

차세대 챗봇 Lab I

Lecture-2 React Hook

Sungja Choi,
School of Computing, Gachon University
2023-1



학습내용

리액트의 기본내용에 대해 학습합니다.

스타일 컴포넌트 적용

UI 컴포넌트 재사용(Button, TextInput)

조건부 렌더링

리액트 훅 (useState, useNavigate, useParams)

JSON 파일 접근

Practice

실습-1 코멘트 컴포넌트 제작

코멘트 컴포넌트를 만들어서 이미지를 온라인으로 가져오고, 이름과 코멘트를 출력합니다. 스타일 적용 및 기본 레이아웃에 대해 학습합니다.

[실행화면]



KimGachon

안녕하세요, 김가천입니다.



CaCaOj

카카오제이입니다~!



GaSoon

가순이입니다. 파이팅!!

Practice

>> Flex는 제시된 방향으로
엘리먼트를 추가
>> justifyContent는 가로-중심축

Coding(1)

```
import React from 'react';

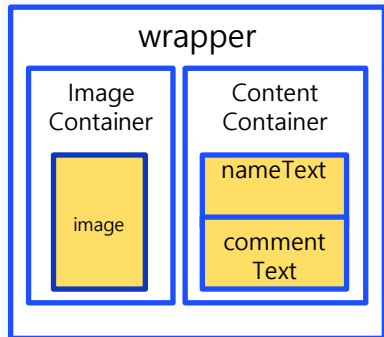
const styles = {
  wrapper: {
    margin: 8,
    padding: 8,
    display: "flex",
    flexDirection: "row",
    border: "1px solid grey",
    borderRadius: 16,
  },
  imageContainer: {},
  image: {
    width: 50,
    height: 50,
    borderRadius: 25,
  },
```

```
  contentContainer: {
    marginLeft: 8,
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
  },
  nameText: { color: "black",
    fontSize: 16,
    fontWeight: "bold",
  },
  commentText: {
    color: "black",
    fontSize: 16,
  },
};
```

Comment.jsx

Practice

스타일 구조



Coding(1)

```
function Comment(props){
  return(
    <div style = {styles.wrapper}>
      <div style={styles.imageContainer}>
        
      </div>
      <div style={styles.contentContainer}>
        <span style={styles.nameText}> {props.name}</span>
        <span style={styles.commentText}>{props.comment}</span>
      </div>
    </div>
  );
}
export default Comment;
```

Practice

Coding(2)

```
import Comment from './Comment';
```

App.js

```
...
```

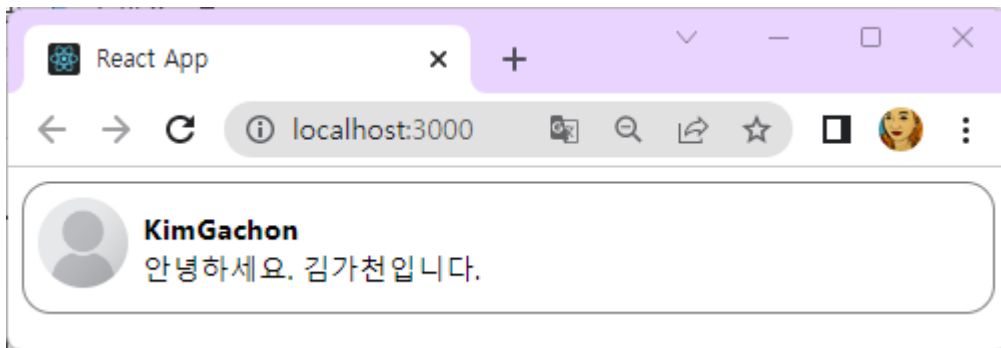
```
<div>
```

```
  <Comment name="KimGachon" comment="안녕하세요. 김가천입니다."
```

```
/>
```

```
</div>
```

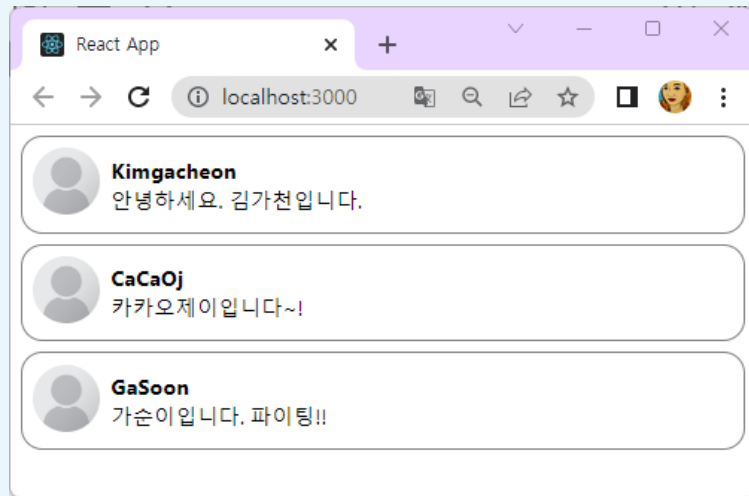
>> npm start



수업시간 내 진행하시기 바랍니다.

코멘트 리스트 페이지 제작

- CommnetList 컴포넌트 추가
- List map을 활용
- 사이버캠퍼스 제출
- 평가(10점)



수업시간 내 진행하시기 바랍니다.

코멘트 리스트 페이지 제작

[REF.] Code reference

```
const users = [  
  {id:1, name: "Kimgacheon", comment: "안녕하세요. 김가천입니다."},  
  {id:2, name: "CaCa0j", comment: "카카오제이입니다~!"},  
  {id:1, name: "GaSoon", comment: "가순이입니다. 파이팅!!"}, ];
```

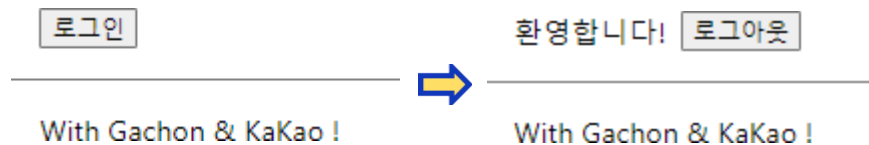
```
{users.map((user)=><Comment key={user.id} name={user.name} comment={user.comment})}
```


Practice

실습-2 랜딩 페이지 제작

툴 바를 만들고 로그인 상태를 관리하도록 합니다.

[실행화면]



Practice

Coding(1)

```
import React from "react";
const styles = {
  wrapper: {
    padding: 16,
    display: "flex",
    flexDirection: "row",
    borderBottom: "1px solid grey",
  },
  greeting: {
    marginRight: 8,
  },
};
```

Toolbar.jsx

Practice

❶ isLoggedIn이 true이면
greeting 스타일을 적
용하여 “환영합니다!”
를 출력

❷ isLoggedIn 이 true이면
로그아웃 버튼을 렌더링
하고 onClickLogout 메소
드를 실행, false이면 로그
인 버튼을 렌더링하고 on
ClickLogin 메소드를 실행

Coding(1)

```
function Toolbar(props) {  
  const { isLoggedIn, onClickLogin, onClickLogout } = props;  
  return (  
    <div style={styles.wrapper}>  
      ❶ {isLoggedIn && <span style={styles.greeting}>환영합니다!</span>}  
      ❷ {isLoggedIn ? (<button onClick={onClickLogout}>  
        로그아웃</button> )  
        : (<button onClick={onClickLogin}>  
        로그인</button>)}  
    </div>  
  ); }  
  
export default Toolbar;
```

Practice

>> Npm

(Node Package Manager)은 Node.js가 설치되면 사용가능한 패키지로써, npm start 명령어를 사용해서 리액트 프로젝트를 실행하게 합니다.

Coding(2)

```
import LandingPage from './LandingPage';
```

App.js

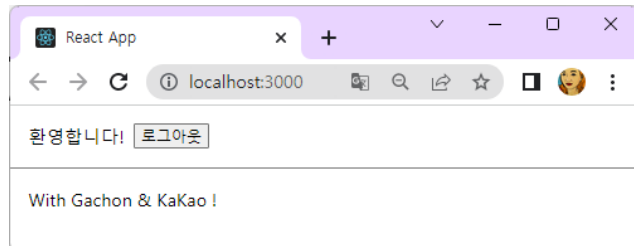
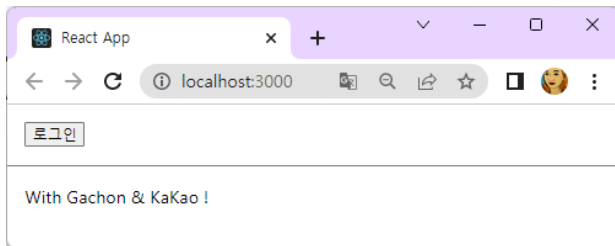
```
...
```

```
<div>
```

```
  <LandingPage />/>
```

```
</div>
```

>> npm start



React Hook

리액트 컴포넌트의 상태를 관리하기 위한 Hook 기법에 대해
학습합니다.

[참조] <https://ko.reactjs.org/>

[REF.]

React Hook

리액트 컴포넌트의 상태를관리 지원하는 기법으로 다양한 훅기술이 제공
주요 훅 유형

Hook Type	기능
useState	상태관리를 하고싶은 변수를 지정하고 set메소드로 변경
useEffect	렌더링을 발생시켜서 상태를 관리
useMemo	기존에 수행한 연산의 결과값을 저장해 두고 동일한 입력이 들어오면 재활용
useContext	필요한 props를 글로벌 하게 사용
useParams	Parameter(파라미터) 값을 URL을 통해서 넘겨서 넘겨받은 페이지에서 사용
useCallback	특정 함수를 재사용하기 위해 사용
useNavigate	양식이 제출되거나 특정 event가 발생할 때, url을 조작할 수 있는 interface를 제공

* 예제와 관련된 훅 기법 위주로 다룹니다.

Practice

>> 상태관리?

리액트 앱에서 변화하는 데이터들을 관리하는 것으로, 상태의 초기값을 저장하거나, 현재 상태의 값을 읽거나, 새로운 데이터로 상태를 업데이트 하는 것입니다.

실습-3 미니 블로그 제작

UI 컴포넌트(Button, TextInput)를 만들어서 재사용 가능하도록 하고, 미니 블로그 형태의 앱을 제작. 앱의 상태관리를 **리액트 훅**을 사용해본다.

[실행화면]


KaKao Academy of Gachon Univ.

글 작성하기

차세대봇

데이터관리

시스템아키텍처



KaKao Academy of Gachon Univ.

뒤로 가기

차세대봇

프론트엔드와 백엔드의 우수한 클라우드 엔지니어를 양성합니다.

댓글

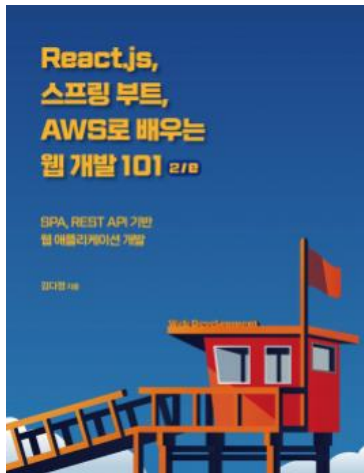
실제로 개발하다보면 map함수를 진짜 많이 쓰는 것 같아요 🤔

적용해보니 코드가 정말 간결해지네요 ㅎㅎ

댓글 작성하기

[REF.]

3장 프론트엔드 개발 참조



미니 블로그 예제의 상태관리

리액트 훅을 사용해서 컴포넌트의 상태관리를 지원합니다.

먼저, 해당 예제에서는 블로그페이지를 작성할 때, 글쓰기 기능에서 입력되는 자료에 대해 상태관리를 지정합니다.

- (1) **useState** 훅을 사용해서 입력창의 정보가 변경될 때마다 관련 상태변수의 정보가 업데이트 되도록 합니다.
- (2) **useNavigate** 훅을 사용해서 페이지 이동을 원활하게 지원해 줍니다.
- (3) **useParams** 훅을 사용하면 페이지 이동으로 발생하는 파라미터를 사용할 수 있습니다.
- (4) **props** 는 컴포넌트들 간 데이터를 전달하고 싶을 때 사용합니다.
(예) `<Name x=10 />` `<->` `function Name(props)`

Practice

>> data.json 파일을 준비
JSON (JavaScript Object
Notation)
데이터를 저장하거나
전송할 때 많이 사용되는
경량의 DATA 교환 형식

Coding(0)

data.json

```
[
  {
    "id": 1,
    "title": "차세대봇",
    "content": "프론트엔드와 백엔드의 우수한 클라우드 엔지니어를 양성합니다.",
    "comments": [
      {
        "id": 11,
        "content": "실제로 개발하다보면 map함수를 진짜 많이 쓰는 것 같아요 😊"
      },
      {
        "id": 12,
        "content": "적용해보니 코드가 정말 간결해지네요 ㅎㅎ"
      }
    ]
  }
],
```

Practice

Coding(0)

```
{
  "id": 2,
  "title": "데이터관리",
  "content": "클라우드 네이티브 환경의 데이터 관리에 대한 사항을 공유합니다.",
  "comments": [
    {
      "id": 21,
      "content": "이렇게 사용하는 방법이 있군요!"
    },
    {
      "id": 22,
      "content": "좋은 글 감사합니다 ㅎㅎ"
    },
    {
      "id": 23,
      "content": "쉬운 설명 감사드립니다 😊"
    },
    {
      "id": 24,
      "content": "바로 코드에 적용해보겠습니다!!"
    }
  ]
},
```

Practice

Coding(0)

```
{
  "id": 3,
  "title": "시스템아키텍처",
  "content": "쿠버네티스에 대한 사항을 공유합니다.",
  "comments": [
    {
      "id": 31,
      "content": "뭔가 어려운 개념이 있는데, 글을 읽고 조금 정리가 된 것 같습니다."
    },
    {
      "id": 32,
      "content": "Hook이 뭔가 했더니 이런거였군요. 알려주셔서 감사합니다 ㅎㅎ"
    },
    {
      "id": 33,
      "content": "처음에 책을 접했을 때 너무 어려웠는데 감사합니다! 🙏"
    }
  ]
}
```

Practice

>> npm i styled-components

- ❶ 스타일드 컴포넌트 표기 방법은 `...` 백틱 사이에 스타일 정의
- ❷ Button 컴포넌트를 호출하는 부모 컴포넌트에서 title과 onClick을 전달받는다.
- ❸ StyledButton 유형의 스타일을 적용하고 onClick 이벤트를 처리

[REF.] 백틱이란?
키보드의 1원쪽에 위치한 ` 문자

Coding(1)

ui/Button.jsx

```
import React from "react";
import styled from "styled-components";
```

```
const StyledButton = styled.button` ❶
  padding: 8px 16px;
  font-size: 16px; border-width: 1px; border-radius: 8px;
  cursor: pointer;
`;
```

```
function Button(props) {
  const { title, onClick } = props; ❷

  return <❸StyledButton onClick={onClick}>{title || "button"}</StyledButton>;
}
```

```
export default Button;
```

Practice

- ❶ ``백틱 안에서 변수를 표현할 때 \$를 붙여준다.
- ❷ TextInput 컴포넌트를 호출하는 부모 컴포넌트에서 height, value, onChange를 전달한다.

Coding(2)

ui/TextInput.jsx

```
import React from "react";
import styled from "styled-components";

const StyledTextarea = styled.textarea`
  width: calc(100% - 32px);
  ${({props}) => props.height && `height: ${props.height}px;`} ❶
  padding: 16px;
  font-size: 16px;
  line-height: 20px;
`;

function TextInput(props) {
  const { height, value, onChange } = props; ❷
  return <StyledTextarea height={height} value={value} onChange={onChange} />;
}
export default TextInput;
```

Practice

>> 주요파일 임포트 및
스타일 지정

Coding(3)

```
import React from "react";  
import styled from "styled-components";
```

list/CommentListItem.jsx

```
const Wrapper = styled.div`  
  width: calc(100% - 32px);  
  padding: 8px 16px;  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
  justify-content: center;  
  border: 1px solid grey;  
  border-radius: 8px;  
  cursor: pointer;  
  background: white;  
  :hover {  
    background: lightgrey;  
  }  
`;  
;
```

```
const ContentText = styled.p`  
  font-size: 16px;  
  white-space: pre-wrap;  
`;  
;
```

Practice

>> props로 전달된 내용을
comment에 저장하고 Co
ntentText 스타일을 적용
해서 내용을 출력하도록
함

Coding(3)

```
function CommentListItem(props) {  
  const { comment } = props; co  
nsole.log(comment);  
  return (  
    <Wrapper>  
      <ContentText>{comment.content}</ContentText>  
    </Wrapper>  
  );  
}  
  
export default CommentListItem;
```

Practice

- ❶ &는 styled component에서 this와 같은 의미로써 마지막 차일드가 아닌 모든 항목에 대해 아래마진을 16px로 지정

Coding(4)

list/CommentList.jsx

```
import React from "react";
import styled from "styled-components";
import CommentListItem from "../CommentListItem";

const Wrapper = styled.div`
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  justify-content: center;

  & > * { ❶
    :not(:last-child) {
      margin-bottom: 16px;
    }
  }
`;
```


Practice

- ❶ Props를 통해 전달받은 내용을 comments에 넣고 map 메소드를 사용하여 CommentListItem을 출력하도록 함

Coding(4)

```
function CommentList(props) {  
  const { comments } = props;  
  
  return (  
    <Wrapper>  
      {comments.map((comment, index) => {❶  
        return <CommentListItem key={comment.id} comment={comment} />;  
      })}  
    </Wrapper>  
  );  
}  
  
export default CommentList;
```

Practice

>> 주요파일 임포트 및
스타일 지정

Coding(5)

```
import React from "react";  
import styled from "styled-components";
```

list/PostListItem.jsx

```
const Wrapper = styled.div`  
  width: calc(100% - 32px);  
  padding: 16px;  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
  justify-content: center;  
  border: 1px solid grey;  
  border-radius: 8px;  
  cursor: pointer;  
  background: white;  
  :hover {  
    background: lightgrey;  
  }  
`;  
;
```

```
const TitleText = styled.p`  
  font-size: 20px;  
  font-weight: 500;  
`;  
;
```

Practice

>> props로 전달된 post와
onClick을 받아서 버튼이
클릭되면 post의 타이틀을
출력함

Coding(5)

```
function PostListItem(props) {  
  const { post, onClick } = props;  
  
  return (  
    <Wrapper onClick={onClick}>  
      <TitleText>{post.title}</TitleText>  
    </Wrapper>  
  );  
}  
  
export default PostListItem;
```

Practice

>> 주요파일 импорт 및
스타일 지정

Coding[6]

```
import React from "react";  
import styled from "styled-components";  
import PostListItem from "../PostListItem";
```

list/PostList.jsx

```
const Wrapper = styled.div`  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
  justify-content: center;  
  
  & > * {  
    :not(:last-child) {  
      margin-bottom: 16px;  
    }  
  }  
`;  
;
```

Practice

>> props로 전달된 posts와
onClickItem을 받아서 map
메소드를 사용해서 PostLi
stItem의 post를 버튼이 클
릭되면 출력함

Coding[6]

```
function PostList(props) {  
  const { posts, onClickItem } = props;  
  console.log(posts);  
  return (  
    <Wrapper>  
      { posts.map( (post, index) => {  
        return (  
          <PostListItem  
            key={post.id}  
            post={post}  
            onClick={() => { onClickItem(post); }}  
          /> ); } )  
        }  
      </Wrapper>  
    );  
  }  
}  
  
export default PostList;
```

Practice

```
>> 주요파일 임포트 및  
스타일 지정  
>> npm i react-router-dom
```

Coding(7)

```
import React from "react";  
import { useNavigate } from "react-  
router-dom";  
import styled from "styled-  
components";  
import PostList from "../list/PostList";  
import Button from "../ui/Button";  
import data from '../data.json';
```

```
const Wrapper = styled.div`  
  padding: 16px;  
  width: calc(100% - 32px);  
  display: flex;  
  flex-direction: column;  
  align-items: center; ju  
  stify-content: center;
```

```
`;
```

page/MainPage.jsx

```
const Container = styled.div`  
  width: 100%;  
  max-width: 720px;  
  
  & > * {  
    :not(:last-child) {  
      margin-bottom: 16px;  
    }  
  }  
`;
```

Practice

React Hook?

함수형 컴포넌트에 state
를 사용하여 상태관리를
가능하도록 한다.

- ❶ useNavigate Hook 으로
해당 URL로 이동
- ❷ Button이 클릭하면 post-
write로 이동
- ❸ item을 클릭하면 /post의
하위 item.id 위치로 이동

Coding(7)

```
function MainPage(props) {  
  const navigate = useNavigate(); ❶  
  
  return (  
    <Wrapper>  
      <Container>  
        <Button title= “글 작성하기” ❷  
          onClick={() => {navigate("/post-write"); }} />  
        <PostList posts={data}  
          ❸ onClickItem={(item) => { navigate(  
            `/post/${item.id}`); }} />  
        </Container>  
      </Wrapper>  
    );  
  }  
  export default MainPage;
```



Practice

>> useState와 useNavigate
Hook 사용을 지정

>> 주요파일 импорт 및
스타일 지정

Coding(8)

```
import React, { useState } from                                     page/PostWritePage.jsx
"react";
import { useNavigate } from "react-
router-dom";
import styled from "styled-
components";
import TextInput from "../ui/TextInput";
import Button from "../ui/Button";

const Wrapper = styled.div`
  padding: 16px;
  width: calc(100% - 32px);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
`;

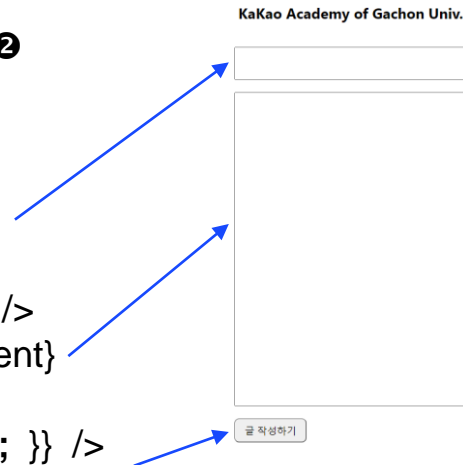
const Container = styled.div`
  width: 100%;
  max-width: 720px;
  & > * {
    :not(:last-child) {
      margin-bottom: 16px;
    }
  }
`;
```


Practice

- 1 useState Hook을 사용하여 상태관리, title 상태 변수를 지정하고, setTitle 메소드를 사용해서 title의 값을 변경, 초기 값은 null로 지정
- 2 content 상태변수와 setContent 메소드를 정의
- 3 입력이 들어와서 변화가 발생하면 setTitle과 setContent 상태관리 메소드로 해당 값 변경

Coding(8)

```
function PostWritePage(props) { c
  onst navigate = useNavigate();
  const [title, setTitle] = useState(""); ①
  const [content, setContent] = useState(""); ②
  return (
    <Wrapper>
      <Container>
        <TextInput height={20} value={title}
          onChange={(event) => {
            ③ setTitle(event.target.value); }} />
        <TextInput height={480} value={content}
          onChange={(event) => {
            setContent(event.target.value); }} />
        <Button title="글 작성하기" onClick={(e) => {console.log({title}, " ",
          {content}); navigate("/"); }} />
      </Container>
    </Wrapper>
  ); } export default PostWritePage;
```



KaKao Academy of Gachon Univ.

Practice

>> 주요파일 импорт 및
스타일 지정

>> useNavigate는
react-router-dom 패
키지의 모듈로써 페이지
이동을 위한 것

>> useParams는
react-router-dom 패
키지의 모듈로써 URL
에 포함된 파라미터에
접근할 수 있다.

>> react-router-dom
1주차 복습

Coding[9]

```
import React, { useState } from
"react";
import { useNavigate, useParams }
from "react-router-dom";
import styled from "styled-
components";
import CommentList from
"../list/CommentList"; im
port TextInput from "../ui/
TextInput";
import Button from "../ui/Button";
import data from "../data.json";
```

```
const Wrapper = styled.div`
padding: 16px;
width: calc(100% - 32px);
display: flex;
flex-direction: column;
```

page/PostViewPage.jsx

```
align-items: center;
justify-content: center; `;
```

```
const Container = styled.div`
width: 100%;
max-width: 720px;

& > * {
  :not(:last-child) {
    margin-bottom: 16px;
  }
} `;
```

Practice

>> 스타일 지정

Coding[9]

```
const PostContainer = styled.div`  
  padding: 8px 16px;  
  border: 1px solid grey;  
  border-radius: 8px;  
`;  
`;
```

```
const TitleText = styled.p`  
  font-size: 28px;  
  font-weight: 500;  
`;  
`;
```

```
const ContentText = styled.p`  
  font-size: 20px;  
  line-height: 32px;  
  white-space: pre-wrap;  
`;  
`;
```

```
const CommentLabel = styled.p`  
  font-size: 16px;  
  font-weight: 500;  
`;  
`;
```

Practice

>> useParams Hook을
사용해서 파라미터의 정
보를 가져와서 활용할 수
있음

Coding(9)

```
function PostViewPage(props) {  
  const navigate = useNavigate();  
  const { postId } = useParams();  
  const post = data.find((item) => {  
    return item.id == postId;  
  });  
  const [comment, setComment] = useState("");  
  return (  
    <Wrapper>  
      <Container>  
        <Button title="뒤로 가기"  
          onClick={() =>{ navigate("/"); }}/>  
        <PostContainer>  
          <TitleText>{post.title}</TitleText>  
          <ContentText>{post.content}</ContentText>  
        </PostContainer>  
      </Container>  
    </Wrapper>  
  );  
}
```

KaKao Academy of Gachon Univ.

뒤로 가기

차세대봇

프론트엔드와 백엔드의 우수한 클라우
드 엔지니어를 양성합니다.

댓글

실제로 개발하다보면 map함수를 진짜 많이 쓰는
것 같아요 🤖

적용해보니 코드가 정말 간결해지네요 ㅎㅎ

댓글 작성하기

Practice

Coding(9)

```
<CommentLabel>댓글</CommentLabel>
<CommentList comments={post.comments} />
<TextInput height={40} value={comment}
  onChange={(event) => {
    setComment(event.target.value);
  }} />
<Button title="댓글 작성하기"
  onClick={() => { navigate("/"); }} />
</Container>
</Wrapper>
);
}
```

→ 댓글

→ 실제로 개발하다보면 map함수를 진짜 많이 쓰는 것 같아요 🥰

→ 적용해보니 코드가 정말 간결해지네요 ㅎㅎ

→ 댓글 작성하기

export default PostViewPage;

Practice

>> 주요파일 импорт 및
스타일 지정

Coding(10)

App.js

```
import React from "react";
import {
  BrowserRouter,
  Routes,
  Route
} from "react-router-dom";
import styled from "styled-components";
// Pages
import MainPage from './page/MainPage';
import PostWritePage from './page/PostWritePage';
import PostViewPage from './page/PostViewPage';

const MainTitleText = styled.p`
  font-size: 24px;
  font-weight: bold;
  text-align: center;
`;
```

Practice

>> BrowserRouter를 사용하여 라우팅 경로를 지정합니다.

post/ :**postId** 에서 볼드 한 부분은 가변적인 값을 나타내며 data.json의 id를 나타냅니다. 경로와 관련된 컴포넌트는 PostViewPage컴포넌트와 관련 있습니다.

클릭이벤트에서 페이지 이동을 위해서 `navigate(`/post/${item.id}`)` 와 같이 코딩하면 됩니다.

Coding(10)

```
function App(props) {  
  return (  
    <BrowserRouter>  
      <MainTitleText>KaKao Academy of Gachon Univ.  
    </MainTitleText>  
    <Routes>  
      <Route index element={<MainPage />} />  
      <Route path="post-write" element={<PostWritePage />} />  
      <Route path="post/:postId" element={<PostViewPage />} />  
    </Routes>  
  </BrowserRouter>  
);  
}  
  
export default App;
```

>> npm start를 하여 결과를 확인합니다.

In-Class Exercises(2)

수업시간 내 진행하는 것을 원칙으로 합니다.

미니 블로그 업그레이드

- 글작성하기를 클릭하여 작성한 내용이 추가되도록 수정
- 개별제출
 - 결과화면 파워포인트로 작성하여 PDF 제출
 - 실행파일 첨부 (node_modules 제외)
- 평가 (10점)

KaKao Academy of Gachon Univ.

차세대 챗봇 프로젝트 설계 🤖

팀별 프로젝트 설계를 위해 자료를 수집하고
정리해드립니다. 즐거운 챗봇 수업이 되도록
합니다.

글 작성하기

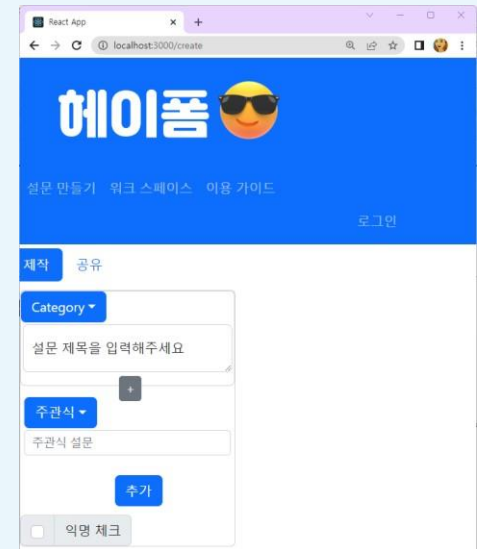
HOMEWORK

수업시간 내 진행하는 것을 원칙으로 합니다.

팀별 홈페이지 업그레이드

- 팀별 홈페이지에 컴포넌트 및 설문페이지를 추가
- 객관식, 주관식, 찬부식 유형의 재사용 컴포넌트를 제작
- 팀별 제출
 - 결과화면 및 주요 코드를 파워포인트로 작성하여 PDF 제출
 - 실행파일 첨부 (node_modules 제외)
- 평가 (10점)

[REF.]



CONNECT.

SOLVE.

CREATE. 