index.tsx

import ReactDom from 'react-dom'

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

import \* as esbuild from 'esbuild-wasm'

import { unpkgPathPlugin } from './plugins/unpkg-path-plugin'; //second set of brackets if for void function

const App =() =>

{

  const [input,setInput] = useState("");

  const [code,setCode] = useState("");

  const ref = useRef<any>();

  // We need to use Service inside onClick so we can do this in two ways

  // Either by use State so it is accessible everywhere or we can use useref

  const startService = async() =>

  {

   ref.current = await esbuild.startService({

   worker:true,

   wasmURL:'./esbuild.wasm'

});

}

  useEffect(()=>

  {

    startService();

  },[]);

  const onClick =async()=>

  {

    if(!ref.current)  //service is ready or not

    {

      return;

    }

    const result=await ref.current.build({

      entryPoints:['index.js'],

      bundle:true,

      write:false,

      plugins:[unpkgPathPlugin(input)],

      define:{'process.env.NODE\_ENV':'"production"',   //we are getting this error

      //in require('react') so we are doing this we are replacing it with string production indside single quote inside double quote

      global:'window',

    },

    })

    console.log(result);

    setCode(result.outputFiles[0].text);

    // const result = await ref.current.transform(input,{

    //   loader:'jsx',  //which language we are converting jsx/typescript

    //   target:'es2015'

    // });

   // setCode(result.code);     //not handle any import or bundling only transpiling

  }

  return <div><textarea onChange={(e)=>

  {

    setInput(e.target.value);

  }}>

    </textarea>

    <div>

      <button onClick={onClick}>Submit</button>

    </div>

<pre>{code}</pre>

  </div>

}

ReactDom.render(<App/>,document.querySelector("#root"));

Unpkgpathplugin

import \* as esbuild from 'esbuild-wasm';

import axios from 'axios'

import localforage from 'localforage';

/\* Whatever instance we are creating here we will need to call that instance for

that key , we can create multiple instances using diff names\*/

const fileCache = localforage.createInstance({

  name:'filecache'

});

(async()=>

{

  await fileCache.setItem('color','red');

  const color = await fileCache.getItem('color');

  const c = await localforage.getItem('color');

  console.log(color);

  console.log(c);

})()

export const unpkgPathPlugin = (inputCode:string) => {

  return {

    name: 'unpkg-path-plugin',

  /\* onResolve and onLoad has a regular expression which tells

  when to run onResolve and when to run onLoad \*\*/

  // Suppose we want to use onLoad function only on the namespace a given in

  // onResolve then in onLoad({filter:/.\*/,namespace:'a'})

  /\* If I give namespace 'b' in onLoad then it will break as the files on

  onResolve has a namespace a \*/

    setup(build: esbuild.PluginBuild) {

      build.onResolve({ filter: /.\*/ }, async (args: any) => {

        console.log('onResolve', args);

        if(args.path==='index.js')

        {

          return {path:args.path,namespace:'a'};

        }

      //always use bracket after return in same line

      /\*return

      {

      }  won't work \*/

      if(args.path.includes('./')||args.path.includes('../'))

      {

        return{

          namespace: 'a',

          path: new URL(args.path,'https://www.unpkg.com'+args.resolveDir+'/').href

        };

      }

    return {namespace:'a',path:`https://unpkg.com/${args.path}`};

    //instead of hard code just return

    // else if(args.path === 'tiny-test-pkg')

    // {

    //   return {path:'https://unpkg.com/tiny-test-pkg@1.0.0/index.js',namespace:'a'}

    // }

     });

      build.onLoad({ filter: /.\*/ }, async (args: any) => {

        console.log('onLoad', args);

/\* import message from 'tiny-test-pkg' does not work as esbuild has no idea

how to fetch this package  \*/

        if (args.path === 'index.js') {

          return {

            loader: 'jsx',

            contents: inputCode,

          };

/\* we can write this is es build as well like import React,{useState} from 'react \*\*/

        /\* I am trying to import package  from unpkg tiny testpkg

        } else {

          return {

            loader: 'jsx',

            contents: 'export default "hi there!"',

          };

       } \*\*/

      }

       /\* Here typescript doesn't understand the result getItem will return

      so to remove this error const cachedResult = await fileCache.getItem(args.path); we use

      <> \*/

     const cachedResult = await fileCache.getItem<esbuild.OnLoadResult>(args.path);

         if(cachedResult)

              return cachedResult;

      //what axios returns from a api

      const{data,request} = await axios.get(args.path);

      //store response if not been visited otherwise return from cache

      console.log(request);

      const result:esbuild.OnLoadResult={

        loader:'jsx',

        contents:data,

        resolveDir:new URL('./',request.responseURL).pathname,   //gives /nested-test-pkg/src/

      }

      await fileCache.setItem(args.path,result);

      return result;

      });

    },

  };

};