Interactive Comments Section

질문사항:

* React query로 데이터를 받아온 후 useState에 data 저장하려고 했지만 실패

ex) const {data} = useQuery({})

const [newArr, setNewArr] = useState([data])

return <div>{newArr.map()}</div>

=>fetch로 data 받아서 useState에 저장하여 해결

-db.json에 user 정보 post후 user가 작성한 comment만 수정,삭제,업데이트 가능하게 하는법

에러:

* localStorage is not defined

CSR과 SSR의 차이로 인한 에러!

Next.js는 client-side 렌더를 하기전에 server-side 렌더를 수행한다.

Next.js에서 제공하는 Server Side Rendering(SSR)에선 client-side에 존재하는 window, document 전역객체를 사용할 수 없다.

그래서 console.log(localStorage)만 코드에 써두고 실행시켜도

ReferenceError: localStorage is not defined

라는 에러가 나온다.

1. typeof window !== 'undefined'

페이지가 client에 마운트될 때까지 기다렸다가 localStorage에 접근해야함.

window 객체가 참조되지 않을 경우, undefined를 반환함.

if (typeof window !== 'undefined') {

localStorage.getItem('accesstoken');

}

2. useEffect

useEffect는 렌더링이 되고 난 후 실행되기때문에 server-side에서는 실행되지 않는 CSR 전용 이벤트라고 생각하자.

useEffect는 렌더링 시 실행되므로, 초기 서버 빌드 시 useEffect 내부 코드는 실행되지 않는다.

useEffect는 client-side에서만 실행되므로 localStorage에 안전하게 접근 가능하다.

* 새로운 배열이 반환되지 않는 이유

새로운 배열 반환이 필요한 경우 forEach가 아닌 map 사용

* 조건 일치하는 값 찾아서 새로운 value 넣는 법

*const updateButton = () => {*

*const arr = newArr.map((*i*) => {*

*if (i.content ===* buttonValue*) {*

*return { ...*i*, content: formik.values.content };*

*}*

*return* i*;*

*});*

*setNewArr(*arr*);*

*setCloseEdit(true);*

*}*;

* formik으로 처리한 input을 수정할 때 기존 값을 인풋에 두고 수정하게 만들고 싶을때는 formik에 초기값 설정을 해당 값으로 지정
* *const formik = useFormik({*
* *initialValues: {*
* *content: i.content,*
* *},*

formik onsubmit 후 인풋창 초기화 하고싶을 때 formik resetForm이용

*const formik = useFormik({*

*initialValues: {*

*content:* ""*,*

*},*

*validationSchema:* commentValidator*,*

*onSubmit: (*values*, {* resetForm *}) => {*

*if (*values *!==* ""*) {*

*// postComments(values);*

*setNewArr([...*newArr*,* values*]);*

*resetForm();*

*}*

*},*

*})*;

* layout page에서 QueryClientProvider client={queryClient} 사용위해서 ‘use client’ 선언,

Meta 정보 사용불가

Meta를 다른 페이지에서 만들어서 불러오기

"use client";

*import* { *Rubik* } *from* "next/font/google";

*import* "./globals.css";

*import* { *QueryClient*, *QueryClientProvider* } *from* "react-query";

*import* { *AuthContextProvider* } *from* "./context/AuthContext";

*import* { *metadata* } *from* "./util/metadata";

*const rubik = Rubik({*

*weight:* "400"*,*

*subsets: [*"latin"*],*

*})*;

*const queryClient = new QueryClient()*;

*export* *default* *function* *RootLayout*({ children }) {

*return* (

    <QueryClientProvider *client=*{queryClient}>

      <AuthContextProvider>

        <html *lang=*"en">

          <head>

            <meta *charSet=*"utf-8" />

            <meta

*name=*"viewport"

*content=*"width=device-width, initial-scale=1"

            />

            <meta *name=*"description" *content=*{*metadata*.*description*} />

            <title>{*metadata*.*title*}</title>

          </head>

          <body *className=*{`${*rubik*.*className*} bg-[#f5f6fa] `}>{children}</body>

        </html>

      </AuthContextProvider>

    </QueryClientProvider>

  );

}

* Vote button 만들 때 state를 localstorage에 저장하면 모든 컴포넌트에 import 시킨 버튼들이 같은 같으로 저장되어 있을 때

Array로 매핑중일 경우 index 번호를 props으로 전달 후 변수에 로컬스토리지 이름을 index와 함께 저장

*import* *React*, { *useEffect*, *useState* } *from* "react";

*import* { *useSearchParams* } *from* "react-router-dom";

*function* *VoteButton*({ index }) {

*const [vote, setVote] = useState(0)*;

*const localStorageKey = `vote${*index*}`*;

*const upVote = () => {*

*setVote((*prevVote*) => {*

*const newVote =* prevVote *+ 1;*

*localStorage.setItem(*localStorageKey*, newVote.toString());*

*return* newVote*;*

*});*

*}*;

*const downVote = () => {*

*setVote((*prevVote*) => {*

*if (*prevVote *> 0) {*

*const newVote =* prevVote *- 1;*

*localStorage.setItem(*localStorageKey*, newVote.toString());*

*return* newVote*;*

*}*

*return* prevVote*;*

*});*

*}*;

*useEffect*(() *=>* {

*const localVote =* JSON*.*parse*(localStorage.getItem(*localStorageKey*))*;

*if* (localVote *!=* null) {

*setVote*(localVote);

    }

  }, []);

*return* (

    <div *className=*"flex flex-col  gap-1 items-center justify-center text-[#5457b6] font-bold bg-[#f5f6fa] p-3 rounded-lg">

      <button

*className=*"font-bold text-gray-400"

*onClick=*{() *=>* *upVote*(upVote)}

      >

        +

      </button>

      {vote}

      <button

*className=*"font-bold text-gray-400"

*onClick=*{() *=>* *downVote*(downVote)}

      >

        -

      </button>

    </div>

  );

}

*export* *default* VoteButton;

*//replies의 reply comment update*

*const handleReplyButton = (*val*) => {*

*const updatedArr = newArr.map((*comment*) => {*

*if (*

*typeof comment.replies !==* "undefined" *&&*

*comment.replies !== null &&*

*comment.replies.length > 0*

*) {*

*const updatedReplies = comment.replies.map((*reply*) => {*

*if (reply.content === val.content) {*

*// 특정 댓글에 대한 처리*

*return {*

*...*reply*,*

*reply: [...(reply.reply || []), formik.values],*

*};*

*}*

*// 특정 댓글이 아닌 경우 그대로 반환*

*return* reply*;*

*});*

*// 특정 댓글이 있는 경우 해당 댓글의 replies를 업데이트*

*return { ...*comment*, replies:* updatedReplies *};*

*}*

*// replies가 없는 경우 그대로 반환*

*return* comment*;*

*});*

*// 업데이트된 댓글 목록을 state에 설정*

*setNewArr(*updatedArr*);*

*setClickReplicesCard(false);*

*}*;

*console*.*log*(newArr);

//replies의 reply삭제

*const removeReplyButton = (*val*) => {*

*let* updatedArr *= newArr.map((*comment*) => {*

*if (*

*comment.replies &&*

*comment.replies.length > 0 &&*

*comment.replies.some((*reply*) => reply.reply && reply.reply.length > 0)*

*) {*

*// 조건을 만족하는 경우에만 해당 comment를 반환*

*let* updatedReplies *= comment.replies.map((*reply*) => {*

*if (reply.reply && reply.reply.length > 0) {*

*// reply가 비어 있지 않은 경우에만 처리*

*let* filteredReply *= reply.reply.filter(*

*(*item*) => item.content !== val.content*

*);*

*return { ...*reply*, reply: [...*filteredReply*] }; // 새로운 배열로 복사*

*} else {*

*return* reply*;*

*}*

*});*

*return { ...*comment*, replies:* updatedReplies *};*

*} else {*

*// 조건을 만족하지 않는 경우에는 기존 comment를 그대로 반환*

*return* comment*;*

*}*

*});*

*// 새로운 배열을 적용*

*setNewArr(*updatedArr*);*

*// Modal 닫기*

*setDeleteModal(false);*

*console.log(*updatedArr*);*

*}*;

*const updateButton = () => {*

*console.log(formik.values);*

*console.log(*newArr*);*

*let* arr *= newArr.map((*i*) => {*

*if (*

*i.replies !== null &&*

*typeof i.replies !==* "undefined" *&&*

*i.replies.length > 0*

*) {*

*let* repliesArr *= i.replies.map((*replies*) => {*

*if (*

*replies.reply !== null &&*

*typeof replies.reply !==* "undefined" *&&*

*replies.reply.length > 0*

*) {*

*let* replyArr *= replies.reply.map((*reply*) => {*

*if (reply.content ===* buttonValue*) {*

*return {* reply*, content: formik.values.content };*

*} else {*

*return* reply*;*

*}*

*});*

*return { ...*replies*, reply:* replyArr *};*

*} else {*

*return* replies*;*

*}*

*});*

*return { ...*i*, replies: [...*repliesArr*] };*

*} else {*

*return* i*;*

*}*

*});*

*setNewArr(*arr*);*

*console.log(*arr*);*

*setCloseEdit(true);*

*}*;