Image upload to firebase

import React, { useState, useRef } from 'react';

import { Modal, Button, Alert } from 'rsuite';

//https://www.npmjs.com/package/react-avatar-editor

import AvatarEditor from 'react-avatar-editor';

import { useModalState } from '../../misc/custom-hooks';

import { storage, database } from '../../misc/firebase';

import { useProfile } from '../../context/profile.context';

import ProfileAvatar from '../ProfileAvatar';

import { getUserUpdates } from '../../misc/helpers';

*const* fileInputTypes = '.png, .jpeg, .jpg';

*const* acceptedFileTypes = ['image/png', 'image/jpeg', 'image/pjpeg'];

*const* isValidFile = *file* *=>* acceptedFileTypes.includes(file.type);

// toBlob method only accepts callbacks, no promises so converting

// to a new promise object

//https://www.udemy.com/course/the-complete-react-bootcamp-2020-w-react-hooks-firebase/learn/lecture/20712466#overview

*const* getBlob = *canvas* *=>* {

  // to create actual promis, need to put callback inside that

  // recives 2 arguments - call this method to actually resolve or reject

  // the promis

  return new Promise((*resolve*, *reject*) *=>* {

    canvas.toBlob(*blob* *=>* {

      if (blob) {

        resolve(blob);

      } else {

        reject(new Error('File process error'));

      }

    });

  });

};

*const* AvatarUploadBtn = () *=>* {

*const* { isOpen, open, close } = useModalState();

*const* { profile } = useProfile();

*const* [img, setImg] = useState(null);

*const* [isLoading, setIsLoading] = useState(false);

*const* avatarEditorRef = useRef();

*const* onFileInputChange = *ev* *=>* {

*const* currFiles = ev.target.files;

    if (currFiles.length === 1) {

*const* file = currFiles[0];

      if (isValidFile(file)) {

        setImg(file);

        open();

      } else {

        Alert.warning(`Wrong file type ${file.type}`, 4000);

      }

    }

  };

*const* onUploadClick = async () *=>* {

    // canvas is coming from the package react-avatar-editor

*const* canvas = avatarEditorRef.current.getImageScaledToCanvas();

    setIsLoading(true);

    try {

      // now that we converted a callback  based method to a promise

      // we can use in this try catch block

*const* blob = await getBlob(canvas);

*const* avatarFileRef = storage

        //Firebase syntax

        .ref(`/profile/${profile.uid}`)

        .child('avatar');

      // need to convert image file to blob format before it can be uploaded to firebase

*const* uploadAvatarResult = await avatarFileRef.put(blob, {

        // cacheControl header actualy cache images in browser for 3 days .

        //https://firebase.google.com/docs/reference/js/firebase.storage.SettableMetadata

        cacheControl: `public, max-age=${3600 \* 24 \* 3}`,

      });

*const* downloadUrl = await uploadAvatarResult.ref.getDownloadURL();

*const* updates = await getUserUpdates(

        profile.uid,

        'avatar',

        downloadUrl,

        database

      );

      await database.ref().update(updates);

      setIsLoading(false);

      Alert.info('Avatar has been uploaded', 4000);

    } catch (err) {

      setIsLoading(false);

      Alert.error(err.message, 4000);

    }

  };

  return (

    <div className="mt-3 text-center">

      <ProfileAvatar

        src={profile.avatar}

        name={profile.name}

        className="width-200 height-200 img-fullsize font-huge"

      />

      <div>

        <label

          htmlFor="avatar-upload"

          className="d-block cursor-pointer padded"

        >

          Select new avatar

          <input

            id="avatar-upload"

            type="file"

            className="d-none"

            accept={fileInputTypes}

            onChange={onFileInputChange}

          />

        </label>

        <Modal show={isOpen} onHide={close}>

          <Modal.Header>

            <Modal.Title>Adjust and upload new avatar</Modal.Title>

          </Modal.Header>

          <Modal.Body>

            <div className="d-flex justify-content-center align-items-center h-100">

              {img && (

                <AvatarEditor

                  ref={avatarEditorRef}

                  image={img}

                  width={200}

                  height={200}

                  border={10}

                  borderRadius={100}

                  rotate={0}

                />

              )}

            </div>

          </Modal.Body>

          <Modal.Footer>

            <Button

              block

              appearance="ghost"

              onClick={onUploadClick}

              disabled={isLoading}

            >

              Upload new avatar

            </Button>

          </Modal.Footer>

        </Modal>

      </div>

    </div>

  );

};

export default AvatarUploadBtn;