React&react-Hooks

- useRef
- React Portals
- useEffect

useRef

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

import Card from '../UI/Card';
import Button from '../UI/Button';
import ErrorModal from '../UI/ErrorModal';
import Wrapper from '../Helpers/Wrapper';
import classes from './AddUser.module.css';

const AddUser = (props) => {
  const nameInputRef = useRef();
  const ageInputRef = useRef();

  const [error, setError] = useState();

  const addUserHandler = (event) => {
    event.preventDefault();
    const enteredName = nameInputRef.current.value;
    const enteredUserAge = ageInputRef.current.value;
    if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {
      setError({
        title: 'Invalid input',
        message: 'Please enter a valid name and age (non-empty values).',
      });
      return;
    }
    if (+enteredUserAge < 1) {
      setError({
        title: 'Invalid age',
        message: 'Please enter a valid age (> 0).',
      });
      return;
    }
    props.onAddUser(enteredName, enteredUserAge);
    nameInputRef.current.value = '';
    ageInputRef.current.value = '';
  };

  const errorHandler = () => {
    setError(null);
  };

  return (
    <Wrapper>
      {error && (
        <ErrorModal
          title={error.title}
          message={error.message}
          onConfirm={errorHandler}
        />
      )}
      <Card className={classes.input}>
        <form onSubmit={addUserHandler}>
          <label htmlFor="username">Username</label>
          <input id="username" type="text" ref={nameInputRef} />
          <label htmlFor="age">Age (Years)</label>
          <input id="age" type="number" ref={ageInputRef} />
          <Button type="submit">Add User</Button>
        </form>
      </Card>
    </Wrapper>
  );
};

export default AddUser;
```

- init useRef
- connect useRef with input
- Reset input
- Wrapper input component by React.forWwordRef
- reAdd ref={ref} to attribute component

```jsx
import React, { useRef } from 'react';
import Input from '../../UI/Input';
import classes from './MealItemForm.module.css';

const MealItemForm = ( props ) => {
  const amountInputRef = useRef();

  const submitHandler = ( event ) => {
    event.preventDefault();
    const enteredAmount = +( amountInputRef.current.value );

  };

  return (
    <form className={ classes.form } onSubmit={ submitHandler }>
      <Input
        ref={ amountInputRef }
        label='Amount'
        input={ {
          id: 'amount_' + props.id, // this changed!
          type: 'number',
          min: '1',
          max: '5',
          step: '1',
          defaultValue: '1',
        } }
      />
      <button type='submit'> +Add</button>
    </form>
  );
};

export default MealItemForm;
```

```jsx
import React from 'react';

import classes from './Input.module.css';

const Input = React.forwardRef( ( props, ref ) => {
  return (
    <div className={ classes.input }>
      <label htmlFor={ props.input.id } >{ props.label }</label>
      <input ref={ ref } { ...props.input } />
    </div>
  );
} );

export default Input;
```

React Portals

```html
<body>
  <div id="overlay"></div>
  <div id="root"></div>
</body>
```

```
import React, { Fragment } from 'react';
import classes from './Modal.module.css';
import ReactDOM from 'react-dom';

const Backdrop = ( props ) => {
  return <div className={ classes.backdrop } />;
};

const ModalOverlay = ( props ) => {
  return <div className={ classes.modal }>
    <div className={ classes.content }>
      { props.children }
    </div>
  </div>;
};

const portalElement = document.getElementById( 'overlays' );

const Modal = ( props ) => {
  return (
    <Fragment>
      { ReactDOM.createPortal( <Backdrop />, portalElement ) }
      { ReactDOM.createPortal( <ModalOverlay>
        { props.children }
      </ModalOverlay>, portalElement ) }
    </Fragment>
  );
};

export default Modal;
```

Add Backdrop and ModalOverlay to Modal inside Fragment

- Create element overlay in index html
- Create Modal Component
- Crate const Backdrop
- Crate const ModalOverlay
- import ReactDOM from 'react-dom'
- Create const portalElement for access element overlay in index html
- Wrapper cart or content by Modal

```
import Modal from './Modal';

const Cart = () => {


  const cartItems = (
    <ul className={ classes[ 'cart-items' ] }>
      { [ { id: 'c1', name: 'Sushi', amount: 2, price: 12.99 } ]
        .map( item => (
          <li>{ item.name }</li>
        ) )
      }
    </ul>
  );

  return (
    <Modal>
      { cartItems }
      <div className={ classes.total }>
        <span>Total Amount</span>
        <span>35.62</span>
      </div>
    </Modal>
  );
};

export default Cart;
```

useEffect -1

➢ If DependencyList is empty, effect done when first rendring

```
import React, { useState, useEffect } from 'react';

import Card from '../UI/Card/Card';
import classes from './Login.module.css';
import Button from '../UI/Button/Button';

const Login = (props) => {
  const [enteredEmail, setEnteredEmail] = useState('');
  const [emailIsValid, setEmailIsValid] = useState();
  const [enteredPassword, setEnteredPassword] = useState('');
  const [passwordIsValid, setPasswordIsValid] = useState();
  const [formIsValid, setFormIsValid] = useState(false);

  useEffect(() => {
    setFormIsValid(
      enteredEmail.includes('@') && enteredPassword.trim().length > 6
    );
  }, [enteredEmail, enteredPassword]);

  const emailChangeHandler = (event) => {
    setEnteredEmail(event.target.value);
  };

  const passwordChangeHandler = (event) => {
    setEnteredPassword(event.target.value);
  };

  const validateEmailHandler = () => {
    setEmailIsValid(enteredEmail.includes('@'));
  };

  const validatePasswordHandler = () => {
    setPasswordIsValid(enteredPassword.trim().length > 6);
  };

  const submitHandler = (event) => {
    event.preventDefault();
    props.onLogin(enteredEmail, enteredPassword);
  };

  return (
    <Card className={classes.login}>
      <form onSubmit={submitHandler}>
        <div
          className={`${classes.control} ${
            emailIsValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="email">E-Mail</label>
          <input
            type="email"
            id="email"
            value={enteredEmail}
            onChange={emailChangeHandler}
            onBlur={validateEmailHandler}
          />
        </div>
        <div
          className={`${classes.control} ${
            passwordIsValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="password">Password</label>
          <input
            type="password"
            id="password"
            value={enteredPassword}
            onChange={passwordChangeHandler}
            onBlur={validatePasswordHandler}
          />
        </div>
        <div className={classes.actions}>
          <Button type="submit" className={classes.btn} disabled={!formIsValid}>
            Login
          </Button>
        </div>
      </form>
    </Card>
  );
};

export default Login;
```

➢ If DependencyList is empty, effect done when first rendring

```
import React, { useState, useEffect } from 'react';

import Login from './components/Login/Login';
import Home from './components/Home/Home';
import MainHeader from './components/MainHeader/MainHeader';

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {
    const storedUserLoggedInInformation =
localStorage.getItem('isLoggedIn');

    if (storedUserLoggedInInformation === '1') {
      setIsLoggedIn(true);
    }
  }, []);

  const loginHandler = (email, password) => {
    localStorage.setItem('isLoggedIn', '1');
    setIsLoggedIn(true);
  };

  const logoutHandler = () => {
    localStorage.removeItem('isLoggedIn');
    setIsLoggedIn(false);
  };

  return (
    <React.Fragment>
      <MainHeader isAuthenticated={isLoggedIn}
onLogout={logoutHandler} />
      <main>
        {!isLoggedIn && <Login onLogin={loginHandler} />}
        {isLoggedIn && <Home onLogout={logoutHandler} />}
      </main>
    </React.Fragment>
  );
}

export default App;
```

➢ component will re-rendered in infinity loop with out useEffect

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

import Login from './components/Login/Login';
import Home from './components/Home/Home';
import MainHeader from './components/MainHeader/MainHeader';

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

    const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

    if (storedUserLoggedInInformation === '1') {
      setIsLoggedIn(true);
    }

  const loginHandler = () => {
    localStorage.setItem('isLoggedIn', '1');
    setIsLoggedIn(true);
  };

  const logoutHandler = () => {
    localStorage.removeItem('isLoggedIn');
    setIsLoggedIn(false);
  };

  return (
    <React.Fragment>
      <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />
      <main>
        {!isLoggedIn && <Login onLogin={loginHandler} />}
        {isLoggedIn && <Home onLogout={logoutHandler} />}
      </main>
    </React.Fragment>
  );
}

export default App;
```

useEffect -2

```
useEffect(() => {
   const identifier = setTimeout(() => {
     console.log('Checking form validity!');
     setFormIsValid(
       enteredEmail.includes('@') && enteredPassword.trim().length > 6
     );
   }, 500);

   return () => {
     console.log('CLEANUP');
     clearTimeout(identifier);
   };
 }, [enteredEmail, enteredPassword]);
```

First when render dom will work the function useeffect, and not work the return .
When change Dependency values the retun will run cleanup function.
This will run as a cleanup process before useEffect executes this function the next time.

useReducer -1

```
const [state, dispatchFn] = useReducer(reducerFn, initialState, initFn);
```

state: The state snapshot used in the component rerender/ reevaluation cycle

dispatchFn: A function that can be used to dispatch a new action (i.e. trigger an update of the state)

initialState: The initial state

initFn: A function to set the initial state programmatically

reducerFn: `(prevState, action) => newState`

A function that is triggered automatically once an action is dispatched (via dispatchFn()) – it receives the latest state snapshot and should return the new, updated state.

useReducer -2

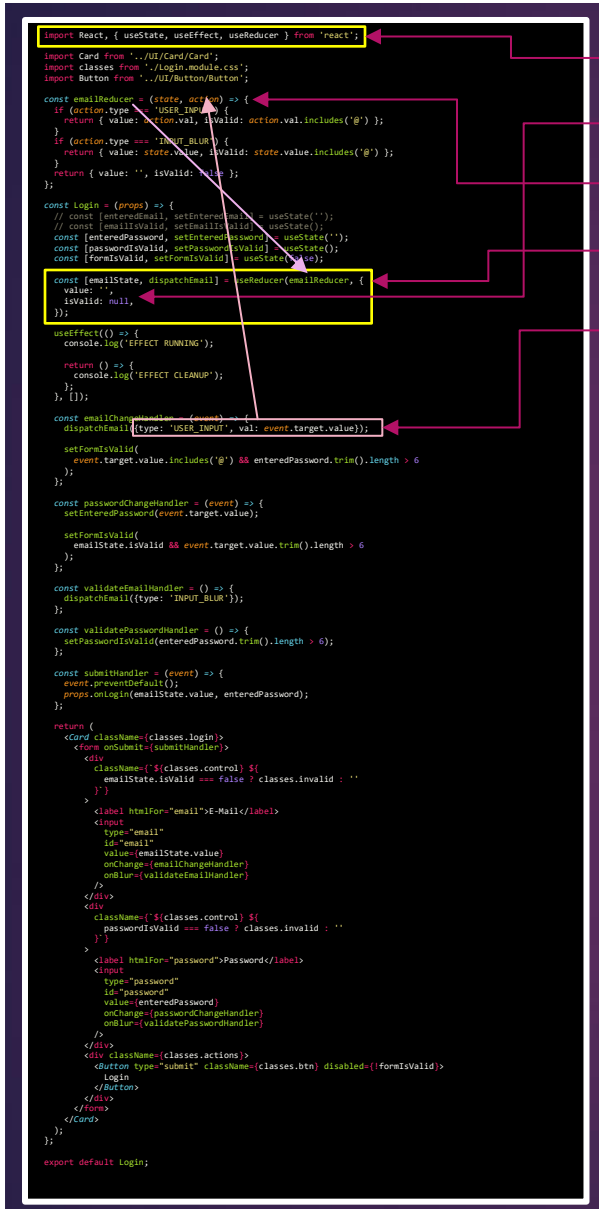
1 - import useReducer

2 - initialState

3 -Reducer Function (prevState, action) => newState

4 – use useReducer

5 – dispatch new action with object type action and new value



```
import React, { useState, useEffect, useReducer } from 'react';

import Card from '../UI/Card/Card';
import classes from './Login.module.css';
import Button from '../UI/Button/Button';

const emailReducer = (state, action) => {
  if (action.type === 'USER_INPUT') {
    return { value: action.val, isValid: action.val.includes('@') };
  }
  if (action.type === 'INPUT_BLUR') {
    return { value: state.value, isValid: state.value.includes('@') };
  }
  return { value: '', isValid: false };
};

const Login = (props) => {
  // const [enteredEmail, setEnteredEmail] = useState('');
  // const [emailIsValid, setEmailIsValid] = useState();
  const [enteredPassword, setEnteredPassword] = useState('');
  const [passwordIsValid, setPasswordIsValid] = useState();
  const [formIsValid, setFormIsValid] = useState(false);

  const [emailState, dispatchEmail] = useReducer(emailReducer, {
    value: '',
    isValid: null,
  });

  useEffect(() => {
    console.log('EFFECT RUNNING');

    return () => {
      console.log('EFFECT CLEANUP');
    };
  }, []);

  const emailChangeHandler = (event) => {
    dispatchEmail({type: 'USER_INPUT', val: event.target.value});

    setFormIsValid(
      event.target.value.includes('@') && enteredPassword.trim().length > 6
    );
  };

  const passwordChangeHandler = (event) => {
    setEnteredPassword(event.target.value);

    setFormIsValid(
      emailState.isValid && event.target.value.trim().length > 6
    );
  };

  const validateEmailHandler = () => {
    dispatchEmail({type: 'INPUT_BLUR'});
  };

  const validatePasswordHandler = () => {
    setPasswordIsValid(enteredPassword.trim().length > 6);
  };

  const submitHandler = (event) => {
    event.preventDefault();
    props.onLogin(emailState.value, enteredPassword);
  };

  return (
    <Card className={classes.login}>
      <form onSubmit={submitHandler}>
        <div
          className={`${classes.control} ${
            emailState.isValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="email">E-Mail</label>
          <input
            type="email"
            id="email"
            value={emailState.value}
            onChange={emailChangeHandler}
            onBlur={validateEmailHandler}
          />
        </div>
        <div
          className={`${classes.control} ${
            passwordIsValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="password">Password</label>
          <input
            type="password"
            id="password"
            value={enteredPassword}
            onChange={passwordChangeHandler}
            onBlur={validatePasswordHandler}
          />
        </div>
        <div className={classes.actions}>
          <Button type="submit" className={classes.btn} disabled={!formIsValid}>
            Login
          </Button>
        </div>
      </form>
    </Card>
  );
};

export default Login;
```

```
import React, { useReducer } from 'react';

const CartContext = React.createContext( {
  items: [],
  totalAmount: 0,
  addItem: ( item ) => { },
  removeItem: ( id ) => { }
} );

export const CartProvider = props => {

  const defaultCartState = {
    items: [],
    totalAmount: 0
  };

  const cartReducer = ( state, action ) => {
    if ( action.type === 'ADD' ) {
      const updateItems = state.items.concat( action.item );
      const updatedTotalAmount = state.totalAmount + action.item.price * action.item.amount;
      return {
        items: updateItems,
        totalAmount: updatedTotalAmount
      };
    }
    return defaultCartState;
  };

  const [ cartState, dispatchStateAction ] = useReducer( cartReducer, defaultCartState );

  const addItemHandler = item => {
    dispatchStateAction( { type: 'ADD', item: item } );
  };
  const removeItemHandler = id => {
    dispatchStateAction( { type: 'REMOVE', id: id } );
  };

  const ctxContext = {
    items: cartState.items,
    totalAmount: cartState.totalAmount,
    addItem: addItemHandler,
    removeItem: removeItemHandler
  };

  return (
    <CartContext.Provider value={ ctxContext }>
      { props.children }
    </CartContext.Provider>
  );
};

export default CartContext;
```

useReducer -3
With usecontext

useContext -1

```
====== auth-context.js ======
import React from 'react';

const AuthContext = React.createContext({
  isLoggedIn: false
});

export default AuthContext;
```

Init context by React.creatContext(initState) — Import React

1 – create folder store > sonthing-context.js

2 – in parent componets in App.js

3 – get values context to target component

```
import React, { useState, useEffect } from 'react';

import Login from './components/Login/Login';
import Home from './components/Home/Home';
import MainHeader from './components/MainHeader/MainHeader';
import AuthContext from './store/auth-context';

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {
    const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

    if (storedUserLoggedInInformation === '1') {
      setIsLoggedIn(true);
    }
  }, []);

  const loginHandler = (email, password) => {
    // We should of course check email and password
    // But it's just a dummy/ demo anyways
    localStorage.setItem('isLoggedIn', '1');
    setIsLoggedIn(true);
  };

  const logoutHandler = () => {
    localStorage.removeItem('isLoggedIn');
    setIsLoggedIn(false);
  };

  return (
    <AuthContext.Provider
      value={{
        isLoggedIn: isLoggedIn,
      }}
    >
      <MainHeader onLogout={logoutHandler} />
      <main>
        {!isLoggedIn && <Login onLogin={loginHandler} />}
        {isLoggedIn && <Home onLogout={logoutHandler} />}
      </main>
    </AuthContext.Provider>
  );
}

export default App;
```

1. Import SomethingContext from something-context.js in to root children's target
2. Wrapper and all target components by SomeThingContext.Provider
3. Add attribute 'value'={{propInitState:dynamicValue}}

ctx useContext

```
import React, { useContext } from 'react';

import AuthContext from '../../store/auth-context';
import classes from './Navigation.module.css';

const Navigation = (props) => {
  const ctx = useContext(AuthContext);

  return (
    <nav className={classes.nav}>
      <ul>
        {ctx.isLoggedIn && (
          <li>
            <a href="/">Users</a>
          </li>
        )}
        {ctx.isLoggedIn && (
          <li>
            <a href="/">Admin</a>
          </li>
        )}
        {ctx.isLoggedIn && (
          <li>
            <button onClick={props.onLogout}>Logout</button>
          </li>
        )}
      </ul>
    </nav>
  );
};

export default Navigation;
```

```
import React from 'react';

import AuthContext from '../../store/auth-context';
import classes from './Navigation.module.css';

const Navigation = (props) => {
  return (
    <AuthContext.Consumer>
      {(ctx) => {
        return (
          <nav className={classes.nav}>
            <ul>
              {ctx.isLoggedIn && (
                <li>
                  <a href="/">Users</a>
                </li>
              )}
              {ctx.isLoggedIn && (
                <li>
                  <a href="/">Admin</a>
                </li>
              )}
              {ctx.isLoggedIn && (
                <li>
                  <button onClick={props.onLogout}>Logout</button>
                </li>
              )}
            </ul>
          </nav>
        );
      }}
    </AuthContext.Consumer>
  );
};

export default Navigation;
```

1. Import SomethingContext from something-context.js in component target
2. Wrapper all element by SomeThingContext.Consumer
3. Get values from provider and set in ctx on consumer

useContext -2

```jsx
import React, { useState, useEffect } from 'react';

const AuthContext = React.createContext({
  isLoggedIn: false,
  onLogout: () => {},
  onLogin: (email, password) => {}
});

export const AuthContextProvider = (props) => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {
    const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

    if (storedUserLoggedInInformation === '1') {
      setIsLoggedIn(true);
    }
  }, []);

  const logoutHandler = () => {
    localStorage.removeItem('isLoggedIn');
    setIsLoggedIn(false);
  };

  const loginHandler = () => {
    localStorage.setItem('isLoggedIn', '1');
    setIsLoggedIn(true);
  };

  return (
    <AuthContext.Provider
      value={{
        isLoggedIn: isLoggedIn,
        onLogout: logoutHandler,
        onLogin: loginHandler,
      }}
    >
      {props.children}
    </AuthContext.Provider>
  );
};

export default AuthContext;
```

Provide custom provider wrapper by init context

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';

import './index.css';
import App from './App';
import { AuthContextProvider } from './store/auth-context';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <AuthContextProvider>
    <App />
  </AuthContextProvider>
);
```

```jsx
import React, { useContext } from 'react';

import Login from './components/Login/Login';
import Home from './components/Home/Home';
import MainHeader from './components/MainHeader/MainHeader';
import AuthContext from './store/auth-context';

function App() {
  const ctx = useContext(AuthContext);

  return (
    <React.Fragment>
      <MainHeader />
      <main>
        {ctx.isLoggedIn && <Login />}
        {ctx.isLoggedIn && <Home />}
      </main>
    </React.Fragment>
  );
}

export default App;
```

```jsx
import React, { useState, useEffect, useReducer, useContext } from 'react';
import AuthContext from '../../store/auth-context';

const Login = (props) => {

  const authCtx = useContext(AuthContext);

  const { isValid: emailIsValid } = emailState;
  const { isValid: passwordIsValid } = passwordState;

  const submitHandler = (event) => {
    event.preventDefault();
    authCtx.onLogin(emailState.value, passwordState.value);
  };
.
.
.
export default Login;
```

- Init context hook in store folder
- Index.js/ wrapped App by custom provider
- Get context from initial context of store
- Get context and update by method it

useImperativeHandle
UI input

```jsx
import React, { useRef, useImperativeHandle } from 'react';

import classes from './Input.module.css';

const Input = React.forwardRef((props, ref) => {
  const inputRef = useRef();

  const activate = () => {
    inputRef.current.focus();
  };

  useImperativeHandle(ref, () => {
    return {
      focus: activate,
    };
  });

  return (
    <div
      className={`${classes.control} ${
        props.isValid === false ? classes.invalid : ''
      }`}
    >
      <label htmlFor={props.id}>{props.label}</label>
      <input
        ref={inputRef}
        type={props.type}
        id={props.id}
        value={props.value}
        onChange={props.onChange}
        onBlur={props.onBlur}
      />
    </div>
  );
});

export default Input;
```

```jsx
  return (
    <Card className={classes.login}>
      <form onSubmit={submitHandler}>
        <div
          className={`${classes.control} ${
            emailState.isValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="email">E-Mail</label>
          <input
            type="email"
            id="email"
            value={emailState.value}
            onChange={emailChangeHandler}
            onBlur={validateEmailHandler}
          />
        </div>
        <div
          className={`${classes.control} ${
            passwordState.isValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="password">Password</label>
          <input
            type="password"
            id="password"
            value={passwordState.value}
            onChange={passwordChangeHandler}
            onBlur={validatePasswordHandler}
          />
        </div>
        <div className={classes.actions}>
          <Button type="submit" className={classes.btn} disabled={!formIsValid}>
            Login
          </Button>
        </div>
      </form>
    </Card>
  );
```

Init UI Component / Input

Use component UI nput in component Login