1. Ilk once Babel-cli, live-server, @babel/preset-react, @babel/preset-env ‘i yuklemeliyiz projemize.

@babel/preset-react

@babel/preset-env

+Bu ikisi kodlarimizi tarayicilarin anlayacagi sekilde ayarlayacak dil paketleridir.

Babel-cli

+babel-cli ise bu iki ek yamayi kontrol ederek donusturmeleri yonetir.

+commmand’da yani terminalde babel kodlarimizi calistirmaya yarar boylelikle onlari belirledigimiz yerdeki dosyalarin icine gomeriz.

live-server

+Hazirladigimiz projeyi local’de izlememize gormemize yarayan modul’dur.

1. Daha sonra package.json doyasina asagidaki kodu script olarak tanimlariz.

babel src/app.js --out-file=public/scripts/app.js --presets=env,react –watch

**--watch =>** babel in surekli src/app.js i denetlemesini soyler ve degisiklikleri aninda guncellemesini saglar.

**--out-file =>** babel’in donusturdugu dosya icerigini nerede hangi isimde bir dosyaya cikarttigi yoldur.

**--presets=env,react =>** Bu kodda babel’in kullanacagi paketleri belirler. Tabi bunlari yuklemeyi unutmayin. (@babel/preset-react, @babel/preset-env) Bu iki paket bu yuzden onemli.

Bu kodla jsx kodlu dosyamizi tarayicilarin anlayacagi dile ceviririz otomatik olarak. Ve kodu calistirdigimizda otomatik olarak takip edip dosyamizi kontrol etmeye baslar.

1. Ve bu adimlardan sonra asagidaki gibi 2 tane script’i tanimliyoruz package.json icinde.

"serve": "live-server public/",

"build-babel": "babel src/app.js --out-file=public/scripts/app.js --presets=env,react --watch",

**NOT**

console.log('app.js is running');

// JSX - Javascript XML

//render edilem template de sadece 1 tane root elemani yani html tagi bulunabilir. Tum tagleri bir tag icinde toplamaliyiz bunun icin

var template = (

        <div>

            <h1>This is JSX from app.js</h1>

            <p>This is some info</p>

            <ol>

                <li>Item 1</li>

                <li>Item 2</li>

                <li>Item 3</li>

            </ol>

        </div>

    );

var appRoot = document.getElementById('app');

//ReactDOM.render ilk parametre olarak eklenecek degeri alir

//ikinci parametre olarak degistirilecek yolu alir.

//Degistirilecek yol yukaridaki appRoot ornegindeki gibi secilir.

ReactDOM.render(template, appRoot);

**FARKLI KULKLANIMLAR**

function getLocation (location) {

    if(location){

        return <p>Location : {location}</p>;

    } else {

        return undefined

    }

}

//Asagidaki sekillerdede kullanimi mevcuttur.

var templateTwo = (

    <div>

//TERNARY OPERATORS

//Burada username eger varsa bu true donecek

//Eger true donerce ilk deger yani user.name yazdirilacak

//Eger false donerse : ‘den sonra yazdigimiz ikinci deger ‘Anonymouse’ yazdirilacak

        <h2>{user.name ? user.name : 'Anonymouse'}</h2>

//LOGICAL OPERATORS

//JSX dosyalarinda {false} {null} {undefined} yazarsak template div icerisinde hicbir goruntu almayiz.

//Agegida && isareti su isimize yariyor. Eger && isaretinden once bir deger yazarsak ve bu dogruysa, && isaretinden sonraki degeri yazdirir. Eger degilse {false} donderir ve buda dogru olmadigi icin hicbirsey gozukmez.

        { (user.age && user.age >= 18) && <p>Age : {user.age}</p> }

        <p>{getLocation(user.location)}</p>

        {getLocation(user.location)}

    </div>

)

**NOT**

//Javascriptte

//!'' true demektir

//!'Mucahid' false demektir

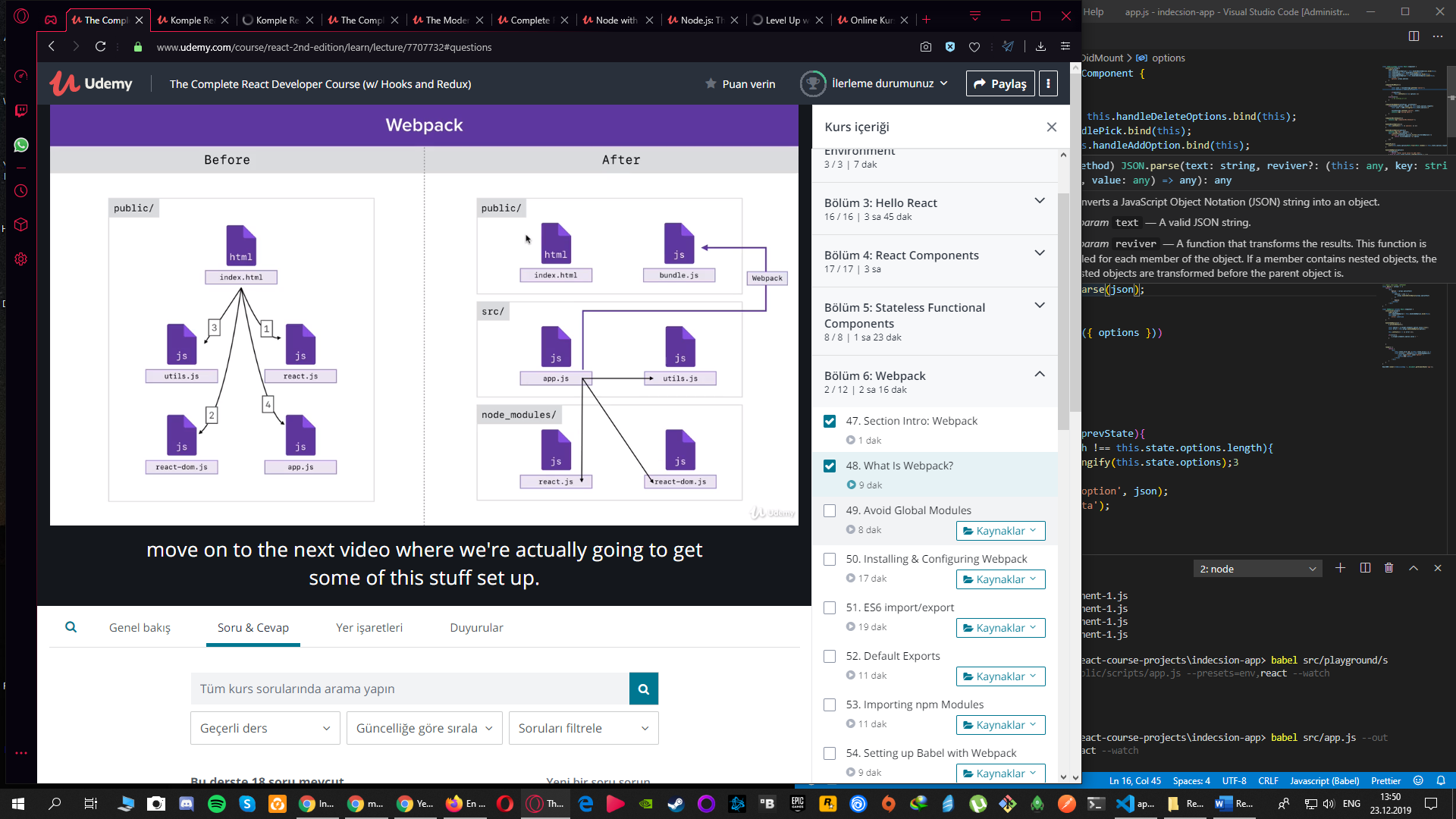
//!!'' false demektir

//!!'Mucahid' true demektir

//!undefined true demektir

//!!undefined false demektir

**WEBPACK CALISMA MANTIGI / ONCESI VE SONRASI**



**Webpack**

+Butun dosyalari tek dosyaya baglayip onu indexe baglamamiza yarayacak

+daha hizli olacak projeler

+daha duzenli tutulacak

Dosyalar saglar bize.

1. Once webpack ’i yukluyoruz projemize. Ve yukledikten sorna asagidaki sekildeki gibi dahil edip scriptlere ekliyoruz.

**--watch =>** webpack ’in surekli denetleme yapmasini soyler ve degisiklikleri aninda guncellemesini saglar.

"serve": "live-server public/",

"build": "webpack --watch",

"build-babel": "babel src/app.js --out-file=public/scripts/app.js --presets=env,react --watch",

**NOT**

Webpack ‘i yukledikten sonra build’deki babel komutunu build-babel yapiyoruz. (Artik ihtiyacimiz kalmadi aslinda isterseniz silebilirsinizde.) Ve build scripti olarak webpack ‘i yukaridaki gibi tanimliyoruz.

Webpack ’i yukleyip Config dosyasini ayarladiktan sonra artik package.json dosyasindaki babel-build scriptine ihtiyac kalmiyor webpack bunu bizim icin zaten yapiyor olacak

**WEBPACK CONFIG DOSYASI**

1. Webpack’i yukledikten sonra webpack.config.js dosyasini olusturuyoruz.

Kullanmak icin node ile birlikte gelen path modulunu REQUIRE yapiyoruz asagidaki gibi.

const path = require('path');

+Ve module.exports yaparak ayarlarimizi icinde EXPORT ederek ayarliyoruz.

module.exports = {

//Burasi ya development olur yada production olur. Gelistirme asamasinda development yayinlama asamasinda productiondur.

    mode: 'development',

    entry: './src/app.js',

    //outputta 2 sey ayarlamamiz tanimlamamiz gerekiyor. Path ve filename

    output: {

        //(./public) bu sekilde tanimlayamayiz burada onun icin asagidaki sekilde tanimlkiyoruz.

        //Bu path yoluna webpack bizim icin tum projeyi bundle.js adinda tek bir dosyada toplayacak.

        path: path.join(\_\_dirname, 'public'),

        //filename istedigimiz herhangi birsey olabilir. Genelde bundle.js kullaniliyor onun icin bende onu kullanacagim

        filename: 'bundle.js'

    },

**IMPORT and EXPORT**

**USER.JS**

//EXPORT ve IMPORT ORNEKLERI

//import './utils'; // CIKTI => 'utils.js is running'

//Bu sekilde import ile sadece dosyayi calistiririz icindekilere ulasamayiz.

//Icindekilere ulasmamiz icin isimlendirilmis yani NAMED EXPORT'lari isimleriyle import etmek gerekmektedir.

//2. 3. 4. gibi daha fazla IMPORT'larda da EXPORT'ta oldugu gibi virgul ve sonrasinda eklenir.

import { square, add, minus } from './utils';

console.log(square(5)); // CIKTI => 25

//CHALLENGE

//DEFAULT EXPORT'lar {} suslu parantezler disinda cagirilirlar ASAGIDAKI isSenior gibi

import isSenior, { isAdult, canDrink } from './utils';

console.log(isAdult(45)); // CIKTI => true

console.log(canDrink(30)); // CIKTI => true

console.log(isSenior(25)); // CIKTI => true

//DEFAULT EXPORT'lar yada asagidaki gibi de IMPORT edilebilir.

import isSenior from './utils';

//Veya dEFAULT EXPORT yerine istedigimiz her hangi bir isimde kullanabiliriz asagidaki gibi. Cunku program biliyor DEFAULT EXPORT hangisi sadece 1 TANE oldugu icin.

import istedigimHerHangiBirSey from './utils';

**IMPORT and EXPORT**

**UTILS.JS**

console.log('utils.js is running');

//Buna user.js'den ulasilmasi icin once asagidaki gibi EXPORT etmemiz gerekiyor.

const square = (x) => x \* x;

// 2 Type EXPORT turu vardir

// DEFAULT EXPORT ve NAMED EXPORT

//Ikinci bir export icin virgul koyulup yanina gondermek istedigimiz degiskeni gonderebiliriz. Asagida 2. olarak bir fonksiyon gonderiyoruz ornek olarak. 3. 4. gibi islemlerdede aynisi gecerli.

export  { square, add, minus };

const add = (a, b) => a + b;

//Ayrica basina EXPORT yazarak da export edebiliriz degiskenleri.

export const minus = (a, b) => a - b;

//Tabi STRING'ler buna dahil degil

//export 'YANLIS KULLANIM'

//Degiskenlerin baslarina export yazarakda EXPORT islemini yapabiliriz.

//Fakat export default degiskenlerin basina yazilmaz. FAKAT CLASS'larin basina yazilabilir.

export const sayi1 = 5; //DOGRU

export const topla = (a, b) => a + b; //DOGRU

export default sayi2 = 7; //YANLIS

export default sayi2; //DOGRU => sayi2 once bir degisken olarak tanimlanir sonra bu sekilde EXPORT edilebilir.

export default topla = (a, b) => a + b; //YANLIS

export default topla; //DOGRU => topla once bir degisken olarak tanimlanir sonra bu sekilde EXPORT edilebilir.

export default class topla { } //DOGRU

//CHALLENGE

const isAdult = (age) => age>=18

const canDrink = (age) => age>=18

const isSenior = (age) => age>=65

export {isAdult, canDrink, isSenior as default};

//DEFAULT EXPORT'lar 1 tane olabilir sadece bir dosyada.

//Ya yukaridaki gibi as default yanina yazilarak normal EXPORT'larla birlikte EXPORT edilebilirler.

**REACT and REACT-DOM**

+React uygulamalari gelistirmemiz ve web sayfalarimizda yayinlamamiz icin kullanmamiz gereken package yani module lerdir. Ister CDN lerle istersek projemize dahil ederek yukleyerek kullanabiliriz.

+HTML sayfasinda kullandigimiz REACT ve REACT-DOM CDN lerini kullanmamak icin bu 2 modulu once yukluyoruz ve sonra asagidaki gibi IMPORT ediyoruz her REACT COMPONENTIN’a. Ve artik HTML sayfasinda ki REACT ve REACT-DOM CDNs lerini silebiliriz.

1. Once **react react-dom** ‘i projemize yukluyoruz ve dahil etmek istedigimiz jsx dosyalarina asagidaki gibi IMPORT ederek kullaniyoruz. Tabi web-pack’le birlikte kullanabilmemiz icin yapmamiz gerekn bir kac ayar daha var webpack.config.js dosyasinin icinde babel ile alakali.

import React from 'react';

import ReactDOM from 'react-dom';

1. Projemize @babel/core, babel-loader modullerini yukleyerek dahil ediyoruz

+Cunku babelimizin yaptigi renderin dogru calisip wepack ile duzenlenip ekrana atilmasi icin loader ile ayarlanmasi gerekiyor. Yani LOADER webpack in davranislarini sekillendirir customize eder ona verilmis bir dosyayi yukletmek istedigimiz zaman.

+loader, module property ’si icinde rule olarak tanimlanir. Birden fazla kural belirlenebilir ama bu projemiz icin suanlik benim sadece 1 tane rule belirlemem yeterli. Bunuda asagidaki gibi yapacagim

+@babel/core, babel-cli gibidir.

+Babel-cli Command’da yani terminalde babel’i calistirmamiza yariyor ve boylelikle onlari belirledigimiz yerdeki dosyalarin icine gomeriz. babel-core ise webpack gibi toolslardan babel i calistirmamiza yarar

+babel-loader ise  bir webpack pluginidir. Ve babelin nasil calisacagini, ve ne zaman nereye babelin cikti dosyalarini servis edecegini, hangi dosyalari sececegini, hangilerini dahil etmeyecegini ogreten bir webpack pluginidir.Babel-core da iste bunun gibi pluginleri calistirir.

    module: {

        //Biz burada kurallari belirtiyoruz. Cok fazla kural koyabiliriz ama simdilik bize burada sadece asagidaki tek kural lazimdi onuda koyduk

        rules: [{

            loader: 'babel-loader',

            //Ne type dosylara calistirmak istedigimizi burada belirtiyoruz. Ben javascript dosyalari calistirmak istedigim icin .js i asagidaki sekilde ifade ettim $ isareti harflerin son sinirini \ isareti harflerin bas sinirini belirtir ve // ifadenin arasina yazilir. Detayli bilgi icin REGULAR EXPRESSION i arayin

            test: /\.js$/,

            //Dahil etmek istemedigimiz dosyalari ayarlariz EXCLUDE'de. Ve biz tum Node\_modules'un dahil olmamasini istiyoruz cunku bu moduller zaten islem gormus ve productiona surulmus dosyalar.

            exclude: /node\_modules/

        }]

    },

+BABEL icin ayarlar boylelikle tamam bundan sonra yapmamiz gereken tek sey .babelrc yani babel config dosyasini ayarlamak.

**BABEL CONFIG DOSYASI**

1. **babelrc** adinda root dizininde yani ana dizinde bir dosya olusturuyoruz. Artik babel-build ’i kullanmayacagimiz icin, webpack ’i ayarladiktan sonra webpack, babel ’a bunu bizim icin yaptiracagi icin ve babel bunu yapmaya calisirken ayarlari otomatik alacagi bir yol ve ayar dosyasi hazirliyoruz asagidaki gibi.

+presets kisminda babel-cli ‘in kullanacagi 2 tane babel donusturuculerini asagidaki gibi ayarliyoruz.

SOURCE MAP // KAYNAK HARITALAMA

1. Webpack.config.js dosyasina asagidaki sekilde devtool: heap-module-eval-source-map bu yazimizi objenin bir elemani olarak ekliyoruz.

devtool: 'cheap-module-eval-source-map',

+Ne ise yarar?

+Eger bunu kullanmazsak bir hata aldigimiz da tarayici konsolunda, tarayici bize bundle.js de olan hata yerini bize gosterecek. Fakat bunu webpack olusturuyor ve duzenliyor bizim duzenlememiz gereken bu dosya olmadigi icin bu yer gosterimi bizim isimize yaramaz. Bize dogru yeri gostermesi icn konsolda bunu ayarlamamiz gerekiyor burada.  +devtool bir stringe ayarlanmasi gerekiyor bunun icin. Bir kac cesit string ayarlamasi yapilabilir. Bazilari development icin iyidir bazilari production kismi icin iyidir. webpack.js.org dev tool section dan bunlara ayrintiuli bakabilirsiniz.

+Productionda farkli kod kullanilacak. Ayrintilardan bakin. Ayrntilar icin buradan bakin https://webpack.js.org/configuration/devtool/

devSERVER

1. Once webpack-dev-server modulumuzu npm install ile projemize ekliyoruz. Ve asagidaki gibi webpack.config.js dosyamizin icinde bir obje elemani olarak ekliyoruz.

devServer: {

        //contentBase'e path vermemiz gerekiyor fakat normal './' seklinde veremeyiz.

        //Tam dizin almamiz gerekiyor loaderde yaptigimiz gibi yapip path ile birlestirerek dizini veriyoruz.

        contentBase: path.join(\_\_dirname, 'public')

    }

ES6 CLASS PROPERTIES

1. Plugini ekleyebilmek icin once babel-plugin-transform-class-properties adinda olan BABEL-PLUGIN modulunu yukluyoruz.

+plugin kisminda ayarladigimiz ise CLASS ’larda degiskenler tanimlayabilmemizi, arrow function ‘lari kullanabilmemize yarayan bir BABEL PLUGIN ‘idir. Tabi bu plugini eklemek icin once 7. Maddede yazan modulu yukluyoruz.

+Ve pluginimizi .babelrc dosyasinda asagidaki sekildeki gibi ekliyoruz.

{

    "presets": [

        "@babel/preset-env",

        "@babel/preset-react"

    ],

    "plugins": [

        "transform-class-properties"

    ]

}

+Ve artik ES6 CLASS PORPERTIES ‘lerimizi kullanabiliriz jsx dosyalarimizda. Asagida OLD SYNTAX ve NEW SYNTAX ile kullanimlarini goruyoruz. Detayli inceleme ReactJS - Aciklamalar.docx dosyasinda ayni baslik altinda mevcuttur.

class OldSyntax {

    constructor() {

        this.name = 'Mike';

        this.getGreeting = this.getGreeting.bind(this);

    }

    getGreeting(){

        return `Hi. I am ${this.name}`;

    }

}

const oldSyntax = new OldSyntax();

console.log(oldSyntax);

//Eger getGreeting'i bind yaparak this le baglamazsak constructoda calismaz. Oysa babel’e ekledigimiz bu plugin sayesinde newSyntax’de buna gerek kalmiyor artik.

console.log(oldSyntax.getGreeting());

class NewSyntax {

    //name'e herhangi bir const var let gibi degisken atamamiza gerek yoktur NEW SYNTAX'de

    name = 'Jen';

    //oldSyntaxde yukaridaki gibi DEGISKEN'de, asagidaki gibi ARROW FONKSIYONDA tanimlayamayiz ama newSyntaxde bunu yapabiliriz.

    //Ve burada ARROW FONKSIYON kullanabildigimiz icin BIND ile this i baglamamiza gerek yok. Cunku ARROW FONKSIYONlar direk parent elementini this olarak kullanirlar.

    getGreeting = () => {

        return `Hi. My name is ${this.name}`;

    }

}

const newSyntax = new NewSyntax();

console.log(newSyntax);

console.log(newSyntax.getGreeting());

//Yukaridaki iki Syntax'de bize ayni sekilde cikti sonucu verir isim farkliliklari haric.

**PROPS Yontemi Ile Template Layout Vermek**

import React from 'react';

import ReactDOM from 'react-dom';

import IndecisionApp from './components/IndecisionApp';

const Layout = (props) => {

    return (

        <div>

            <p>Header</p>

            {props.content}

            <p>Footer</p>

        </div>

    );

}

const template = (

    <div>

        <h1>Page Title</h1>

        <p>This is my page</p>

    </div>

)

ReactDOM.render(<Layout content={template} />, document.getElementById('app'));

**PROPS CHILDREN YONTEMI**

+Sadece <Layout /> yerine 2’li <layout></layout> kullaniyoruz. Ve arasina ister degisken isterse HTML kodlarindan istedigimizi yaziyoruz. Ve daha sonra bunu Layout icinden props.children ile yakaliyor ve kullaniyoruz asagida ki gibi.

+Template veya herhangi bir istedigimiz isimde yukaridaki gibi bir degisken tanimlayip {template} yazip layout’un icine koyabilir ve bunu yine props.children ile yakalayip kullanabilirdik. Tercih sizin.

import React from 'react';

import ReactDOM from 'react-dom';

import IndecisionApp from './components/IndecisionApp';

const Layout = (props) => {

    return (

        <div>

            <p>Header</p>

            {props.children}

            <p>Footer</p>

        </div>

    );

}

ReactDOM.render(

    (

        <Layout>

            <p>This is inline</p>

        </Layout>

    ),

    document.getElementById('app'));

**REACT MODALS**

1. Once projemize **react-modal** ‘I npm install yaparak dahil ediyoruz.

+Modal yapisi asagida goreceginiz gibi Component’larla hemen hemen neredeyse aynidir.

+Moodal Componentlarin calismasi icin 2 tane PROPS'a ihtiyaci vardir. isOpen ve contentLabel

+**isOpen** => Bu bize modalin gosterilsin veya gosterilmesini ayarlamamizi saglar. Gosterilmesini istiyorsak TRUE yazmamiz gerekiyor.

+**contentLabel** => Buda modalimizi kullanicilara iyi bir isimle ortaya cikarabilmek icin gerekiyor. Tarayicida gozukmez. Modalimizin icindeki TITLE ile ayni olabilir. Accessibility ayarlari aktif edilirken gosterilebilir.

+++Daha cok fazla modal PROPS’u vardir. Simdilik sadece bu kadar. Usteki ikisi gerekli olanlardir. Alta ise ogrendikce yeni PROPS’lar ekleyecegim.

+**onRequestClose** => Bu PROPS ise bir FONKSIYON alir isOpen gibi. Bu bize ESC ye bastigimizda veya MODAL disina tikladigimizda olacaklari yapmamizi belirler. Asagida okay butonuna ayarladigim fonksiyonla artik bu props da bize esc ye basinca ve modal disina tiklayinca kapatilmasini saglayacak modalin.

Ornek MODAL Yapisi

import React from 'react';

import Modal from 'react-modal';

const OptionModal = (props) => (

        <Modal

            isOpen={!!props.selectedOption}

            contentLabel="Selected Option"

            onRequestClose={props.handleClearSelectedOption}

        >

            <h3>Selected Option</h3>

            { props.selectedOption && <p> {props.selectedOption} </p> }

            <button onClick={props.handleClearSelectedOption}>Okay</button>

        </Modal>

);

export default OptionModal;

**STYLE and CSS**

1. React projelerimize Style ve CSS dosyalarimizi yuklememiz icin once css-loader, style-loader ‘I npm install ile projemize dahil ediyoruz.

+Simdi bunlari kural olarak webpack.confing.js de rules icine ayarlamamiz gerekiyor asagida ki gibi. Burada gordugunuz gibi 2 tane modul yukledim. Bunun icin loader yerine ikisinide use arrayinin icine tanimliyorum asagida ki gibi.

{

            test: /\.css$/,

            use: [

                'style-loader',

                'css-loader'

            ]

        }

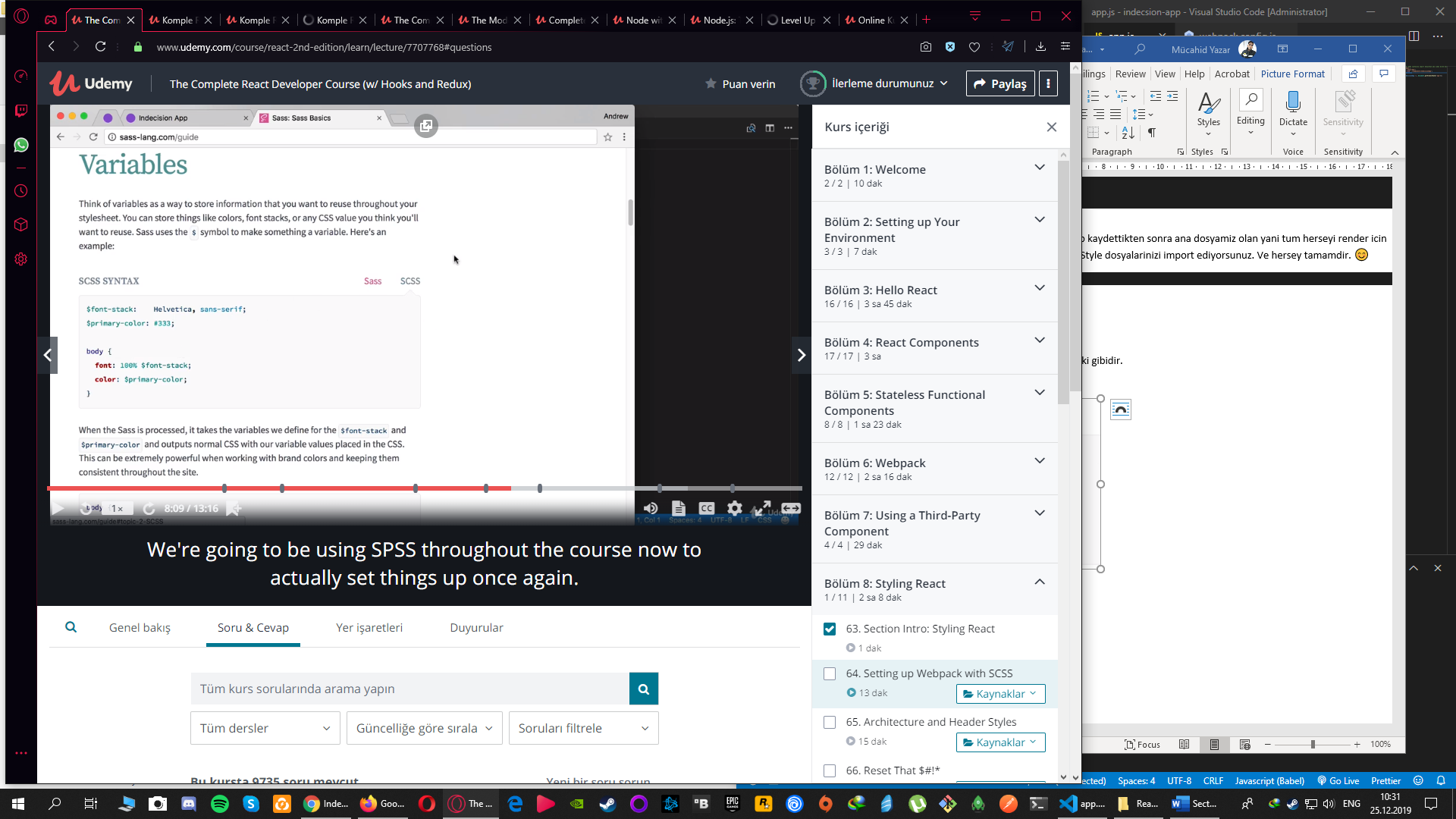
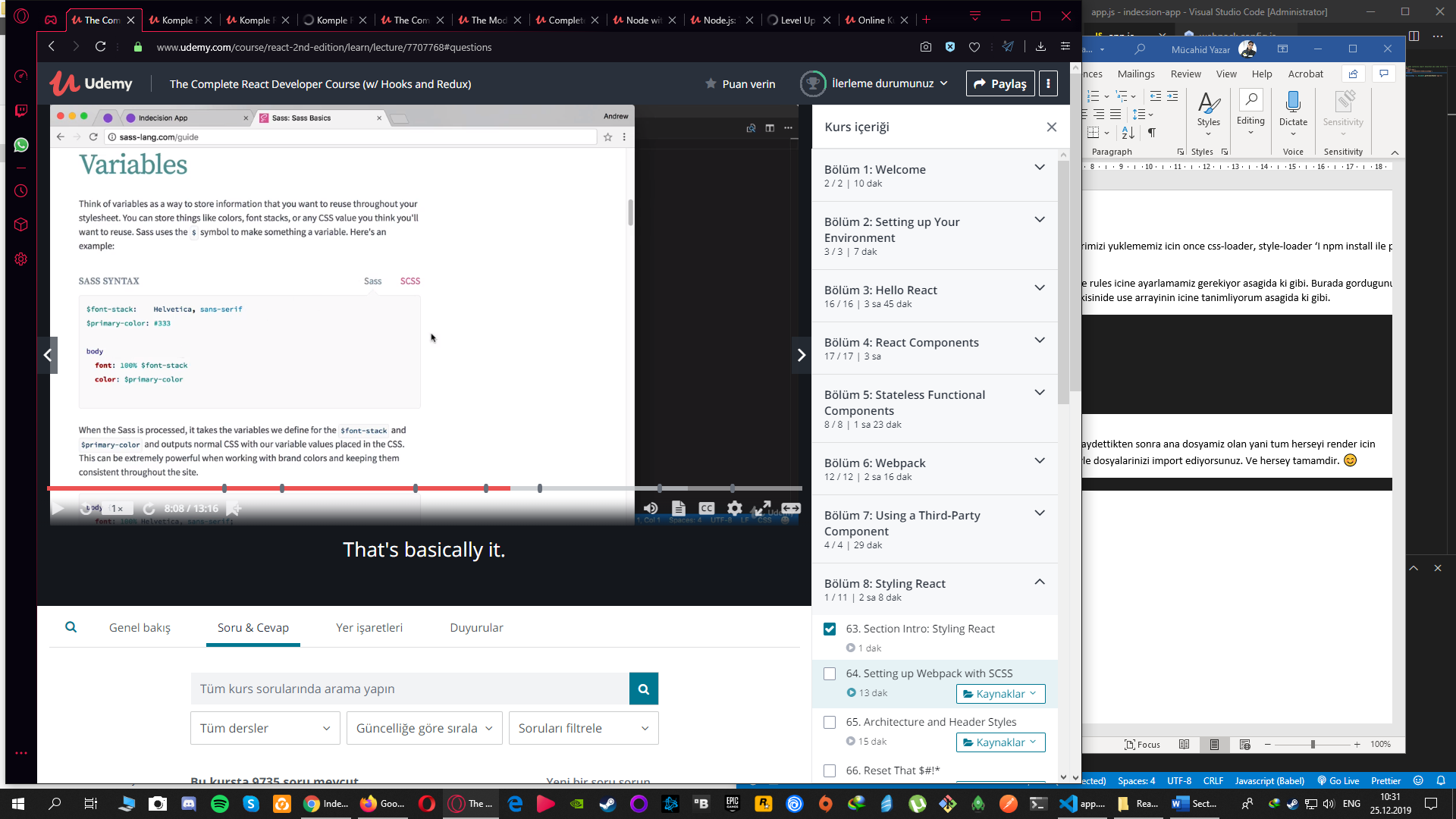
+Yukarida ki webpack.config.js ayarlarini yapip kaydettikten sonra ana dosyamiz olan yani tum herseyi render icin gonderdiginiz dosyaya gelip asagidaki sekilde Style dosyalarinizi import ediyorsunuz. Ve hersey tamamdir. 😊

import './styles/styles.css'

+Bu sistemde aslinda CSS doyalarimiz Javascript’in calistirmasini bekliyor bizim css dosyalarimizin.

**SASS SCSS**

+Ikiside ayni TOOL’un urunudur. Farki asagidaki gibidir.



1. Kurs boyunca biz SCSS kullanacagiz o yuzden

\*webpack.config.js ‘de .css diye ayarladigimiz uzantiyi .scss diye

\*Ve style icindeki dosyamizida .scss olarak

\*Ve son olarakda ana dizinde app’de STYLE veya CSS dosyalarimizi import ettigimiz yerdeki import yolunda yazan css uzantili yolu .scss olarak

degistiriyoruz.

$brand-color: blue;

\* {

    color: $brand-color;

}

+SCSS dosyalarimizda yukaridaki gibi $ isaretli ayarlamalar yapacagiz bunlari tarayicimiz tanimaz. Bunu projemize eklememiz gerekiyor. Bunun icin birisi babel-loader biriside @babel/core benzeri 2 tane modul yukleyecegiz.

1. Projemize sass-loader, node-sass modullerini npm I yaparak yukluyoruz. Ve sass-loader’I daha once olsuturdugumuz use rules’inin icine koyuyoruz ve goruntu asagidaki gibi oluyor. (Aslinda arkada olan sass-loader, node-sass ‘i kullanarak convert ediyor dosyalari)

{

      test: /\.s?css$/,

      use: [

            'style-loader',

            'css-loader',

            'sass-loader'

      ]

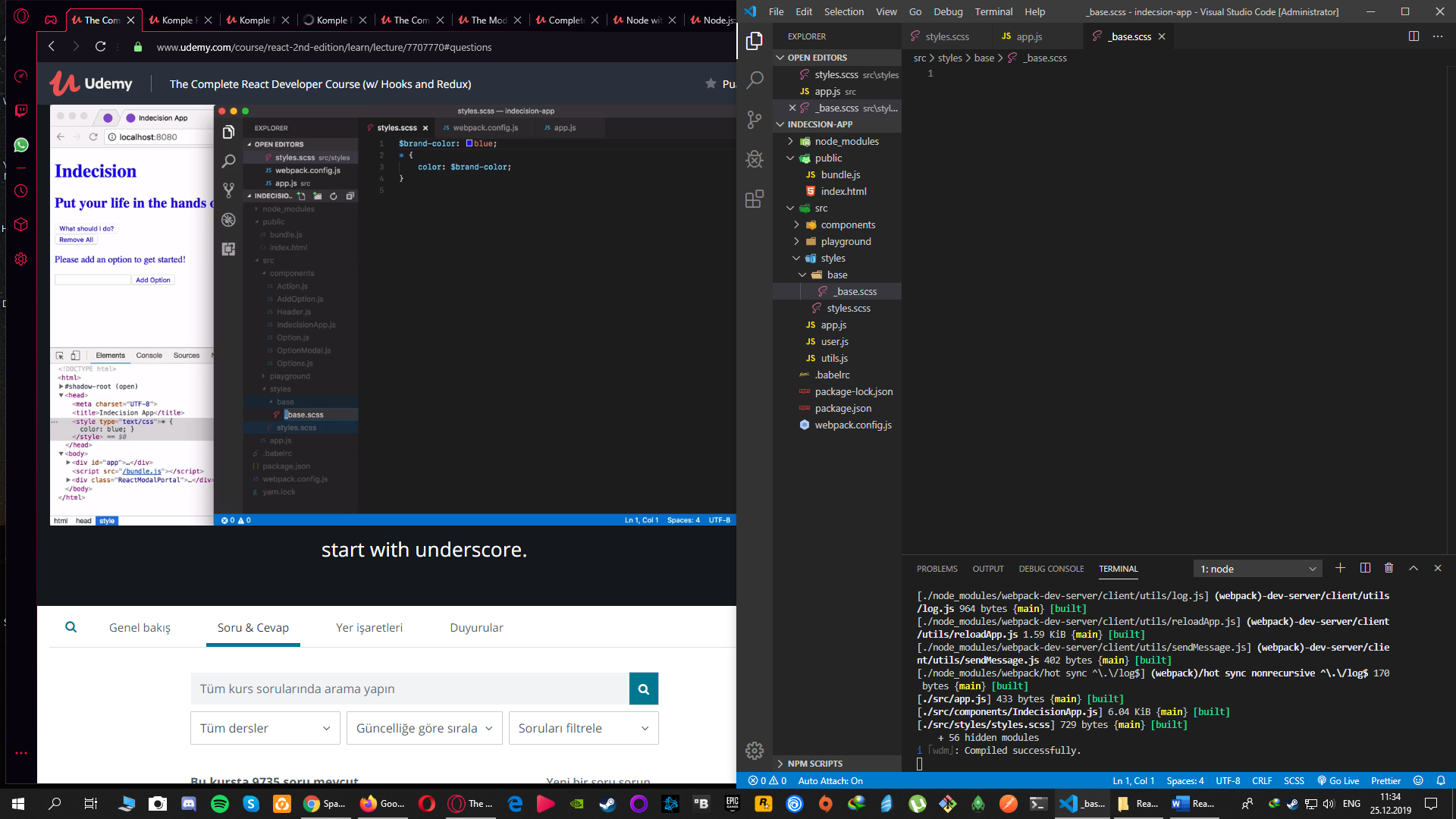
}

+Ve artik dev-server’I baslatabiliriz.

+Styles klasorunde ilk klasor isim.scss seklinde yazilir. Fakat ayri bir klasordeki isimler \_isim.scss seklinde cagrilir. Asagidaki sekilde ki gibi.

+\_base.scss ‘i styles.scss ‘de import etmemiz gerekiyor bunun icinde style.scss icinde asagidaki sekilde IMPORT etmemiz gerekiyor.

@import './base/\_base.scss'



+SCSS Nested Selectors’lari destekler asagidaki gibi. Yani ic ice gecmis Selector leri.

+Bu sayede h1 .header icindeki her h1’e degerini aktarir ama .header disindaki h1’lara aktarmaz.

.header {

    font-size: 2.2rem;

    background: #20222b;

    color: white;

    margin-bottom: 4.8rem;

    padding: 1.6rem 0;

    h1 {

        color: red;

    }

}

//BEM Block Element Modifier

.header\_\_title {

    font-size: 3.2rem;

    margin: 0;

}

+ BEM Block Element Modifier

+Asagida ki gibi bir componentimiz varsa BEM yapisini da istiyorsak eger kullanabiliriz.

+BEM’ler asagidaki gibi className icine tanimlanir. Cok duzenli bir sira oldugu icin aradiginizi bulmak kolaydir.

//---REACT FUNCTIONAL COMPONENT

const Header = (props) => (

    <div className="header">

        <h1 className="header\_\_title">{props.title}</h1>

        <h2>{props.subtitle}</h2>

    </div>

)

BEM ve daha fazlasi icin <http://getbem.com/>

**AXIOS**

React te back ende istek yaparken bu modulu JSX dosyalarimizin icinde kullaniriz.

Ornek AXIOS KULLANIMLARI

deleteExercise = id => {

        axios.delete('http://127.0.0.1:3000/exercises/' + id)

            .then(res => console.log(res))

            .catch(err => console.error(err));

*this*.setState({

            exercises:*this*.state.exercises.filter(exercise => exercise.\_id !== id)

        });

    }

componentDidMount() {

        axios.get('http://127.0.0.1:3000/users')

            .then(res => {

*if*(res.data.length > 0) {

*this*.setState({

                        users: res.data.map((user) => user.username)

                    })

                }

            });

    }

componentDidMount() {

        axios.post('http://127.0.0.1:3000/users/'+*this*.props.match.params.id)

            .then(res => {

*this*.setState({

                    username: res.data.username,

                    description: res.data.description,

                    duration: res.data.duration,

                    date: new Date(res.data.date)

                });

            })

            .catch((err) => console.error(err));

        axios.get('http://127.0.0.1:3000/users')

            .then(res => {

*if*(res.data.length > 0) {

*this*.setState({

                        users: res.data.map((user) => user.username)

                    })

                }

            });

    }

**REDUX**

1. Redux yazmaya baslamadan once her zaman importumuzu hemen alt satirdaki gibi yapiyoruz.

*import* { createStore }*from* 'redux';

1. Redux ‘da createStore ile STATE ‘ler asagidaki gibi olusturuyoruz.
2. State ’e asagidaki gibi default bir deger atabiliriz.
3. Action ile dispatch ‘lerimizde tanimladigimiz objelerimizin atrgumentlerini yakalayabiliriz.
4. SWITCH veya IF gibi kosullarimizla action.type ‘i yani dispatch ‘imizin TYPE argument ‘ini kosula baglayabiliriz.

const store = createStore((state = { count: 0 }, action) => {

*switch* (action.type) {

*case* 'INCREMENT':

            const incrementBy = typeof action.incrementBy === 'number' ? action.incrementBy : 1;

*+Ust satirda diyoruz ki Eger action.incrementBy 'in type i number yani sayi ise dogruysa,*

*const incrementBy = actin.incrementBy 'in sahip oldugu sayi olsun.*

*Sayi degil baska birseyse 1 olsun.*

*+Ve daha sorna bunu istersek asagida count'ada ekleyebiliriz. Eklemezzsek zaten hicbir anlam ifade etmez.*

*return* {

                count: state.count + incrementBy

            };

*case* 'DECREMENT':

            const decrementBy = typeof action.decrementBy === 'number' ? action.decrementBy : 2

*return* {

                count: state.count - decrementBy

            };

*case* 'SET':

*return* {

                count: action.count

            }

*case* 'RESET':

*return* {

                count: 0

            };

*default*:

*return* state;

    }

});

1. Yukarinin calisabilmesi icin asagidaki gibi DegiskenAdi**.dispatch** ‘lerle veri gondermemiz gerektigini, ve bunlari console.log ile tek tek yazdirmamiz gerektigini veya tek tek yazdirmak yerine DegiskenAdi**.subscribe** ile surekli guncellemeleri kontrol eder hale getirmemiz gerektigini unutmayalim.
2. **degiskenAdi.subscribe (store.subscribe)** (Ustteki ornek referans alinmistir.)

* Bu fonksiyon state i izlememizi ve degisikliklerde refesh almamizi saglar.
* subscribe icine argument olarak bir fonksiyon alir ve her degisiklikte bu fonksiyon calisir.
* Eger .subscribe tanimladiysak, daha once soyledigim gibi ayriyeten .dispatch ‘lerin altina console.log birakmaya gerek yoktur.
* Cunku olusturdugumuz degiskenAdi.subscribe her degisiklik oldugunda zaten konsola bunu yazdiracak

1. degiskenAdi.subscribe fonksiyonunu bir degiskene atayip daha sonra o degiskeni her hanagi bir yerde kullandigimiz zaman altindaki tum islemleri gecersiz kilar ve islemleri durdurur.

const unsubscribe = store.subscribe(() =>{

    console.log(store.getState());

});

1. **degiskenAdi.dispatch (store.dispatch)** (Ustteki ornek referans alinmistir.)

* action ‘larimiza deger gondermemize yariyor.
* Type: zorunlu nir degerdir. Yanlis yazili veya girilmezse REDUX hata verir.
* Ve ayrica type altinda istedigimiz isimde bir degerde belirleyebiliriz. Asagidaki orneklerde oldugu gibi, decrementBy, incrementBy… gibi…

store.dispatch({

    type: 'INCREMENT',

    incrementBy: 5

});

store.dispatch({

    type: 'INCREMENT'

});

store.dispatch({

    type: 'RESET',

});

store.dispatch({

    type: 'DECREMENT',

    decrementBy: 'Mucahid'

});

store.dispatch({

    type: 'DECREMENT',

    decrementBy: 8

});

store.dispatch({

    type: 'SET',

    count: 155

})

*//Alttakiler calismayacak cunku unsubscribe cagirdik hemen ustte*

*//BKZ MADDE 9*

unsubscribe();

store.dispatch({

    type: 'DECREMENT'

});

1. ACTION GENERATORS

const store = createStore((state = { count: 0 }, action) => {

*switch* (action.type) {

*case* 'INCREMENT':

*return* {

                count: state.count + action.incrementBy

            };

*case* 'DECREMENT':

*return* {

                count: state.count - action.decrementBy

            };

*case* 'SET':

*return* {

                count: action.setBy

            }

*case* 'RESET':

*return* {

                count: action.resetBy

            };

*default*:

*return* state;

    }

});

* Action generetorler, action object ler donen generatorlerdir.
* store.dispatch ‘lerde type degerlerinde, yazim hatasi yaptigimiz zaman bir hata vermez bize. (type da degil)
* fakat incrementCount 'u, bir store.dispatch ile kullandigimiz zaman eger yanlis bir harf yazarsak hemen konsolda hata aliriz. Bu bizim icin daha kullanislidir.

*//FONSIYON PARANTEZINI UNUTMAYIN*

*//RETURN PARANTEZINI UNUTMAYIN*

const incrementCount = ( payload = { } ) => ({

    type: 'INCREMENT',

    incrementBy: payload.incrementBy === 'number' ? payload.incrementBy : 1

})

store.dispatch(incrementCount({incrementBy: 10}));

* Ilk basta incrementCount icinde bir obje gondermek istedigimzide calismaz.Bunu fixlemek icin IncrementCount ‘u parantez icinde bir degiskene, yani yuakridaki gibi PAYLOAD veya ismini bizim belirleyecegimiz herhangi bir isimde degiskene atariz yukaridaki gibi.
* Daha sonra incrementCount icinde incrementBy tanimlariz ve karsisindaki gibi bir degerlendirme yapariz ve sorun cozulur. :)
* incrementBy ‘i bu sekilde tanimladiktan sonra o artik her zaman orada olacaktir fonksiyon ile cagrildiginda. Case icinde artik daha once yukaridaki orneklerde yaptigimiz gibi incrementBy tanimlamaya gerek yoktur.

const decrementCount = ( { decrementBy = 1 } = { } ) => ({

    type: 'DECREMENT',

    decrementBy

})

store.dispatch(decrementCount());

store.dispatch(decrementCount({ decrementBy: 5}));

* Yukaridaki gibi **DEFAULT** degerde belirtebiliriz. En alttaki 2 tane .dispatch lerinden birisinde decrementBy icin deger belirlenmemis birisinde ise belirlenmistir. Belirlenmemis olan ust satirdaki **DEFAULT** olarak atanmis deger 1’i alip kullanacaktir. Belirlenmis olan ise icinde yazan 5 ‘i alip kullanacaktir.
* Eger default deger atanmasaydi belirlenmsi olan yine sorun cikarmazdi fakat belirlenmemis olan HATA verirdi.

**ORNEK REDUX with ACTIONS**

*import* { createStore, combineReducers }*from* 'redux';

*import* uuid*from* 'uuid';

*// ADD\_EXPENSE*

const addExpense  = (

    {

        description = '',

        note = '',

        amount = 0,

        createdAt =0

    } = {}

) => ({

    type: 'ADD\_EXPENSE',

    expense: {

        id: uuid(),

        description,

        note,

        amount,

        createdAt

    }

});

*// REMOVE\_EXPENSE*

const removeExpense = ({ id } = { }) => ({

    type: 'REMOVE\_EXPENSE',

    id

})

*// EDIT\_EXPENSE*

const editExpense = ( id, updates ) => ({

    type: 'EDIT\_EXPENSE',

    id,

    updates

})

*// SET\_TEXT\_FILTER*

const setTextFilter = ( text = '' ) => ({

    type: 'SET\_TEXT\_FILTER',

    text

})

*// SORT\_BY\_DATE*

const sortByDate = () => ({

    type: 'SORT\_BY\_DATE',

    sortBy: 'Date'

})

*// SORT\_BY\_AMOUNT*

const sortByAmount = () => ({

    type: 'SORT\_BY\_AMOUNT',

    sortBy: 'Amount'

})

*// SET\_START\_DATE*

const setStartDate = (startDate) => ({

    type: 'SET\_START\_DATE',

    startDate

});

*// SET\_END\_DATE*

const setEndDate = (endDate) => ({

    type: 'SET\_END\_DATE',

    endDate

})

*// Expenses Reducer*

const expensesReducerDefaultState = [];

const expensesReducer = (state = expensesReducerDefaultState, action) => {

*switch* (action.type) {

*case* 'ADD\_EXPENSE':

*//Burasi return state.concat(actiob.expense) 'de olabilirdi.*

*return* [

                ...state, action.expense

            ];

*case* 'REMOVE\_EXPENSE':

*return* state.filter((expense) => (

                expense.id !== action.id

            ));

*case* 'EDIT\_EXPENSE':

*return* state.map((expense) =>{

*if* (expense.id === action.id){

*return* {

                        ...expense,

                        ...action.updates*//Bu yani action.updates ile gelen herseyi expense'in uzerine yazdirir. Ve map de o sekilde bir dizi olusturur.*

                    }

                }*else* {

*return* expense;

                }

            })

*default*:

*return* state;

    }

}

const filtersReducerDefaultState = {

    text: '',

    sortBy: 'amount',

    startDate: undefined,

    endDate: undefined

};

const filtersReducer = (state = filtersReducerDefaultState, action) => {

*switch* (action.type) {

*case* 'SET\_TEXT\_FILTER':

*return* {

                ...state,

                text: action.text

            }

*case* 'SORT\_BY\_DATE':

*return* {

                ...state,

                sortBy: action.sortBy

            }

*case* 'SORT\_BY\_AMOUNT':

*return* {

                ...state,

                sortBy: action.sortBy

            }

*case* 'SET\_START\_DATE':

*return* {

                ...state,

                startDate: action.startDate

            }

*case* 'SET\_END\_DATE':

*return* {

                ...state,

                endDate: action.endDate

            }

*default*:

*return* state;

    }

}

*// Get vısıble expenses*

*//Bu 2 argument alacak. 1. si tamamlanmis filtrelenmis ve siralanmis sortinglenmis dizi olacak yani expenses, Ve ikincisidde filters lari bilmeliyiz. Ve bunlari kullanarak gorunecek expensesleri aliriz.*

*//Ve daha sonra asagidaki .subscvribe icinde bunu tanimliyorum*

const getVisibleExpenses = (expenses, {text, sortBy, startDate, endDate}) => {

*return* expenses.filter((expense) => {

        const startDateMatch = typeof startDate !== 'number' || expense.createdAt >= startDate;

        const endDateMatch = typeof endDate !== 'number' || expense.createdAt <= endDate;

        const textMatch = expense.description.toLowerCase().includes(text.toLowerCase());

*return* startDateMatch && endDateMatch && textMatch;

    }).sort((a, b) => {

*if*(sortBy === 'date'){

*return* a.createdAt < b.createdAt ? 1 : -1*//Eger bu kosul saglanirsa 1 olur ve b ilk siralanir, saglanmazsa -1 olur ve a ilk siralanir.*

        }

*if*(sortBy === 'amount'){

*return* (a.amount > b.amount) ? -1 : 1

        }

    })

}

*// Store creatıon*

*//Combinereducers'lar bumbuyu reducer 'lar yerine kucuk multiple reducur lara ayirmamiza yariyor uygulamamizi.*

*//createStore icine combineReducers 'larimizi fonksiyon olarak yaziyoruz. Ve o fonksiyonun icindede argumet olarak bir obje olucak*

*//Bu objeler 1 value ve 1 ke y olmak zorundadir. Ama bu ornek icin demoState 'den dolayi 2 VALUE ve 2 KEY 'muz vardir.*

*//Bunlardan expenses ve filters VALUE 'dir*

*//yukaridaki tanimladigimiz expensesReducer ise KEY 'dir.*

*// Store Creation*

const store = createStore(

    combineReducers({

        expenses: expensesReducer,

        filters: filtersReducer

    })

);

store.subscribe(() => {

    const state = store.getState();

    const visibleExpenses = getVisibleExpenses(state.expenses, state.filters);

    console.log(visibleExpenses);

});

const expenseOne = store.dispatch(addExpense({ description: 'Rent', amount: 100, createdAt: 1000}));

const expenseTwo = store.dispatch(addExpense({ description: 'Coffee', amount: 300, createdAt: -1000}));

store.dispatch(removeExpense({ id: expenseOne.expense.id }));

store.dispatch(editExpense(expenseTwo.expense.id, {amount : 500}));

store.dispatch(setTextFilter('rent'));

store.dispatch(sortByDate());

store.dispatch(sortByAmount());

Alt satirla siralamada birisi gozukur. 125 expenseONe ve expenseTwo icin ikisininde 125 olarak ayarlanir. Fakat expenseOne createdAt 125 den buyuk oldugu icin siralama fonskyionumuzdan true doner ve gosterilir bu dispatch le.

store.dispatch(setStartDate(125));

store.dispatch(setStartDate());

store.dispatch(setEndDate(1250));

const demoState = {

    expenses: [{

        id: 'd2a5sd1a5das2das',

        description: 'January Rent',

        note: 'This was the final payment for that address',

        amount: 54500,

        createdAt: 0

    }],

    filters: {

        text: 'rent',

        sortBy: 'amount',*//date or amount, you can choose*

        startDate: undefined,

        endDate: undefined

    }

};

**HOC – Higher Order Component**

//Higher Order Component (HOR)

//A Component (HOC) that renders another component (Regular Basit Component)

//HOC 'in Hedefi Amaci

// Reuse code

// Render hijacking

// Prop manipulation

// Abstract state

import React from 'react';

import ReactDOM from 'react-dom';

const Info = (props) => (

    <div>

        <h1>Info</h1>

        <p>The info is : {props.info}</p>

    </div>

);

const withAdminWarning = (WrappedComponent) => {

    return (props) => (

        <div>

            {

                props.isAdmin && <p>This is private info. Please don't share</p>

            }

            <WrappedComponent {...props} />

        </div>

    )

};

const requireAuthentication = (WrappedComponent) => {

    return (props) => (

        <div>

            {

                props.isAuthenticated ? <WrappedComponent {...props} /> : <p>Please Login</p>

            }

        </div>

    )

}

const AdminInfo = withAdminWarning(Info);

const AuthInfo = requireAuthentication(Info);

//info props 'unu  {...props} ile WrappedComponent 'da yakaladik.

//ReactDOM.render(<AdminInfo isAdmin={true} info="These are the details" />, document.getElementById('app'));

ReactDOM.render(<AuthInfo isAdmin={false} isAuthenticated={true} info="These are the details" />, document.getElementById('app'));

* Yukaridaki HOC Component imiz bize reuse code yani kodlari tekrar kullanmamizi sagliyor.

**PROVIDER AND CONNECT**

* Provider bize tum componentlarimiza STORE ile etkilesime girmemize ve uygulamamizi yapmamizi, ayaga kaldirmamizi sagliyor.
* Ilk once **react-redux** ‘u yukluyoruz ve asagidaki gibi Provider ‘imizi import ediyoruz. Provider’imizi render yapilan yerde gonderiyoruz.

import { Provider } from 'react-redux';

* Import yaptiktan sonra

-bir degisken tanimliyoruz, Ornek asagida ki jsx

-sonra bu degiskene <Provider></Provider> taglarimizi tanimliyoruz.

-sonra asil render ettigimiz AppRouter ’i bu Provider taglerinin arasina alarak render’inin yapilmasini sagliyoruz.

* Bu Provider ‘a store props ‘u tanimliyoruz. Ve bu props ‘a import ettigimiz, createStore ve reduceStore ‘leri tanimladigimiz .js dosyasindan gelen ve configureStore() olarak const store’ye atadigimiz, **store** degiskenini yaziyoruz.

const jsx = (

    <Provider store={store}>

        <AppRouter />

    </Provider>

)

ReactDOM.render(jsx , document.getElementById('app'));

Ulasmak icin

* Once Provider ile erisime actigimiz state ‘e ulasmak istedigimiz Component ‘e gidiyoruz. Ornek: ExpenseList.js
* Ve yine react-redux kutuphanesinden asagida ki gibi connect ‘i import ediyoruz.

import { connect } from 'react-redux';

* connect() ile Provider 'ile bagladigimiz olaydan oturu bir fonksiyon cagiriyor. Ve bu fonksiyonada ExpenseList argumanini veriyoruz.
* Ve connect icinde bir fonksiyon tanimlariz asagidaki gibi veya bir degiskene tanimlariz bu fonksiyonu ve connect ‘in icine bu degiskeni yazariz. Bu fonksiyon icinde bizim componentimizin neye ihtiyaci varsa ulasabilmesini saglariz.
* Bu fonksiyonumuz ilk argumet olarak state alir. Ve icinde bir return islemi gerceklesir.
* Istersek icine hardcoded olarak name:’Andrew’ tanimlariz. Ve bunu ExpenseList component ‘inda props.name olarak yakalayabiliriz. Ve export isleminden sonra Andrew ‘in basildigini gorebiliriz.
* Tabi once export default connect(FONKSIYON)(ExpenseList) olarak export islemini gerceklestiriyoruz.
* Veya name:'Andrew' hardcoded yerine state’e erisip icinden tanimladigimiz store ‘nin expenses ve filters reducer ‘larini yakalayip props olarak ExppenseList ‘e gonderir ve yakalayabiliriz asagidaki gibi
* Ve ExpenseList component ‘inda props 'la expenses 'i yakalayalim ve kullanalim.
* Expenses icerisinde objeler barindiran bir dizi oldugu icin [0] ile dizi elemanin birisine ulasip id ile id sini yazdirabilir yada length ile uzunlugunu yazdirmayi deneyebiliriz ornek olarak.

const mapStateToProps = (state) => {

    return {

        expenses: state.expenses,

        filters: state.filters

    }

}

export default connect(mapStateToProps)(ExpenseList);

//REGEX101 // Regular Expression Kullanimi

// \d = Bu kod ile sayilari yakalayip eeslesmesinda kullanabiliriz. Ornek 1095

// eger yukaridaki kodun yaninda ^ olmazsa harfler varkende eslesme yapar

// ^\d yani bununla harler varken eslesme yapmaz harfler yokken eslesme yapar sayialri kabul ederiz. Tabi buda ilk sayi ile eslesir sadece => 1

// Sadece ilk sayiyla eslesme sorununu ise \* kullanarak cozeriz => ^\d\* => 1095

// 1095.95655 oldugunda ise noktadan sonralari almaz yukaridaki kodumuz

// Bunun icin ^\d\*()? diyerek ne olacak diyoruz ve ^\d\*(\.\d)? nokta olacak sonrasindada d ile digit yani numara gelecek diyoruz. Fakat buda tum sayilari alir noktadan sonra sinirsiz sekilde. Bunu istemiyoruz

// Bunun icinde ^\d\*(\.\d{0,2})?$ ve {curly braces} lari acariz en az 0 en cok 2 karakter olsun deriz noktadan sonra

// degisken adi.match(/ ^\d\*(\.\d{0,2})?$ /) yaparak eslestirmede kullanabiliriz

Setting Up A Date Picker

MomentJS den saat format duzenleyiciyi alacagiz

React-dates /airbnb.io buradan Tarih secici uygulamamizi alacagiz

react-addons-shallow-compare => Bu ise react-dates in calismasi icin gerekli yuklemek zorunda oldugumuz moduldur.

* Asagidaki modulleri yukleyerek baslayalim.

moment

react-dates

react-addons-shallow-compare

MomentJS / [www.momentjs.com](http://www.momentjs.com)

Yukledikten sonra import ederiz kullanacagimiz yerde

import moment from 'moment';

Sonra bir degiskene tanimlariz momenti asagidaki gibi

Daha sonra bu degiskene bir format tanimlariz ve kullaniriz asagidaki gibi.

Detayli format kullanimi icin sitesine bakin.

const now = moment();

console.log(now.format('MMM Do, YYYY'));

Ornek

console.log(now.format());

CIKTI => 2019-12-31T16:06:42+03:00

PURE FUNCTIONS

*//ITS NOT PURE FUNCTION*

const a = 5;

const add = (b) => {

*return* a + b

}

*//PURE FUNCTION*

const add = (a, b) => {

*return* a + b

}

*//IT'S NOT PURE FUNCTION*

*//Cunku burada result'da degisebilir oldugundan PURE FUNCTION degildir.*

*//PURE FUNTION 'lar disaridan degistirilemezler, mudahaleye ugrayamazlar.*

let result;

const add = (a, b) => {

    result = a + b;

}

SPREAD OPERATORS - ARRAY

*//Bu names'i degistirir. Ve bundan sonra names 3 kisilik bir dizi olur.*

const names = ['Andrew', 'Steve'];

names.push('Einde');

*//Bu ise Spread Operatordur. Name'i degistirmez. Tekrar name i print yaptirsak 'gs' ve ''fb' yi goruruz sadece. Sadece kodun yazildigi an degistirir ve okumaya yarar.*

const teams = ['Gs', 'Fb']

names.concat('Ts');

*//[...names] yazarak'da tum diziyi yazdirabiliriz.*

*//ayni zamanda asagidaki gibi kullanarakda isim ekleyebiliriz. Bu Spread Operatordur.*

*//Spread operatorler names'i degistirmez. Yani tekrar print yapsak 'Jen' eklenmemis olurdu.*

*//Buda reducer 'larda bunu kullanabilecegimiz anlamina geliyor.*

[...names, 'Jen']

SPREAD OPERATORS - OBJECT

const user = {

    name: 'Jen',

    age: 24

};

console.log({

    ...user,

    location: 'Istanbul'

})

* Objects lerde Spread operatorleri yukaridaki gibi kullanabiliriz ama … hata verebiliyor onun icin once **babel-plugin-transform-object-rest-spread** pluginini indirip .babelrc deki pluginlere basinda **babel-plugin** olmadan ekliyoruz.
* Console.log’da eklemeler yapabiliriz yukarida ki gibi. Bu degisiklikler gercek degisiklik degildir. Reducurlara uygundr

Javascript Siralama

function compare(a, b) {

*if*(a nin once siralanmasini istiyorsak return -1 olmali) {

*return* -1

    }

*if*(b nin once siralanmasini istiyorsak return 1 olmali) {

*return* 1

    }

*if*(a must be equal to b) {

*return* 0

    }

}

*//Diziler icin basit siralama icin sort() yeterli*

compare.sort()

1. REDUCER FUNCTION

* Reducer Function ‘larimiz yukarida anlatilan gibi PURE FUNCTION ‘lardir.
* Never Change STATE and ACTIONS

*// ADD\_EXPENSE*

const addExpense  = (

    {

        description = '',

        note = '',

        amount = 0,

        createdAt =0

    } = {}

) => ({

    type: 'ADD\_EXPENSE',

    expense: {

        id: uuid(),

        description,

        note,

        amount,

        createdAt

    }

});

*// REMOVE\_EXPENSE*

const removeExpense = ({ id } = { }) => ({

    type: 'REMOVE\_EXPENSE',

    id

})

*// EDIT\_EXPENSE*

const editExpense = ( id, updates ) => ({

    type: 'EDIT\_EXPENSE',

    id,

    updates

})

*// SET\_TEXT\_FILTER*

const setTextFilter = ( text = '' ) => ({

    type: 'SET\_TEXT\_FILTER',

    text

})

*// SORT\_BY\_DATE*

const sortByDate = () => ({

    type: 'SORT\_BY\_DATE',

    sortBy: 'Date'

})

*// SORT\_BY\_AMOUNT*

const sortByAmount = () => ({

    type: 'SORT\_BY\_AMOUNT',

    sortBy: 'Amount'

})

*// SET\_START\_DATE*

const setStartDate = (startDate) => ({

    type: 'SET\_START\_DATE',

    startDate

});

*// SET\_END\_DATE*

const setEndDate = (endDate) => ({

    type: 'SET\_END\_DATE',

    endDate

})

REFESH YAPMADAN YONLENDIRMEK

                props.history.push('/');

Resources: Importing SVG In React

In the previous lesson, you may have seen the strange syntax: **import** { ReactComponent **as** Logo }

This is a a new special syntax when importing SVG in React. The ReactComponent import name is special and tells Create React App that you want a React component that renders an SVG, rather than its filename. You can read more about it here, but keep in mind that this is a React library special syntax:

<https://facebook.github.io/create-react-app/docs/adding-images-fonts-and-files>

React Routers

//Switch yalnizca tek sayfaya yonlendirilme yapilmasini saglar. Yanlislikla 2 sayfada ayni url kullanilirsa yanlizca ilk siradaki sayfaya gider. Bu sekilde multiple yonlendirmelerin onune geceriz. Bir nevi bizim icin daha fazla kontrol saglar.