

REACT NATIVE TOOLBOX

01 - 08

©NIR CHEN



SYLLABUS

- **-01 Expo**
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

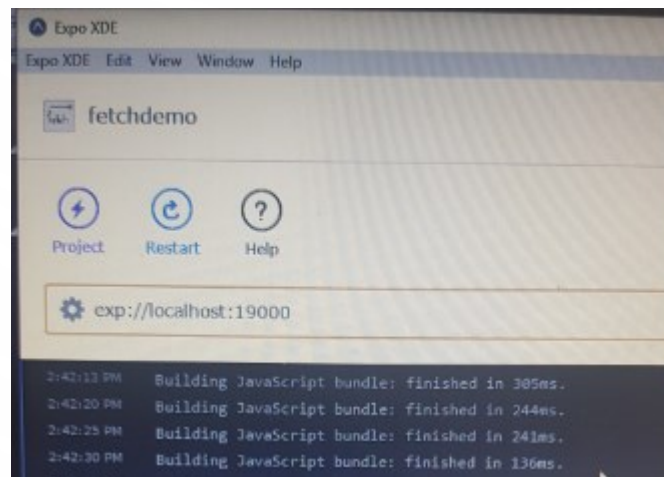
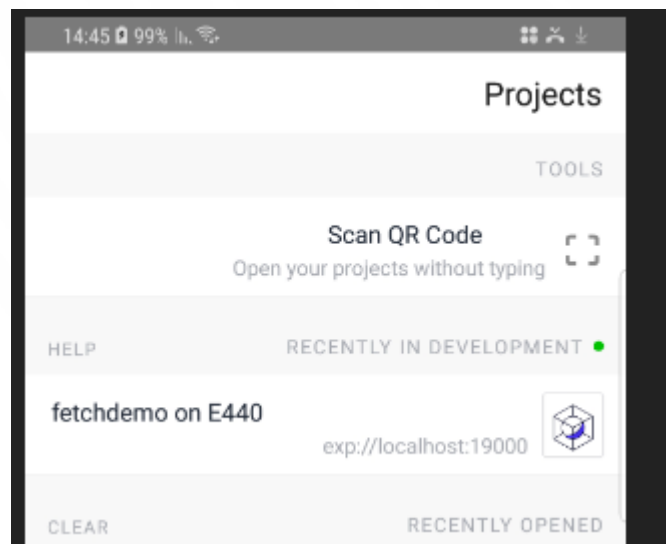
©NIR CHEN

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- 14 React Elements Lib

EXPO

1. הכי מהיר זה לעבוד בLOCALHOST כאשר המכשיר וגם המחשב מחוברים על אותה רשת WIFI.

2. חובה לחבר את המכשיר למחשב עם כבל USB ולאפשר DEBUGGING ("איתור באגים של USB") דרך הUSB ב"אפשרויות למפתחים"



SYLLABUS

- **00 Debug**
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

DEBUG

1. לשקשק\לנענע\לנער את המכשיר ואז לבחור בתפריט את "Debug JS Remotely"
2. יפתח הכרום בצורה אטומטית ואז יש ללחוץ F12
3. יש לבחור את הטאב של Sources
4. כאשר נגיע לBREAKPOINT הקוד יעצר בשורה המתאימה ואז אפשר לדאבג רגיל.

DEBUG IN VS CODE

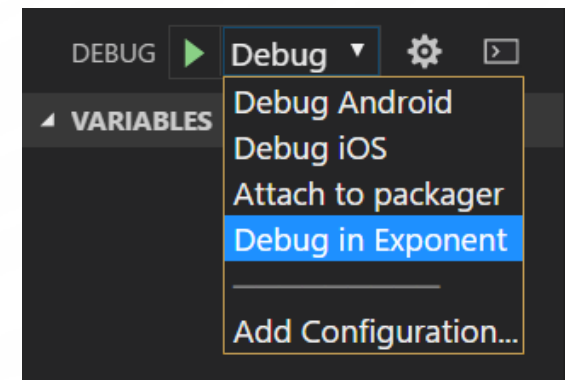
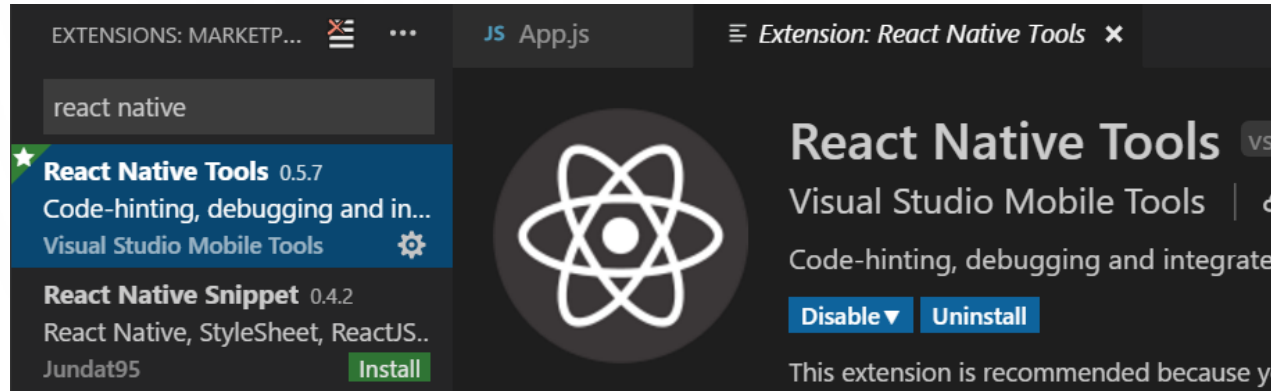
- <https://github.com/Microsoft/vscode-react-native/blob/master/doc/expo.md>

• ניתן להתקין תוסף שמאפשר לדאבג במקום
בכרום בתוך ה- visual code עצמו.

• להתקין את התוסף react native tools

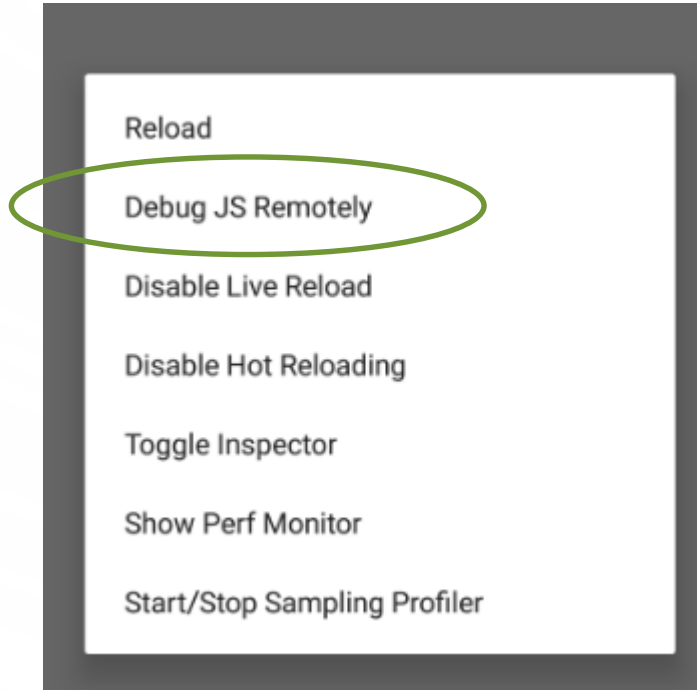
• לודא שמותקן `npm install -g react-native-cli`

• לבחור



DEBUG IN VS CODE

- לחכות למסך שייציג QR לסריקה – לוקח זמן, סבלנות 😊
- ברגע שמסתיים – לשקשק את המכשיר ולבחור את האופציה "Debug JS Remotely"
- לדאבג 😊



SYLLABUS

- 00 Debug
- **01 APK Creation**
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

APK CREATION

1. אם רוצים את האפליקציה על המכשיר אבל בתוך האפליקציה של EXPO ניתן רק לעשות PUBLISH .

- https://www.youtube.com/watch?v=JAKO1-F0Cgs&index=10&list=PL06z42zB6YZ_G3sjHluv6uj9bA76c9v7V&t=7s

1. אם רוצים כאפליקציה נפרדת יש לייצר APK

2. להתקין `npm install exp-cli`

3. להתקין `npm install -g exp`

APK CREATION CONT'

4. exp login

5. exp start (נתקע אצלי באמצע ועדין עובד בסוף)

6. exp build:android , בבחירה נא לבחור אופציה 1. לוקח הרבה זמן ~יותר מעשר דקות

7. ניתן לראות את הסטטוס ע"י exp build:status (אצלי לא מראה כלום) וניתן גם לראות את הסטטוס דרך האתר שלהם ע"י הלינק שמקבלים. (כן עובד אצלי)

8. אם הצליח להסתיים לוקאלית תקבלו APK, אם לא ניתן להוריד מהאתר ברגע שהסתיים.

9. נותר להתקין על הטלפון ע"י למשל התוכנה APKINSTALLER ב
[/http://apkinstaller.com/downloads](http://apkinstaller.com/downloads)

10. install apk on device

©NIR CHEN

<https://www.youtube.com/watch?v=N2qCAF0LBMY>

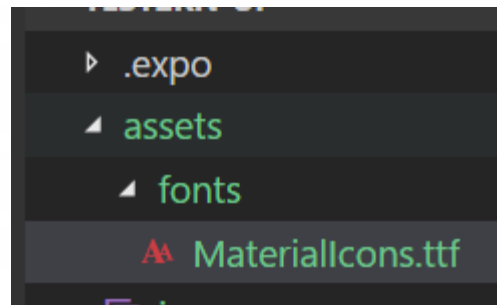
SYLLABUS

- 00 Debug
- 01 APK Creation
- **02 React Native Material UI**
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

REACT NATIVE MATERIAL UI

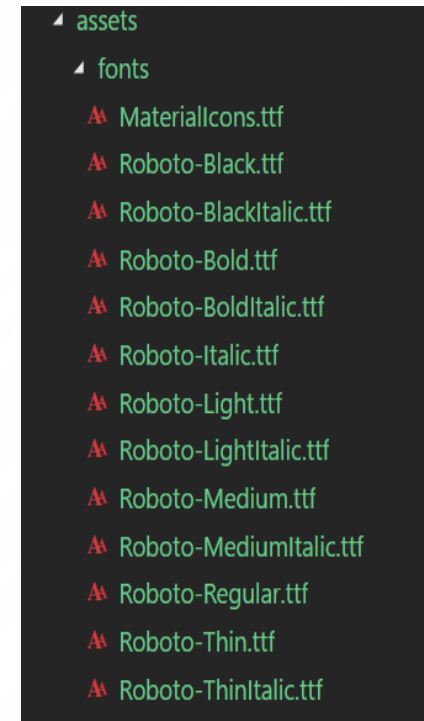
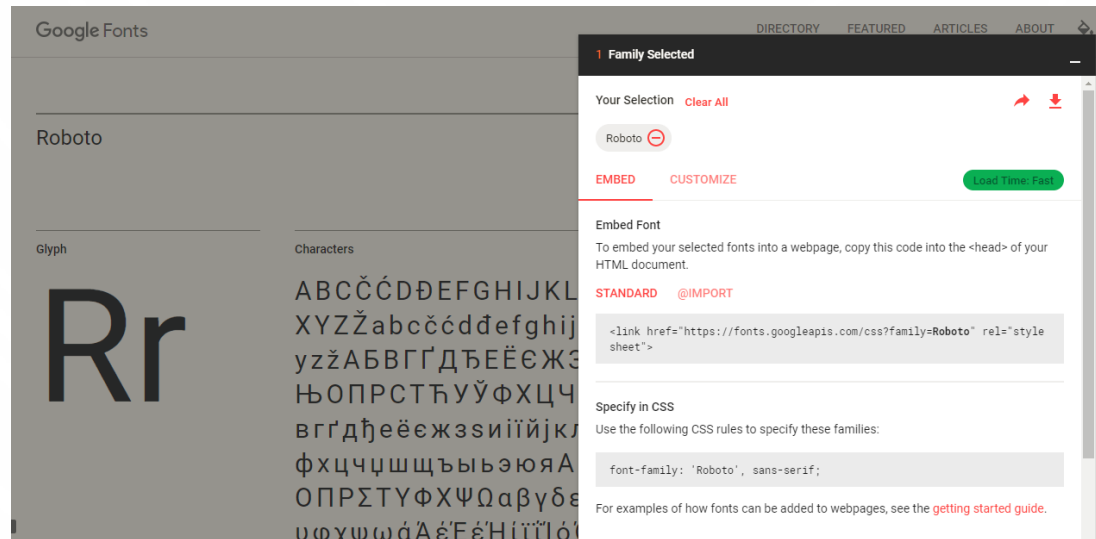
- <https://www.npmjs.com/package/react-native-material-ui>
- npm **install** react-native-material-ui --save
- npm i react-native-cli
- react-native link react-native-vector-icons
- copy from : ./node_modules/react-native-vector-icons/Fonts/MaterialIcons.ttf

to : assets/fonts



REACT NATIVE MATERIAL UI

- This project uses Roboto as the main font for text. Make sure to add Roboto to your project



REACT NATIVE MATERIAL UI

- Warp every thing in `<ThemeProvider>`
- `import { Button, ThemeProvider } from 'react-native-material-ui';`
- `onPress`

```
btnPrimaryPress(){
  alert("primary pressed!");
}

render() {
  return (
    <ThemeProvider>
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!7 {new Date().to
      <Button primary text="Primary" onPress={this.btnPrimaryPress} />
      <Button accent text="Accent" />
    </View>
  );
}
```

REACT NATIVE MATERIAL UI

- link to demo site: <https://github.com/xotahal/react-native-material-ui-demo-app/tree/master/src>
- Icons list: <https://blog.foswiki.org/System/MaterialIcons>
- **react-native-material-design** is similar BUT tested only for android and not for IOS!!!

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- **03 Fetch**
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

FETCH

- נעשה אותו דבר כמו בREACT רגיל. עובד רגיל בEXPO
- דוגמאות לקריאות ל WEB SERVICE וגם לWEB API

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- **04 Geolocation**
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

GEOLOCATION

- מתוך האתר יש ל-API כל מיני יכולות

```
navigator.geolocation.getCurrentPosition(  
  (position) => {  
    const output=  
      'latitude=' + position.coords.latitude +  
      '\nlongitude=' + position.coords.longitude +  
      '\naltitude=' + position.coords.altitude +  
      '\nheading=' + position.coords.heading +  
      '\nspeed=' + position.coords.speed  
  
    alert(output);  
  },  
  (error) => alert(error.message),  
  { enableHighAccuracy: true, timeout: 20000, maximumAge: 1000 }  
);
```

SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- **05 Camera**
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

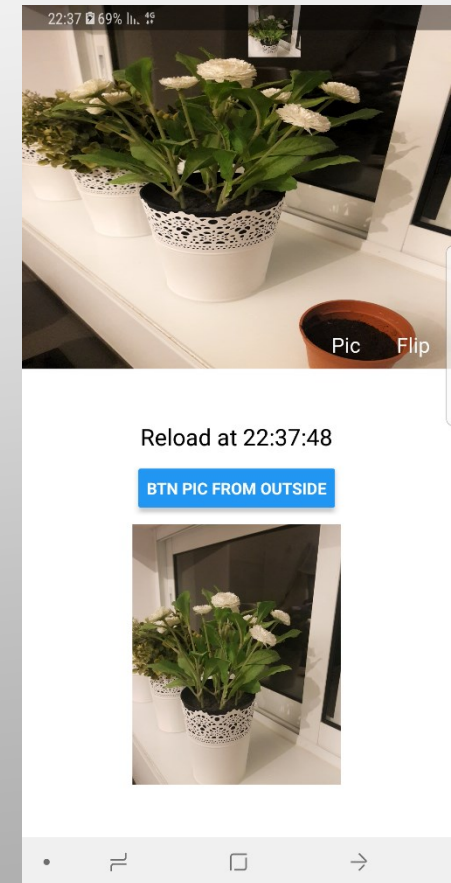
```

import { Camera, Permissions, takePictureAsync } from 'expo';
...
this.state = {
  hasCameraPermission: null,
  type: Camera.Constants.Type.back,
  picUri: 'https://facebook.github.io/react-native/docs/assets/favicon.png'
};
...
async componentWillMount() {
  const { status } = await Permissions.askAsync(Permissions.CAMERA);
  this.setState({ hasCameraPermission: status === 'granted' });
}

btnPic = async () => {
  debugger;
  let photo2 = await this.camera.takePictureAsync(); ←
  //alert(photo2.uri);
  this.setState({ picUri: photo2.uri });
  Vibration.vibrate();
}
...
onPress={() => {this.setState({
  type: this.state.type === Camera.Constants.Type.back ? Camera.Constants.Type.front : Camera.Constants.Type.back });

```

CAMERA



SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- **06 Navigation**
- 07 Image Upload
- 08 Push Notification

©NIR CHEN

```
import { createStackNavigator, createAppContainer } from
'react-navigation';
import FirstPage from './Pages/FirstPage';
import SecondPage from './Pages/SecondPage';
import TabbedPageNavigator from './Pages/TabbedPage';
```

```
class App extends React.Component {
  render() {
    return (
      <AppNavigator />
    );
  }
}
```

```
const AppNavigator = createStackNavigator(
{
  First: FirstPage,
  Second: SecondPage ,
  TabbedPage: TabbedPageNavigator
},
{
  initialRouteName: 'FirstPage',
}
);
```

```
export default createAppContainer(AppNavigator);
```

STACK NAVIGATOR

<https://reactnavigation.org/> •

V3 •

npm install --save react-navigation •

אחרי זה עוד פעם npm install •

```
<TouchableOpacity onPress={() => {
  this.props.navigation.navigate('Second');
}}>
<Text style={{ ... }} >
  Goto Second Page!</Text>
</TouchableOpacity>
```



TABBED NAVIGATOR

```
import { createBottomTabNavigator } from 'react-navigation';  
  
import TabbedAlternatePage from './TabbedAlternatePage';  
import TabbedSecondAlternatePage from './TabbedSecondAlternatePage';  
  
import Ionicons from 'react-native-vector-icons/Ionicons';  
  
class TabbedPage extends React.Component {  
  render() {  
    return (  
      <View style={styles.container}>  
        <Text style={{ color: 'red', fontSize: 28, margin: 15 }}>tabbed Page!</Text>  
      </View>  
    );  
  }  
}
```

©NIR CHEN

V2 •


```

const TabbedPageNavigator = createBottomTabNavigator(
{
  Tabbed_Page: TabbedPage,
  TabbedAlternatePage: TabbedAlternatePage,
  'Tabbed Second Alternate Page': TabbedSecondAlternatePage
},
{
  navigationOptions: ({ navigation }) => ({
    tabBarIcon: ({ focused, tintColor }) => {
      const { routeName } = navigation.state;
      let iconName;
      if (routeName === 'Tabbed_Page') {
        iconName = `ios-information-circle${focused ? '' : '-outline'}`;
      } else if (routeName === 'TabbedAlternatePage') {
        iconName = `ios-options${focused ? '' : '-outline'}`;
      }

      // You can return any component that you like here! We usually use an
      // icon component from react-native-vector-icons
      return <Ionicons name={iconName} size={25} color={tintColor} />;
    },
  })),
},

```

TABBED NAVIGATOR

V2 •

```

tabBarOptions: {
  activeTintColor: 'tomato',
  inactiveTintColor: 'gray',
  labelStyle: {fontSize: 15}
},
});

```

DRAWER NAVIGATOR

```
import {createDrawerNavigator} from 'react-navigation';
```

```
import MyHomeScreen from './Pages/Home';  
import MyDetailsScreen from './Pages/Details';
```

```
export default class App extends React.Component {  
  render() {  
    return (  
      <MyApp/>  
    );  
  }  
}
```

```
const MyApp = createDrawerNavigator({  
  Home: {  
    screen: MyHomeScreen,  
  },  
  Details: {  
    screen: MyDetailsScreen,  
  },  
});
```



V2 •

SYLLABUS

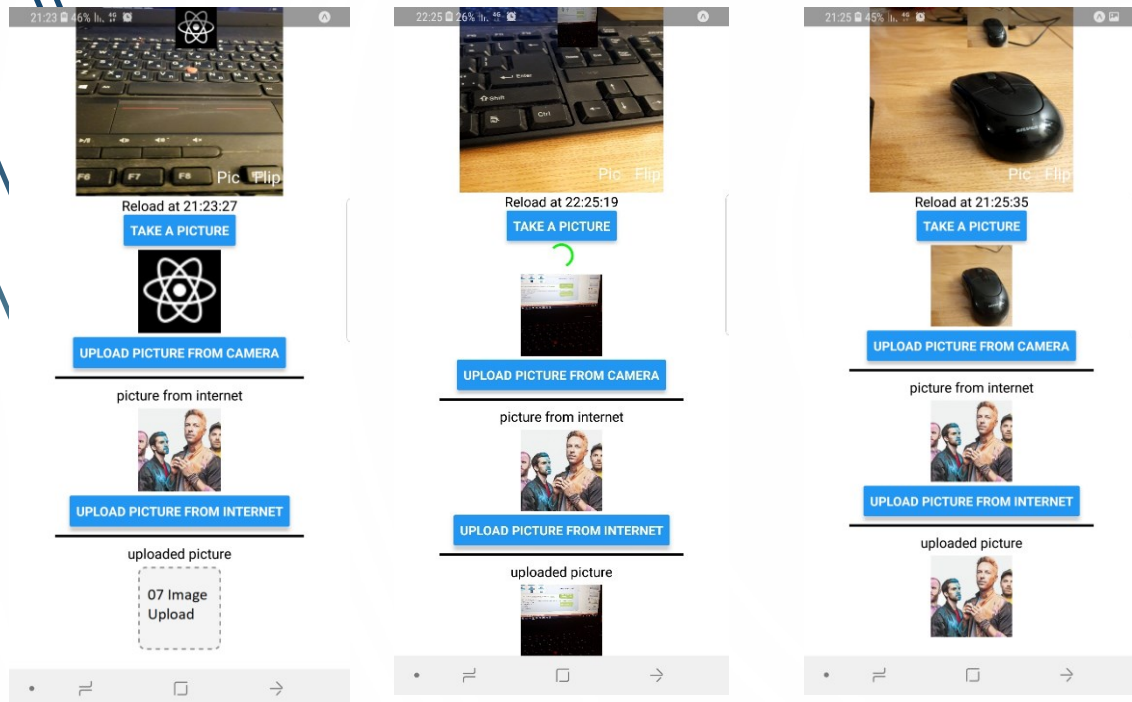
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- **07 Image Upload**
- 08 Push Notification

©NIR CHEN

WEB SERVICE

IMAGE UPLOAD – FETCH, CLIENT SIDE

- תמונה מהמצלמה: נצלם תמונה כרצף של ביטים בבסיס 64 ונשלח אותו לשרת כמחרוזת ארוכה בליווי השם של התמונה.
- תמונה מהאינטרנט: נשלח לשרת את הURL ואת השם של התמונה והשרת ב #C יוריד את התמונה וישמור אותה אצלו.



```
let photo = await this.camera.takePictureAsync({
  quality: 0.1,
  base64: true,
});
this.setState({
  pic64base: photo.base64,
  picName64base: 'image1_' + new Date().getTime() + '.jpg',
  picUri: `data:image/gif;base64,${photo.base64}`,
});
```

להקטין את גודל התמונה
שצולמה. 1 מקסימום 0
מינימום

חשוב!!! ב-RN כאשר מזמנים תמונה היא נשמרת תחת השם שלה ב-CACHE ולכן אם מזמנים תמונה עם אותו שם תופיע התמונה הראשונה שוב פעם. לכן כאשר נעלה את התמונה לשרת נצטרך ליצר לה שם חדש בכל פעם. פה אני מייצר את השם החדש. השם החדש מורכב מתחילית אשר ביקשנו ועוד מספר שנוצר מהתיקים של המחשב.

WEB SERVICE

IMAGE UPLOAD – FETCH, CLIENT SIDE

```
uploadBase64ToASMX = () => {  
  this.setState({ animate: true });  
  let urlAPI =  
    'http://ruppinmobile.tempdomain.co.il/site01 /webservice.asmx/ImgUpload';  
  
  fetch(urlAPI, {  
    method: 'POST',  
    body: JSON.stringify({  
      base64img: this.state.pic64base,  
      base64imgName: this.state.picName64base,  
    }},
```

©NIR CHEN

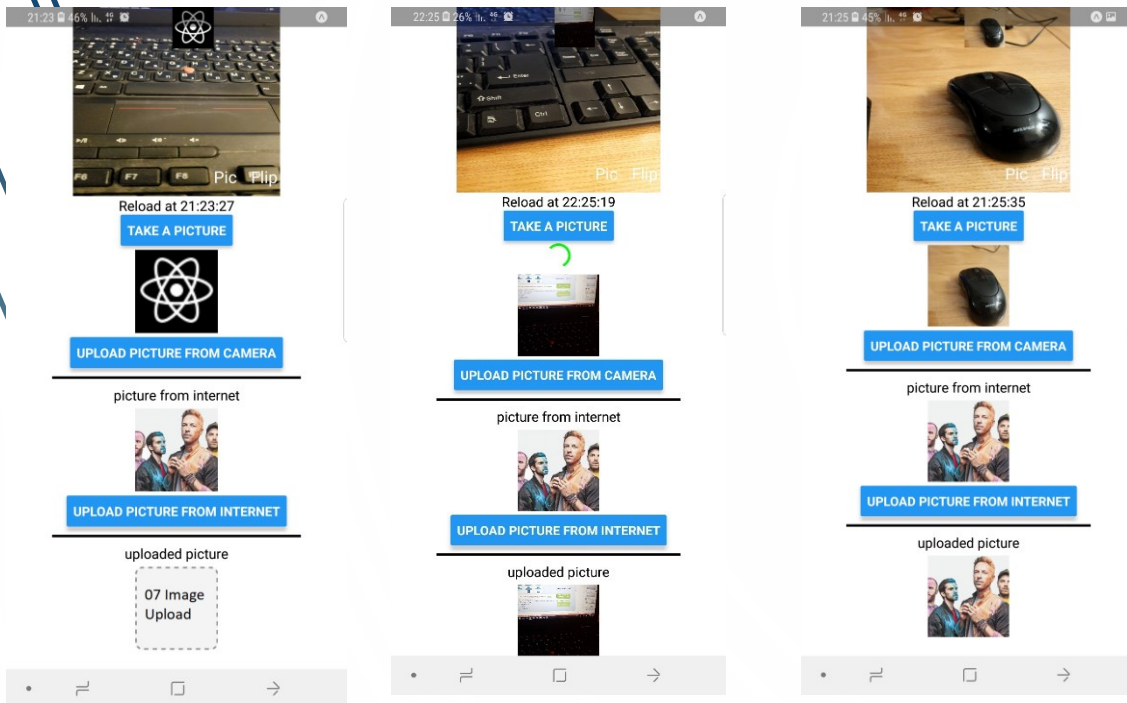
WEB SERVICE

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
[WebMethod]
public string ImgUpload(string base64img, string base64imgName)
{
    //for example - pay attention the first '/' is part of the image!
    //
    //File.AppendAllText(Server.MapPath("images/file1.txt"), base64imgName + "\r\n");
    File.WriteAllBytes(Server.MapPath("images/"+ base64imgName), Convert.FromBase64String(base64img));
    return new JavaScriptSerializer().Serialize(new { res = "OK" });
}
```

WEB API

IMAGE UPLOAD – FETCH, CLIENT SIDE



...

```
let photo = await this.camera.takePictureAsync({quality : 0.7});
```

...

```
imageUpload = (imgUri, picName) => {
```

```
  let urlAPI = "http://185.60.170.14/plesk-site-preview/ruppinmobile.ac.il/site01/uploadpicture";
```

```
  let datal = new FormData();
```

```
  datal.append('picture', {
```

```
    uri: imgUri,
```

```
    name: picName,
```

```
    type: 'image/jpeg'
```

```
  });
```

להקטין את גודל התמונה
שצולמה. 1 מקסימום 0
מינימום

מיקום הקוד של צד השרת

```
const config = {  
  method: 'POST',  
  body: data,  
}
```

```
fetch(urlAPI, config)  
  .then((responseData) => {  
    let res = responseData._bodyText;  
    let picNameWOExt = picName.substring(0, picName.indexOf("."));  
    let imageNameWithGUID = res.substring(res.indexOf(picNameWOExt), res.indexOf(".jpg")+4);  
    if (responseData.status === 201) {  
      this.setState({  
        uploadedPicUri: { uri: this.uploadDirURL + imageNameWithGUID },  
      });  
    }  
    else {  
      alert('error uploading ...');  
      ...  
    }  
  })  
  .catch(err => {  
    alert('err upload= ' + err);  
  })  
}
```

IMAGE UPLOAD – FETCH, CLIENT SIDE

חשוב!!! ב-RN כאשר מזמנים תמונה היא נשמרת תחת השם שלה ב-CACHE ולכן אם מזמנים תמונה עם אותו שם תופיע התמונה הראשונה שוב פעם. לכן כאשר נעלה את התמונה לשרת נצטרך ליצר לה שם חדש בכל פעם. זאת ניתן לעשות בצד השרת. ולקבל את השם החדש לצד הלקוח. פה אני מחלץ את השם החדש. השם החדש מורכב מתחילית אשר ביקשנו כאשר שלחנו את התמונה לשרת ועוד GUID שנוצר בשרת

בדוגמה ניתן לראות איך להשתמש ב
ActivityIndicator

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
[EnableCors(origins: "*", headers: "*", methods: "*")]  
public class ValuesController : ApiController  
{  
    ...  
    [Route("uploadpicture")]  
    public Task<HttpResponseMessage> Post()  
    {  
        string outputForNir="start---";  
        List<string> savedFilePath = new List<string>();  
        if (!Request.Content.IsMimeMultipartContent())  
        {  
            throw new HttpResponseException(HttpStatusCode.UnsupportedMediaType);  
        }  
        string rootPath = HttpContext.Current.Server.MapPath("~/uploadFiles");  
        var provider = new MultipartFileStreamProvider(rootPath);  
        var task = Request.Content.ReadAsMultipartAsync(provider).  
            ContinueWith<HttpResponseMessage>(t =>  
            {  
                if (t.IsCanceled || t.IsFaulted)  
                {  
                    Request.CreateErrorResponse(HttpStatusCode.InternalServerError, t.Exception);  
                }  
            })  
    }
```

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
foreach (MultipartFormData item in provider.FileData)
{
    try
    {
        outputForNir += " ---here";
        string name = item.Headers.ContentDisposition.FileName.Replace("\\", "");
        outputForNir += " ---here2=" + name;

        //need the guid because in react native in order to refresh an image it has to have a new name
        string newFileName = Path.GetFileNameWithoutExtension(name) + "_" + Guid.NewGuid() + 
        Path.GetExtension(name);
        //string newFileName = name + "" + Guid.NewGuid();
        outputForNir += " ---here3" + newFileName;


        //delete all files begining with the same name
        string[] names = Directory.GetFiles(rootPath);
        foreach (var fileName in names)
        {
            if (Path.GetFileNameWithoutExtension(fileName).IndexOf(Path.GetFileNameWithoutExtension(name)) != -
            1)
            {
                File.Delete(fileName);
            }
        }

        //File.Move(item.LocalFileName, Path.Combine(rootPath, newFileName));
        File.Copy(item.LocalFileName, Path.Combine(rootPath, newFileName), true);
        File.Delete(item.LocalFileName);
        outputForNir += " ---here4";
    }
}
```

IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
Uri baseuri = new Uri(Request.RequestUri.AbsoluteUri.Replace(Request.RequestUri.PathAndQuery, string.Empty));
    outputForNir += " ---here5" ;
    string fileRelativePath = "~/uploadFiles/" + newFileName;
    outputForNir += " ---here6 imageName=" + fileRelativePath;
    Uri fileFullPath = new Uri(baseuri, VirtualPathUtility.ToAbsolute(fileRelativePath));
    outputForNir += " ---here7" + fileFullPath.ToString();
    savedFilePath.Add(fileFullPath.ToString());
}
catch (Exception ex)
{
    outputForNir += " ---exception=" + ex.Message;
    string message = ex.Message;
}
}

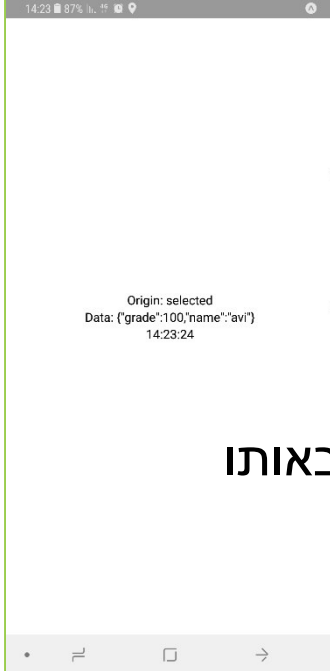
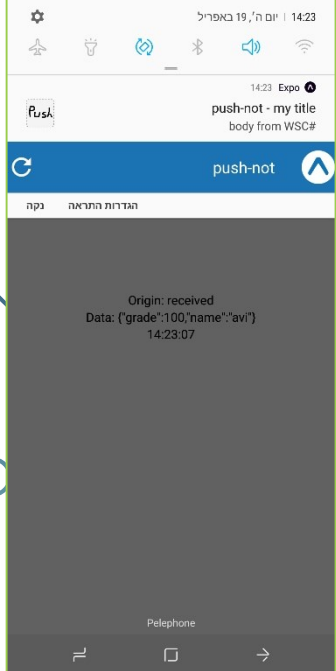
return Request.CreateResponse(HttpStatusCode.Created, "nirchen " + savedFilePath[0] + "!" +
provider.FileData.Count + "!" + outputForNir + ":)");
});
return task;
}
```



SYLLABUS

- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- **08 Push Notification**

©NIR CHEN



PUSH NOTIFICATION

- נעשה שימוש בספריה של EXPO בכדי לקבל יכולות של PN עבור ANDROID ו-IOS באותו קוד. כמו כן לא צריך לייצר משתמש של APPLE.

• <https://docs.expo.io/versions/latest/guides/push-notifications>

- שימו לב שצריך לאפשר את ההתראות במכשיר. בחלק ממכשירי ה- ANDROID לא הצלחתי לאפשר זאת ויכול להיות שצריך לחפש "לשחק" הכן מאפשרים את ההתראות.
- אין צורך לפתוח חשבון בשירות ענן כלשהוא. EXPO כבר מבצעים הכל עבורינו.
- בכדי לשלוח הודעה כל מה שצריך הוא לשלוח הודעת POST לשרת של EXPO שמעביר את ההודעה לשירות ענן בכדי לשלוח PN.

- יש מערכת ששולחת הודעות ישירות מהאתר של EXPO בכדי לבחון את הקוד בצד לקוח

<https://expo.io/dashboard/notifications>

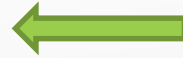
☒ Play sound

JSON DATA

```
{"grade": 100, "name": "avi"}
```

```
import { Permissions, Notifications } from 'expo';
```

```
export default async function registerForPushNotificationsAsync() {  
  const { status: existingStatus } = await Permissions.getAsync(  
    Permissions.NOTIFICATIONS  
  );  
  let finalStatus = existingStatus;
```



```
  // only ask if permissions have not already been determined, because iOS won't necessarily prompt the user a second time.
```

```
  if (existingStatus !== 'granted') {  
    // Android remote notification permissions are granted during the app install, so this will only ask on iOS  
    const { status } = await Permissions.askAsync(Permissions.NOTIFICATIONS);  
    finalStatus = status;  
  }
```

```
  // Stop here if the user did not grant permissions  
  if (finalStatus !== 'granted') {  
    return;  
  }
```

```
  // Get the token that uniquely identifies this device  
  let token = await Notifications.getExpoPushTokenAsync();
```



```
  //alert(token);  
  // POST the token to your backend server from where you can retrieve it to send push notifications.  
  return {  
    token  
  };  
}
```

PN – CLIENT SIDE REGISTRATION

• פה צריך לדאוג ל:

- קבלת הרשאה ליכולת קבלת התראות
- קבלת מספר ייחודי מזהה של מכשיר הטלפון

PN – CLIENT SIDE RECEIVING MSG

```
...
import { Notifications } from 'expo';
import registerForPushNotificationsAsync from './registerForPushNotificationsAsync';

export default class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      notification: {},
    };
  }

  componentDidMount() {
    registerForPushNotificationsAsync();
    this._notificationSubscription = Notifications.addListener(this._handleNotification);
  }

  _handleNotification = (notification) => {
    this.setState({ notification: notification });
  };

  render() {
    return (
      <Text>Origin: {this.state.notification.origin}</Text>
      <Text>Data: {JSON.stringify(this.state.notification.data)}</Text>
    );
  }
}
```

- פה נעשה רישום של פונקציה לקבלת התראה
- יש אפשרות לראות בהתראה עצמה בין היתר
TITLE, BODY,
BADGE - באייקון
- DATA - באפליקציה

SYLLABUS

- -01 Expo
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

©NIR CHEN

- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- **14 React Elements Lib**

REACT NATIVE ELEMENTS UI

```
import { Button, ThemeProvider } from 'react-native-elements';
import Icon from 'react-native-vector-icons/FontAwesome';
```

```
const MyApp = () => {
  return (
    <ThemeProvider> ←
    <Button
      icon={
        <Icon
          name="arrow-right"
          size={15}
          color="white"
        />
      }
      iconRight
      title="Button with right icon"
    />
  </ThemeProvider>
);
};
```

- <https://react-native-training.github.io/react-native-elements/>
- `npm install --save react-native-elements`

MORE EXPO CAPABILITIES!!!

SDK API REFERENCE

Introduction

Accelerometer

Admob

Amplitude

AppLoading

ART

Asset

Audio

AuthSession

AV

BarCodeScanner

BlurView

Branch

Brightness

Calendar

Camera

Constants

Contacts

DeviceMotion

DocumentPicker

ErrorRecovery

FacebookAds

Facebook

FaceDetector

FileSystem

Fingerprint

Font

GestureHandler

GLView

Google

Gyroscope

ImageManipulator

ImagePicker

IntentLauncherAndroid

KeepAwake

LinearGradient

Localization

Localization

Location

Lottie

Magnetometer

MailComposer

MapView

Notifications

Payments

Pedometer

Permissions

Print

registerRootComponent

ScreenOrientation

SecureStore

Segment

Speech

SQLite

Svg

takeSnapshotAsync

Updates

Video

WebBrowser