import React from "https://cdn.skypack.dev/react"

import ReactDOM from "https://cdn.skypack.dev/react-dom"

import { Provider, connect } from "https://cdn.skypack.dev/react-redux"

import { createStore, combineReducers, applyMiddleware } from "https://cdn.skypack.dev/redux"

import PropTypes from "https://cdn.skypack.dev/prop-types"

import redux from "https://cdn.skypack.dev/redux"

let interval = null;

const estado = {

break: 5,

sessionMinutes: 25,

minutes: 25,

seconds: 0,

session: true,

estado: true

}

function pad(d) { return (d < 10) ? '0' + d.toString() : d.toString(); }

const DOWNBREAK = 'DOWNBREAK', UPBREAK = 'UPBREAK', DOWNSESSION = 'DOWNSESSION', UPSESSION = 'UPSESSION', STARTSTOP = 'STARTSTOP', RESET = 'RESET', CHANGESECOND = 'CHANGESECOND';

const downBreak = (state) => {

return {

type: DOWNBREAK,

value: state // ojo que aca va state!! no state.break

}

};

const upBreak = (state) => {

return {

type: UPBREAK,

value: state

}

};

const downSession = (state) => {

return {

type: DOWNSESSION,

value: state

}

};

const upSession = (state) => {

return {

type: UPSESSION,

value: state

}

};

const startStop = (state) => {

return {

type: STARTSTOP,

value: state

}

};

const reset = (state) => {

return {

type: RESET,

value: state

}

};

const changeSecond = (state) => {

return {

type: CHANGESECOND,

value: state

}

};

function tratarCronometro(state, action) {

//Se termina la sesión o el brake

if (action.value == 0 && state.minutes == 0) {

let beep = new Audio('https://media.geeksforgeeks.org/wp-content/uploads/20190531135120/beep.mp3');

beep.id = "beep";

beep.play();

if (state.session == true)

return { break: state.break, sessionMinutes: state.sessionMinutes, minutes: state.break, seconds: 0, session: !state.session, estado: state.estado }; // comienza brake

else return { break: state.break, sessionMinutes: state.sessionMinutes, minutes: state.sessionMinutes, seconds: 0, session: !state.session, estado: state.estado }; // comienza sesión

}

//Continúo con el brake o la sesión

if (action.value == 0) return { break: state.break, sessionMinutes: state.sessionMinutes, minutes: state.minutes-1, seconds: 59, session: state.session, estado: state.estado };

else return { break: state.break, sessionMinutes: state.sessionMinutes, minutes: state.minutes, seconds: state.seconds-1, session: state.session, estado: state.estado };

}

const clocksReducer = (state = estado, action) => {

switch (action.type) {

case DOWNBREAK:

if (action.value >= 1) return { break: action.value-1, sessionMinutes: state.sessionMinutes, minutes: state.minutes, seconds: state.seconds, session: state.session, estado: state.estado }; else return state;

case UPBREAK:

if (action.value <= 59) return { break: action.value+1, sessionMinutes: state.sessionMinutes, minutes: state.minutes, seconds: state.seconds, session: state.session, estado: state.estado }; else return state;

case DOWNSESSION:

if (action.value >= 1) return { break: state.break, sessionMinutes: action.value-1, minutes: action.value-1, seconds: state.seconds, session: state.session, estado: state.estado }; else return state;

case UPSESSION:

if (action.value <= 59) return { break: state.break, sessionMinutes: action.value+1, minutes: action.value+1, seconds: state.seconds, session: state.session, estado: state.estado }; else return state;

case STARTSTOP:

return { break: state.break, minutes: state.minutes, sessionMinutes: state.sessionMinutes, seconds: state.seconds, session: state.session, estado: !action.value };

case CHANGESECOND:

return tratarCronometro(state, action);

case RESET:

return { break: 5, sessionMinutes: 25, minutes: 25, seconds: 0, session: true, estado: true };

default:

return state;

}

};

class Tecla extends React.Component {

constructor(props) {

super(props);

this.state = {

on: true,

text: "Start"

}

}

render() {

return (

<div>

<button id={this.props.id} onClick = {this.props.onClick}>{this.props.value}</button>

</div>

);

}

}

class Display extends React.Component {

constructor(props) {

super(props);

document.addEventListener('imprimir', (texto) => {this.imprimirPantalla(texto)});

}

imprimirPantalla(texto) {

if (this.props.id == "time-left") {

this.props.text = 9;//texto.detail.text;

this.forceUpdate();

}

}

pad(d) {

return (d < 10) ? '0' + d.toString() : d.toString();

}

render() {

return (

<div>

<label id={this.props.id}>{this.props.text}</label>

</div>

);

}

}

class Panel extends React.Component {

constructor(props) {

super(props);

this.state = {

}

this.iniciarFrenar = this.iniciarFrenar.bind(this);

this.iniciarReset = this.iniciarReset.bind(this);

}

iniciarFrenar() {

this.props.submitStartStop(this.props.estado);

if (this.props.estado == true) {

interval = setInterval(() => {

this.props.submitChangeSecond(this.props.seconds);}, 1000);

}

else interval = clearInterval(interval);

}

iniciarReset() {

if (this.props.estado == false) {

this.props.submitReset();

interval = clearInterval(interval);

}

}

render() {

return (

<div id="container">

<div id="panelBreakLength">

<Display id="break-label" text="Break Length"/>

<Display id="break-length" text={this.props.break}/>

<Tecla id="break-decrement" value="Down Break" onClick={() => this.props.submitDownBreak(this.props.break)}/>

<Tecla id="break-increment" value="Up Break" onClick={() => this.props.submitUpBreak(this.props.break)}/>

</div>

<div id="panelSessionLength">

<Display id="session-label" text="Session Length"/>

<Display id="session-length" text={this.props.sessionMinutes}/>

<Tecla id="session-decrement" value="Down Session" onClick={() => this.props.submitDownSession(this.props.sessionMinutes)}/>

<Tecla id="session-increment" value="Up Session" onClick={() => this.props.submitUpSession(this.props.sessionMinutes)}/>

</div>

<div id="panelSession">

<Display id="timer-label" text={this.props.session ? "Session" : "Break"}/>

<Display id="time-left" text={this.props.minutes + ':' + pad(this.props.seconds)}/>

<Tecla id="start-stop" value={this.props.estado ? "Start" : "Stop"} onClick={this.iniciarFrenar}/>

<Tecla id="reset" value="Reset" onClick={this.iniciarReset}/>

</div>

</div>

);

}

};

const mapStateToProps = (state) => {

return {

break: state.break,

sessionMinutes: state.sessionMinutes,

minutes: state.minutes,

seconds: state.seconds,

session: state.session,

estado: state.estado

}

};

const mapDispatchToProps = (dispatch) => {

return {

submitDownBreak: (state) => {

dispatch(downBreak(state))

},

submitUpBreak: (state) => {

dispatch(upBreak(state))

},

submitDownSession: (state) => {

dispatch(downSession(state))

},

submitUpSession: (state) => {

dispatch(upSession(state))

},

submitChangeSecond: (state) => {

dispatch(changeSecond(state))

},

submitStartStop: (state) => {

dispatch(startStop(state))

},

submitReset: (state) => {

dispatch(reset(state))

}

}

};

const Container = connect(mapStateToProps, mapDispatchToProps)(Panel);

const store = createStore(clocksReducer);

class AppWrapper extends React.Component {

render() {

return (

<Provider store={store}>

<Container/>

</Provider>

);

}

};

ReactDOM.render(<AppWrapper/>, document.getElementById('root'));