용법

시나리오 1: 서울 날씨 확인 (초기 화면)

1. **앱 실행**
   - `npm run dev` 실행
   - 브라우저에서 `http://localhost:5173` 접속

2. **초기 화면**
   - 자동으로 서울(Seoul) 날씨 표시
   - 이유: `useEffect`에서 자동 로드

3. **표시 정보 확인**
   ```
   Seoul, KR
   ```

[날씨 아이콘]

15°C

맑음

체감 온도: 13°C

습도: 60%

풍속: 3.5 m/s

기압: 1013 hPa

```

---

#### 시나리오 2: 다른 도시 검색

1. **검색창 클릭**
   - 입력 필드에 포커스

2. **도시 이름 입력**
   - 영문으로 입력
   - 예시:
     - 국내: Seoul, Busan, Incheon, Daegu, Gwangju
     - 해외: Tokyo, Paris, London, New York, Sydney

3. **검색 실행**
   - 방법 1: "검색" 버튼 클릭
```

- 방법 2: Enter 키 누르기

4. **로딩 표시**

```
[회전하는 스피너]
날씨 정보를 불러오는 중...
```

5. **결과 표시**

- 0.5~2초 후 날씨 정보 표시
- 부드러운 애니메이션 효과

시나리오 3: 에러 처리

케이스 1: 도시 이름 미입력

```
입력: (빈 문자열)
결과: 도시 이름을 입력해주세요
```

케이스 2: 존재하지 않는 도시

```
입력: asdfqwer
결과: 날씨 정보를 가져올 수 없습니다. 도시 이름을 확인해주세요.
```

```

**케이스 3: 네트워크 오류**

```
상황: 인터넷 연결 끊김

결과: 날씨 정보를 가져올 수 없습니다...

```

---

### 🌏 검색 가능한 도시 예시

#### 한국 주요 도시

| 한글 | 영문 입력 |

|------|-----------|

| 서울 | Seoul |

| 부산 | Busan |

| 인천 | Incheon |

| 대구 | Daegu |

| 대전 | Daejeon |

| 광주 | Gwangju |

| 울산 | Ulsan |

| 수원 | Suwon |

| 제주 | Jeju |

| 춘천 | Chuncheon |
```

세계 주요 도시

| 도시 | 국가 |

|------|------|

| Tokyo | 일본 |

| Beijing | 중국 |

| Shanghai | 중국 |

| Hong Kong | 홍콩 |

| Singapore | 싱가포르 |

| Bangkok | 태국 |

| New York | 미국 |

| Los Angeles | 미국 |

| London | 영국 |

| Paris | 프랑스 |

| Berlin | 독일 |

| Rome | 이탈리아 |

| Moscow | 러시아 |

| Dubai | UAE |

| Sydney | 호주 |

🎨 UI 요소 설명

1. 제목 영역

```

🌞 날씨 앱

```

- 고정 위치

- 흰색 텍스트

- 그림자 효과

2. 검색 영역

- **입력 필드**

  - 흰색 배경

  - 둥근 모서리

  - 포커스 시 살짝 확대

  - 플레이스홀더: "도시 이름을 입력하세요..."

- **검색 버튼**

  - 흰색 배경, 보라색 텍스트

  - 호버 시 위로 이동

  - 로딩 중 비활성화

  - 텍스트 변경: "검색" ↔ "검색 중..."

3. 날씨 카드

- **헤더**

  - 좌측: 도시명, 국가 코드

  - 우측: 날씨 아이콘 (100x100px)

- **온도 표시**

  - 큰 글자 (4rem)
  - 보라색 (#667eea)
  - 섭씨 단위 (℃)

- **날씨 설명**
  - 중간 크기 (1.5rem)
  - 회색 (#4a5568)
  - 예: "맑음", "흐림", "비"

- **상세 정보 그리드**
  - 2x2 그리드 레이아웃
  - 각 항목: 그라데이션 배경
  - 라벨 + 값 구조

📱 반응형 디자인

데스크톱 (600px 초과)
- 카드 너비: 최대 500px
- 온도: 4rem
- 그리드: 2열

모바일 (600px 이하)
- 카드 너비: 화면에 맞춤
- 온도: 3rem

- 그리드: 1열 (세로 배치)

- 패딩 축소

테스트 방법:

1. 브라우저 개발자 도구 열기 (F12)

2. 디바이스 툴바 토글 (Ctrl+Shift+M)

3. 다양한 기기 크기로 테스트

    - iPhone SE (375px)

    - iPad (768px)

    - Desktop (1920px)

---

8. 프로젝트 구조 상세 분석

🗀 전체 디렉토리 구조

```
openweather1/
 |
 ├──── 🗀 public/                  # 정적 파일 (변환 없이 그대로 제공)
 |     └──── vite.svg              # Vite 로고 (파비콘)
 |
 ├──── 🗀 src/                     # 소스 코드
 |     ├──── App.jsx               # 메인 React 컴포넌트
 |     ├──── App.css               # App 컴포넌트 스타일
```

```
|   ├── main.jsx                # 애플리케이션 진입점
|   └── index.css               # 전역 스타일
|
├── 📁 node_modules/            # 설치된 패키지 (자동 생성, Git 제외)
|   ├── react/
|   ├── axios/
|   ├── vite/
|   └── ... (200+ 패키지)
|
├── index.html                  # HTML 템플릿
├── package.json                # 프로젝트 설정 및 의존성
├── package-lock.json           # 정확한 패키지 버전 (자동 생성)
├── vite.config.js              # Vite 빌드 설정
├── README.md                   # 프로젝트 설명
├── GUIDE.md                    # 상세 코드 가이드
└── GETTING_STARTED_DETAILED.md  # 이 문서
```

📄 각 파일/폴더 역할

`public/` 폴더

역할:

- 빌드 과정을 거치지 않는 정적 파일 저장

- 파일이 그대로 배포됨

예시:

- 이미지 (logo.png, banner.jpg)

- 폰트 파일

- robots.txt

- favicon.ico

사용 방법:

```html
<!-- HTML에서 -->
<img src="/logo.png" alt="Logo" />

<!-- React에서 -->
<img src="/logo.png" alt="Logo" />
```

주의:

- `/`로 시작 (루트 경로)

- `public` 폴더명은 생략

`src/` 폴더

소스 코드가 위치하는 메인 폴더

`src/main.jsx`

역할: React 앱의 시작점

코드:

```javascript
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

설명:

1. `createRoot`: React 18의 새로운 렌더링 API
2. `document.getElementById('root')`: HTML의 `<div id="root">` 찾기
3. `<StrictMode>`: 개발 모드 경고 활성화
4. `<App />`: App 컴포넌트 렌더링

`src/index.css`

역할: 전체 앱에 적용되는 글로벌 스타일

코드:

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', ...;
  -webkit-font-smoothing: antialiased;
}

#root {
  min-height: 100vh;
}
```

설명:
- `*`: 모든 요소에 적용 (전역 리셋)
- `box-sizing`: 너비 계산 방식 통일

- `font-family`: 시스템 기본 폰트

- `#root`: 앱 컨테이너 최소 높이 설정

`src/App.jsx`

역할: 메인 애플리케이션 로직

크기: 약 120줄

주요 부분:

1. State 관리 (4개)

2. API 호출 함수

3. Form 제출 처리

4. useEffect (초기 로드)

5. JSX 렌더링

자세한 설명은 GUIDE.md 참조

`src/App.css`

역할: App 컴포넌트 전용 스타일

크기: 약 200줄

주요 스타일:

- 레이아웃 (Flexbox)
- 검색 폼
- 날씨 카드
- 애니메이션
- 반응형 디자인

`node_modules/` 폴더

역할:

- npm으로 설치한 모든 패키지 저장
- 약 200~300개의 패키지

크기:

- 보통 100~300 MB
- React, Vite 등 포함

주의:

- ❌ 직접 수정 금지
- ❌ Git에 커밋하지 않음 (.gitignore)
- ✅ 삭제 후 `npm install`로 재생성 가능

구조 예시:

```
node_modules/
├── react/
│   ├── package.json
│   ├── index.js
│   └── cjs/
├── axios/
├── vite/
└── ...
```

`index.html`

역할: HTML 템플릿, 앱의 진입점

코드:

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
    <title>날씨 앱 (JSX) - OpenWeather</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

중요 요소:
- `<div id="root">`: React가 렌더링될 위치
- `<script type="module">`: ES6 모듈 사용
- `<meta name="viewport">`: 모바일 반응형 설정

`package.json`

역할: 프로젝트 설정 파일

주요 필드:
```json
{
  "name": "openweather1",                // 프로젝트 이름
  "version": "0.0.0",                    // 버전
  "type": "module",                      // ES 모듈 사용
```

```
  "scripts": { ... },                    // 실행 명령어
  "dependencies": { ... },               // 프로덕션 의존성
  "devDependencies": { ... }             // 개발 의존성
}
```

---

9. 기술 스택 상세 설명

⚛️ React

정의: 사용자 인터페이스를 만들기 위한 JavaScript 라이브러리

특징:
- 🧩 컴포넌트 기반
- ⚡ Virtual DOM으로 빠른 렌더링
- 🔄 선언적 프로그래밍
- 🌐 대규모 생태계

핵심 개념:
1. **컴포넌트**: 재사용 가능한 UI 조각
2. **JSX**: JavaScript + XML
3. **Props**: 컴포넌트 간 데이터 전달
4. **State**: 동적 데이터 관리
5. **Hooks**: 함수 컴포넌트에서 React 기능 사용

⚡ Vite

정의: 차세대 빌드 도구

장점:

- ⚡ 매우 빠른 시작 속도
- 🔥 즉각적인 HMR
- 📦 최적화된 빌드

🌐 OpenWeatherMap API

제공 데이터:

- 현재 날씨
- 5일 예보
- 날씨 아이콘
- 다양한 단위 지원

---

10. 문제 해결 완벽 가이드

🔧 설치 관련 문제

문제 1: "npm을 찾을 수 없습니다"

해결:

1. Node.js 설치 확인
2. 컴퓨터 재시작
3. 터미널 재시작

문제 2: 패키지 설치 오류

해결:

```bash
# 캐시 정리
npm cache clean --force

# 재설치
rm -rf node_modules package-lock.json
npm install
```

🚀 실행 관련 문제

문제 3: 포트 사용 중

해결:

```bash
# 다른 포트로 실행
npm run dev -- --port 3000
```

🌐 API 관련 문제

문제 4: API 키 오류

해결:

1. API 키 재확인
2. 활성화 대기 (최대 2시간)
3. 브라우저에서 직접 테스트

---

11. 커스터마이징 방법

🎨 디자인 변경

배경색 변경:

```css
.app {
  background: linear-gradient(135deg, #ff6b6b 0%, #feca57 100%);
}
```

폰트 변경:

```css
body {
  font-family: 'Noto Sans KR', sans-serif;
}
```

⚙️ 기능 추가

기본 도시 변경:

```javascript
useEffect(() => {
  fetchWeather('Tokyo') // 도쿄로 변경
}, [])
```

화씨 단위 사용:

```javascript
units: 'imperial'  // metric → imperial
```

---

12. 추가 학습 자료

📚 공식 문서

- [React 공식 문서](https://ko.react.dev)
- [Vite 공식 문서](https://vitejs.dev)
- [Axios 문서](https://axios-http.com)
- [OpenWeatherMap API](https://openweathermap.org/api)

🎓 온라인 강의

- 생활코딩 - React
- FreeCodeCamp
- React 공식 튜토리얼

💻 연습 프로젝트

1. 할 일 목록
2. 계산기
3. 영화 검색 앱
4. 채팅 앱

🎉 마치며

축하합니다! 이 가이드를 통해 React 프로젝트의 모든 것을 배우셨습니다.

🚀 다음 단계

1. 프로젝트 실행해보기
2. 코드 수정하며 실험하기
3. 자신만의 프로젝트 만들기

행복한 코딩 되세요! 🎉