**왜 userObj 를 계속 뿌렸는가**

1. uid를 참조하고 싶으면 authService.currentUser.uid 를 사용하면 됐음
2. 근데도 userObj로 계속 뿌린 이유는, 소스를 하나로 통일 시키기 위해
3. userObj 를 사용하는 모든 소스를 userObj로 통일 시키면, userObj에 변화가 생겼을 때 모든 관련된 소스를 리렌더링 시킬 수 있기 때문.

**하지만 전 장에서 바꿨는데 안됐다.**

1. 단순히 리렌더링이 제대로 안됐기 때문.
2. 따라서 새로고침하는 함수를 추가해보자.

//App.js

const refreshUser = () => {

setUserObj(authService.currentUser);

};

* 이후 profile로 쭉쭉 넘겨서 displayName을 바꾸고 refreahUser를 실행하도록 함

**근데도 안된다**

1. userObj의 크기가 너무 크기때문에, 리액트가 변화를 제대로 감지하고 못하는 것.
2. 해결책 1
   1. userObj에 필요한 정보만 담자.
   2. 쓴 것. uid, displayName, updateProfile

 useEffect(() => {

        authService.onAuthStateChanged((user) => {

            if (user) {

                setUserObj({

                    displayName: user.displayName,

                    uid: user.uid,

                    updateProfile: (args) => user.updateProfile(args),

                });

            }

            setInit(true);

        });

    }, []);

    const refreshUser = () => {

        const user = authService.currentUser;

            setUserObj((prev) => ({

                displayName: user.displayName,

                uid: user.uid,

                updateProfile: (args) => user.updateProfile(args),

            }));

    };

* 이렇게하면 리액트가 충분히 전부 검사할 수 있어서 리렌더링이 일어남.

1. 해결책 2
   1. userObj에 전부 담되, 다른 방법으로 refresh하자
   2. 바로 setUserObj(Object.assign({}, user)); 로 하여 새로운 객체를 만들어 user를 집어넣고, 그 새로운 객체를 userObj에 집어넣는 것이다.
   3. 간단한 방법이지만, 오류가 많아서 첫번째방법이 더 좋다