













웹 프로그래밍 깃허브 정리

<https://github.com/bumsoo1/web>

소프트웨어학과
2021763063
정범수

0311~0527

| | | | |
|---|----------------------|-------------------------|--|
|  bumsoo1 | Update readme.md | 9467ebb · 6 minutes ago |  32 Commits |
|  0311 | Add files via upload | 3 weeks ago | |
|  0318 | Update readme.md | 18 minutes ago | |
|  0324 | Update readme.md | 13 minutes ago | |
|  0401 | Update readme.md | 9 minutes ago | |
|  0408 | Update readme.md | 6 minutes ago | |
|  0429 | Add files via upload | last month | |
|  0513 | Add files via upload | last month | |
|  0520 | Add files via upload | 3 weeks ago | |
|  0527 | Update readme.md | 2 weeks ago | |
|  README.md | Update README.md | 28 minutes ago | |

README

2025 Web Programming

| 날짜 | 파일 |
|------|----------------------|
| 0311 | 0311 |
| 0318 | 0318 |
| 0324 | 0324 |
| 0401 | 0401 |
| 0408 | 0408 |
| 0429 | 0429 |
| 0513 | 0513 |
| 0520 | 0520 |
| 0527 | 0527 |

0311

Readme 수정



bumsoo1 Add files via upload

Code

Blame

18 lines (16 loc) · 484 Bytes



Code 55% faster with GitHub Copilot

```
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>나의 소개</title>
7 </head>
8 <body>
9
10   <div class="container">
11     <h1>안녕하세요! 저는 홍길동입니다.</h1>
12     <p>직업: 대학생</p>
13     <p>관심사: 프로그래밍, 웹개발</p>
14     <p>저의 웹 페이지에 방문해 주셔서 감사합니다!</p>
15   </div>
16
17 </body>
18 </html>
```

Commits on May 26, 2025

Add files via upload



bumsoo1 authored 3 weeks ago

Create readme.md



bumsoo1 authored 3 weeks ago

End of commit history for this file

0318

Readme 수정

```

1  import React from "react";
2
3  const styles = {
4    wrapper: {
5      margin: 8,
6      padding: 8,
7      display: "flex",
8      flexDirection: "row",
9      border: "1px solid grey",
10     borderRadius: 16,
11   },
12   imageContainer: {},
13   image: {
14     width: 50,
15     height: 50,
16     borderRadius: 25,
17   },
18   contentContainer: {
19     marginLeft: 8,
20     display: "flex",
21     flexDirection: "column",
22     justifyContent: "center",
23   },
24   nameText: {
25     color: "black",
26     fontSize: 16,
27     fontWeight: "bold",
28   },
29   CommentText: {
30     color: "black",
31     fontSize: 16,

```

```

32 },
33 };
34
35 function Comment(props) {
36   return (
37     <div style={styles.wrapper}>
38       <div style={styles.imageContainer}>
39         
40       </div>
41
42       <div style={styles.contentContainer}>
43         <span style={styles.nameText}>{props.name}</span>
44         <span style={styles.commentText}>{props.comment}</span>
45       </div>
46     </div>
47   )
48 }
49
50
51 export default Comment;

```

```

1  import React from "react";
2  import Comment from "./comment";
3
4  const commentse = [
5    {
6      name: "이인재",
7      comment: "안녕하세요, 소름입니다.",
8    },
9    {
10     name: "유재석",
11     comment: "리액트 재미있어요~!",
12   },
13   {
14     name: "강인경",
15     comment: "저도 리액트 배워 보고 싶어요!!",
16   },
17 ];
18
19 function CommentList(props){
20   return (
21     <div>
22       {commentse.map((comment) => {
23         return (
24           <Comment name={comment.name} comment={comment.comment} />
25         )
26       })}
27     </div>

```

```

1  import React from "react";
2  function Clock1(props) {
3    return (
4      <div>
5        <h1>안녕, 리액트!</h1>
6        <h2>현재 시간: {new Date().toLocaleTimeString()}</h2>
7      </div>
8    );
9  }
10 export default Clock1;

```

React Elements

개요

- 가상 DOM은 자바스크립트 객체입니다.
- 가상 DOM은 리액트 엘리먼트로 이루어져 있고, 브라우저 DOM은 실제 DOM 엘리먼트로 구성되어 있습니다.

리액트 엘리먼트란?

- 리액트 앱을 이루는 가장 기본적인 단위입니다.
- 리액트 엘리먼트는 브라우저 DOM을 만들기 위해 **개발자와 브라우저 DOM을 이어주는 오직고** 역할을 합니다.
- 엘리먼트는 View에 렌더링할 내용을 React에 알려주기 위한 수단으로, React 애플리케이션을 구성하는 **가장 작은 블록**입니다.

구성

- React 엘리먼트는 다음과 같은 객체(object) 형태로 정의됩니다.
 - type: 필드: HTML 태그의 이름을 값으로 가집니다.
 - props: 필드: 그 외 속성들을 값으로 전달받습니다.
- React는 이 객체를 읽어들이 실제 DOM을 구성하고 최신 상태로 업데이트하는 데 사용합니다.

특징

- React 엘리먼트는 **일반 객체(plain object)**로 손쉽게 생성할 수 있습니다.
- 하지만 **불변 객체(immutable object)**이기 때문에, 일단 생성된 후에는 상태나 속성을 변경할 수 없습니다.
- 따라서 React에서 **나를 업데이트**하는 방법은 새로운 엘리먼트를 생성하고, 이를 `render()` 메서드에 전달하는 것입니다.

엘리먼트 렌더링 (Element Rendering)

ReactDOM이란?

- ReactDOM은 **나를 실제로 브라우저에 렌더링할 때** 사용하는 라이브러리입니다.

render 함수

- ReactDOM의 `render()` 함수는 리액트 엘리먼트와 해당 엘리먼트의 모든 **자식 엘리먼트를 루트(root) DOM 노드**에 렌더링합니다.

Commits on Jun 16, 2025

Update readme.md

bumsoo1 authored 24 minutes ago

Commits on May 26, 2025

Add files via upload

bumsoo1 authored 3 weeks ago

Create readme.md

bumsoo1 authored 3 weeks ago

End of commit history for this file

0324

Readme 수정

```
1 import React, { useState } from 'react';
2
3 const Header = () => {
4   return (
5     <header>
6       <h1>나의 웹사이트</h1>
7       <nav>
8         <ul>
9           <li><a href="#home">홈</a></li>
10          <li><a href="#about">소개</a></li>
11          <li><a href="#contact">연락처</a></li>
12        </ul>
13      </nav>
14    </header>
15  );
16
17 const Footer = () => {
18   return (
19     <footer>
20       <p>&copy; 2025 나의 웹사이트</p>
21     </footer>
22   );
23
24 const Sidebar = () => {
25   return (
26     <aside>
27       <h3>사이드바</h3>
28       <ul>
29         <li><a href="#link1">링크 1</a></li>
30
31         <li><a href="#link2">링크 2</a></li>
32         <li><a href="#link3">링크 3</a></li>
33       </ul>
34     </aside>
35   );
36
37 const Article = ({ article, onClick, author, date }) => {
38   return (
39     <div onClick={() => onClick(article)} className="article">
40       <h3>{article.title}</h3>
41       <p>{article.excerpt}</p>
42       <p>작성자: {author}</p>
43       <p>작성일: {date}</p>
44     </div>
45   );
46
47 const ArticleList = ({ articles, onArticleClick }) => {
48   return (
49     <section>
50       <h2>기사 목록</h2>
51       <div className="articles">
52         {articles.map((article, index) => (
53           <Article
54             key={index}
55             article={article}
56             onClick={onArticleClick}
57             author={article.author}
58             date={article.date}
59
60           />
61         ))}
62       </div>
63     </section>
64   );
65
66 const ArticleDetail = ({ article }) => {
67   if (!article) {
68     return <p>기사를 선택해주세요.</p>
69   }
70   return (
71     <section>
72       <h2>{article.title}</h2>
73       <p>{article.content}</p>
74     </section>
75   );
76
77 const App = () => {
78   const [selectedArticle, setSelectedArticle] = useState(null);
79   const articles = [
80     {
81       title: '기사 1',
82       excerpt: '기사 1의 요약 내용',
83       content: '기사 1의 내용',
84       author: '홍길동',
85       date: '2025-03-31',
86     },
87     {
88       title: '기사 2',
89       excerpt: '기사 2의 요약 내용',
90       content: '기사 2의 내용',
91       author: '김철수',
92       date: '2025-03-30',
93     },
94     {
95       title: '기사 3',
96       excerpt: '기사 3의 요약 내용',
97       content: '기사 3의 내용',
98       author: '이영희',
99       date: '2025-03-29',
100    },
101  ];
102
103 const handleArticleClick = (article) => {
104   setSelectedArticle(article);
105 };
106
107 export default App;
```

프로퍼티 (Props)

개념

- 프로퍼티(Props)는 properties의 줄임말입니다.
- 컴포넌트 간에 데이터를 전달하기 위한 React의 중요한 기능입니다.

데이터 흐름

- 상위 컴포넌트 → 하위 컴포넌트 로 값을 전달합니다.
- 단방향 데이터 흐름(one-way data flow) 을 갖습니다.

자식 컴포넌트 입장에서

- 자식 컴포넌트는 props를 읽기 전용(read-only) 으로만 사용할 수 있습니다.
- 즉, 전달받은 값을 직접 수정할 수 없습니다.

컴포넌트 간 통신

- React 컴포넌트는 props를 통해 서로 통신합니다.
- 모든 부모 컴포넌트는 props를 줌으로써 정보 전달이 가능합니다.

⚠ 일반 변수를 사용하는 경우

화면에 변화가 없어요!

- 일반 변수를 사용하면 화면은 변경되지 않습니다.
- 하지만 그렇다고 해서 값이 변경되지 않은 것은 아닙니다.
 - F12 개발자 도구 콘솔에서 확인해보면, count 값은 정상적으로 변경됩니다.
 - plus() 또는 minus() 함수를 실행한 후 console.log(count) 로 확인 가능!

? 그럼 왜 화면은 그대로일까요?

- 그 이유는 바로 "일반 변수는 변경되어도 React가 이를 감지하지 못하기 때문" 입니다.
- React는 렌더링을 제어하는 메커니즘을 갖고 있는데, 일반 변수는 이 메커니즘과 연결되어 있지 않아요.

일반 변수는 값이 바뀌더라도
자동으로 화면을 재렌더링하지 않습니다!

Commits on Jun 16, 2025

Update readme.md

bumsoo1 authored 22 minutes ago

Create readme.md

bumsoo1 authored 25 minutes ago

Commits on Mar 31, 2025

Delete 0324/readme.md

bumsoo1 authored on Mar 31

Add files via upload

bumsoo1 authored on Mar 31

Create readme.md

bumsoo1 authored on Mar 31

End of commit history for this file

0401 Readme 수정

```
1 import React from 'react';
2 import './App.css';
3
4 const App = () => {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <h1>안녕하세요, 저는 [이름]입니다!</h1>
9         <p>저에 대해 더 알아보세요!</p>
10      </header>
11
12      <section className="about-me">
13        <h2>자기소개</h2>
14        <p>
15          안녕하세요! 저는 [이름]이고, [직업/학문]에 관심이 많은 사람입니다.
16          [취미/특기]를 좋아하고, 항상 새로운 도전과 배움을 즐깁니다.
17        </p>
18      </section>
19
20      <section className="skills">
21        <h2>나의 기술</h2>
22        <ul>
23          <li>HTML, CSS, JavaScript</li>
24          <li>React, Node.js</li>
25          <li>Python, Django</li>
26        </ul>
27      </section>
28
29      <section className="contact">
30        <h2>연락처</h2>
31        <p>저와 연락하고 싶으시면, 아래 이메일로 연락 주세요:</p>
32        <p>
33          <a href="mailto:youremail@example.com">youremail@example.com</a>
34        </p>
35      </section>
36
37      <footer>
38        <p>© 2025 [이름]. All Rights Reserved.</p>
39      </footer>
40    </div>
41  );
42 };
43
44 export default App;
```

📚 인공지능 및 생성형 AI 이해부터 활용까지

인공지능의 이해

🚀 생성형 AI가 주목받는 이유

- 기술적 변화
 - 데이터 인식에서 '창조' 영역 확장
 - GPT-3 → GPT-3.5 (2년), GPT-3.5 → GPT-4 (4개월)로 발전 속도 가속화
- 산업적 영향
 - 대형 언어 모델 기반 서비스와 애플리케이션 급증

🧠 인공지능의 역사

- 초기 AI: 체스, 수학 문제 등 단순 알고리즘
- 발전 방향: 딥러닝 + 대량 데이터로 복잡한 문제 해결
- 대표 분야: 자연어 처리(NLP), 컴퓨터 비전(CV)

챗봇 AI 비교

💡 ChatGPT

- 정확하고 구체적인 응답
- 개인 설정 및 메모리 기능 관리 가능
- 음성 인식 지원

🌐 Gemini (구글)

- Google 앱들과 연동 우수
- 이메일, 드라이브, 지도 등과 통합

📄 Claude

- 한국어 성능 우수
- 코딩과 텍스트 생성에 최적화

💚 CLOVA X (네이버)

- 문서 업로드 및 AI 지우개 기능
- 네이버 생태계와 연계

발표자료 제작도 AI로 똑딱

🖼️ AI 발표 도구 예시

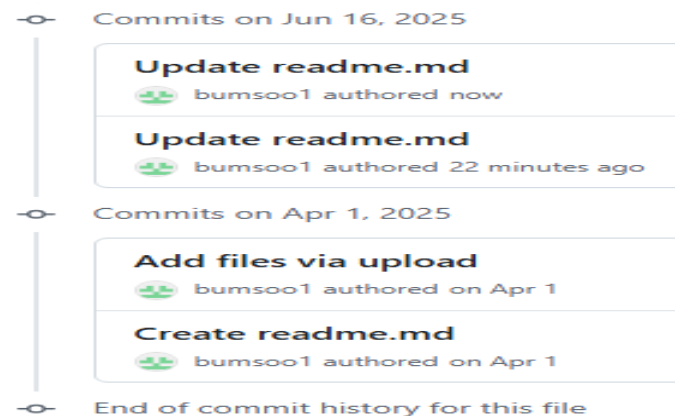
- Gamma.app: 흐름 중심
- Beautiful.ai: 템플릿 다양, 디자인 자동화

✅ 장점

- 시간 절약 ⌚
- 전문가 수준 디자인 🎨
- 직관적 사용성 🧑🏻💻
- 협업과 실시간 피드백 👥

🔄 제작 단계

- 기획 및 문서화
- 도구 선택
- 슬라이드 구성
- 피드백 → 수정



0408

Readme 수정

```
1 import React, { useState } from 'react';
2
3 function TutorialCard({ tutorial, onSelect, onSave, isSaved }) {
4   return (
5     <div
6       onClick={() => onSelect(tutorial)}
7       className="p-4 border rounded-lg cursor-pointer hover:shadow-md transition"
8     >
9       <h3 className="text-lg font-bold">{tutorial.title}</h3>
10      <p>제작자: {tutorial.creator}</p>
11      <p>주제: {tutorial.topic}</p>
12      <button
13        onClick={(e) => {
14          e.stopPropagation();
15          onSave(tutorial);
16        }}
17        className="mt-2 text-sm text-blue-500 hover:underline"
18      >
19        {isSaved ? '📁 저장됨' : '☆ 품하기'}
20      </button>
21    </div>
22  );
23 }
24
25 export default function TutorialListPage() {
26   const [tutorials] = useState([
27     { id: 1, title: 'React로 TOOD 앱 만들기', creator: '코딩가이', topic: 'React' },
28     { id: 2, title: '파이썬으로 데이터 분석', creator: 'Data왕', topic: 'Python' },
29     { id: 3, title: 'AI 모델 기초', creator: 'AI마스터', topic: 'AI' },
30   ]);
```

```
31
32   const [filterTopic, setFilterTopic] = useState('');
33   const [selectedTutorial, setSelectedTutorial] = useState(null);
34   const [savedTutorials, setSavedTutorials] = useState([]);
35
36   const filtered = tutorials.filter(t =>
37     filterTopic === '' || t.topic === filterTopic
38   );
39
40   const toggleSave = (tutorial) => {
41     setSavedTutorials(prev =>
42       prev.some(t => t.id === tutorial.id)
43         ? prev.filter(t => t.id !== tutorial.id)
44         : [...prev, tutorial]
45     );
46   };
47
48   return (
49     <div className="p-6 max-w-4xl mx-auto">
50       <h1 className="text-2xl font-bold mb-4">코딩 튜토리얼 추천</h1>
51
52       <select
53         value={filterTopic}
54         onChange={(e) => setFilterTopic(e.target.value)}
55         className="mb-4 p-2 border"
56       >
```

```
57     <option value="">전체 보기</option>
58     <option value="React">React</option>
59     <option value="Python">Python</option>
60     <option value="AI">AI</option>
61   </select>
62
63   <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
64     {filtered.map(t => (
65       <TutorialCard
66         key={t.id}
67         tutorial={t}
68         onSelect={setSelectedTutorial}
69         onSave={toggleSave}
70         isSaved={savedTutorials.some(st => st.id === t.id)}
71       >
72     ))}
73   </div>
74
75   {selectedTutorial && (
76     <div className="mt-6 p-4 border-t">
77       <h2 className="text-xl font-semibold">선택한 튜토리얼</h2>
78       <p>제목: {selectedTutorial.title}</p>
79       <p>제작자: {selectedTutorial.creator}</p>
80       <p>주제: {selectedTutorial.topic}</p>
81     </div>
82   )}
83 </div>
84 );
85 }
```

Commits on Jun 16, 2025

Update readme.md

bumsoo1 authored 23 minutes ago

Commits on Apr 14, 2025

Update readme.md

bumsoo1 authored on Apr 14

Add files via upload

bumsoo1 authored on Apr 14

Create readme.md

bumsoo1 authored on Apr 14

End of commit history for this file

React Hook의 등장 배경

리액트 컴포넌트의 두 가지 형태

- 함수형 컴포넌트 (Functional Component)
- 클래스형 컴포넌트 (Class Component)

리액트 초기 컴포넌트 사용 방식

- 일반적으로는 함수형 컴포넌트를 사용
- 하지만 다음과 같은 기능이 필요할 때만 클래스형 컴포넌트를 사용해야 했음:
 - 상태(state) 관리
 - 생명주기 메서드 (Lifecycle Methods) 사용

클래스형 컴포넌트의 주요 단점

| 단점 | 설명 |
|---------------|----------------------|
| 복잡한 코드 구성 | 컴포넌트 재사용성이 낮음 |
| 최적화 어려움 | 컴파일 단계에서 코드 최적화가 어려움 |
| 최신 기술 적용의 어려움 | 새로운 패턴이나 기술 반영이 비효율적 |

그래서 등장한 것이 바로...

React Hook (리액트 훅)

클래스형 컴포넌트의 단점을 보완하고, 함수형 컴포넌트에서도 상태 관리와 생명주기 기능을 사용할 수 있도록 만들어진 혁신적인 기능입니다!

React Hook이 해결해준 것

- 간결한 코드로 컴포넌트 작성 가능
- 함수형 컴포넌트에서도 상태 및 생명주기 로직 처리 가능
- 최신 기술 및 패턴을 더 유연하게 적용

Hook은 함수형 컴포넌트를 더욱 강력하게 만들어주는 도구이며, React 개발의 새로운 표준으로 자리잡았습니다.

useState 훅(React Hook) 완전 정리

0429

```
1  function ButtonComponent() {  
2      function handleClick() {  
3          alert("버튼이 클릭되었습니다!");  
4      }  
5      return <button onClick={handleClick}>클릭</button>;  
6  }  
7  export default ButtonComponent;
```


```
1  function MyButton(props) {  
2      const handleDelete = (id, event) => {  
3          console.log(id, event.target);  
4      };  
5      return (  
6          <button onClick={(event) => handleDelete(1, event)}>  
7              삭제하기  
8          </button>  
9      );  
10 }  
11 export default MyButton;
```

```
1  function InputComponent() {  
2      function handleChange(event) {  
3          console.log("입력값:", event.target.value);  
4      }  
5      return <input type="text" onChange={handleChange} />;  
6  }  
7  export default InputComponent;
```

```
1  function Parent() {  
2      function handleParentClick() {  
3          alert("부모 요소 클릭!");  
4      }  
5      function handleChildClick(event) {  
6          event.stopPropagation();  
7          alert("자식 요소 클릭!");  
8      }  
9      return (  
10         <div onClick={handleParentClick} style={{ padding: "20px", background: "#ddd" }}>  
11             <button onClick={handleChildClick}>클릭</button>  
12         </div>  
13     );  
14 }  
15 export default Parent;
```

Commits on May 19, 2025

Add files via upload

 bumsoo1 authored last month

Create readme.md

 bumsoo1 authored last month

End of commit history for this file

0513

```
1 import React from "react";
2
3 const students = [
4   {
5     id: 1,
6     name: "Inje",
7   },
8   {
9     id: 2,
10    name: "Steve",
11  },
12  {
13    id: 3,
14    name: "Bill",
15  },
16  {
17    id: 4,
18    name: "Jeff",
19  },
20 ];
21 function AttendanceBook(props) {
22   return (
23     <ul>
24       {students.map((student, index) => {
25         return <li key={student.id}>{student.name}</li>
26       })}
27     </ul>
28   );
29 }
30
31 export default AttendanceBook;
```

```
1 import React, { useState } from "react";
2 import TemperatureInput from "../temperatureinput";
3
4 function BoilingVerdict(props) {
5   if (props.celsius >= 100) {
6     return <p>물이 끓습니다.</p>;
7   }
8   return <p>물이 끓지 않습니다.</p>;
9 }
10 function toCelsius(fahrenheit) {
11   return ((fahrenheit - 32) * 5) / 9;
12 }
13 function toFahrenheit(celsius) {
14   return (celsius * 9) / 5 + 32;
15 }
16 function tryConvert(temperature, convert) {
17   const input = parseFloat(temperature);
18   if (Number.isNaN(input)) {
19     return "";
20   }
21   const output = convert(input);
22   const rounded = Math.round(output * 1000) / 1000;
23   return rounded.toString();
24 }
25 function Calculator(props) {
26   const [temperature, setTemperature] = useState("");
27   const [scale, setScale] = useState("c");
28   const handleCelsiusChange = (temperature) => {
29     setTemperature(temperature);
30     setScale("c");
31   };
32   const handleFahrenheitChange = (temperature) => {
33     setTemperature(temperature);
34     setScale("f");
35   };
36   const celsius =
37     scale === "f" ? tryConvert(temperature, toCelsius) : temperature;
38   const fahrenheit =
39     scale === "c" ? tryConvert(temperature, toFahrenheit) : temperature;
40
41   return (
42     <div>
43       <TemperatureInput
44         scale="c"
45         temperature={celsius}
46         onTemperatureChange={handleCelsiusChange}
47       />
48       <TemperatureInput
49         scale="f"
50         temperature={fahrenheit}
51         onTemperatureChange={handleFahrenheitChange}
52       />
53       <BoilingVerdict celsius={parseFloat(celsius)} />
54     </div>
55   );
56 }
57
58 export default Calculator;
```

```
1 const scaleNames = {
2   c: "섭씨",
3   f: "화씨",
4 };
5 function TemperatureInput(props) {
6   const handleChange = (event) => {
7     props.onTemperatureChange(event.target.value);
8   };
9   return (
10    <fieldset>
11      <legend>
12        온도를 입력해주세요 (단위: {scaleNames[props.scale]}):
13      </legend>
14      <input value={props.temperature} onChange={handleChange} />
15    </fieldset>
16  );
17 }
18 export default TemperatureInput;
```

```
1 function Card(props) {
2   const { title, backgroundColor, children } = props;
3   return (
4     <div
5       style={{
6         margin: 8,
7         padding: 8,
8         borderRadius: 8,
9         boxShadow: "0px 0px 4px grey",
10        backgroundColor: backgroundColor || "white",
11      }}
12     >
13       {title && <h1>{title}</h1>}
14       {children}
15     </div>
16   );
17 }
18 export default Card;
```

```
1 import Card from "../Card";
2 function ProfileCard(props) {
3   return (
4     <Card title="Inje Lee" backgroundColor="#4ea04e">
5       <p>안녕하세요, 소플입니다.</p>
6       <p>저는 리엑트를 사용해서 개발하고 있습니다.</p>
7     </Card>
8   );
9 }
10 export default ProfileCard;
```

Commits on May 19, 2025

Add files via upload

 bumsoo1 authored last month

Create readme.md

 bumsoo1 authored last month

End of commit history for this file

0520

```
1 import React from 'react';
2 import styled from 'styled-components';
3 const TodoTemplateBlock = styled.div`
4 width: 512px;
5 height: 768px;
6 position: relative; /* 추후박스하단에 추가버튼을 위치시키기
7 위한설정*/
8 background: white;
9 border-radius: 16px;
10 box-shadow: 0 0 8px 0 rgba(0, 0, 0, 0.04);
11 margin: 0 auto; /* 페이지중앙에 나타나도록 설정*/
12 margin-top: 96px;
13 margin-bottom: 32px;
14 display: flex;
15 flex-direction: column;
16 `;
17 function TodoTemplate({ children }) {
18   {
19     return <TodoTemplateBlock>{children}</TodoTemplateBlock>;
20   }
21   export default TodoTemplate;
```

```
1 import React from 'react';
2 import styled from 'styled-components';
3 const TodoListBlock = styled.div`
4 flex: 1;
5 padding: 20px 32px;
6 padding-bottom: 48px;
7 overflow-y: auto;
8 background: gray; /* 사이즈 조정이 잘 되고 있는지 확인하기 위한 임시 스타일 */
9 `;
10 function TodoList() {
11   return <TodoListBlock>TodoList</TodoListBlock>;
12 }
13 export default TodoList;
```

```
1 import React from 'react';
2 import styled from 'styled-components';
3 import { useTodoState } from './TodoContext';
4 const TodoHeadBlock = styled.div`
5 padding-top: 48px;
6 padding-left: 32px;
7 padding-right: 32px;
8 padding-bottom: 24px;
9 border-bottom: 1px solid #e9ecef;
10 h1 {
11   margin: 0;
12   font-size: 36px;
13   color: #343a40;
14 }
15 .day {
16   margin-top: 4px;
17   color: #868e96;
18   font-size: 21px;
19 }
20 .tasks-left {
21   color: #20c997;
22   font-size: 18px;
23   margin-top: 40px;
24   font-weight: bold;
25 }
```

```
26 `;
27 function TodoHead() {
28   const todos = useTodoState();
29   const undoneTasks = todos.filter(todo => !todo.done);
30   const today = new Date();
31   const dateString = today.toLocaleDateString('ko-KR', {
32     year: 'numeric',
33     month: 'long',
34     day: 'numeric'
35   });
36   const dayName = today.toLocaleDateString('ko-KR', { weekday: 'long' });
37   return (
38     <TodoHeadBlock>
39       <h1>{dateString}</h1>
40       <div className="day">{dayName}</div>
41       <div className="tasks-left">할 일 {undoneTasks.length}개 남음
42     </div>
43   </TodoHeadBlock>
44 );
45 }
46 export default TodoHead;
```

✓ Todo App 컴포넌트 목록

할 일 관리 앱을 구성하기 위한 컴포넌트 목록과 각 컴포넌트의 역할을 정리한 문서입니다.

TodoTemplate

역할: 투두리스트의 레이아웃을 설정하는 컴포넌트

- 페이지 중앙에 위치
- 그림자가 적용된 흰색 박스를 렌더링
- 전체 앱의 틀을 담당

31 TodoHead

역할: 날짜와 남은 할 일 수 표시

- 오늘의 날짜와 요일을 표시
- 아직 완료되지 않은 할 일 개수를 보여줌

✓ TodoItem

역할: 개별 할 일 정보를 표시하고 조작

- 좌측 원 클릭 시 완료 상태 토글
- 완료 시:
 - 체크 표시
 - 텍스트 색상 연해짐
- 마우스 호버 시 휴지통 아이콘 표시
 - 클릭 시 항목 삭제

+ TodoCreate

역할: 새로운 할 일을 추가할 수 있게 하는 입력 폼 제공

- TodoTemplate 하단에 초록색 원 버튼 표시
- 버튼 클릭 시 입력 폼 표시
- 버튼 다시 클릭 시 폼이 사라짐

Commits on May 26, 2025

Add files via upload

bumsoo1 authored 3 weeks ago

Update readme.md

bumsoo1 authored 3 weeks ago

Create readme.md

bumsoo1 authored 3 weeks ago

End of commit history for this file

0527

🌐 라우팅 (Routing)

- 웹 애플리케이션에서 라우팅이라는 개념은 사용자가 요청한 URL에 따라 알맞는 페이지를 보여주는 것을 의미합니다.
- 웹 애플리케이션을 만들 때 하나의 페이지로 구성할 수도 있고, 여러 페이지로 구성할 수도 있습니다.

📌 예시

- 일정 관리 애플리케이션 → 하나의 페이지로 충분
- 블로그 애플리케이션 → 여러 페이지로 구성

📁 블로그 애플리케이션 페이지 구성 예시

- 📄 글쓰기 페이지 : 새로운 포스트를 작성하는 페이지
- 📁 포스트 목록 페이지 : 블로그에 작성된 여러 포스트의 목록을 보여주는 페이지
- 📄 포스트 읽기 페이지 : 하나의 포스트를 상세히 보여주는 페이지

🔗 라우팅 시스템의 필요성

여러 페이지로 구성된 웹 애플리케이션을 만들 때 페이지별로 컴포넌트를 분리해가며 프로젝트를 관리하기 위해 라우팅 시스템이 필요합니다.

이를 통해 URL 경로마다 각기 다른 컴포넌트를 연결하고, 사용자의 요청에 맞는 화면을 보여줄 수 있습니다.

🖨 SPA (Single Page Application)

- 싱글 페이지 애플리케이션(SPA) 이란, 하나의 페이지로 이루어진 웹 애플리케이션입니다.

📌 MPA와 SPA의 차이점

- MPA (Multi Page Application)
사용자인터랙션이 별로 없는 정적인 페이지는 전통적인 MPA 방식이 적합합니다.
- SPA (Single Page Application)
사용자인터랙션이 많고 다양한 정보를 제공하는 모던 웹 애플리케이션에는 SPA 방식이 더 적합합니다.

🔗 SPA의 동작 방식

- HTML 파일을 한 번만 받아온 뒤, 이후에는 필요한 데이터만 서버에서 받아와서 화면을 동적으로 업데이트합니다.
- 페이지를 이동할 때도 서버에서 새 HTML을 불러오는 것이 아니라, 클라이언트에서 필요한 부분만 바꿔서 보여주는 방식입니다.

👤 사용자 경험

- 기술적으로는 한 페이지만 존재하지만, 사용자가 경험하기에는 여러 페이지가 존재하는 것처럼 느낄 수 있습니다.
- 이를 위해 라우팅 시스템이 동작하여, URL 경로에 따라 다른 화면을 보여줍니다.

🔗 SPA에서 라우팅 동작 방식

- 리액트에서는 React Router 같은 라이브러리를 사용하여 라우팅을 구현합니다.
- 사용자가 링크를 클릭해 이동할 때.
 - 서버에 새 HTML을 요청하는 것이 아니라
 - 브라우저의 History API를 이용해 주소만 변경하고 기존 웹 애플리케이션을 유지한 채 라우팅 설정에 따라 다른 컴포넌트를 렌더링합니다.

이를 통해 페이지 전환 속도가 빠르고 자연스러운 UX를 제공할 수 있습니다.

🌐 URL 파라미터와 쿼리스트링

웹 애플리케이션에서 페이지 주소를 정의할 때 유동적인 값을 전달해야 할 때가 있습니다. 이때 사용하는 방법이 URL 파라미터와 쿼리스트링(Query String) 입니다.

📌 URL 파라미터

- 주소의 경로에 유동적인 값을 넣는 형태
- 주소 ID, 이름 등을 사용하여 특정 데이터를 조회할 때 사용

📄 예시

→ velopert 값이 URL 파라미터로 전달됨

📌 쿼리스트링 (Query String)

- 주소의 뒷부분에 ? 문자열 이후에 key=value 형태로 값을 정의
- 여러 값을 전달할 때는 & 로 구분

📄 예시

→ page=1, keyword=react 값이 쿼리스트링으로 전달됨

📊 두 방식의 차이

| 구분 | 설명 | 예시 |
|----------|---------------------------|--------------------------------|
| URL 파라미터 | 주소 경로에 값을 포함 | /profile/velopert |
| 쿼리스트링 | ? 이후에 key=value 형태로 값을 전달 | /articles?page=1&keyword=react |

📄 사용 용도

- URL 파라미터
 - 특정 데이터 조회 (ex. 사용자 ID, 게시물 ID 등)
- 쿼리스트링
 - 데이터 조회에 필요한 옵션 전달 (ex. 키워드 검색, 페이지네이션, 정렬 방식 등)

🚀 useNavigate

React Router에서 페이지 이동을 제어할 때 사용하는 훅입니다. useNavigate 를 통해 이동할 때 숫자 값이나 옵션을 줄 수 있습니다.

📌 navigate 함수 동작

- 숫자 타입을 전달하면 앞으로/뒤로 이동
 - navigate(-1) → 한 번 뒤로 가기
 - navigate(-2) → 두 번 뒤로 가기
 - navigate(1) → 앞으로 한 번 가기 (단, 뒤로 가기를 한 상태여야 앞으로 가기가 가능)
- 페이지 이동 시 옵션 설정
 - replace: true 옵션을 사용하면 현재 페이지를 히스토리에 남기지 않고 이동

📄 코드 예제

```
import { useNavigate } from 'react-router-dom';

const MyComponent = () => {
  const navigate = useNavigate();

  const goArticles = () => {
    navigate('/articles', { replace: true });
  };

  return (
    <button onClick={goArticles}>게시글 목록 페이지로 이동</button>
  );
};
```

🔗 Commits on Jun 5, 2025

Update readme.md

bumsoo1 authored 2 weeks ago

Create readme.md

bumsoo1 authored 2 weeks ago

🔗 End of commit history for this file

점수

12점

0415 미제출과 늦게 제출한 레포트들이 있어서
12점이라고 생각하였습니다.