Some custom hooks used in the react firebase chat course

<https://www.udemy.com/course/the-complete-react-bootcamp-2020-w-react-hooks-firebase/learn/lecture/20705462#overview>

import { useState, useCallback, useEffect, useRef } from 'react';

import { database } from './firebase';

export *function* useModalState(*defaultValue* = false) {

*const* [isOpen, setIsOpen] = useState(defaultValue);

*const* open = useCallback(() *=>* setIsOpen(true), []);

*const* close = useCallback(() *=>* setIsOpen(false), []);

  return { isOpen, open, close };

}

export *const* useMediaQuery = *query* *=>* {

*const* [matches, setMatches] = useState(

    () *=>* window.matchMedia(query).matches

  );

  useEffect(() *=>* {

*const* queryList = window.matchMedia(query);

    setMatches(queryList.matches);

*const* listener = *evt* *=>* setMatches(evt.matches);

    queryList.addListener(listener);

    return () *=>* queryList.removeListener(listener);

  }, [query]);

  return matches;

};

export *function* usePresence(*uid*) {

*const* [presence, setPresence] = useState(null);

  useEffect(() *=>* {

*const* userStatusRef = database.ref(`/status/${uid}`);

    userStatusRef.on('value', *snap* *=>* {

      if (snap.exists()) {

*const* data = snap.val();

        setPresence(data);

      }

    });

    return () *=>* {

      userStatusRef.off();

    };

  }, [uid]);

  return presence;

}

export *function* useHover() {

*const* [value, setValue] = useState(false);

*const* ref = useRef(null);

*const* handleMouseOver = () *=>* setValue(true);

*const* handleMouseOut = () *=>* setValue(false);

  useEffect(

    () *=>* {

*const* node = ref.current;

      if (node) {

        node.addEventListener('mouseover', handleMouseOver);

        node.addEventListener('mouseout', handleMouseOut);

      }

      return () *=>* {

        node.removeEventListener('mouseover', handleMouseOver);

        node.removeEventListener('mouseout', handleMouseOut);

      };

    },

    // eslint-disable-next-line react-hooks/exhaustive-deps

    [ref.current] // Recall only if ref changes

  );

  return [ref, value];

}