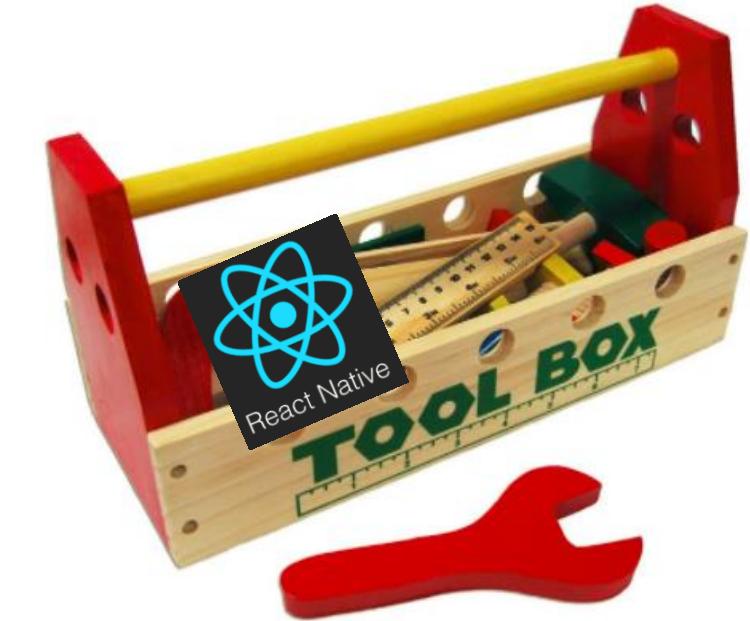




# REACT NATIVE TOOLBOX

01 - 15

©NIR CHEN



# SYLLABUS

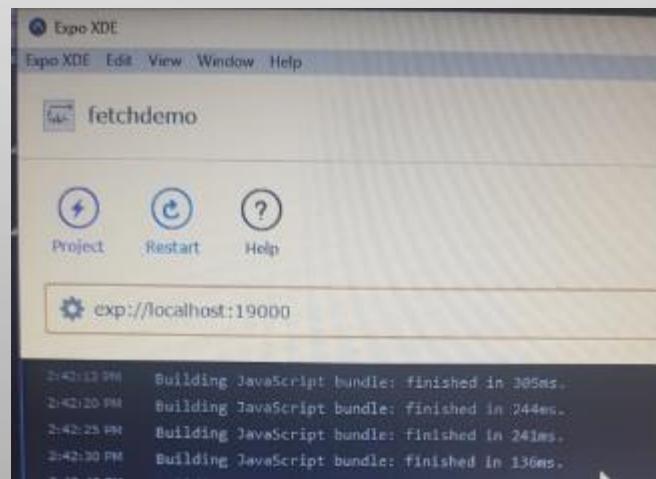
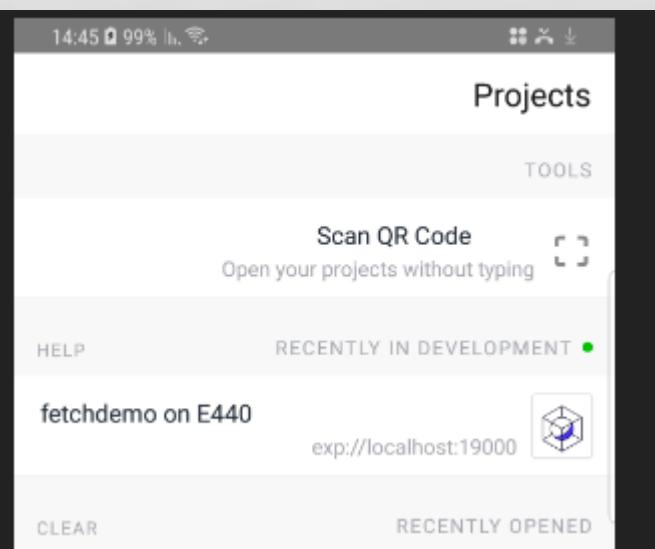
- **-01 Expo**
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

©NIR CHEN

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

# EXPO

- הכי מהיר זה לעבוד בLOCALHOST כאשר המחשב וגם המחשב מחוברים על אותה רשת WIFI.
- חובה לחבר את המחשב למחשב עם כבל USB ולאפשר DEBUGGING ("איתור באגים של USB") דרך הUSB ב"אפשרויות למפתחים"



# SYLLABUS

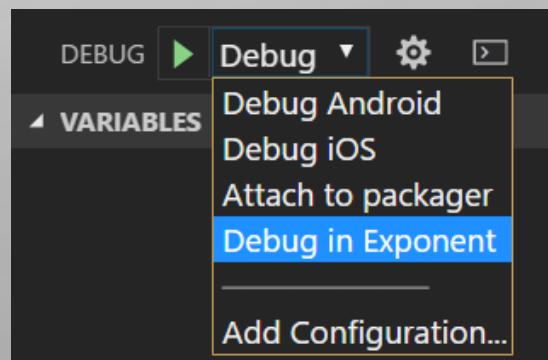
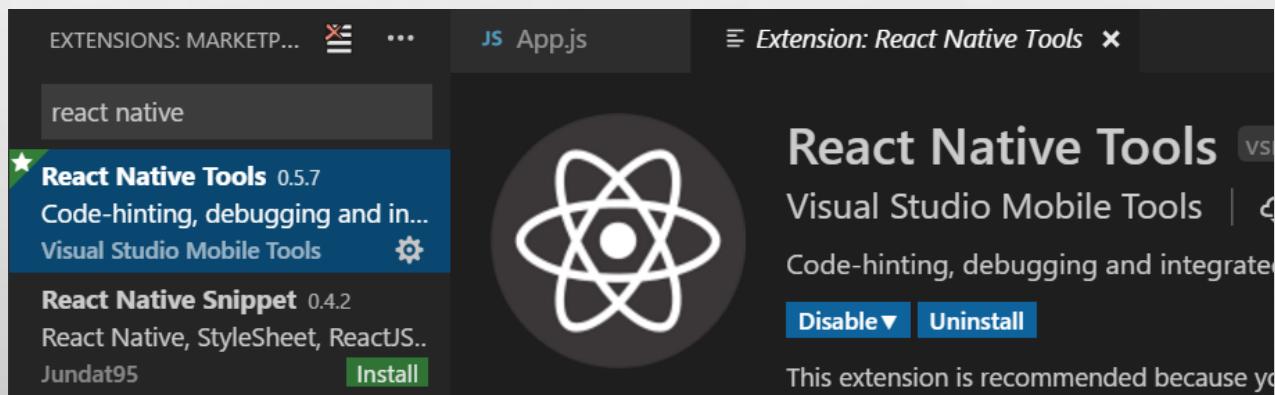
- **00 Debug**
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification

# DEBUG

1. לשקש **לנענען** נעה את המכשיר ואז לבחור בתפריט את "Debug JS Remotely"
2. יפתח הכרום בצורה אוטומטית ואז יש ללחוץ F12
3. יש לבחור את הטעב של Sources
4. כאשר נגיע ל**POINT BREAK** הקוד יעצר בשורה המתאימה ואז אפשר לדאגג רגיל.

# DEBUG IN VS CODE

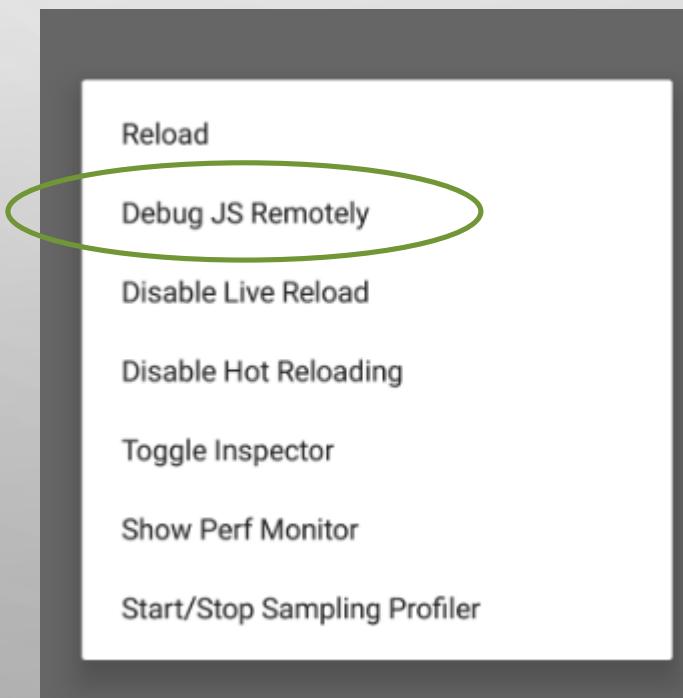
- <https://github.com/Microsoft/vscode-react-native/blob/master/doc/expo.md>
- ניתן להתקין תוסף שמאפשר לדאגג במקום בכרום בתוך ה- **visual code** עצמו.
- להתקין את התוסף **react native tools**
- לוודא שמותקן **npm install -g react-native-cli**



לבחירה

# DEBUG IN VS CODE

- לחכמת למסר שיציג QR לסריקה – לוקח זמן, סבלנות😊
- ברגע שמסתים – לשקשק את המכשיר ולבחר את האופציה "Debug JS Remotely"



"Remotely"  
• לדאג😊

# 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\\_G3sjHluv6u9bA76c9v7V&t=7s](https://www.youtube.com/watch?v=JAkO1-F0Cgs&index=10&list=PL06z42zB6YZ_G3sjHluv6u9bA76c9v7V&t=7s)

1. אם רוצים כAPPLICATION נפרדת יש ליצור APK
2. להתקין `npm install exp-cli`
3. להתקין `npm install -g exp`

# APK CREATION CONT'

4. exp login
5. exp start (נתקע אצלך 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 (אצלך לא מראה כלום) ונitin גם לראות את הסטטוס דרך האתר שלהם ע"י הLINK שמקבלים. (כן עובד אצלך)
8. אם הצליח להסתמיכם לוקאלית תקבלו APK, אם לא ניתן להוריד מהאתר ברגע שהסתמיכם.
9. נותר להתקין על הטלפון ע"י למשל התוכנה APKINSTALLER ב <http://apkinstaller.com/downloads>

## 10.install apk on device

©NIR CHEN

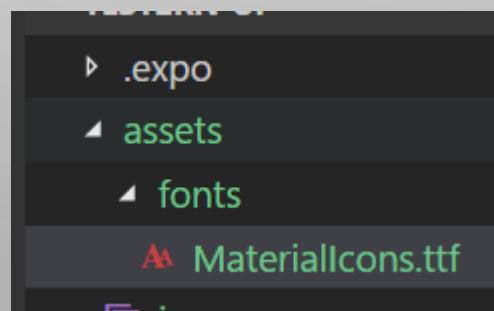
<https://www.youtube.com/watch?v=N2qCAFOLBMY>

# 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

The image consists of three side-by-side screenshots. The left screenshot shows the Google Fonts website with 'Roboto' selected. The middle screenshot shows a mobile application interface with large, bold 'Rr' characters. The right screenshot shows a file browser listing various Roboto font files.

**Google Fonts Screenshot:**

- Family Selected: Roboto
- Embed Font (Standard): `<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">`
- Specify in CSS: `font-family: 'Roboto', sans-serif;`

**Mobile Application Screenshot:**

**File Browser Screenshot:**

- assets
- fonts
- Roboto-Black.ttf
- Roboto-BlackItalic.ttf
- Roboto-Bold.ttf
- Roboto-BoldItalic.ttf
- Roboto-Italic.ttf
- Roboto-Light.ttf
- Roboto-LightItalic.ttf
- Roboto-Medium.ttf
- Roboto-MediumItalic.ttf
- Roboto-Regular.ttf
- Roboto-Thin.ttf
- Roboto-ThinItalic.ttf

# 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>
    </ThemeProvider>
  );
}
```

# 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 בEXPRESS
- דוגמאות לקריאה לWEB API ו גם לWEB SERVICE

# 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

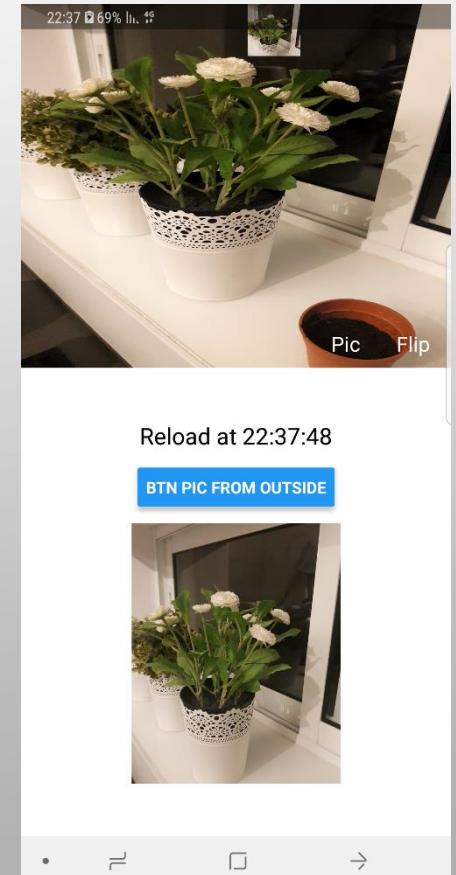
```

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={ONRICHEN}=>{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

```
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);
```

©NIR CHEN

# STACK NAVIGATOR

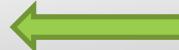
<https://reactnavigation.org/> •

V3 •

npm install --save react-navigation •

npm install --save react-navigation •

```
<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>  
    );  
  }  
}
```

V2 •

# TABBED NAVIGATOR

```
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} />;  
      },  
    },  
  },  
);  
©NIR CHEN
```

V2 •

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

# DRAWER NAVIGATOR

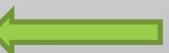
V2 •

```
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,
  },
});
```



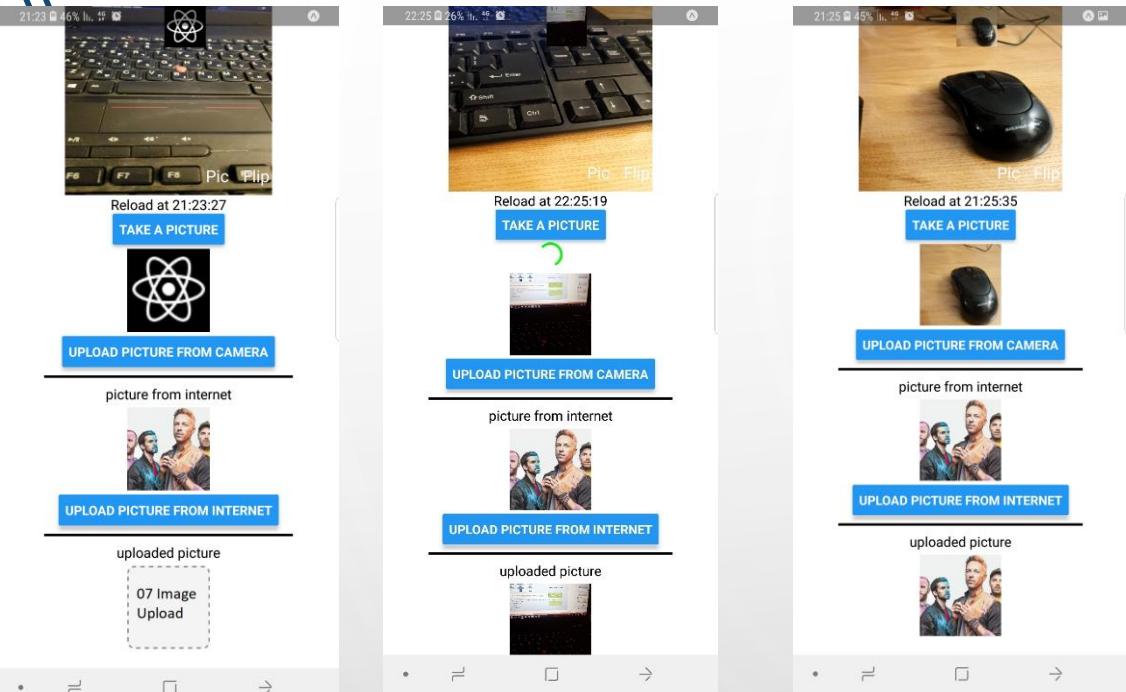
©NIR CHEN

# 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

# WEB SERVICE

## IMAGE UPLOAD – FETCH, CLIENT SIDE



- תמונה מהצלמה: נצלם תמונה כרץ של ביטים בסיס 64
- ונסלח אותו לשרת כמחוזת אורך בלויי השם של התמונה.
- תמונה מראינטנט: נשלח לשרת את הURL ואת השם של התמונה והשרות ב C# יוריד את התמונה ויישמור אותה אצלך.

```
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!!! בRN כאשר מזמינים תמונה היא נשמרת תחת השם  
שלה ב CACHE וכך אם מזמינים תמונה עם אותו שם תופיע  
התמונה הראשונה שוב פעם. לכן כאשר נעלמת התמונה  
לשרת נדרש ליצור לה שם חדש בכל פעם. פה אני מייצר את  
שם החדש. השם החדש מורכב מתחלית אשר ביקשנו ועוד  
מספר שמנצ' מהטייקים 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

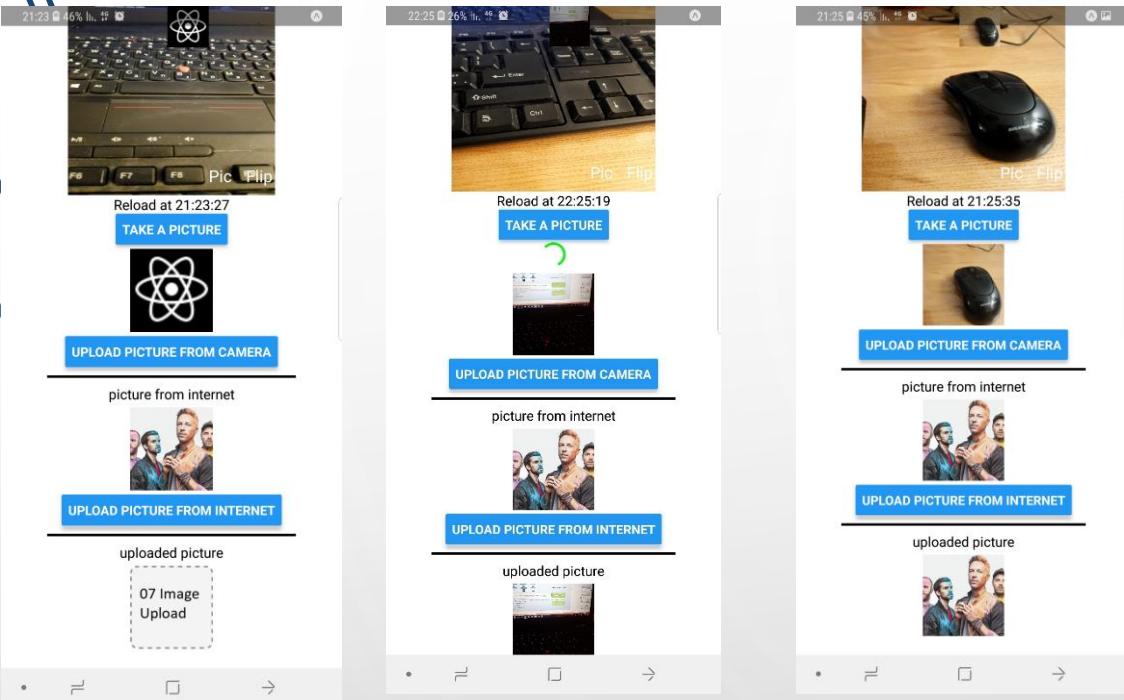
## IMAGE UPLOAD – C# WEB API, SERVER SIDE

```
[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 data1 = new FormData();  
  data1.append('picture', {  
    uri: imgUri,  
    name: picName,  
    type: 'image/jpg'  
  });  
};
```

להקטין את גודל התמונה  
שצולמה. 1 מקסימום 0  
מינימום

מקום הקוד של צד השרת

# IMAGE UPLOAD – FETCH, CLIENT SIDE

```
...  
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);  
})  
}
```

חשוב!!! בRN כאשר מזמינים תמונה היא נשמרת תחת השם  
שלה ב CACHE ולכן אם מזמינים תמונה עם אותו שם תופיע  
התמונה הראשונה שוב פעם. אך כאשר נעלמת התמונה  
לשרת נוצר ליצר לה שם חדש בכל פעם. זאת ניתן לעשות  
בצד השירות. ולקבל את השם החדש לצד הליקו. פה אני  
מחלץ את השם החדש. השם החדש מורכב מתחלית אשר  
ביקשנו כאשר שלחנו את התמונה לשרת ועוד GUID שנוצר  
ברשת

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 (MultipartFileData 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 inamge 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

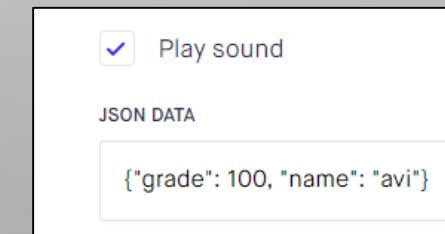
- 00 Debug
- 01 APK Creation
- 02 React Native Material UI
- 03 Fetch
- 04 Geolocation
- 05 Camera
- 06 Navigation
- 07 Image Upload
- 08 Push Notification



# PUSH NOTIFICATION

- נעשה שימוש בספריה של EXPO ב כדי 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>
- שימוש לב שוצרק לאפשר את ההתראות במכשיר. בחלק ממכシリ ה- ANDROID לא הצלחתי לאפשר זאת ויכול להיות שצריך לחפשו "לשחק" הכן מאפשרים את ההתראות.
- אין צורך לפתח חשבון בשירות ענן כלשהו. EXPO כבר מבצעים הכל עבורינו.
- ב כדי לשלוח הודעה כל מה שצריך הוא לשלוח הודעה POST לשרת של EXPO שמעביר את הודעה לשירות ענן ב כדי לשלוח PN.

יש מערכת שלוחת הודעה ישירות מהאתר של EXPO ב כדי לבדוק את הקוד בצד לקוח



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

# PN – CLIENT SIDE REGISTRATION

```
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 (
    ©NIR CHEN
    token
  );
}
```

- פה צריך לדאוג ל:
- קבלת הרשות ליכולה לקבלת 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 לקליטת התראה
- יש אפשרות לראות בהתראה עצמה בין היתר TITLE, BODY, BADGE - באיךון
- באפליקציה - DATA

# 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 UI Lib

# COMPASS

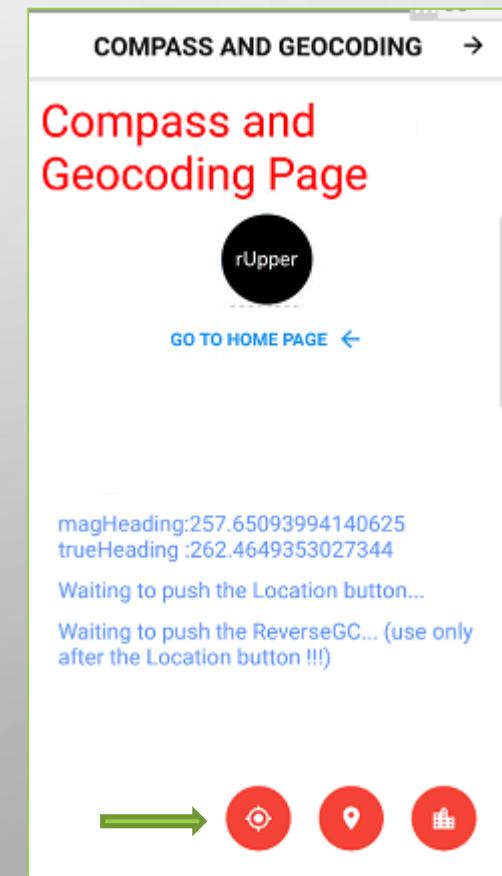
```
import { Location, Permissions } from 'expo';
...
btnHeading = async () => {
  let { status } = await Permissions.askAsync(Permissions.LOCATION);
  if (status !== 'granted') {
    alert('Permission to access location was denied');
  }

  let heading = await Location.getHeadingAsync({});
  this.setState({ heading });
};
```

©NIR CHEN

- ניתן לקבל מידע לקבל הכוון שהטלפון מפנה אליו.
- כיוון אמיתי וכיוון מגנטי.

magHeading:275.7325439453125  
trueHeading :280.5465393066406



# 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 UI Lib

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

```
...
```

```
btnLocation = async () => {
```

```
  let { status } = await Permissions.askAsync(Permissions.LOCATION);
```

```
  if (status !== 'granted') {
```

```
    this.setState({errorMessage: 'Permission to access location was denied',});
```

```
}
```

```
let location = await Location.getCurrentPositionAsync({});
```

```
this.setState({ location });
```

```
};
```

```
btnReverseGC = async () => {
```

```
  let { status } = await Permissions.askAsync(Permissions.LOCATION);
```

```
  if (status !== 'granted') {
```

```
    this.setState({ errorMessage: 'Permission to access location was denied',});
```

```
}
```

```
if (this.state.location) {
```

```
  let reverseGC = await Location.reverseGeocodeAsync(this.state.location.coords);
```

```
  this.setState({ reverseGC });
```

```
}else{
```

```
  alert('You must push the Location button first in order to get the loca
```

```
can get the reverse geocode for the latitude and longitude!');
```

```
}
```

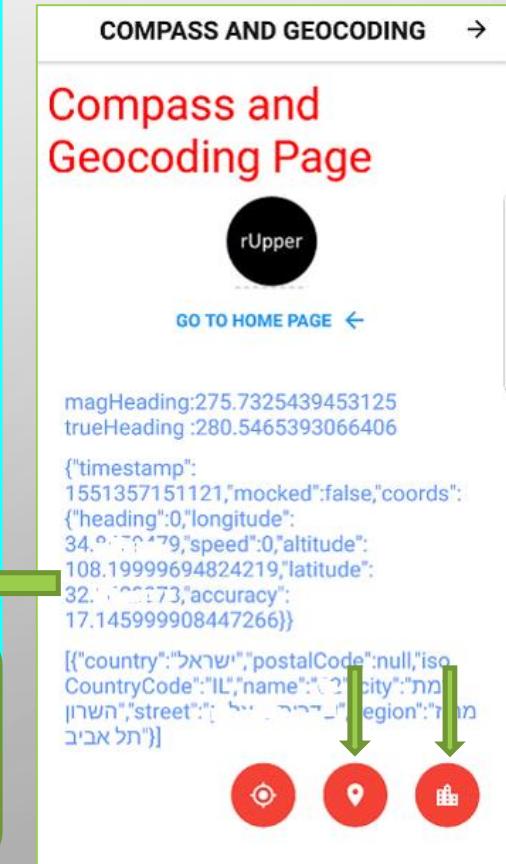
```
};
```

# GEOCODING

- ניתן לקבל מידע אודות כתובות עברו נ.צ. קיימ.

עברו כתובות

- ניתן לקבל גם הFOR - נ.צ. עברו כתובות (לא בדוגמה זו)



# 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

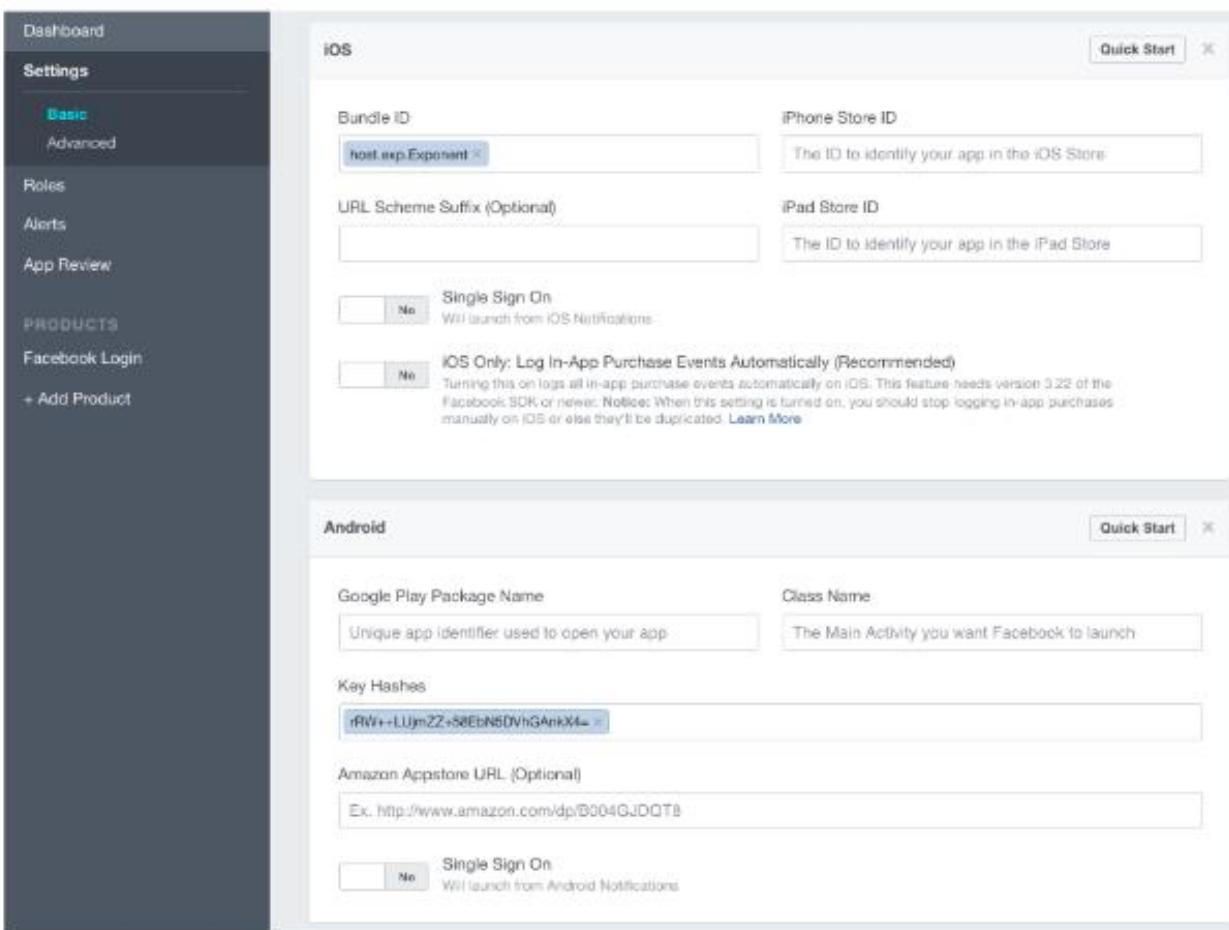
- 09 Compass
- 10 Geocoding
- **11 Facebook**
- 12 Image gallery
- 13 Sms
- 14 React Elements UI Lib

- צריך ליצור פרויקט ב-FB developer API כדי לקבל appId

- <https://docs.expo.io/versions/latest/sdk/facebook/>

- The Expo client app

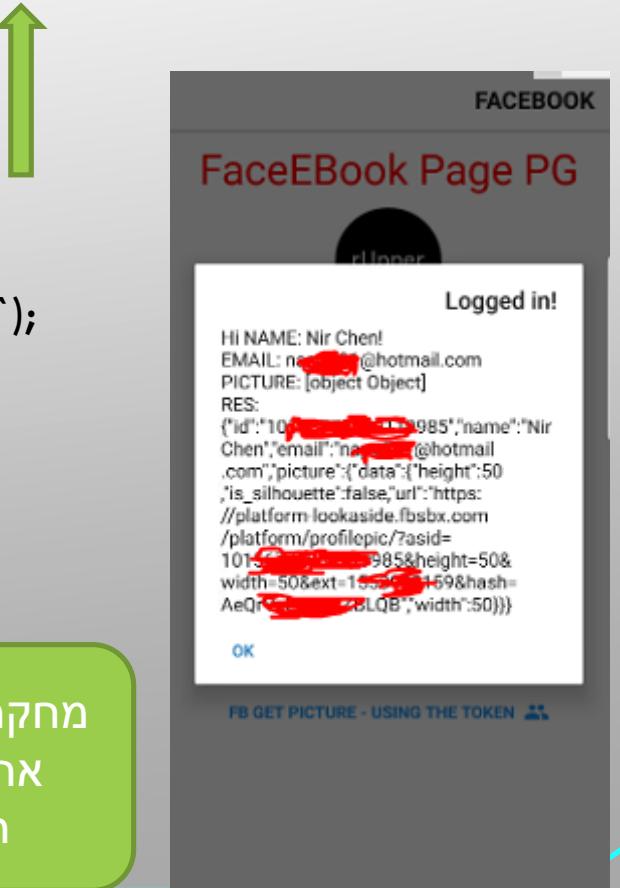
- Add `host.exp.Exponent` as an iOS Bundle ID. Add `rRW++LUjmZZ+58EbN5DVhGANkX4=` as an Android key hash. Your app's settings should end up including the following under "Settings > Basic":



# FB

- צריך לקבל token ייחודי בצד' ל לעבוד עם ה API שלהם.

```
import { Facebook } from 'expo';
...
btnLoginFB = async () => {
  const { type, token, expires, permissions, declinedPermissions, } = await Facebook.logInWithReadPermissionsAsync(this.appId,
{ permissions: ['public_profile'], });
  if (type === 'success') {
    //after getting the token we can use a simple fetch against the facebook API
    // Get the user's name using Facebook's Graph API
    const response = await
fetch(`https://graph.facebook.com/me?fields=id,name,email,picture&access_token=${token}`);
    let res = await response.json();
    this.setState({ token: token });
    Alert.alert('Logged in!', `Hi NAME: ${res.name}!\nEMAIL: ${res.email}\nPICTURE:
${res.picture}\nRES:${JSON.stringify(res)}`);
  } else {
    // type === 'cancel'
  }
};
```

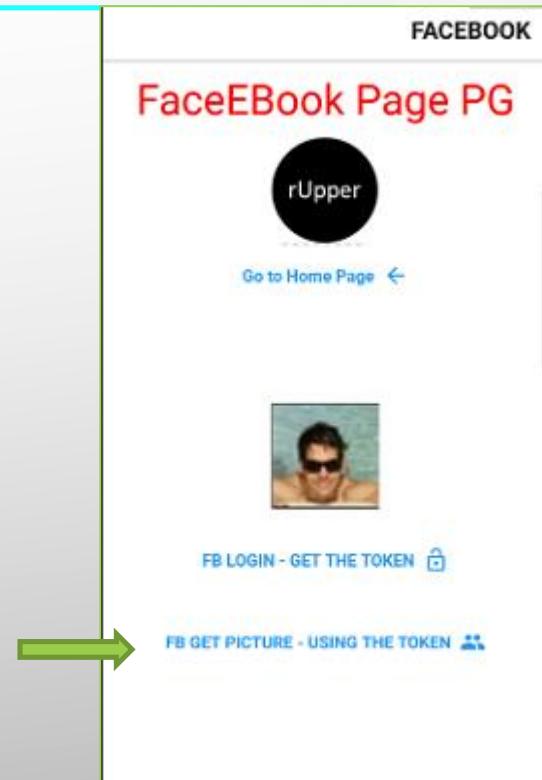


מחקתי מהדוגמה  
את הפרטים  
ה אישיים

# FB

```
btnFetch_PersonPicture = () => {
  // POST adds a random id to the object sent
  fetch(`https://graph.facebook.com/me?fields=picture&access_token=${this.state.token}`, {
    method: 'POST',
    body: '',
    headers: {
      "Content-type": "application/json; charset=UTF-8"
    }
  })
  .then(response => response.json())
  .then(json => {
    if (json != null) {
      this.setState({ photoUrl: json.picture.data.url });
      alert(`picture= ${json.picture}\npicture.data.url= ${json.picture.data.url}\nRES=${JSON.stringify(json)}`);
    } else {
      this.setState({ lblErr: true });
    }
  });
}
```

©NIR CHEN



- פה אני מקבל מראת תמונה

# 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

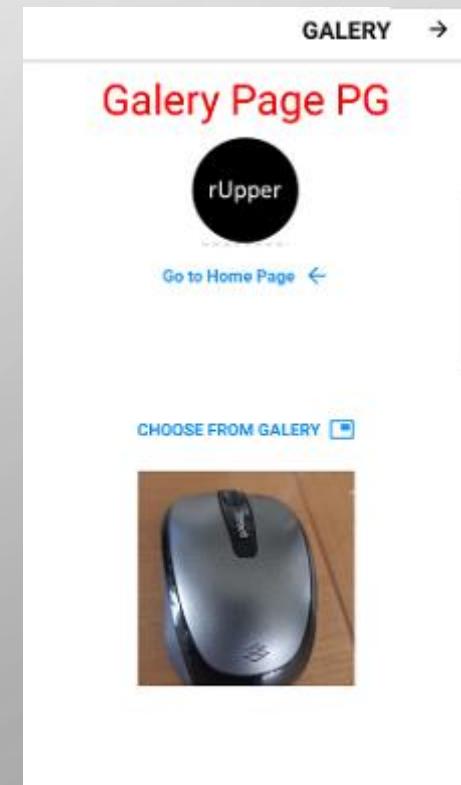
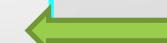
- 09 Compass
- 10 Geocoding
- 11 Facebook
- **12 Image gallery**
- 13 Sms
- 14 React Elements UI Lib

# IMAGE GALLERY

- ניתן לבחור תמונה מה갤ריה

```
import { ImagePicker } from 'expo';
...
btnOpenGalery = async () => {
  let result = await ImagePicker.launchImageLibraryAsync({
    //allowsEditing: true,
    //aspect: [4, 3],
  });

  if (!result.cancelled) {
    this.setState({ image: result.uri });
  }
};
```



# 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

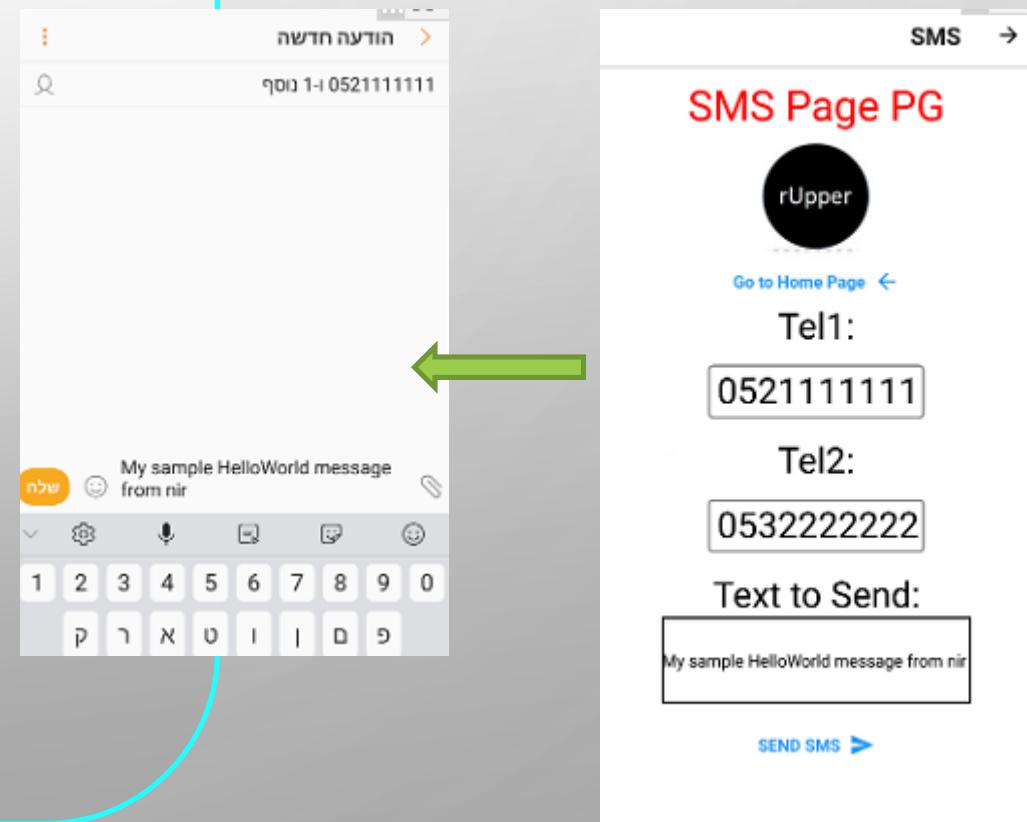
- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- **13 Sms**
- 14 React Elements UI Lib

# SMS

```
import { SMS } from 'expo';
...
btnSendSms = async () => {
  const isAvailable = await SMS.isAvailableAsync();
  if (isAvailable) {
    //alert('send');
    const { result } =
      await SMS.sendSMSAsync(
        [this.state.txtTel1, this.state.txtTel2],
        this.state.txtTextValue);
    // alert(result);
    this.setState({ txtTextValue: result });
  } else {
    alert('misfortune... there\'s no SMS available on this device');
  }
};
```

©NIR CHEN

- ניתן לפתח את מסך הוסעת ה SMS עם רשימת טלפונים והודעה מוכנה מהאפליקציה שלנו.
- זה לא שולח מהאפליקציה שלנו אלא רק פותח את אפליקציית ה SMS-ים הקיימת



# 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

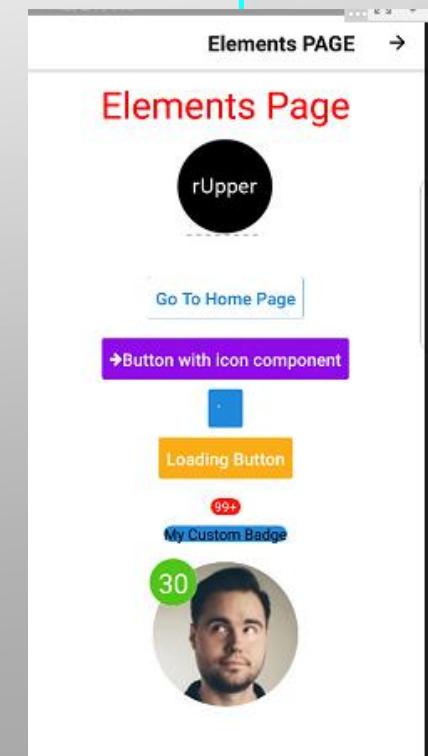
- 09 Compass
- 10 Geocoding
- 11 Facebook
- 12 Image gallery
- 13 Sms
- **14 React Elements UI Lib**

# 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"
    />
  );
};
```

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



# 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 UI Lib
- 15 Linking

Open a url

```
Linking.openURL('https://expo.io');  
WebBrowser.openBrowserAsync('https://expo.io');
```

...

```
Linking.openURL('mailto:support@expo.io?subject=Congrats Snoopy&body=Enjoy your  
stay,%0ARegards');
```

...

```
Linking.openURL('tel:+123456789');
```

....

```
Linking.openURL('sms:+123456789');
```

...

```
//this will open the app through whatsapp  
//1. need to publish the app first!  
//2. when clicked in whatsapp this will go to the published page on expo.io  
// then you can open the app in the expo if installed.
```

```
btnWhatsapp = () => {  
  let text = "hello ";  
  text = 'https://expo.io/@nirc/get-a-ride-demo';  
  let phoneNumber = '+972523333333';  
  Linking.openURL(`whatsapp://send?text=${text}&phone=${phoneNumber}`);  
}
```

©NIR CHEN

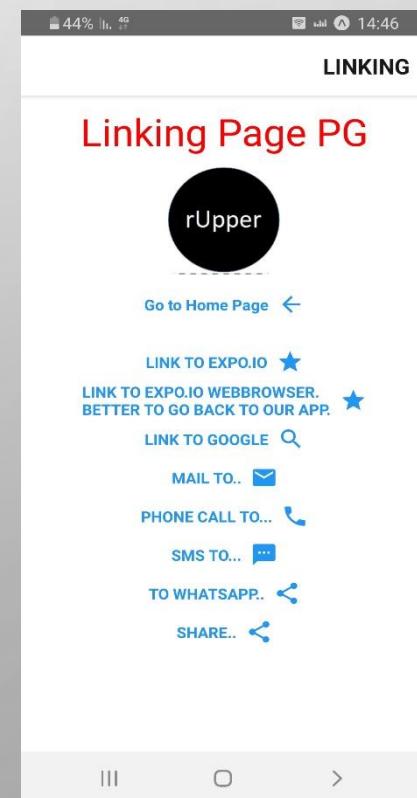
Send  
whatsapp

Send mail

Open dialer

Send sms

- כל מה שקשר ליצירת קשר עם אפליקציות  
חיצונית

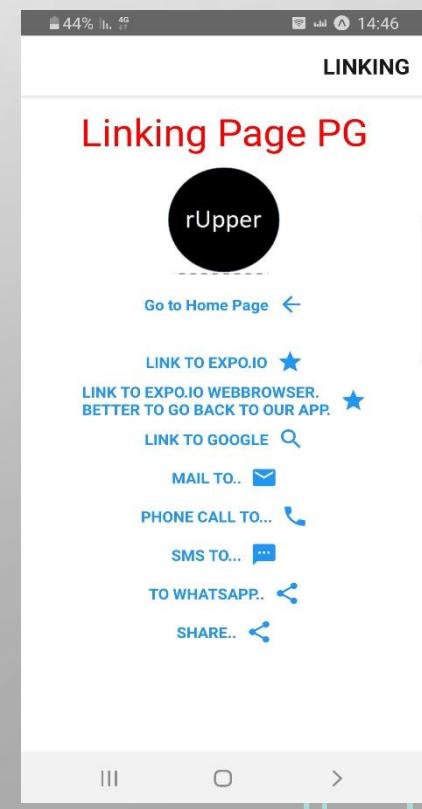


# LINKING

```
btnShare = () => {
  //let text = Expo.Linking.makeUrl();
  let text = 'https://expo.io/@nirc/get-a-ride-demo';
  Share.share({
    message: "Click Here to View More! " + text,
    url: text,
    title: 'nir has invited you to join this activity',
  })
  .then((result) => {
    console.log(result)
    if (result === 'dismissedAction') {
      return
    }
  })
  .catch((error) => console.log(error))
}
```

Open all sharable apps

- כל מה שקיים ליצירת קשר עם אפליקציות חיצונית



# MORE EXPO CAPABILITIES!!!

## SDK API REFERENCE

Introduction

Accelerometer

Admob

Amplitude

AppLoading

ART

Asset

Audio

AuthSession ←

AV

BarCodeScanner

BlurView

Branch

Brightness

Calendar

Camera

Constants

©NIR CHEN

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

