**AUXILLIARY ( HOC – Higher Order Component )**

Auxilliary.js

import React from "react";

const Auxilliary = props => props.children;

in Component

import React, { Component } from "react";

import Auxilliary from "../../../hoc/Auxilliary";

import classes from "./Person.css";

class Person extends Component {

  render() {

    return (

      <Auxilliary>

        <p onClick={this.props.click}>

          I'm {this.props.name} and I am {this.props.age} years old!

        </p>

        <p>{this.props.children}</p>

        <input

          type="text"

          onChange={this.props.changed}

          value={this.props.name}

        />

      </Auxilliary>

    );

  }

}

export default Person;

**React Fragments**

Veya asagidaki sekilde React kutuphanesinden Reactin yeni gelen ozelligi Fragment i cagiripda asagidaki gibi kullanabiliriz.

import React, { Component } from "react";

import Auxilliary from "../../../hoc/Auxilliary";

import classes from "./Person.css";

class Person extends Component {

  render() {

    return (

      <React.Fragment>

        <p onClick={this.props.click}>

          I'm {this.props.name} and I am {this.props.age} years old!

        </p>

        <p>{this.props.children}</p>

        <input

          type="text"

          onChange={this.props.changed}

          value={this.props.name}

        />

      </React.Fragment>

    ); }}

export default Person;

**HOC – Higher Order Component**

import React from 'react';

const withClass = props => (

//Buradaki props.classes asagida classes olarak cagriliyor.

//Ister className yerine kullan ister style ister baska birsey.

    <div className={props.classes}>

        {props.children}

    </div>

);

export default withClass;

import React, { Component } from "react";

…

import WithClass from "../hoc/WithClass";

class App extends Component {

  …

  render() {

    return (

//Eski Hali = <div className={classes.App}>

      <WithClass classes={classes.App}>

…

      </WithClass>

    ); }}

export default App;

**setState** => Onceki state ile ilgili bir degisiklik oldugunda prevState kullanimi onemlidir. Dogru kullanim sekli budur. Tabi prevstate yerine istediginiz bir isimde verebilirsiniz.

    this.setState(prevState => ({

      persons: persons,

      changeCounter: prevState.changeCounter + 1

    }));

**PropTypes**

Propslarin type lerini belirledigimiz npm moduludur. Create-react-app ile birlikte gelmiyor ayrica yarn add prop-types yazarak kurmamiz gerekiyor. Library yazarken veya buyuk projelerde diger yazilimcilarin kodlari daha iyi anlayabilmesi icin kullanilabilir.

Kullanacagimiz yerde asagidaki gibi import ediyoruz yukledikten sonra

import React, { Component } from "react";

import PropTypes from "prop-types";

import Auxilliary from "../../../hoc/Auxilliary";

class Person extends Component {

  render() {

    return (

      <Auxilliary>

        <p onClick={this.props.click}>

          I'm {this.props.name} and I am {this.props.age} years old!

        </p>

        <p>{this.props.children}</p>

        <input

          type="text"

          onChange={this.props.changed}

          value={this.props.name}

        />

      </Auxilliary>

    );

  }

}

Ve buradada yukarida kullanilan propslarin hangi ozelliklere sahip olmasi gerektigini, zorunlumu yoksa zorunlu degilmi olmasi gerektigini belirtebiliyoruz.

Person.propTypes = {

  click: PropTypes.func,

  name: PropTypes.string.isRequired,

  age: PropTypes.number,

  changed: PropTypes.func

};

export default Person;

**Using REFs**

Asagidaki gibi ref icinde fonksiyon kurup daha sonra onu componentDidMount ta kullanabiliriz. Asagida son input kutusuna odaklanan ve ref kullanilan bir ornek var.

…

class Person extends Component {

  componentDidMount() {

    this.inputElement.focus();

  }

  render() {

    return (

      <Auxilliary>

        …

        <input

          type="text"

          ref={inputEl => {

            this.inputElement = inputEl;

          }}

          onChange={this.props.changed}

          value={this.props.name}

        />

      </Auxilliary> );}}

…

export default Person;

yada

Once asagidaki gibi bir constructor olusturup icinde this.inputElementRef adinda bir degisken tanimlayip React.createRef()’e esitleriz.

Daha sonra input icinde ref ile bunu cagiririz.

Ve componentDidMount ‘ta site yuklendikten sonra bunu kullanip referansa focus olmasini isteriz.

…

class Person extends Component {

  constructor(props) {

    super(props);

    this.inputElementRef = React.createRef();

  }

  componentDidMount() {

    this.inputElementRef.current.focus();

  }

  render() {

    return (

      <Auxilliary>

        …

        <input

          type="text"

          ref={this.inputElementRef}

          onChange={this.props.changed}

          value={this.props.name}

        />

      </Auxilliary> );}}

…

export default Person;

**REF with React Hooks**

Once import ile reactin icine {useRef} i cagiriyorum Sonra const ile degisken olusturup useRef(null) ‘ a esitliyorum. Null initial degeridir. Daha sonra button icinde bu degiskeni ref ‘e atiyorum.

Daha sonra useEffect ile componentDidMount gibi sayfa renderi bittikten sonra calisan halinde tanimladigim bu degiskeni .current.click yaparak kullaniyorum. Bu sayede artik bu butona sayfa her yenilediginde otomatik olarak tiklaniyor.

…

const cockpit = props => {

  const toggleButtonRef = useRef(null);

  useEffect(() => {

    console.log("[Cockpit.js] useEffect");

    toggleButtonRef.current.click();

//componentWillUnmount icin bu

    return () => {

      console.log("[Cockpit.js] cleanup work in useEffect");

    };

  }, []);

  const assignedClasses = [];

  …

  return (

    <div className={classes.Cockpit}>

      …

      <button

        ref={toggleButtonRef}

        className={btnClass}

        onClick={props.clicked}

      >

    </div>

  );

};

export default React.memo(cockpit);

**HINTS**

* Degisik bir className birlestirme yontemi

className={["button", props.btnType].join(" ")}

+ koyarak sayiya cevirmek

this.state.loadedPost.id !== +this.props.match.params.id)

Buradaki + isareti this.props.match.params.id yi sayiya cevirir.