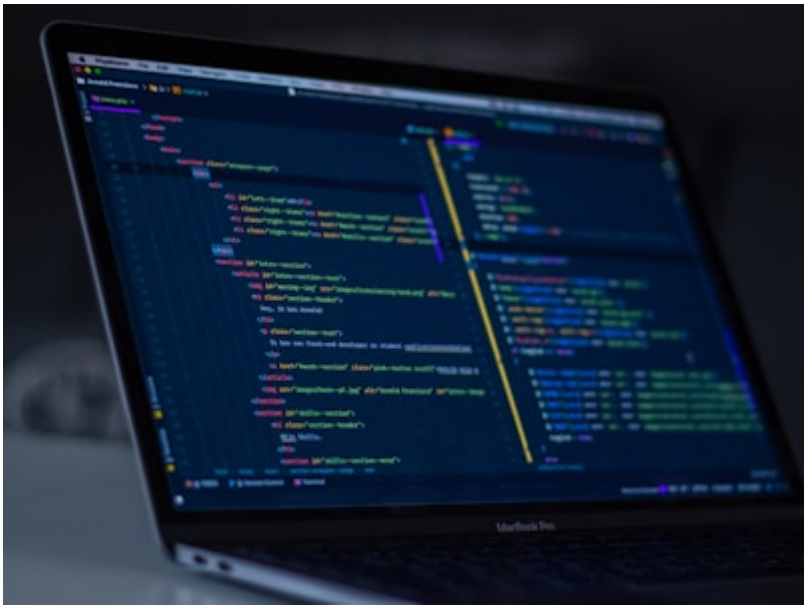
🎯 React 기초 및 컴포넌트 개발 (2부)

State와 이벤트로 인터랙티브한 웹 만들기

강사명: 조성호
소요시간: 30분

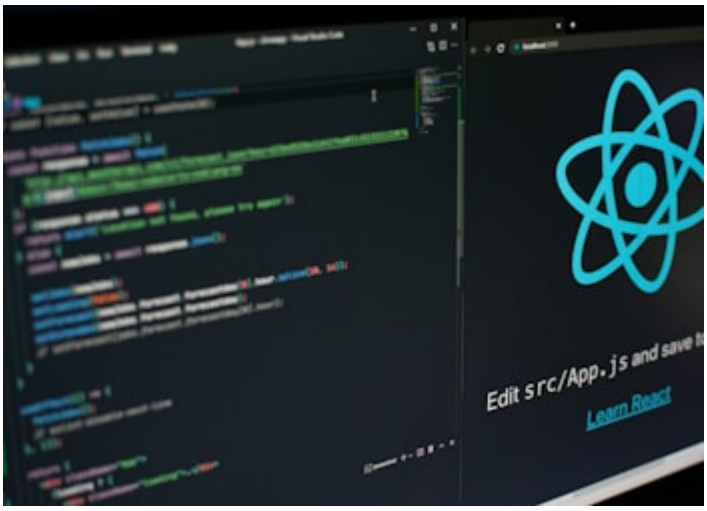


📋 6-2교시 학습 목표

- **useState Hook** 마스터하여 상태 관리
- **이벤트 처리** 및 사용자 상호작용 구현
- **조건부 렌더링** 고급 기법 활용
- **실전 프로젝트**: 다이나믹 포트폴리오 완성

🔗 6-1교시 복습

- React 필요성 및 장점 이해
- 개발 환경 설정 (Node.js, Create React App)
- JSX 문법 기초 마스터
- 컴포넌트 개념 및 Props 전달
- Header 컴포넌트 구현



State와 useState Hook

State란?

State는 컴포넌트가 가지고 있는 **변경 가능한 데이터**입니다.

Props vs State

구분	Props	State
데이터 소유	부모 컴포넌트	자기 자신
변경 가능성	읽기 전용	변경 가능
전달 방향	부모 → 자식	컴포넌트 내부
용도	데이터 전달	동적 상태 관리

State가 필요한 경우

- 사용자 입력 (폼, 검색어)
- 토글 상태 (메뉴, 테마)
- 카운터, 타이머
- API 데이터 로딩 상태

---

useState Hook 기본 사용법

기본 문법

```
import { useState } from 'react';

function Counter() {
    // [현재값, 변경함수] = useState(초기값)
    const [count, setCount] = useState(0);

    const handleIncrement = () => {
        setCount(count + 1);
    };

    const handleDecrement = () => {
        setCount(count - 1);
    };

    return (
        <div>
            <h2>카운터: {count}</h2>
            <button onClick={handleIncrement}>+</button>
            <button onClick={handleDecrement}>-</button>
        </div>
    );
}
```

다양한 데이터 타입의 State

```
function StateExamples() {
    // 문자열
    const [name, setName] = useState('');
```

```
    // 불린
    const [isVisible, setIsVisible] = useState(false);

    // 배열
    const [items, setItems] = useState([]);

    // 객체
    const [user, setUser] = useState({
        name: '',
        email: '',
        age: 0
    });

    return (
        // 컴포넌트 JSX
    );
}
```

State 업데이트 패턴

1. 단순 값 업데이트

```
const [count, setCount] = useState(0);

// 직접 값 설정
setCount(10);

// 이전 값 기반 업데이트 (권장)
setCount(prevCount => prevCount + 1);
```

2. 객체 State 업데이트

```
const [user, setUser] = useState({
    name: '',
    email: '',
    age: 0
});

// ❌ 잘못된 방법 (기존 객체 직접 수정)
user.name = 'John';

// ☑ 올바른 방법 (새 객체 생성)
setUser({
    ...user,        // 기존 속성 복사
    name: 'John'    // 특정 속성만 변경
});
```

```
// 또는
setUser(prevUser => ({
    ...prevUser,
    name: 'John'
}));
```

3. 배열 State 업데이트

```
const [todos, setTodos] = useState([]);

// 항목 추가
setTodos([...todos, newTodo]);

// 항목 삭제
setTodos(todos.filter(todo => todo.id !== targetId));

// 항목 수정
setTodos(todos.map(todo =>
    todo.id === targetId
        ? { ...todo, completed: !todo.completed }
        : todo
));
```

이벤트 처리

React에서의 이벤트

SyntheticEvent

React는 브라우저 이벤트를 **SyntheticEvent**로 감싸서 제공합니다.

```
function Button() {
    const handleClick = (event) => {
        event.preventDefault();    // 기본 동작 방지
        event.stopPropagation();   // 이벤트 버블링 중단

        console.log('클릭됨!');
        console.log('이벤트 타입:', event.type);
        console.log('클릭된 요소:', event.target);
    };

    return <button onClick={handleClick}>클릭하세요</button>;
}
```

일반적인 이벤트 종류

```
function EventExamples() {
    return (
        <div>
            {/* 클릭 이벤트 */}
            <button onClick={handleClick}>클릭</button>

            {/* 입력 이벤트 */}
            <input onChange={handleChange} onFocus={handleFocus} />

            {/* 폼 이벤트 */}
            <form onSubmit={handleSubmit}>
                <input type="submit" value="제출" />
            </form>

            {/* 마우스 이벤트 */}
            <div
                onMouseEnter={handleMouseEnter}
                onMouseLeave={handleMouseLeave}
            >
                호버 영역
            </div>
        </div>
    );
}
```

폼 다루기

제어 컴포넌트 (Controlled Component)

```
function ContactForm() {
    const [formData, setFormData] = useState({
        name: '',
        email: '',
        message: ''
    });

    const handleChange = (event) => {
        const { name, value } = event.target;
        setFormData({
            ...formData,
            [name]: value
        });
    };

    const handleSubmit = (event) => {
        event.preventDefault();
        console.log('폼 데이터:', formData);

        // 여기서 API 호출 등 처리
```

```
            alert('메시지가 전송되었습니다!');

            // 폼 리셋
            setFormData({
                name: '',
                email: '',
                message: ''
            });
        };

        return (
            <form onSubmit={handleSubmit}>
                <input
                    type="text"
                    name="name"
                    placeholder="이름"
                    value={formData.name}
                    onChange={handleChange}
                    required
                />

                <input
                    type="email"
                    name="email"
                    placeholder="이메일"
                    value={formData.email}
                    onChange={handleChange}
                    required
                />

                <textarea
                    name="message"
                    placeholder="메시지"
                    value={formData.message}
                    onChange={handleChange}
                    required
                />

                <button type="submit">전송</button>
            </form>
        );
    }
```

🎯 조건부 렌더링

다양한 조건부 렌더링 패턴

1. if 문 사용

```
function UserProfile({ user }) {
    if (!user) {
        return <div>로딩 중...</div>;
    }

    if (user.role === 'admin') {
        return (
            <div>
                <h2>관리자: {user.name}</h2>
                <button>관리 패널</button>
            </div>
        );
    }

    return (
        <div>
            <h2>사용자: {user.name}</h2>
            <p>{user.email}</p>
        </div>
    );
}
```

2. 삼항 연산자

```
function LoginButton({ isLoggedIn, onLogin, onLogout }) {
    return (
        <button onClick={isLoggedIn ? onLogout : onLogin}>
            {isLoggedIn ? '로그아웃' : '로그인'}
        </button>
    );
}

function Dashboard({ user }) {
    return (
        <div>
            <h1>대시보드</h1>
            {user ? (
                <div>
                    <p>안녕하세요, {user.name}님!</p>
                    <UserMenu user={user} />
                </div>
            ) : (
                <div>
                    <p>로그인이 필요합니다.</p>
                    <LoginForm />
                </div>
            )}
        </div>
    );
}
```

3. 논리 AND 연산자

```
function Notifications({ notifications }) {
    return (
        <div>
            <h2>알림</h2>
            {notifications.length > 0 && (
                <div className="notification-badge">
                    {notifications.length}
                </div>
            )}

            {notifications.length === 0 && (
                <p>새로운 알림이 없습니다.</p>
            )}

            {notifications.map(notification => (
                <div key={notification.id} className="notification">
                    {notification.message}
                </div>
            ))}
        </div>
    );
}
```

📋 리스트 렌더링과 Key

리스트 렌더링 기본

배열 데이터 렌더링

```
function SkillList() {
    const skills = [
        { id: 1, name: 'HTML', level: 90 },
        { id: 2, name: 'CSS', level: 85 },
        { id: 3, name: 'JavaScript', level: 80 },
        { id: 4, name: 'React', level: 75 }
    ];

    return (
        <div className="skills">
            <h2>기술 스택</h2>
            {skills.map(skill => (
                <div key={skill.id} className="skill-item">
                    <span className="skill-name">{skill.name}</span>
                    <div className="skill-bar">
```

```
                        <div
                            className="skill-progress"
                            style={{ width: `${skill.level}%` }}
                        />
                    </div>
                    <span className="skill-level">{skill.level}%</span>
                </div>
            ))}
        </div>
    );
}
```

Key의 중요성

왜 Key가 필요한가?

React는 리스트가 변경될 때 어떤 항목이 **추가, 삭제, 수정**되었는지 알아야 합니다.

```
// ❌ key가 없으면 경고 발생
{items.map(item => (
    <div>{item.name}</div>
))}

// ❌ 인덱스를 key로 사용 (권장하지 않음)
{items.map((item, index) => (
    <div key={index}>{item.name}</div>
))}

// ☑ 고유한 ID를 key로 사용 (권장)
{items.map(item => (
    <div key={item.id}>{item.name}</div>
))}
```

Key 선택 가이드라인

```
// 1. 데이터에 고유 ID가 있는 경우
{users.map(user => (
    <UserCard key={user.id} user={user} />
))}

// 2. 고유 ID가 없는 경우 - 조합 사용
{items.map(item => (
    <Item key={`${item.category}-${item.name}`} item={item} />
))}

// 3. 정적 리스트인 경우에만 인덱스 사용
{['월', '화', '수', '목', '금'].map((day, index) => (
    <Day key={index} name={day} />
))}
```

💻 실전 프로젝트: 인터랙티브 포트폴리오

Theme Toggle 구현

ThemeProvider 컴포넌트

```
import { useState, createContext, useContext } from 'react';

const ThemeContext = createContext();

export function ThemeProvider({ children }) {
    const [isDark, setIsDark] = useState(false);

    const toggleTheme = () => {
        setIsDark(!isDark);
    };

    return (
        <ThemeContext.Provider value={{ isDark, toggleTheme }}>
            <div className={isDark ? 'dark-theme' : 'light-theme'}>
                {children}
            </div>
        </ThemeContext.Provider>
    );
}

export const useTheme = () => {
    const context = useContext(ThemeContext);
    if (!context) {
        throw new Error('useTheme must be used within ThemeProvider');
    }
    return context;
};
```

Header에 테마 토글 추가

```
import { useTheme } from '../context/ThemeContext';

function Header() {
    const { isDark, toggleTheme } = useTheme();

    return (
        <header className="header">
            <div className="container">
                <div className="logo">
                    <h1>조성호</h1>
```

```
                    <span>Frontend Developer</span>
                </div>

                <nav className="nav">
                    <a href="#about">About</a>
                    <a href="#skills">Skills</a>
                    <a href="#projects">Projects</a>
                    <a href="#contact">Contact</a>
                </nav>

                <button className="theme-toggle" onClick={toggleTheme}>
                    {isDark ? '🌞' : '🌙'}
                </button>
            </div>
        </header>
    );
}
```

동적 스킬 차트

SkillBar 컴포넌트

```
import { useState, useEffect } from 'react';
import './SkillBar.css';

function SkillBar({ name, level, color }) {
    const [animatedLevel, setAnimatedLevel] = useState(0);

    useEffect(() => {
        const timer = setTimeout(() => {
            setAnimatedLevel(level);
        }, 500);

        return () => clearTimeout(timer);
    }, [level]);

    return (
        <div className="skill-bar">
            <div className="skill-info">
                <span className="skill-name">{name}</span>
                <span className="skill-percentage">{level}%</span>
            </div>
            <div className="skill-track">
                <div
                    className="skill-progress"
                    style={{
                        width: `${animatedLevel}%`,
                        backgroundColor: color,
                        transition: 'width 1.5s ease-out'
                    }}
```

```
                />
            </div>
        </div>
    );
}

export default SkillBar;
```

Skills 컴포넌트

```
import SkillBar from './SkillBar';

function Skills() {
    const skills = [
        { name: 'HTML', level: 90, color: '#e34c26' },
        { name: 'CSS', level: 85, color: '#1572b6' },
        { name: 'JavaScript', level: 80, color: '#f7df1e' },
        { name: 'React', level: 75, color: '#61dafb' },
        { name: 'Git', level: 70, color: '#f05032' }
    ];

    return (
        <section id="skills" className="skills">
            <div className="container">
                <h2>기술 스택</h2>
                <div className="skills-grid">
                    {skills.map(skill => (
                        <SkillBar
                            key={skill.name}
                            name={skill.name}
                            level={skill.level}
                            color={skill.color}
                        />
                    ))}
                </div>
            </div>
        </section>
    );
}

export default Skills;
```

프로젝트 필터링

ProjectFilter 컴포넌트

```
import { useState } from 'react';
```

```
function ProjectFilter() {
    const [activeFilter, setActiveFilter] = useState('all');
    const [projects, setProjects] = useState([
        {
            id: 1,
            title: 'Portfolio Website',
            category: 'web',
            technologies: ['HTML', 'CSS', 'JavaScript'],
            image: '/images/portfolio.jpg'
        },
        {
            id: 2,
            title: 'Todo App',
            category: 'app',
            technologies: ['React', 'CSS'],
            image: '/images/todo.jpg'
        },
        {
            id: 3,
            title: 'Weather App',
            category: 'app',
            technologies: ['React', 'API'],
            image: '/images/weather.jpg'
        }
    ]);

    const categories = ['all', 'web', 'app', 'mobile'];

    const filteredProjects = activeFilter === 'all'
        ? projects
        : projects.filter(project => project.category === activeFilter);

    return (
        <section id="projects" className="projects">
            <div className="container">
                <h2>프로젝트</h2>

                <div className="filter-buttons">
                    {categories.map(category => (
                        <button
                            key={category}
                            className={`filter-btn ${activeFilter === category ?
'active' : ''}`}
                            onClick={() => setActiveFilter(category)}
                        >
                            {category.charAt(0).toUpperCase() + category.slice(1)}
                        </button>
                    ))}
                </div>

                <div className="projects-grid">
                    {filteredProjects.map(project => (
                        <div key={project.id} className="project-card">
                            <img src={project.image} alt={project.title} />
```

```
                            <div className="project-info">
                                <h3>{project.title}</h3>
                                <div className="tech-tags">
                                    {project.technologies.map(tech => (
                                        <span key={tech} className="tech-tag">
                                            {tech}
                                        </span>
                                    ))}
                                </div>
                            </div>
                        </div>
                    ))}
                </div>
            </div>
        </section>
    );
}

export default ProjectFilter;
```

연락처 폼 구현

ContactForm 컴포넌트

```
import { useState } from 'react';

function ContactForm() {
    const [formData, setFormData] = useState({
        name: '',
        email: '',
        subject: '',
        message: ''
    });

    const [isSubmitting, setIsSubmitting] = useState(false);
    const [submitStatus, setSubmitStatus] = useState(null);

    const handleChange = (e) => {
        setFormData({
            ...formData,
            [e.target.name]: e.target.value
        });
    };

    const handleSubmit = async (e) => {
        e.preventDefault();
        setIsSubmitting(true);

        try {
            // 실제로는 여기서 API 호출
```

```
            await new Promise(resolve => setTimeout(resolve, 2000));

            setSubmitStatus('success');
            setFormData({
                name: '',
                email: '',
                subject: '',
                message: ''
            });
        } catch (error) {
            setSubmitStatus('error');
        } finally {
            setIsSubmitting(false);
        }
    };

    return (
        <section id="contact" className="contact">
            <div className="container">
                <h2>연락하기</h2>

                {submitStatus === 'success' && (
                    <div className="alert alert-success">
                        메시지가 성공적으로 전송되었습니다!
                    </div>
                )}

                {submitStatus === 'error' && (
                    <div className="alert alert-error">
                        전송 중 오류가 발생했습니다. 다시 시도해주세요.
                    </div>
                )}

                <form onSubmit={handleSubmit} className="contact-form">
                    <div className="form-group">
                        <input
                            type="text"
                            name="name"
                            placeholder="이름"
                            value={formData.name}
                            onChange={handleChange}
                            required
                            disabled={isSubmitting}
                        />
                    </div>

                    <div className="form-group">
                        <input
                            type="email"
                            name="email"
                            placeholder="이메일"
                            value={formData.email}
                            onChange={handleChange}
                            required
```

```
                            disabled={isSubmitting}
                        />
                    </div>

                    <div className="form-group">
                        <input
                            type="text"
                            name="subject"
                            placeholder="제목"
                            value={formData.subject}
                            onChange={handleChange}
                            required
                            disabled={isSubmitting}
                        />
                    </div>

                    <div className="form-group">
                        <textarea
                            name="message"
                            placeholder="메시지"
                            rows="5"
                            value={formData.message}
                            onChange={handleChange}
                            required
                            disabled={isSubmitting}
                        />
                    </div>

                    <button
                        type="submit"
                        className="submit-btn"
                        disabled={isSubmitting}
                    >
                        {isSubmitting ? '전송 중...' : '메시지 보내기'}
                    </button>
                </form>
            </div>
        </section>
    );
}

export default ContactForm;
```

🙌 6-2교시 마무리

오늘 배운 내용 정리

- **useState Hook**으로 상태 관리 마스터
- **이벤트 처리** 및 사용자 상호작용 구현
- **조건부 렌더링** 다양한 패턴 활용

- **리스트 렌더링**과 Key 속성 이해
- **실전 프로젝트**: 인터랙티브 포트폴리오 완성

React 학습 로드맵

기초 단계 (완료!)

- ☑ 컴포넌트와 JSX
- ☑ Props와 State
- ☑ 이벤트 처리
- ☑ 조건부/리스트 렌더링

중급 단계

- useEffect Hook (생명주기)
- 커스텀 Hook
- Context API (전역 상태)
- React Router (페이지 라우팅)

고급 단계

- 성능 최적화 (memo, useMemo, useCallback)
- 상태 관리 라이브러리 (Redux, Zustand)
- 테스팅 (Jest, React Testing Library)
- TypeScript with React