**Jegyzet**

##### Alap parancsok

parancsok -npm install

npx create-react-app <nev>

cd <nev megadása>

npm run start vagy npm start az indításhoz (lehet több portot futtatni egy szerre)

##### App js

return belülre html felépítéssel és kapcsos jellel lehet átváltani java scriptre vagy react részeket ezután hozzuk létre egy mappát (ajánlott „komponesek” nevezni mert az a hivatalos neve) az src belül majd későbbiekben itt hozzuk létre a funkciót ahhoz, hogy kattintasra vagy valami egyébre meg változtassuk a honlapot (ezt a funkciót tovább kell küldeni minden részen keresztül )

return (

    <div className="App">

      <header className="App-header">

      <h1>LightOut</h1>

      </header>

      <article>

        <Jatekter list={LAMPAMODELL.GetLista()} katt={katt} />

      </article>

      <footer>Dobrocsi Kornél</footer>

    </div>

  );

Itt hívjuk meg a többi részt „import”-tal amit a meg felelő részeken példányosítunk

pl : a return belül különlegesen kell csinálni a ez fölött látható módon.

import { useState } from 'react';

import './App.css';

import Jatekter from './komponensek/jatekter';

import LampaModell from './komponensek/LampaModell';

##### Komponesek

komponensen belül létre hozzuk a részeket példa képpen a játékteret ahol az általunk be adott listát props ként érjük el(a név változtatható) majd a map segítségégével végig megyünk a listán és kisebb elemekre szedjük szét ezen kívül . Ha ezen belül létre hoztunk még elemet amivel szeretnénk valamilyen funkciót készíteni akkor itt is át kell vezetni a kapcsolatot.

function Jatekter(props){

    function katt(index) {

        props.katt(index);

    }

    return(

        <div className="jatekter">

        {props.list.map((elem,index)=>{

            return <Elem ertek={elem} index={index} key={index} katt={katt}  />

        })}

        </div>

    )

}

A legbelsőbb részben hozzuk létre a funkciót pontosság és használ hatóság kedvéért illetve itt is meg adhatjuk milyen információ van benne

function Elem(props){

    function katt(){

        props.katt(props.index)

    }

return(

    <div className="elem" onClick={katt}>

        <p>{props.ertek}</p>

    </div>

)

egy külön modelben ajánlatos elvégezni bármilyen bonyolultabb műveletet és azt meg példányosítani az App.js.

const tmodell=new TModell()

function App() {

  const [list,setList]=useState(tmodell.getList());

  function katt(index){

    tmodell.setAllapot()

    setList(tmodell.getList())

    console.log(list)

  }

##### BrowserRouter

A borwserRouter az app.js-ben kell létrehozni a return belül ahol a Routon belül adjuk meg a path és element segítségével ehezz bekell importálni „import { BrowserRouter, Routes, Route } from "react-router-dom"” és a „ReactDom from React-doom/client” is (NoPage a nem elérhetőséget jelzi.)

ez nem mutathat az src kivül

<BrowserRouter>

      <Routes>

        <Route path="/" element={<Layout />}>

          <Route index element={<jatekok />} />

          <Route path="TicTacToe" element={<TicTacToe />} />

          <Route path="LightOutApp" element={<LightOutApp />} />

          <Route path="\*" element={<NoPage />} />

        </Route>

      </Routes>

    </BrowserRouter>

ezután létre kell hozni a layout amiben linkként kell majd megadni a következő részeket és azt hogy miképp írja ki a Linkeket.

<nav>

        <ul>

          <li>

            <Link to="/">Játékok</Link>

          </li>

          <li>

            <Link to="/TicTacToe">Tic Tac Toe</Link>

          </li>

          <li>

            <Link to="/LightOutApp">Light On</Link>

          </li>

        </ul>

      </nav>

      <Outlet />

ezután létre hozzuk a pages mappát ahová tesszük a különböző oldalakat amit szeretnénk linkelni a feladatokat illetve a NoPage is és hogy mit írjon ki.