CSS Framework React Bootstrap

CSS Framework를 쓰는 이유는 ?

↓

기능을 만드는데 더욱 집중하기 위해서

CSS FRAMEWORK 종류 for React JS

1. Material UI

2.React Bootstrap

3.Semantic UI

4.Ant Design

5. Materialize

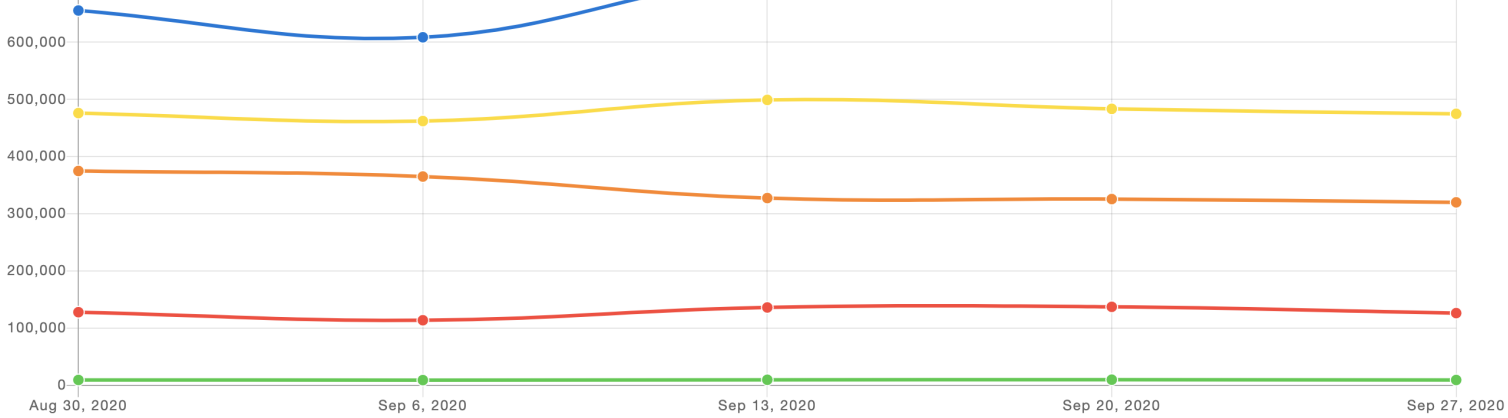
.
.
.

https://ant.design/

↓

npm install antd --save

import 'antd/dist/antd.css'



react-bootstrap.github.io

npm install react-bootstrap bootstrap --save

import 'bootstrap/dist/css/bootstrap.min.css'

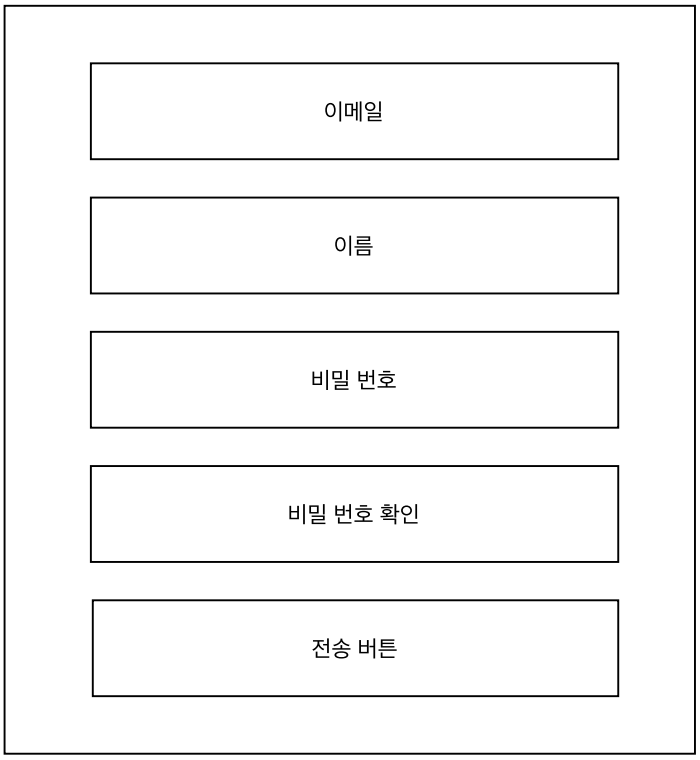
react-icons.github.io/react-icons/

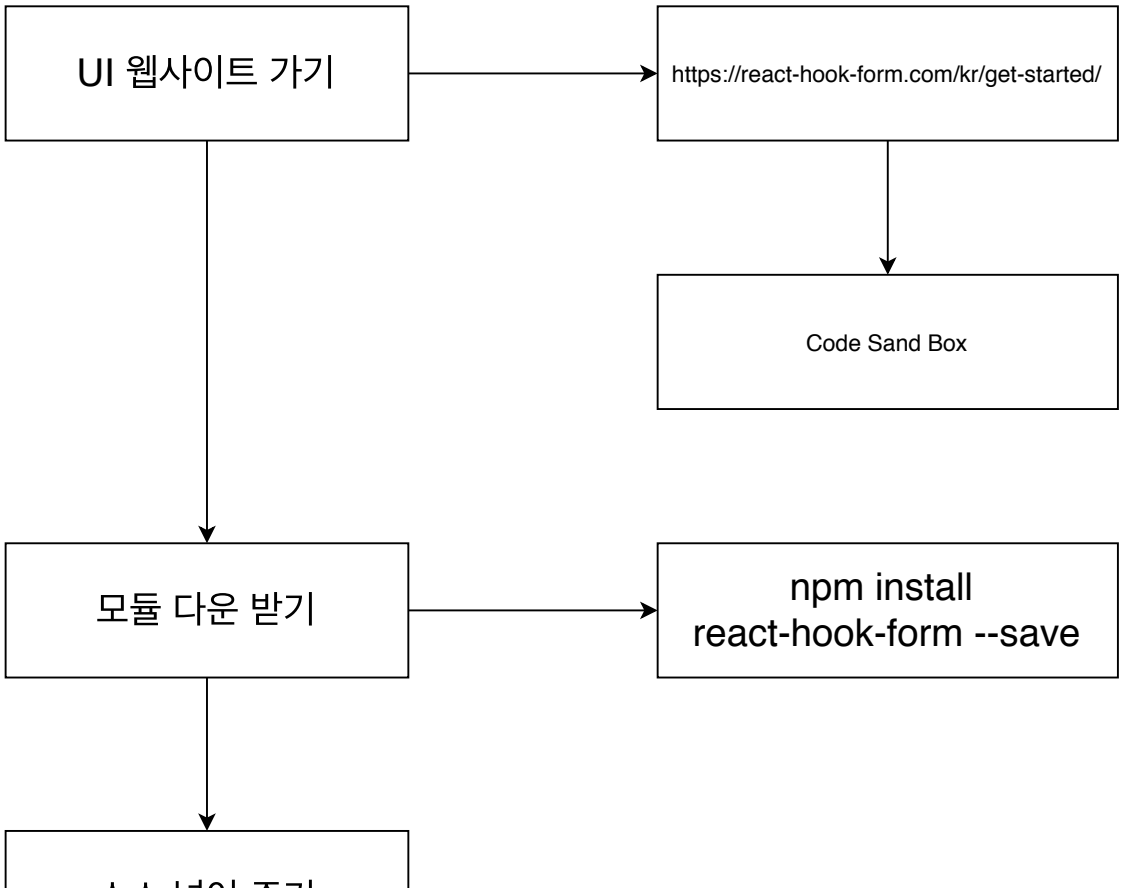
npm install react-icons --save

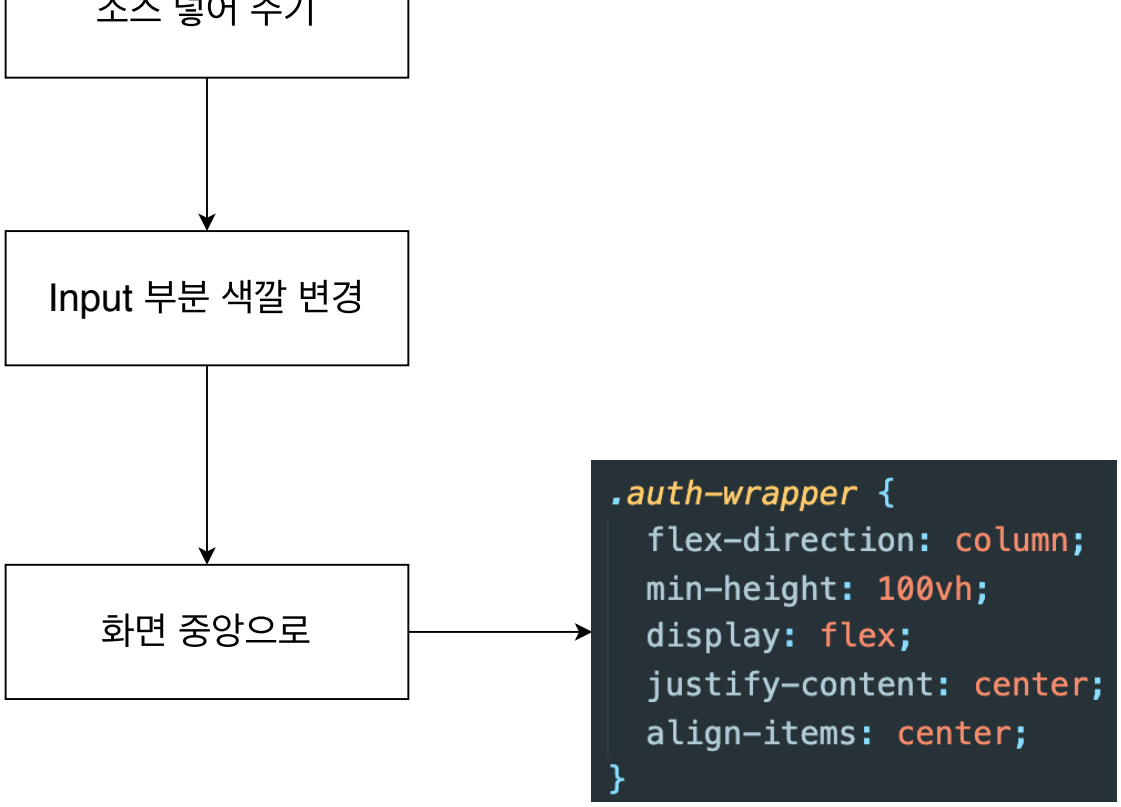
Usage

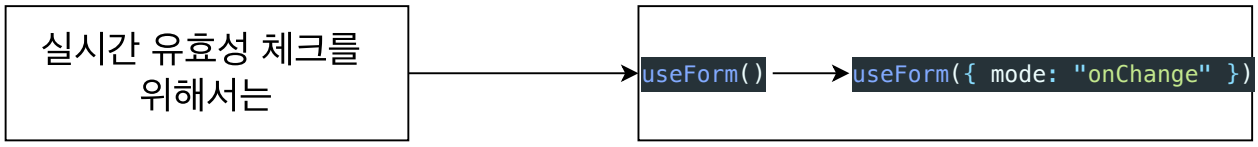
```
import { FaBeer } from 'react-icons/fa';

class Question extends React.Component {
  render() {
    return <h3> Lets go for a <FaBeer />? </h3>
  }
}
```

회원 가입 페이지 UI 만들기






Input 부분 색깔 변경
화면 중앙으로
.auth-wrapper {
flex-direction: column;
min-height: 100vh;
display: flex;
justify-content: center;
align-items: center;
}


실시간 유효성 체크를
위해서는
useForm()
useForm({ mode: "onChange" })

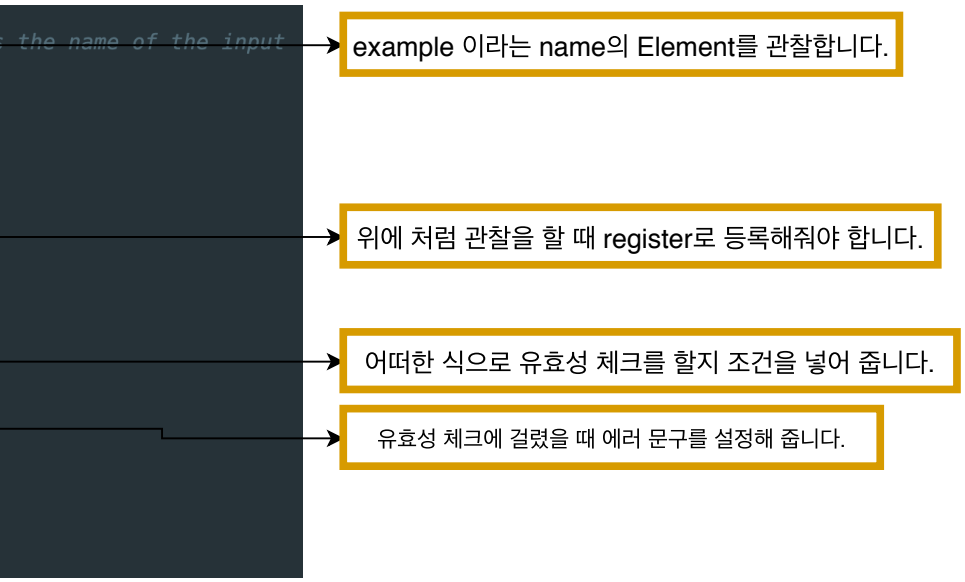
유효성 체크 with " react-hook-form " for 회원 가입

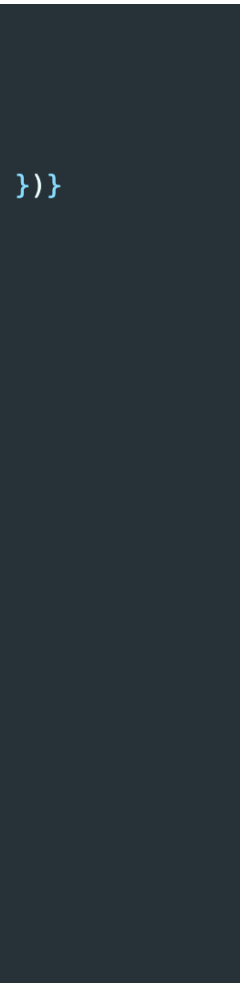
react-hook-form 사용법

```
console.log(watch("example")); // you can watch individual input by pass

return (
    <div className="auth-wrapper">
        <form onSubmit={handleSubmit(onSubmit)}>
            <label>Example</label>
            <input name="example" defaultValue="test" ref={register} />
            <label>ExampleRequired</label>
            <input
                name="exampleRequired"
                ref={register({ required: true, maxLength: 10 })}
            />
            {errors.exampleRequired && <p>This field is required</p>}
            <input type="submit" />
        </form>
    </div>
)
```

각 필드에 맞게
유효성 체크 넣어주기

```
<label>Email</label>
<input
    name="email"
    type="email"
    ref={register({ required: true, pattern: /^\S+@\S+$/i
/>
{errors.email && <p>This email field is required</p>}

<label>Name</label>
<input
    name="name"
    ref={register({ required: true, maxLength: 10 })}
/>
{errors.name && errors.name.type === "required"
    && <p> This name field is required</p>}
{errors.name && errors.name.type === "maxLength"
    && <p> Your input exceed maximum length</p>}

<label>Password</label>
<input
    name="password"
    type="password"
    ref={register({ required: true, minLength: 6 })}
/>
{errors.password && errors.password.type === "required"
```


the name of the input
example 이라는 name의 Element를 관찰합니다.
위에 처럼 관찰을 할 때 register로 등록해줘야 합니다.
어떠한 식으로 유효성 체크를 할지 조건을 넣어 줍니다.
유효성 체크에 걸렸을 때 에러 문구를 설정해 줍니다.


})}

```
    && <p> This name field is required</p>}
{errors.password && errors.password.type === "minLength"
    && <p> Password must have at least 6 characters</p>}

<label>Password Confirm</label>
<input
    name="password_confirm"
    ref={register({
        required: true,
        validate: (value) =>
            value === password.current
    })}
/>
{errors.password_confirm && errors.password_confirm.type
    && <p> This password confirm field is required</p>}
{errors.password_confirm && errors.password_confirm.type
    && <p>The passwords do not match</p>}
```

이 부분은 password와 password_confirm이 같은지 체크해주는 부분

password.current는 어떻게 나온 것인가?

useRef 란?

특정 DOM 선택하기

보통 Javascript 에서는 → getElementById, querySelector 같은 DOM Selector 함수를 사용해서 DOM을 선택

리액트에서는... → ref라는 것을 이용해서 DOM을 선택

- 클래스 컴포넌트 → React.createRef
- 함수형 컴포넌트 → useRef

DOM을 직접 선택해야 할 경우들

1. 엘리먼트 크기를 가져와야 할 때

```
=== "required"

=== "validate"
```

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }
  render() {
    return <div ref={this.myRef} />;
  }
}
```

```
function MyComponent() {
    const myRef = useRef(null);

    return (
        <div ref={myRef} />
    );
```

1. 엘리먼트 크기를 가져와야 할 때

2. 스크롤바 위치를 가져와야 할 때

3. 엘리먼트에 포커스를 설정 해줘야 할 때 등등등

UseRef를 이용해서 어떻게 Password와 Password Confirm이 같은지 알 수 있는지?

```
const password = useRef();
password.current = watch("password");
```

```
<input
    name="password_confirm"
    ref={register({
        required: true,
        validate: (value) =>
            value === password.current
    })}
/>
```

1. ref 생성

2. watch 를 이용하여 password 필드 값 가져오기

3. 가져온 password 값을 ref.current에 넣어주기

```
}
```

firebase에서 이메일로 유저 생성

Submit 버튼을 눌렀을 때
onSubmit 이벤트에 따른 함수 처리

↓

firebase에서
이메일과 비번을 이용해서
유저 생성

```
let createdUser = await firebase
    .auth()  // auth 서비스에 접근
    .createUserWithEmailAndPassword(data.email, data.password)
console.log(createdUser);
```

↓

에러 발생

```
▼POST https://www.googleapis.com/identitytoolkit/v3/relyi  auth.js:204
ngparty/signupNewUser?key=AIzaSyBkypNwgU6YDOOOcoqlm-DEitN-8p1zOd4 400
```

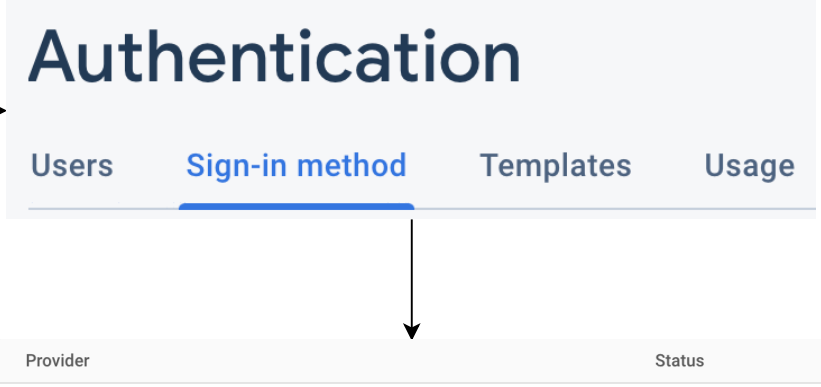
↓

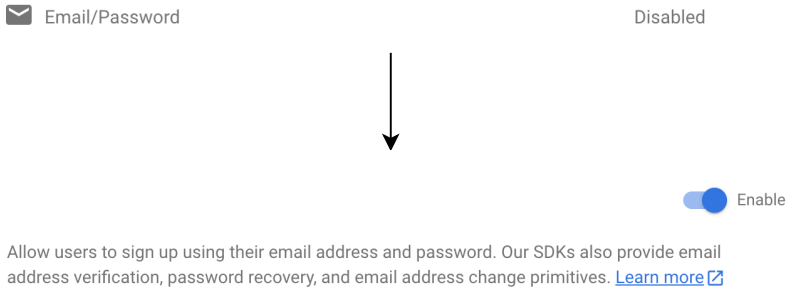
try catch 로 에러 처리

```
try {
    let createdUser = await firebase
        .auth()  // auth 서비스에 접근
        .createUserWithEmailAndPassword(data.email, data.password)
    console.log(createdUser);

} catch (error) {
    console.log(error)
    setErrorsFromSubmit(error.message)
    setTimeout(() => {
        setErrorsFromSubmit("")
    }, 5000);
}
```

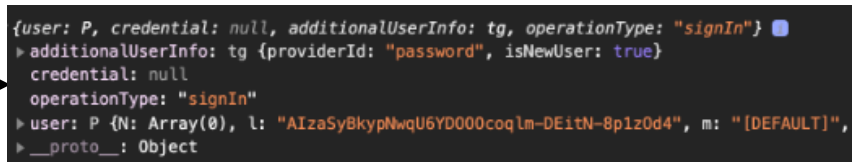
↓

firebase에서 이메일로 회원 가입 할 수 있게 허락해주기

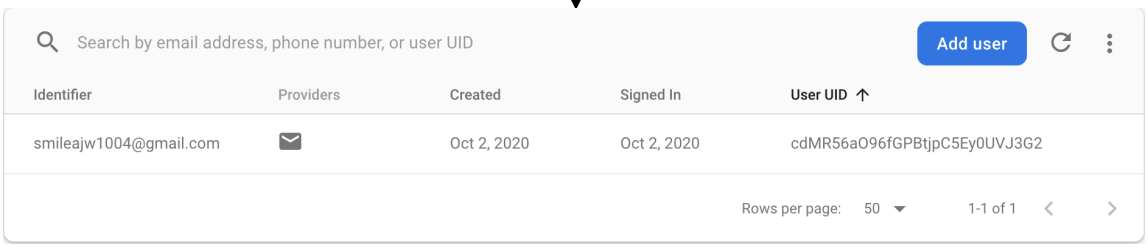




다시 로그인
firebase에서 많은 정보를
받아옴



firebase 사이트에 가서도
유저가 잘 생성 되었는지 확인



프로세스 처리 중 Button을 누르지 못하게 막기

Submit 버튼을 눌렀을 때
firebase에서
유저 생성을 처리 중 ...
이때는 버튼 더는 못 누르게 해주기

↓

loading state 생성

```
const { register, handleSubmit, watch, errors } = useForm();
const password = useRef();
password.current = watch("password");
const [errorsFromSubmit, setErrorsFromSubmit] = useState("")
const [loading, setLoading] = useState(false)
```

↓

loading state 컨트롤

```
try {
    setLoading(true)
    let createdUser = await firebase
        .auth()  // auth 서비스에 접근
        .createUserWithEmailAndPassword(data.email, data.password)
    console.log(createdUser);
    setLoading(false)
} catch (error) {
    setErrorsFromSubmit(error.message)
    setLoading(false)
    setTimeout(() => {
        setErrorsFromSubmit("")
    }, 5000);
}
```

↓

loading button UI

```
<input type="submit" disabled={loading} />
```

Firebase에 생성한 유저에 상세 정보 추가하기

firebase에서 유저를 생성할 때
Email과 password 정보만
이용해서 생성 ...
그러면 이름이나 유저 사진
같은 것은???

업데이트 해야 할 곳은 ?

```
console.log(createdUser);
```

업데이트 하는 방법은 ?

```
.updateProfile
```

```
//Firebase에서 이메일과 비밀번호로 유저 생성
let createdUser = await firebase
    .auth()  // auth 서비스에 접근
    .createUserWithEmailAndPassword(data.email, data.password)

//Firebase에서 생성한 유저에 추가 정보 입력
await createdUser.user
    .updateProfile({
        displayName: data.name,
        photoURL: `http://gravatar.com/avatar/${md5(
            createdUser.user.email
        )}?d=identicon`
    })
console.log(createdUser);
```

MD5 ?

유니크한 값을 가지려고...

```
var md5 = require('md5');

console.log(md5('message'));
```

This will print the following

78e731027d8fd50ed642340b7c9a63b3

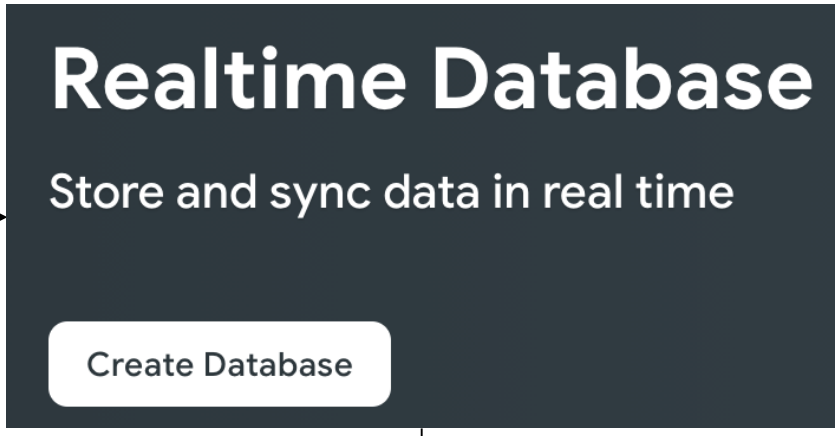
넣어준 값이 잘 들어갔는지 확인

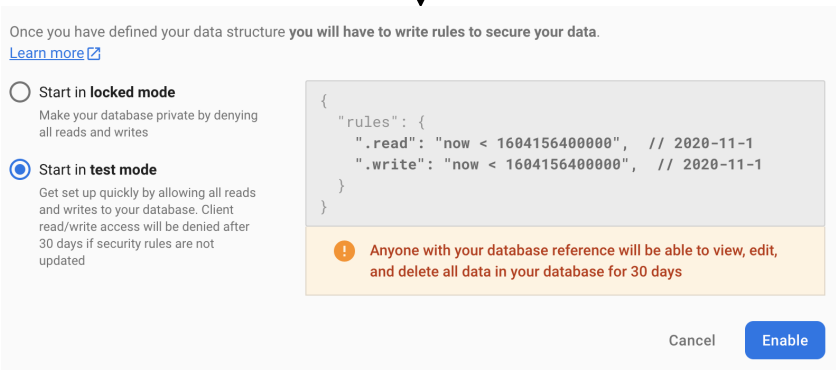
```
phoneNumber: null
photoURL: "http://gravatar.com/avatar/5ad50146659489f581b53a6120105f2c?d=identicon"
```

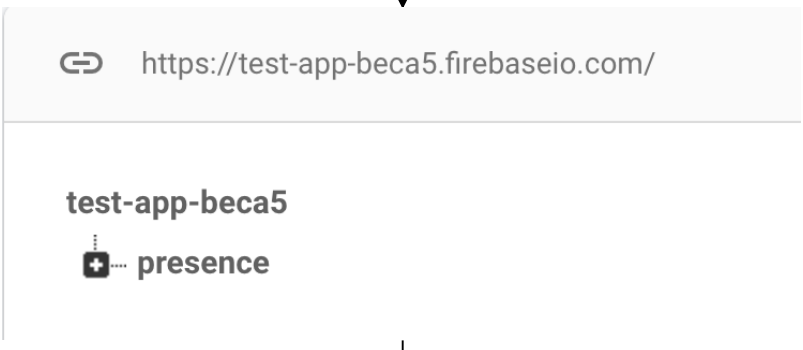
Firebase에 생성한 유저 데이터베이스에 저장

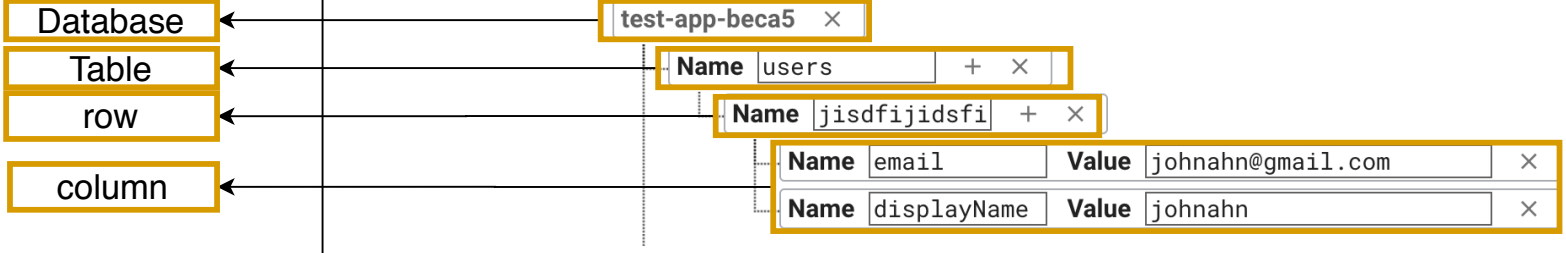
생성 된 유저를
이제는 데이터베이스에
저장해주겠습니다.

저장하는 곳은 ?









저장 하는 방법은 ?

MYSQL

INSERT INTO users (email, displayName, photoURL)
VALUES (johnahn@naver.com, johnahn, gravatar...)

MongoDB

UserModel.create({ email: "johnahn@naver.com",
displayName: "johnahn",
photoURL: "gravatar" })

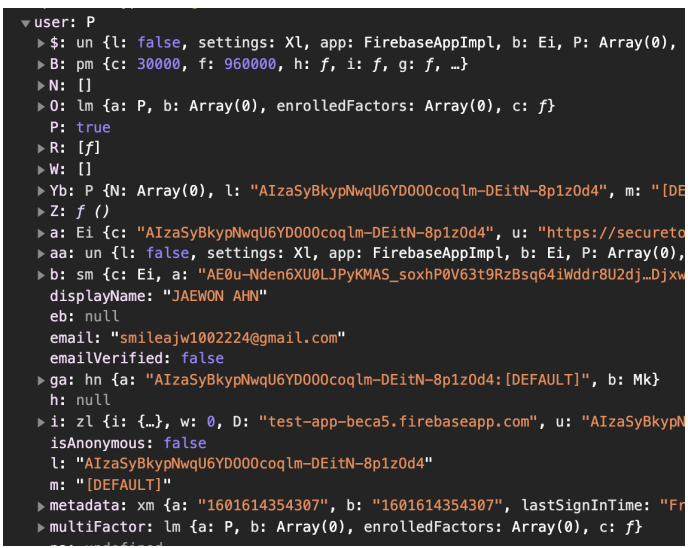
Firebase

firebase.database().ref("users").child(userId).set({
name: displayName,
image: photoURL})

실제 코드

users 테이블 없으면 생성

```
await firebase.database().ref("users").child(createdUser.user.uid).set({
    name: createdUser.user.displayName,
    image: createdUser.user.photoURL
});
```



넣어준 값이 잘 들어갔는지 확인


```
(method) firebase.database.Database.ref(path?: string): firebase.datab
ase.Reference
```

Returns a `Reference` representing the location in the Database corresponding to the provided path. If no path is provided, the `Reference` will point to the root of the Database.

@example

```
// Get a reference to the root of the Database
var rootRef = firebase.database().ref();
```

@example

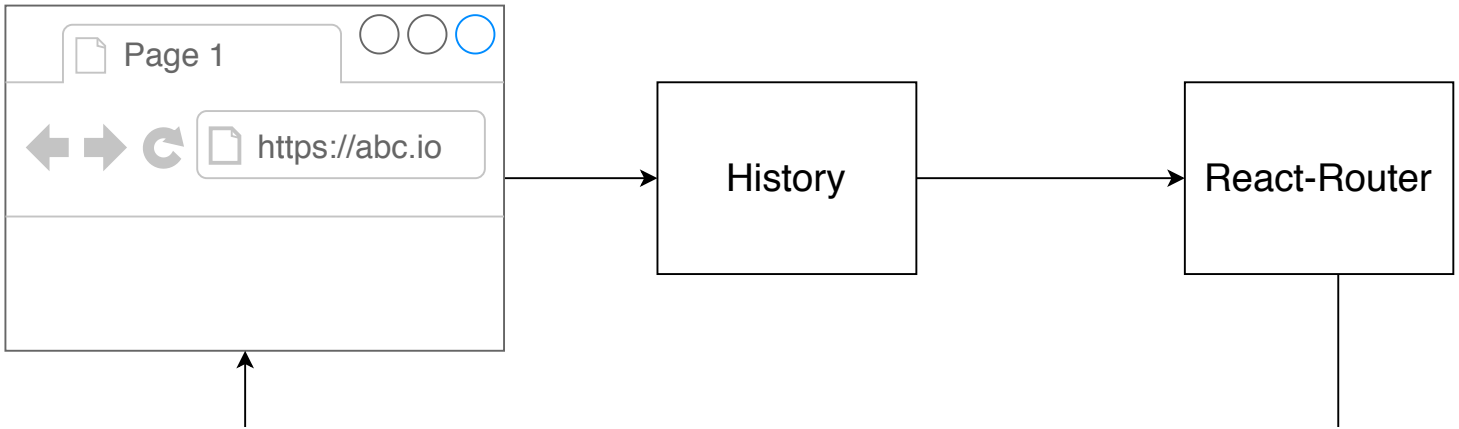
```
// Get a reference to the /users/ada node
var adaRef = firebase.database().ref("users/ada");
// The above is shorthand for the following operations:
```

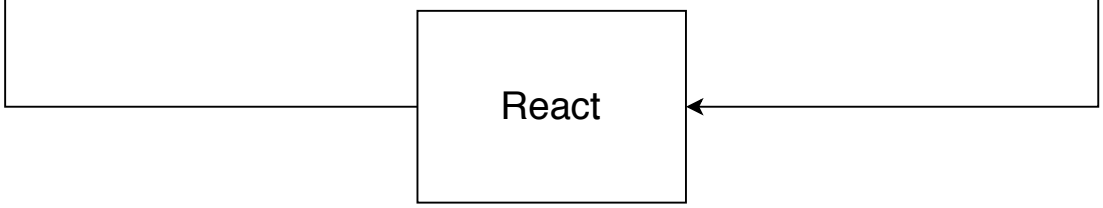
React Router Dom

이제 회원 가입 페이지를 다 만들었기 때문에 로그인 페이지로 이동해야 합니다.
페이지 이동은 어떻게 해주면 될까요 ?
원래는 <a> 태그를 이용하면 되지만 react에서는 react router dom이란것을 이용해줄수도 있습니다.

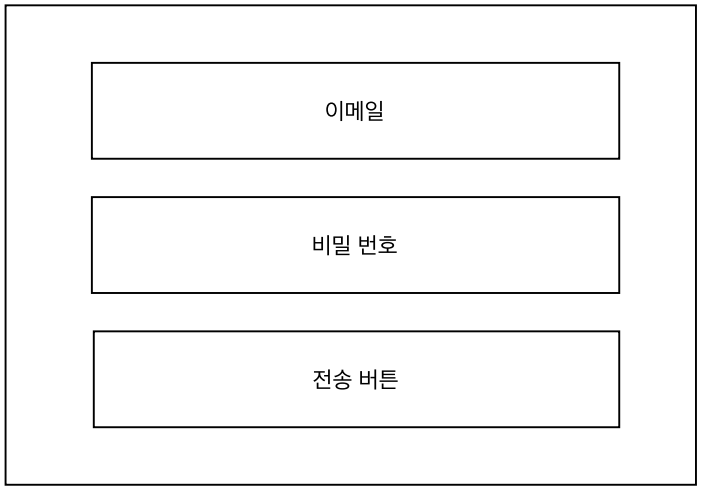
일반 <a> 태그 이용

React Router Dom 이용




React

로그인 페이지 만들기



전체적인 기능이나
UI는 회원가입 페이지와 비슷

필요한 코드들 회원가입 페이지
에서 가져오기

로그인 하기

```
try {
    setLoading(true)

    await firebase
        .auth() // auth 서비스에 접근
        .signInWithEmailAndPassword(data.email, data.password)

    setLoading(false)
} catch (error) {
    setErrorsFromSubmit(error.message)
    setLoading(false)
    setTimeout(() => {
        setErrorsFromSubmit("")
    }, 5000);
}
```

인증된 이후의 페이지 이동 & useHistory

로그인 된 유저

Chatting Page

로그인 된 사람만 이용 가능

Redirect To

Login Page
Register Page

로그인 안 된 사람만 이용 가능

구현을 위해서는

HOC 파일 새로 만들어서 구현

App.js (routing 페이지)에서 구현

인증이 이미 된 사람은
채팅 페이지로 보내주기

```
useEffect(() => {
  firebase.auth().onAuthStateChanged(user => {
    if (user) {
      props.history.push("/");
    } else {
      props.history.push("/login");
    }
  });
}, [props.history])
```

로그인 된 사람의 정보...

```
TypeError: Cannot read property 'push' of undefined
```

에러 문구

```
at Object.next (App.js:17)
at subscribe.ts:104
at subscribe.ts:233
```

History 객체가 없어서 나오는 에러

https://reactrouter.com/web/api/history

해결책으로는

Class Component

withRouter

Functional Component

useHistory

바뀐 코드

```
let history = useHistory();
console.log('history', history)
useEffect(() => {
  firebase.auth().onAuthStateChanged(user => {
    if (user) {
      history.push("/");
    } else {
      history.push("/login");
    }
  });
}, [history])
```

또 에러...

```
TypeError: Cannot read property 'push' of undefined
  at Object.next (App.js:19)
  at subscribe.ts:104
  at subscribe.ts:233
```

왜 그럴까 ...?

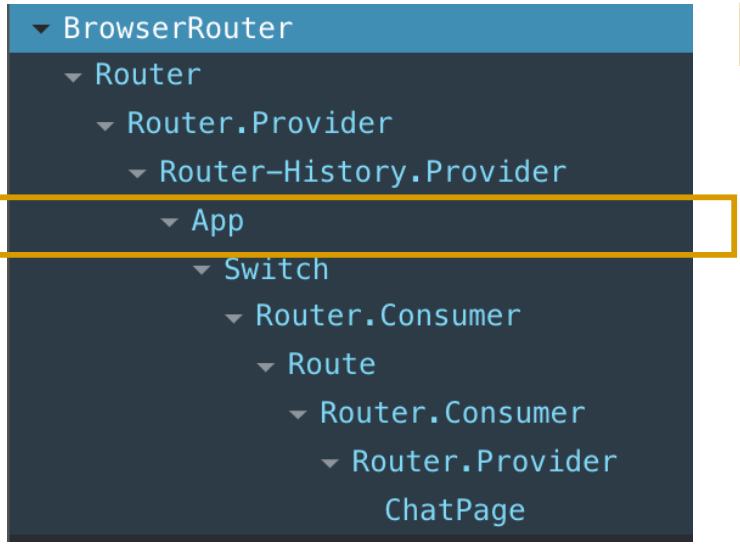
Main Component

React Router Context가 있어야지

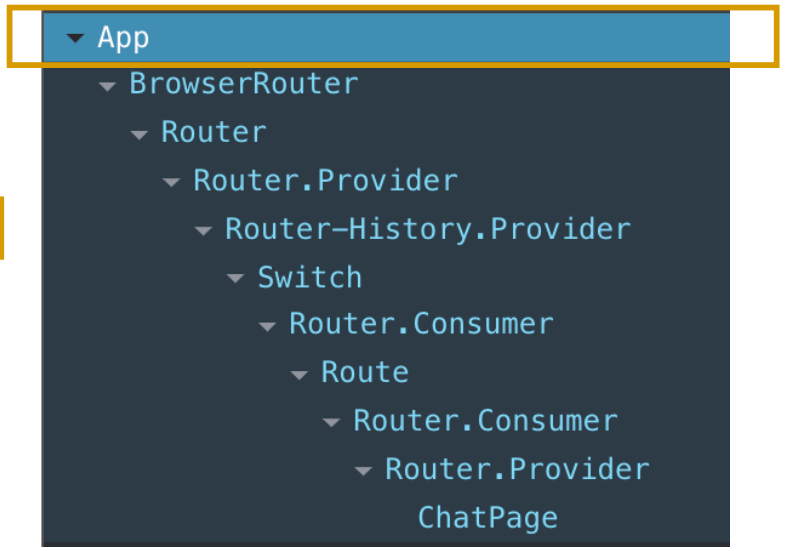
Sub-Component

useHistory 사용가능

React Router Context가 먼저 있고 난 뒤
App.js에서 useHistory를 쓸 때



App.js안에서
React Router Context가 있을 때



해결책 ! → BrowserRouter를 index.js에서 감싸주기

```
import {
  BrowserRouter as Router,
} from "react-router-dom";
ReactDOM.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Redux 스토어에 로그인 유저 정보 저장

로그인한 유저의 정보를 어디서나 쓸 수 있게 Redux 스토어에 넣어주겠습니다.

dispatch

```
import { useDispatch, useSelector } from 'react-redux';
import { setUser } from './redux/actions/user_action';
```

```
if (user) {
  history.push("/");
  dispatch(setUser(user))
} else {
  history.push("/login");
}
```

action

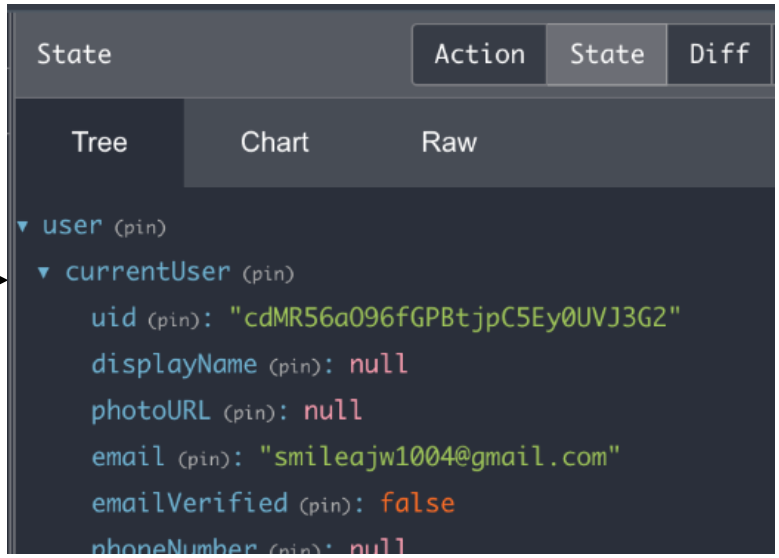
```
import { SET_USER } from './types';

export function setUser(user) {
    return {
        type: SET_USER,
        payload: user
    }
}
```

```
const initialUserState = {
    currentUser: null,
    isLoading: true
};
```

reducer

```
export default function (state = initialUserState, action) {
    switch (action.type) {
        case SET_USER:
            return {
                ...state,
                currentUser: action.payload,
                isLoading: false
            }
        default:
            return state;
    }
}
```

store



react component

```
const currentUser = useSelector(state => state.user.currentUser)
console.log('currentUser', currentUser)
```

Redux 데이터 Flow(strict unidirectional data flow)