<https://console.developers.google.com/>

adresınden yenı proje olusturuyoruz.

Daha sonra

Oauth izin ekranindan Harici olarak isteklerimizi tum Google kullanicilari icin aciyoruz.

Ve proje adimizi yazarak Save yapiyoruz.

-Oauth istemci kimligi olustur yapiyoruz

-Burada Authorized JS origins yani Yetkilendirilmis JS kaynaklarina projemizi kullanacagimiz web sitesinin adresini yaziyoruz. Google’da sadece buradan gelecek isteklere cevap verecek boylelikle. Ve Olustur diyoruz.

-Google bize istemci anahtari ve gizli keyi veriyor. Gizli keyi kullanmayacagiz. Gizli keyi sadece Eger JS Oauth unu kullanmayacak olsaydik sadece Oauth servislerini kullanacak olsaydik kullanacakdir. Simdi sadece ISTEMCI anahtarini kullanacagiz. Bu ID yi kopyalayin bir kenarda tutun.

-Daha sonra index.html dosyamiza asagida ki satiri ekliyoruz

<script src="https://apis.google.com/js/api.js">

-Sonra component ‘larimizin icine GoogleAuth.js adinda dosya olusturuyoruz. Ve bunu Header icinde Navbar icine <GoogleAuth /> ile ekliyoruz.

GoogleAuth ‘un ici

import React from 'react';

class GoogleAuth extends React.Component {

    componentDidMount() {

        //Clientimizi yukluyoruz

        window.gapi.load('client:auth2', () => {

            //Ve buradan artik client imizi initialize yapiyoruz.

//Initialize yapmamiz icin icinde bir obje dondurmemiz gerekiyor. Ve bu objede client id olarak api keyimizi buraya giriyoruz. Scopeda ise ulasmak yani erismek istedigimiz degerleri belirytiyoruz.

            window.gapi.client.init({

                clientId: '929277223060-0vat68bshncij4ja85pbt6m4v2buai4u.apps.googleusercontent.com',

                scope: 'email'

            })

        });

    }

    render() {

        return (

            <div>Google Auth</div>

        )

    }

}

export default GoogleAuth;

Kullanacagimiz metodlar ayrintili olarak bu documentation da bulunamktadir.

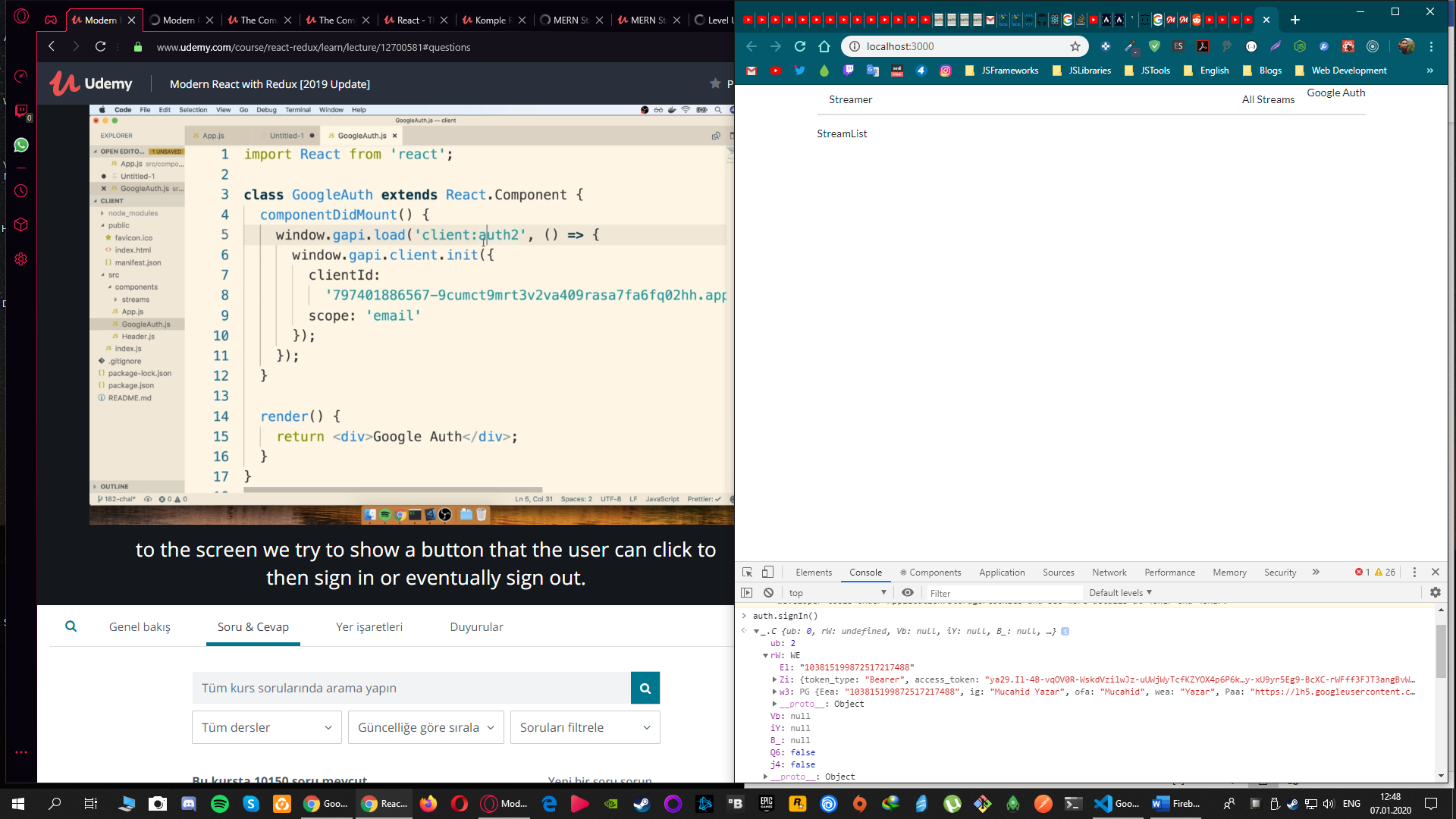
<https://developers.google.com/identity/sign-in/web/reference>

Yukaridaki kodlardan sonra artik gapi.auth2.getAuthInstance() metoduyla isSignedin, signIn ve signOut gibi metodlarina erisebiliriz.

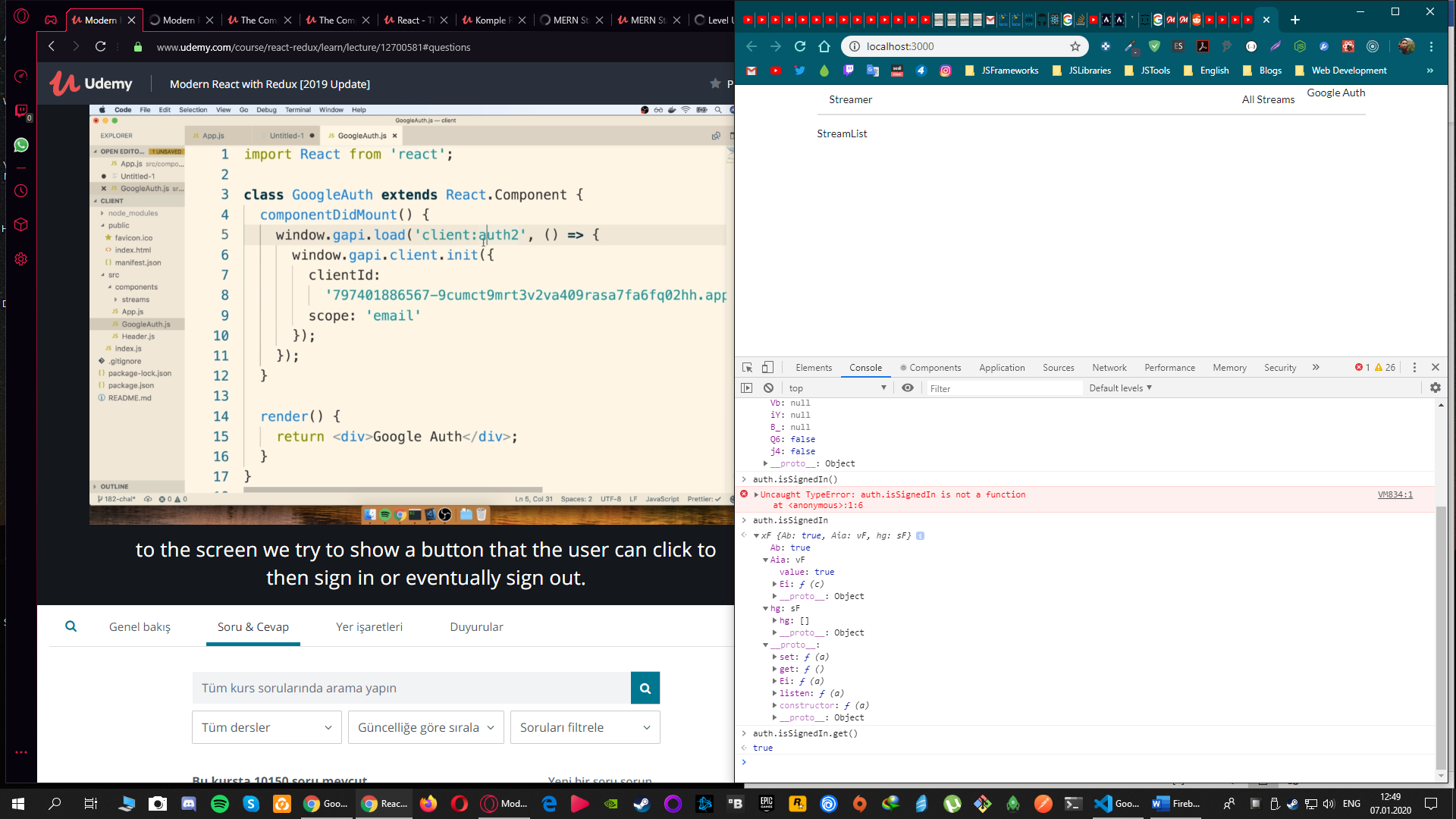
// MANUAL OLARAK DENEMEK

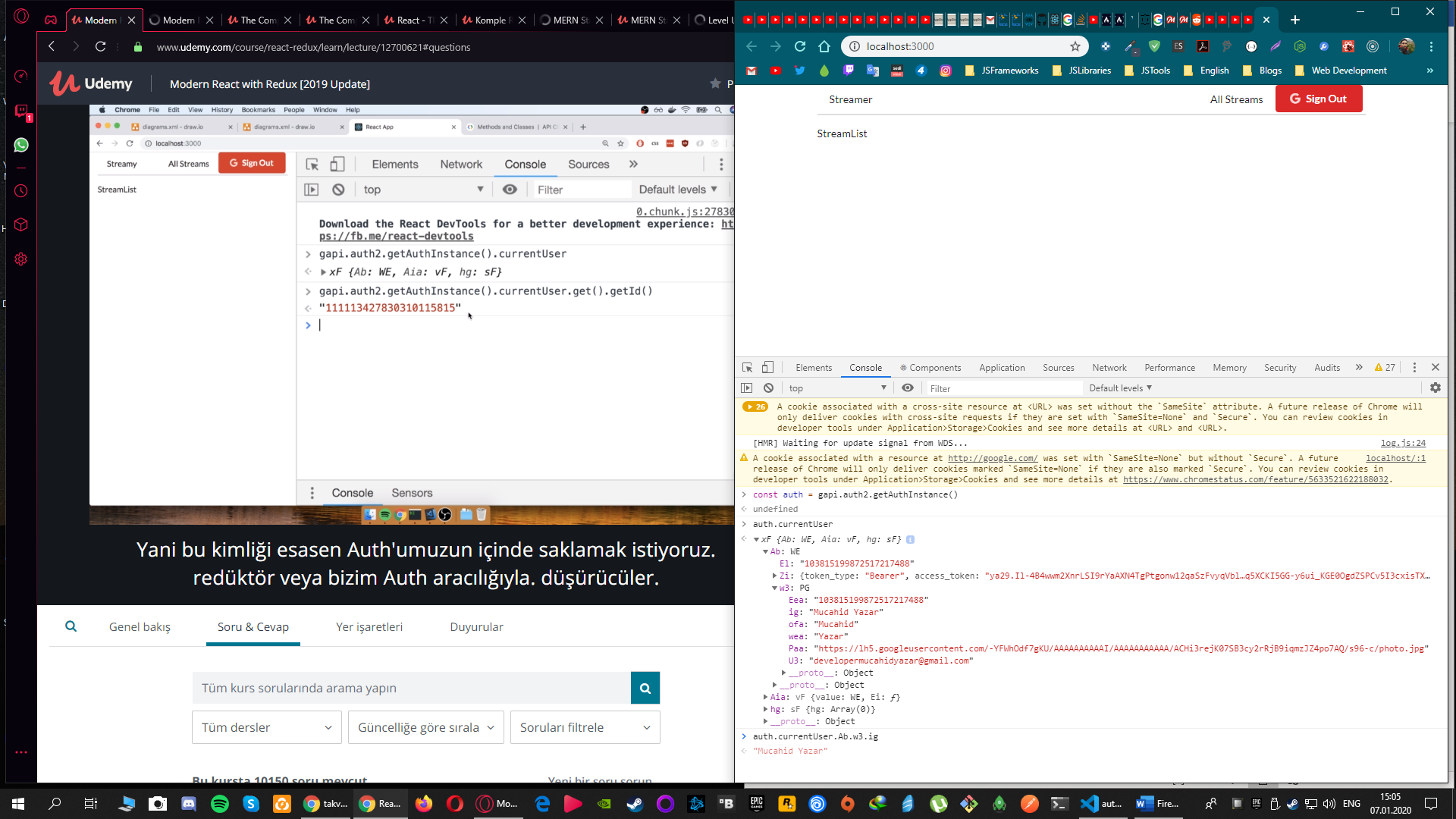
* Terminali acip const auth = gapi.auth2.getAuthInstance() diyelim
* Sonra

**auth.signIn() -**



**Auth.isSignedIn / Auth.isSignedIn.get()**





* Ve artik library mizden signIn yapilmis mi oldugunu cekip true ve false olarak this.setState ile yazdirip uygulamamizda asagidaki gibi fonksiyon olusturarak kontrol sagliyoruz.

import React from 'react';

class GoogleAuth extends React.Component {

    state = {

        isSignedIn: null

    }

    componentDidMount() {

        //Clientimizi yukluyoruz

        window.gapi.load('client:auth2', () => {

            //Ve buradan artik client imizi initialize yapiyoruz.

            //Initialize yapmamiz icin icinde bir obje dondurmemiz gerekiyor. Ve bu objede client id olarak api keyimizi buraya giriyoruz. Scopeda ise ulasmak yani erismek istedigimiz degerleri belirytiyoruz.

            window.gapi.client.init({

                clientId: '929277223060-0vat68bshncij4ja85pbt6m4v2buai4u.apps.googleusercontent.com',

                scope: 'email'

            }).then(() => {

                //Burada auth degiskenini gapi.auth2 ile getAuthInstance ozelliklerini kazandiriyoruz. Ve artik sigin signout issigned gibi islemleri this.auth ile ulasabilecegiz asagidaki gibi

                this.auth = window.gapi.auth2.getAuthInstance();

                this.setState({ isSignedIn: this.auth.isSignedIn.get()})

            })

        });

    }

    renderAuthButton = () => {

        if(this.state.isSignedIn === null) {

            return <div>I don't know if we are signed in</div>;

        } else if (this.state.isSignedIn) {

            return <div>I am signed in</div>

        } else {

            return <div>I am not signed in</div>

        }

    }

    render() {

        return (

            <div>{this.renderAuthButton()}</div>

        )

    }

}

export default GoogleAuth;

GAPI with REDUX

TERCIH EDILMEYEN YONTEM

