React Native

Juha Hinkula
Github: juhahinkula

React Native

- Framework for building mobile apps with Javascript and React
- Developed by Facebook
- https://facebook.github.io/react-native/
- React Native uses same UI elements as native Android and IOS apps
- See the apps that are using React Native
  https://facebook.github.io/react-native/showcase.html

React Native

Comparison to other frameworks

- **Native Android and IOS**
  + Performance
  + Native look & feel
  + Security
  - Different codebase & technologies between platforms

- **Hybrid App** (Cordova, Phonegap, etc.)
  - Performance
  - Weak native look and feel
  - Security
  + One codebase & technology

- **React Native**
  + Performance
  + Native look and feel
  + One codebase & technology

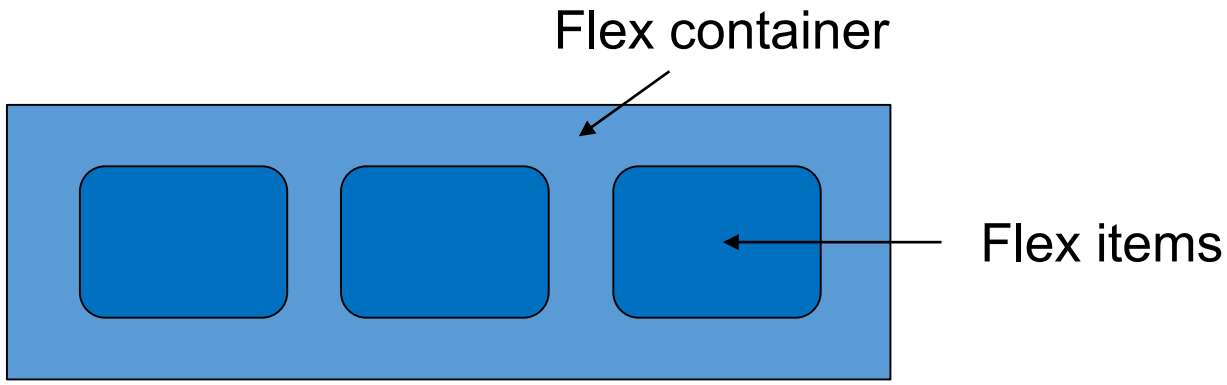
- React Native provides possibility to write native code when high performance is needed or some functionalities are missing from React Native

React Native

- Web elements are not used with React Native
- React Native has a lot of mobile components available which can be used to create apps
- Some of the most common components
    - <View> - container that supports layout with flexbox
    - <Button> - basic button component
    - <Image> - component for displaying images
    - <TextInput> - component for text input

React Native: Flexbox

- Layouts can be defined by using flexbox
- Flexbox works same way in React Native as it works with CSS in HTML
- Parent container becomes Flex container and all its childs becomes Flex items.



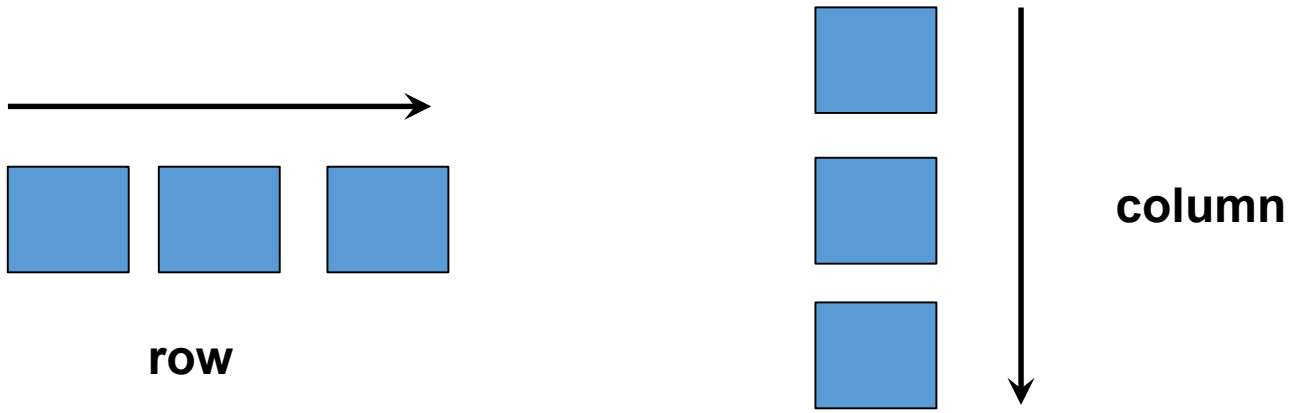
Haaga-Helia

React Native: Flexbox

- Some common attributes used in flexbox with React Native are

`flexDirection`

- Defines the direction how components are organized inside the container (horizontally or vertically). Default is horizontally.
- flexDirection also defines the **primary axis**



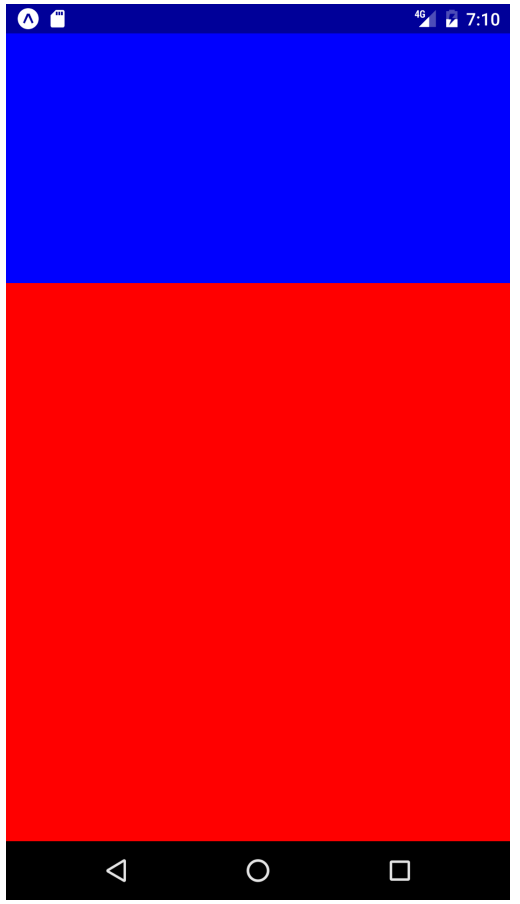
Haaga-Helia

React Native: Flexbox

`flex`

- Defines how the space is divided between multiple flex containers

```
<View style={{height: 100, flex: 1}}>
 <View style={{flex: 1}}>
   Some components goes here - 1/3 space
 </View>
 <View style={{flex: 2}}>
   Some components goes here - 2/3 space
 </View>
</View>
```



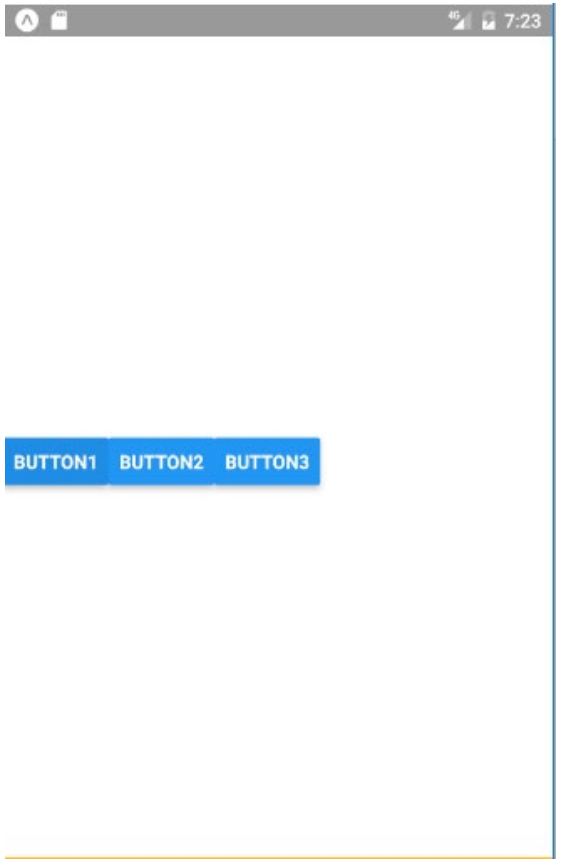
Haaga-Helia

React Native: Flexbox

`alignItems`

- Defines the alignment of childrens in the secondary axis. If flexDirection is row the then the secondary axis is column and vice versa.
- Options: `center, flex-start, flex-end, stretch`

```
<View style={{flex: 1, flexDirection: 'row', alignItems: 'center'}}>
  <Button title="Button1" onPress={this.buttonPressed}/>
  <Button title="Button2" onPress={this.buttonPressed}/>
  <Button title="Button3" onPress={this.buttonPressed}/>
</View>
```



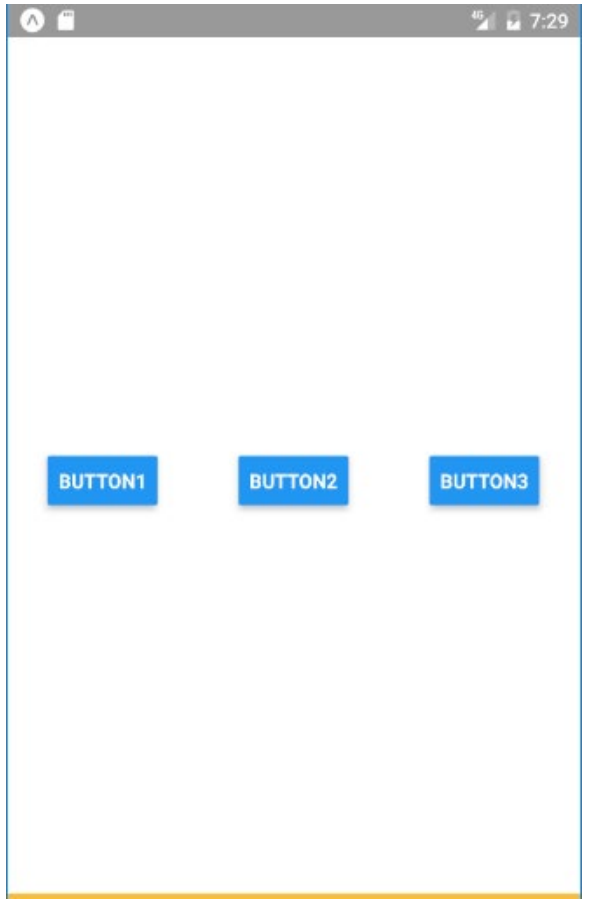
Haaga-Helia

React Native: Flexbox

`justifyContent`

- Defines the distribution of childrens in the primary axis.
- Options: `center`, `flex-start`, `flex-end`, `space-around`, `space-between`

```
<View style={{flex: 1, flexDirection: 'row', alignItems: 'center',
 justifyContent: 'space-around'}}>
  <Button title="Button1" onPress={this.buttonPressed}/>
  <Button title="Button2" onPress={this.buttonPressed}/>
  <Button title="Button3" onPress={this.buttonPressed}/>
</View>
```



Haaga-Helia

React Native: Hello World (With Expo CLI)

- Easiest way to create a new React Native app is using Expo CLI
- Install Expo CLI

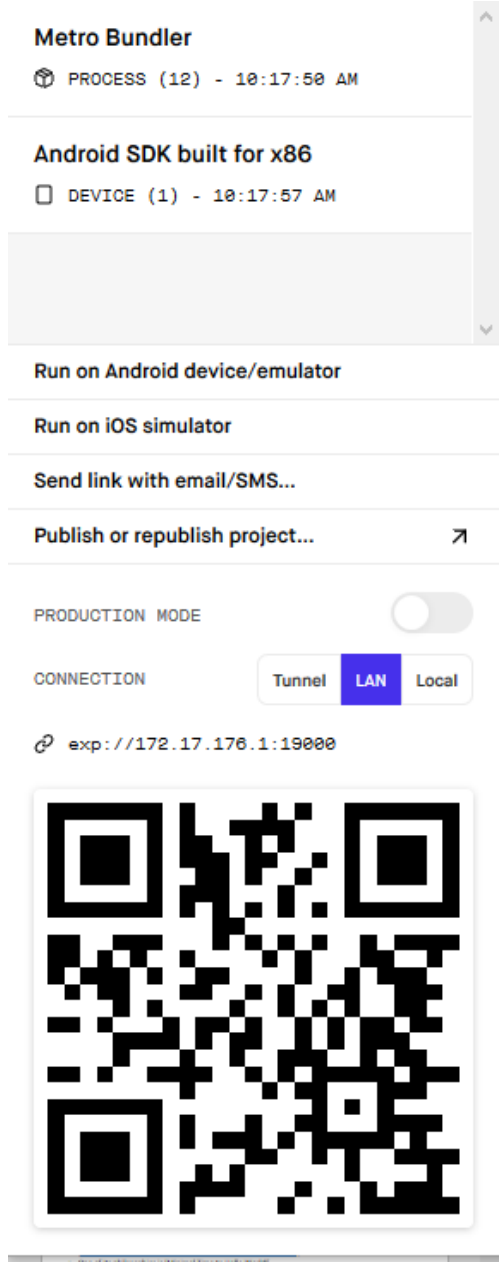
```
npm install -g expo-cli
```

- Create your first app and run it

```
expo init yourAppName
cd yourAppName
expo start OR yarn start
```

React Native: Hello World

- Expo opens Metro Bundler in your browser after you run your app.
- Read the QR code from the Metro Bundler or terminal (Note! Your device and computer should be in the same network. Create a hotspot from your phone and connect your laptop to this network)
- Expo app is installed to your android or ios device when you run your app in the device.
- You can also run your app in emulator or USB connected device.



Haaga-Helia

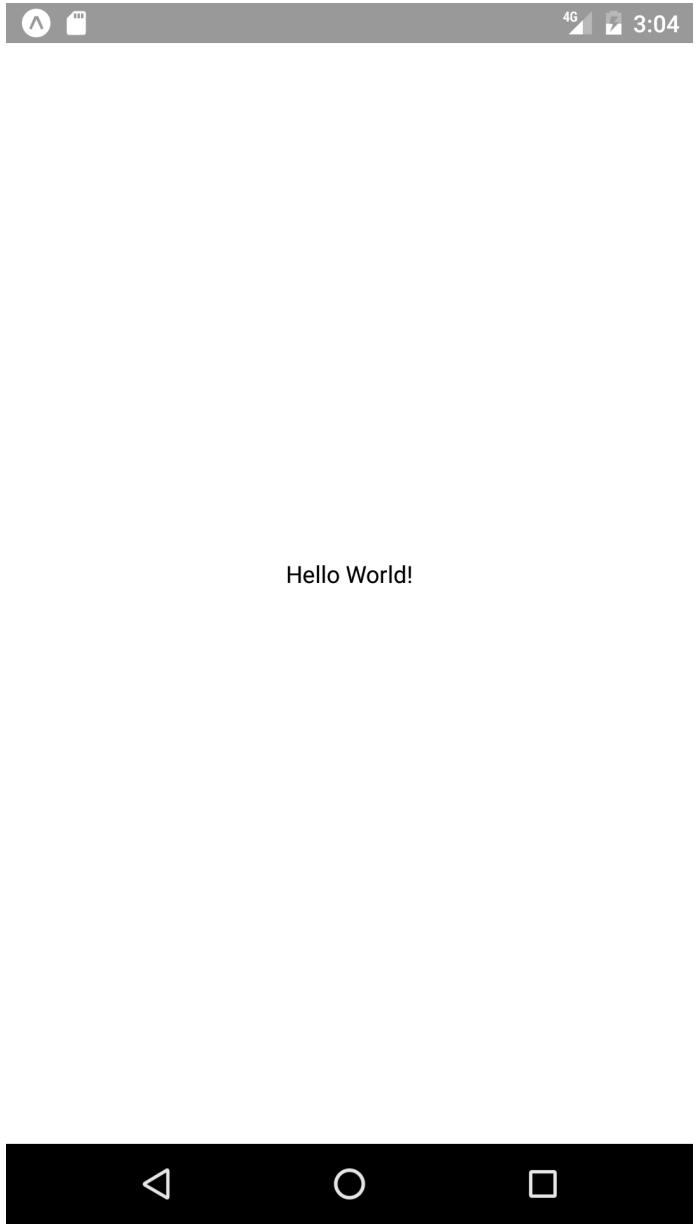
React Native: Hello World

- Modify return statement in **App.js** file

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Hello World!</Text>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
 flex: 1,
 backgroundColor: '#fff',
 alignItems: 'center',
 justifyContent: 'center',
 },
});
```



Haaga-Helia

React Native: Basic components

- **Text** – component for displaying text
- Add Text component to import

```
import { Text, View } from 'react-native';
```

- Add Text component to the return statement

```
return (
  <View style={styles.container}>
    <Text>This is text</Text>
  </View>
);
```

React Native: Basic components

- **Button** – basic button component

- Add Button component to import

```
import { StyleSheet, Text, View, Button } from 'react-native';
```

- Add Button component to the return statement

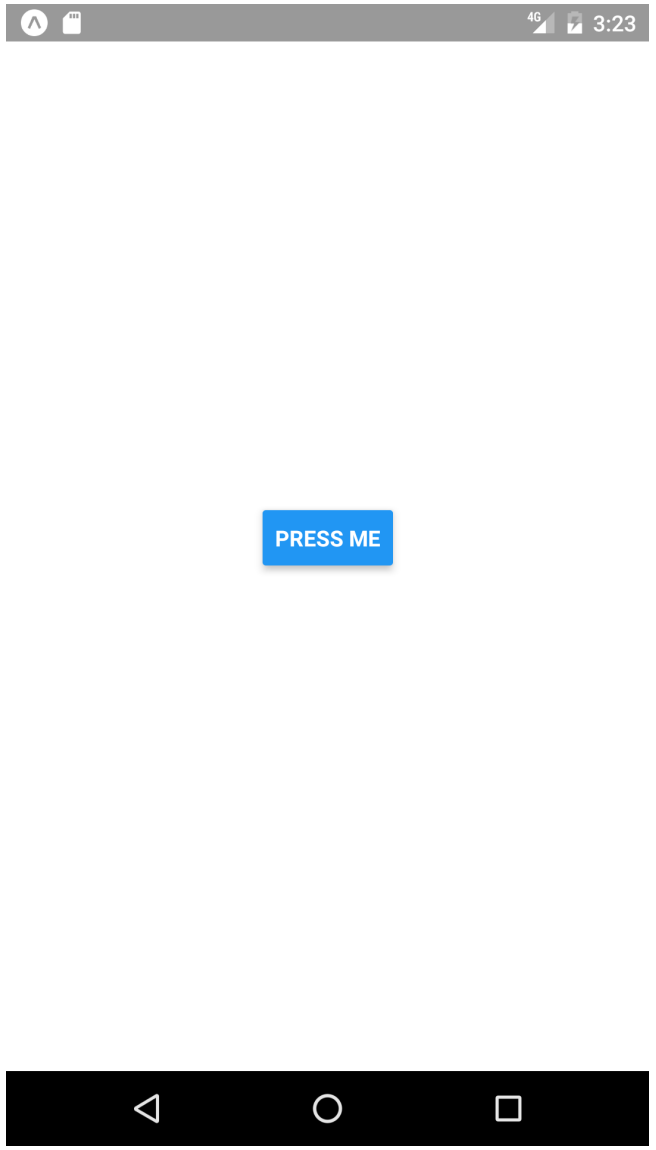
```
return (
  <View style={styles.container}>
    <Button onPress={buttonPressed} title="Press me" />
  </View>
);
```

- Pressing the button will now call `buttonPressed` function which we have to create next

React Native: Basic components

- Pressing button will show an alert. Note! You have to import also Alert.

```
const buttonPressed = () => {
  Alert.alert('Button pressed');
}
```



React Native: Basic components

- **TextInput** – component for text input
- Add TextInput component to import

```
import { View, Button, Alert, TextInput } from 'react-native';
```

- Add new state where typed input is saved

```
const [text, setText] = useState('');
```

React Native: Basic components

- Add TextInput component to render method

```
<TextInput
  style={{width: 200, borderColor: 'gray', borderWidth: 1}}
  onChangeText={(text) => setText(text)}
  value={text}
/>
```

- Typing text to TextInput component will set written text to the text state

```
onChangeText={(text) => setText(text)}
```

- Show text state in alert

```
buttonPressed = () => {
 Alert.alert('You typed:'+ text);
}
```

React Native: Basic components

- **Image** – component for displaying images

- Add Image component to import

```
import { StyleSheet, Image } from 'react-native';
```

- Add Image component to the container

```
<Image style={{width:250, height: 100}}
source={require('./img/haaga-helia.jpg')} />
```

- Image can be local image or image from remote URI
- In the case of remote URI image the source is defined in following way

```
source={{uri: 'IMAGE_URI'}}
```

Haaga-Helia
University of Applied Sciences Ltd.

Haaga-Helia

React Native: Stylings

- Core components has a property called `style` that can be used for styling. Style is a javascript object

Example:

```
<Text style={{ fontSize:18, color: 'red' }}>Red text</Text>
```

- The better way is to create Stylesheets

Example:

- Import StyleSheet component

```
import {StyleSheet, Text} from 'react-native';
```

React Native: Stylings

- Create StyleSheet

```
const styles = StyleSheet.create({
 alerttext: {
   fontSize: 18,
   color: 'red'
 },
});
```

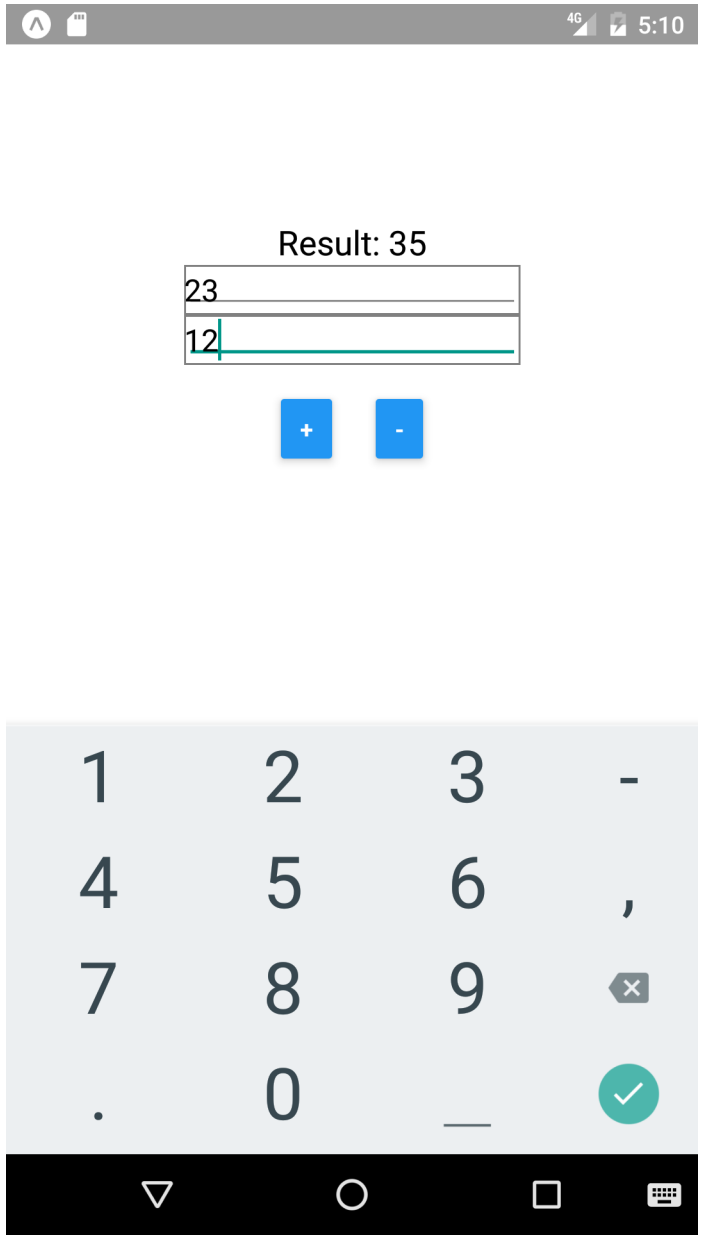
- Use Stylesheet

```
<Text style={styles.alerttext}>Red text</Text>
```

Exercise 1

Calculator

- Create simple calculator which contains:
  - Two TextInputs with numeric keyboard
  - Buttons for calculating sum and subtraction. When buttons are pressed the calculated answer is shown in the Text element



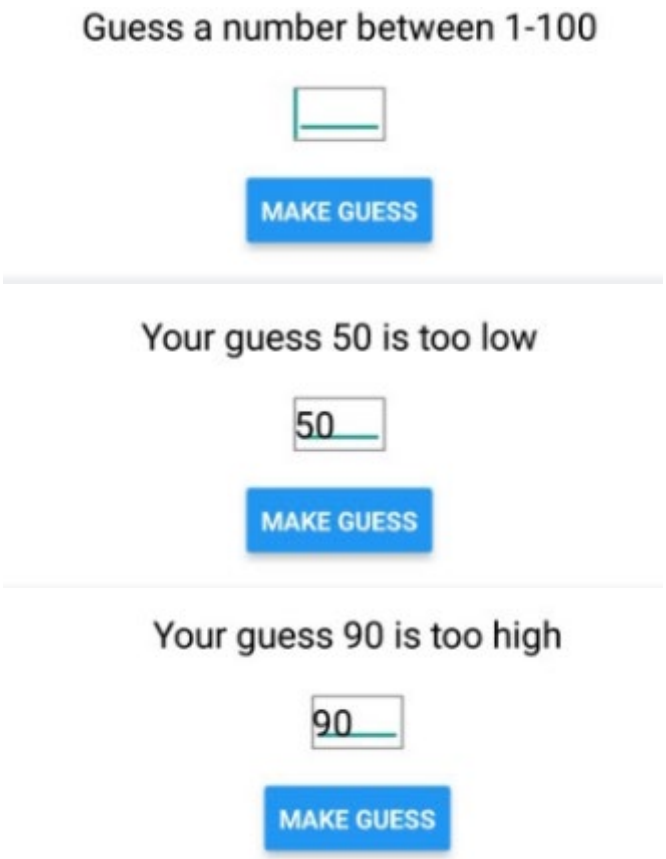
Haaga-Helia

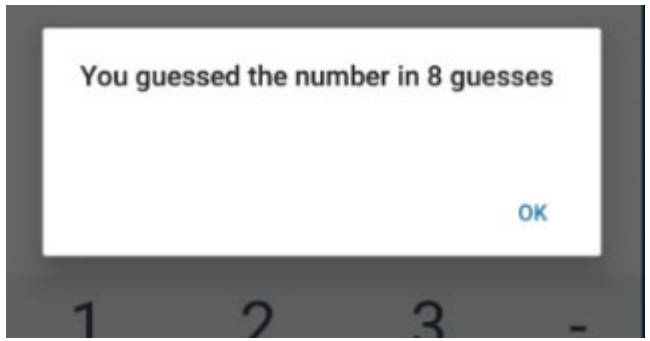
Exercise 2

Number Guessing Game

- Create Number Guessing game where user have to guess the secret number between 1-100 (see the screenshots)
  - Random number between 1-100

```
Math.floor(Math.random() * 100) + 1
```





Haaga-Helia

React Native: Basic components

- **FlatList** – list component with some nice features

- Add Flatlist component to import

```
import { StyleSheet, Text, View, Button,
TextInput, FlatList } from 'react-native';
```

- Add new states that are used for textinput and listitems

```
const [text, setText] = useState('');
const [data, setData] = useState([]);
```

- Add FlatList to the return statement

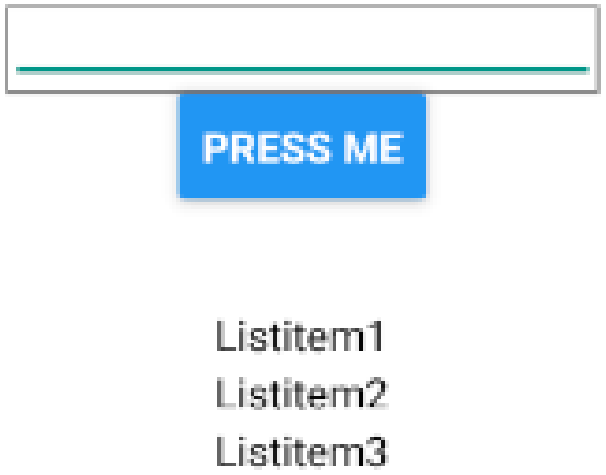
```
<FlatList
  data={data}
  renderItem={({item}) =>
        <Text>{item.key}</Text>}
/>
```

- renderItem defines how listitems are rendered

React Native: Basic components

- Add new text from the TextItem component to the data state when button is pressed

```
const buttonPressed = () => {
  setData([...data, {key: text}]);
  setText('');
}
```



Note! `...` = Spread syntax

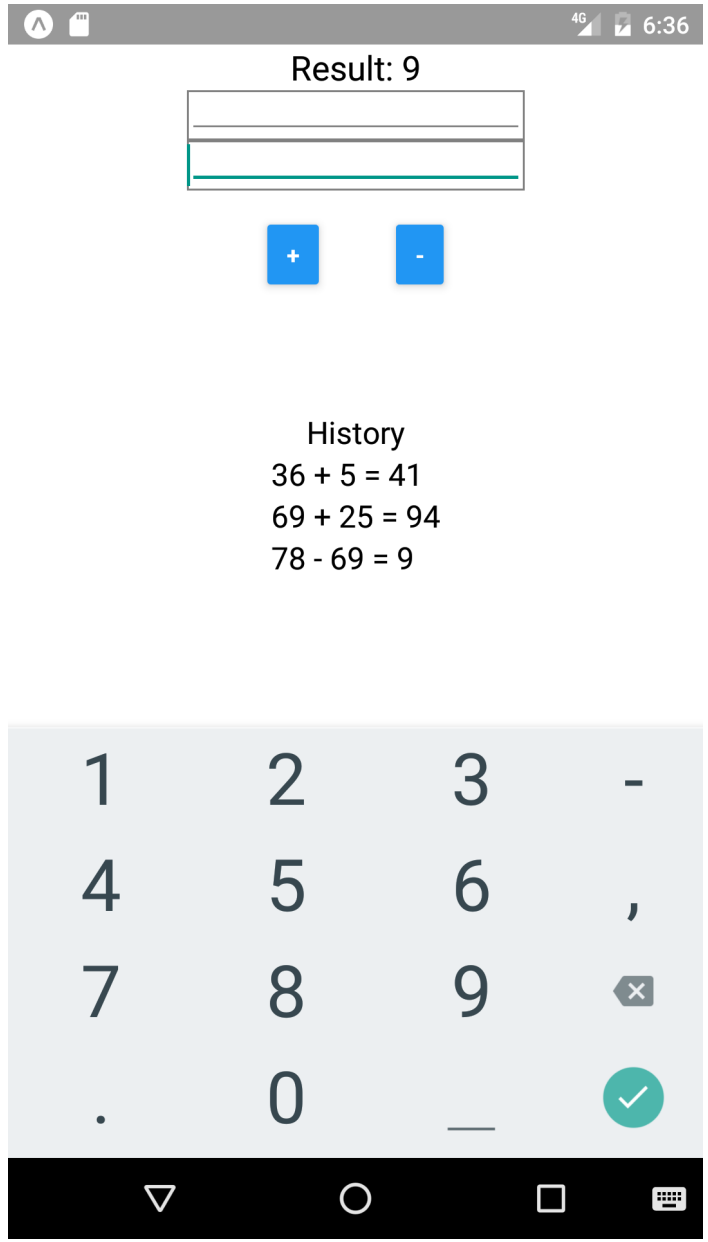
Example:

```
let myCars = ['Volvo', 'Toyota'];
let allCars = ['Nissan', ...myCars, 'Audi'];
// ["Nissan", "Volvo", "Toyota", "Audi"]
```

Exercise 3

Expand your calculator

- Add FlatList component that shows calculation history. Note! Data is not saved anywhere. When the app is restarted the history is cleared.



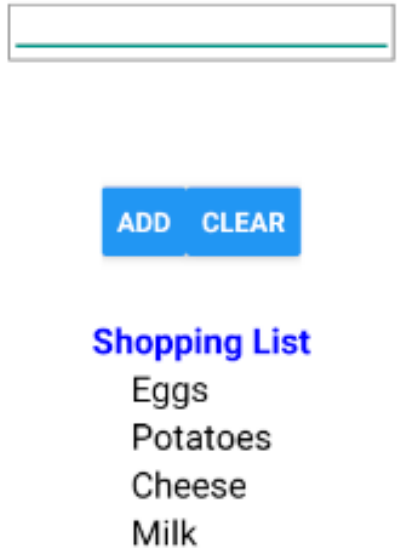
Haaga-Helia

Exercise 4

Shopping List

- Create a simple shopping list app.

  Note! Data is not saved anywhere. It is only shown in the list component. Clear button will clear all items from the list.



React Native: Navigation

- React Navigation provides navigation solution for IOS and Android.
- Stack, Tab and Drawer navigations are available in React Navigation
- https://reactnavigation.org/

Example:

- Install React Navigation

`npm install react-navigation` OR **`yarn add react-navigation`**

Tab navigation

- Import createAppContainer and createBottomTabNavigator (in App.js)

```
import {createAppContainer, createBottomTabNavigator} from 'react-navigation';
```

- Container (createAppContainer) links your navigator to the app environment.

React Native: Navigation

- Create class HomeScreen.js for the first page

```
// This is the first page
import React from 'react';
import { View, Text } from 'react-native';

const HomeScreen = () => {
 return (
  <View>
    <Text>This is HomeScreen</Text>
  </View>
 );
};

export default HomeScreen;
```

Note! You can hide Expo statusbar by adding following row inside your View: `<StatusBar hidden={true} />`

React Native: Navigation

- Create class SettingScreen.js for the second page

```
// Add also imports & export
// This is the second page
const SettingScreen = () => {
  return (
   <View>
    <Text>This is SettingScreen</Text>
   </View>
  );
};
```

- Create Tab Navigator in App.js

```
const AppNavigator = createBottomTabNavigator({
      Home: {screen: HomeScreen},
      Settings: {screen: SettingScreen}
});
```

React Native: Navigation

- Create Container

```
const AppContainer = createAppContainer(AppNavigator);
```

- App component renders the container

```
export default function App() {
  return (
    <AppContainer />
  );
}
```

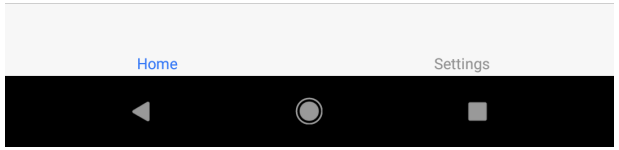
- Import HomeScreen and SettingScreen to the App component

```
import HomeScreen from './HomeScreen';
import SettingScreen from './SettingScreen';
```

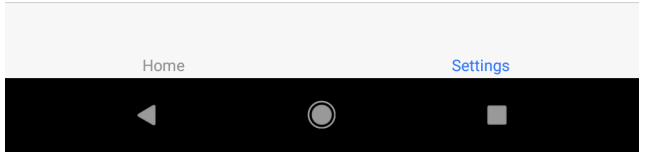
React Native: Navigation

Home screen

Settings screen



Haaga-Helia



React Native: Navigation

- Let’s add icons to our tab navigation
- Import icons from Expo

```
import { Ionicons } from '@expo/vector-icons';
```

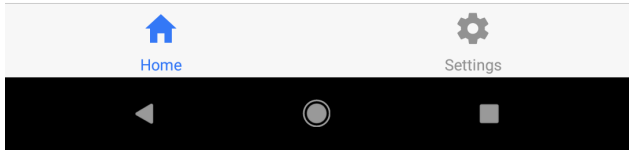
- Navigation can be customized using `navigationOptions`

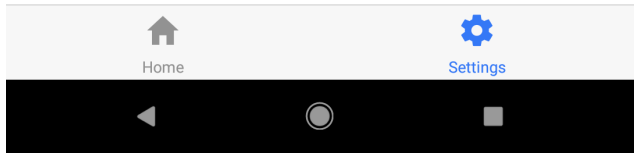
```
const MyApp = createBottomTabNavigator(
{
  Home: {screen: HomeScreen},
  Settings: {screen: SettingScreen}
},
{
  navigationOptions: ({ navigation }) => ({
   tabBarIcon: ({ focused, tintColor }) => {
    const { routeName } = navigation.state;
    if (routeName === 'Home') {
      return <Ionicons name='md-home' size={25} color={tintColor} />;
    } else if (routeName === 'Settings') {
      return <Ionicons name='md-settings' size={25} color={tintColor} />;
    }
   }
  })
});
```

React Native: Navigation

Home screen

Settings screen





Haaga-Helia

React Native: Navigation

Stack navigation

- Import createAppContainer and createStackNavigator (in App.js)

```
import {createAppContainer, createStackNavigator} from 'react-navigation';
```

- Use createStackNavigator instead of Tab Navigation (App.js)

```
const MyApp = createStackNavigator({
  Home: {screen: HomeScreen},
  Settings: {screen: SettingScreen}
});
```

Haaga-Helia

React Native: Navigation

- Add a button for navigation and title to your HomeScreen component (Add title also to SettingScreen)

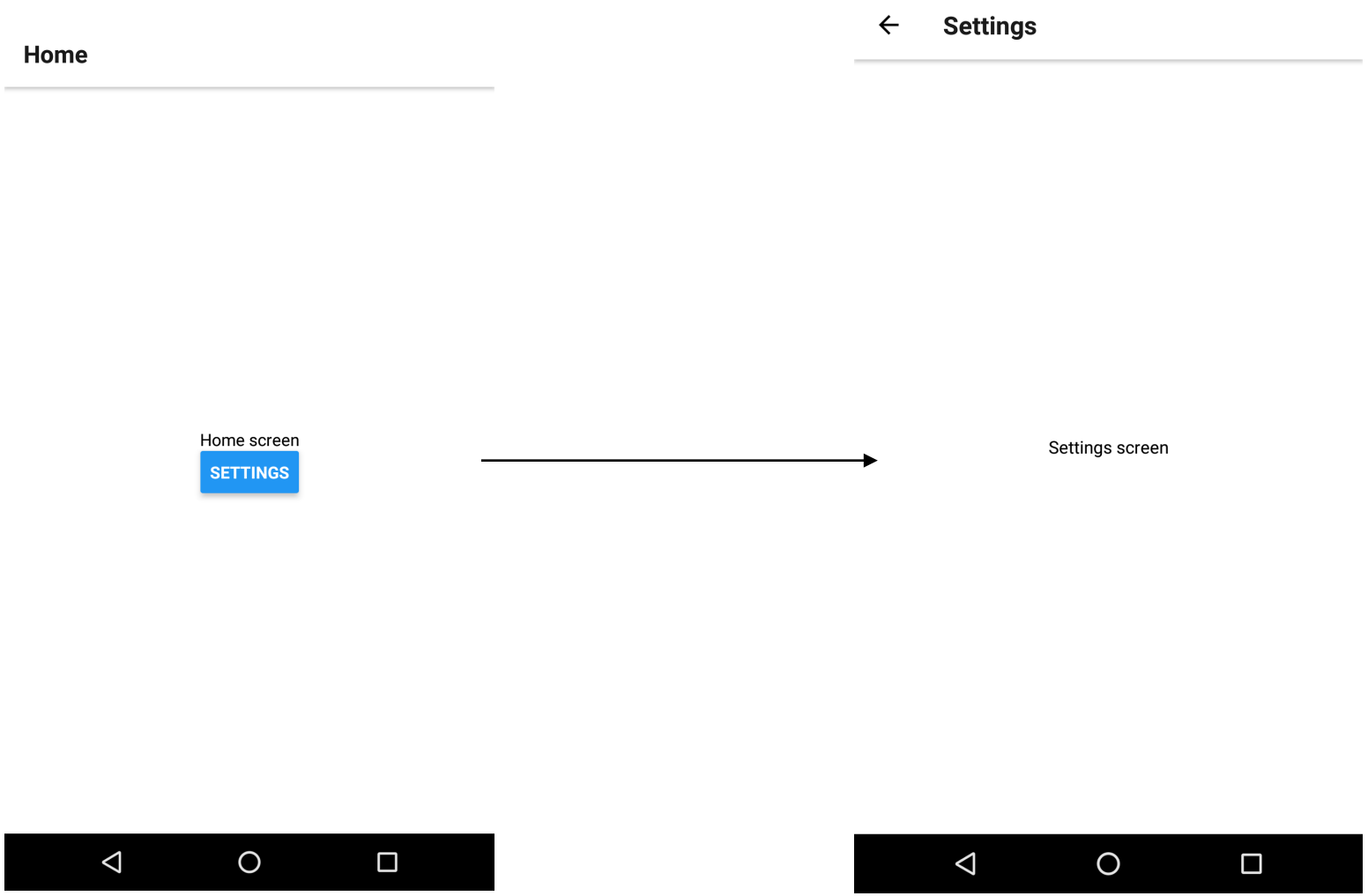
```
// Add also imports
// This is the first page
const HomeScreen = (props) => {
  static navigationOptions = {title: 'Home',};

  const { navigate } = props.navigation;
  return (
   <View>
     <Text>Hello screen</Text>
     // Button navigates to Settings page
     <Button onPress={() => navigate('Settings')} title="Settings"/>
   </View>
  );
}
HomeScreen.navigationOptions = ({navigate}) => ({title: 'Home'});

export default HomeScreen;
```

Note! You can hide Expo statusbar by adding following row inside your View: `<StatusBar hidden={true} />`

React Native: Navigation



React Native: Navigation

- Passing parameters between pages

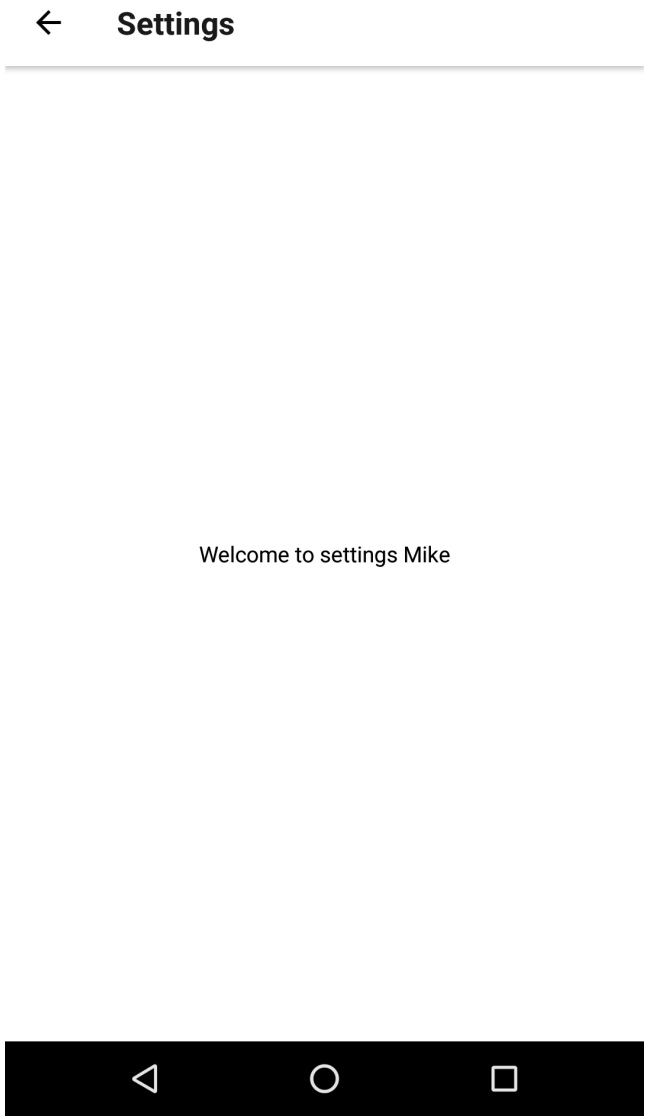
  Parameters can be passed in navigation props.

```
// HomeScree
const { navigate } = props.navigation;
return (
  <View style={styles.container}>
    <Text>Home screen</Text>
    <Button onPress={() => navigate('Settings', {user: 'Mike'})}
        title="Settings"/>
  </View>
);
```

React Native: Navigation

- Settings screen can now get the passed params

```
//SettingsScreen
const { params } = props.navigation.state;
return(
  <View style={styles.container}>
    <Text>Welcome to settings {params.user}</Text>
  </View>
);
```

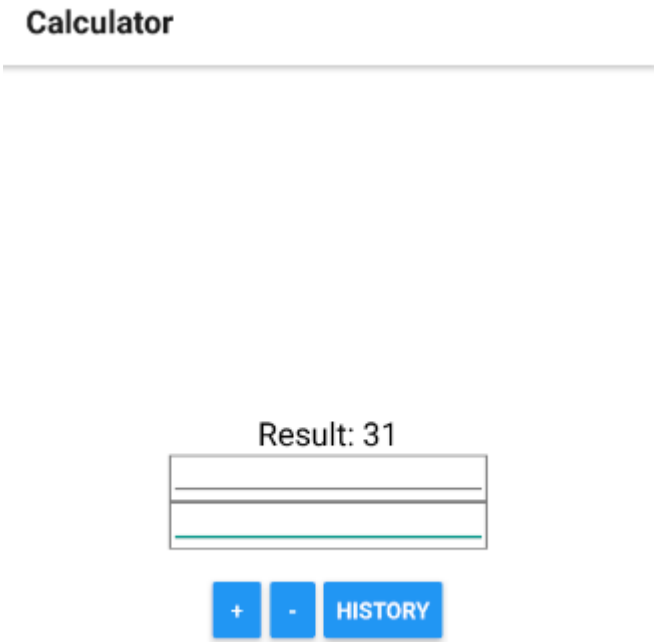


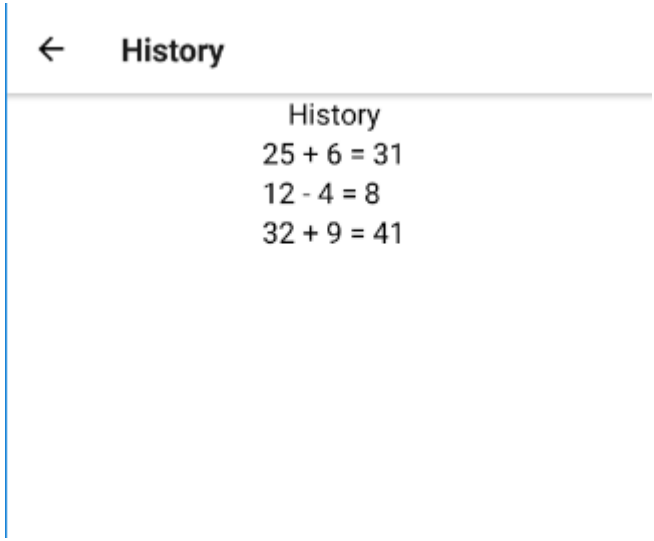
Exercise 5

Calculator (multipage)

- Split your calculator into two pages: Calculator and History
    1. Calculator page that contains calculator and button to navigate to History page
    2. History page shows the calculation history

Note! Calculator page and History page should be in separate component classes.





Haaga-Helia

React Native: Saving data

AsyncStorage

- AsyncStorage is asynchronous and persistent key-value pair storage system

Usage:

- Import AsyncStorage

```
import { AsyncStorage, Alert } from 'react-native';
```

- Saving data – **Note!** Value must be string or it should be ‘stringified’ before saving using JSON.stringify(). For reading you should then use JSON.parse().

```
try {
  await AsyncStorage.setItem('someKey', 'This is the value');
} catch (error) {
  Alert.alert('Error saving data');
}
```

Note! `await` can be used only inside `async` function

React Native: Saving data

- Read data from AsyncStorage

```
readData = async () => {
  try {
    let value = await AsyncStorage.getItem('someKey');
  } catch (error) {
    Alert.alert('Error reading data');
  }
}
```

Note! `await` can be used inside `async` function

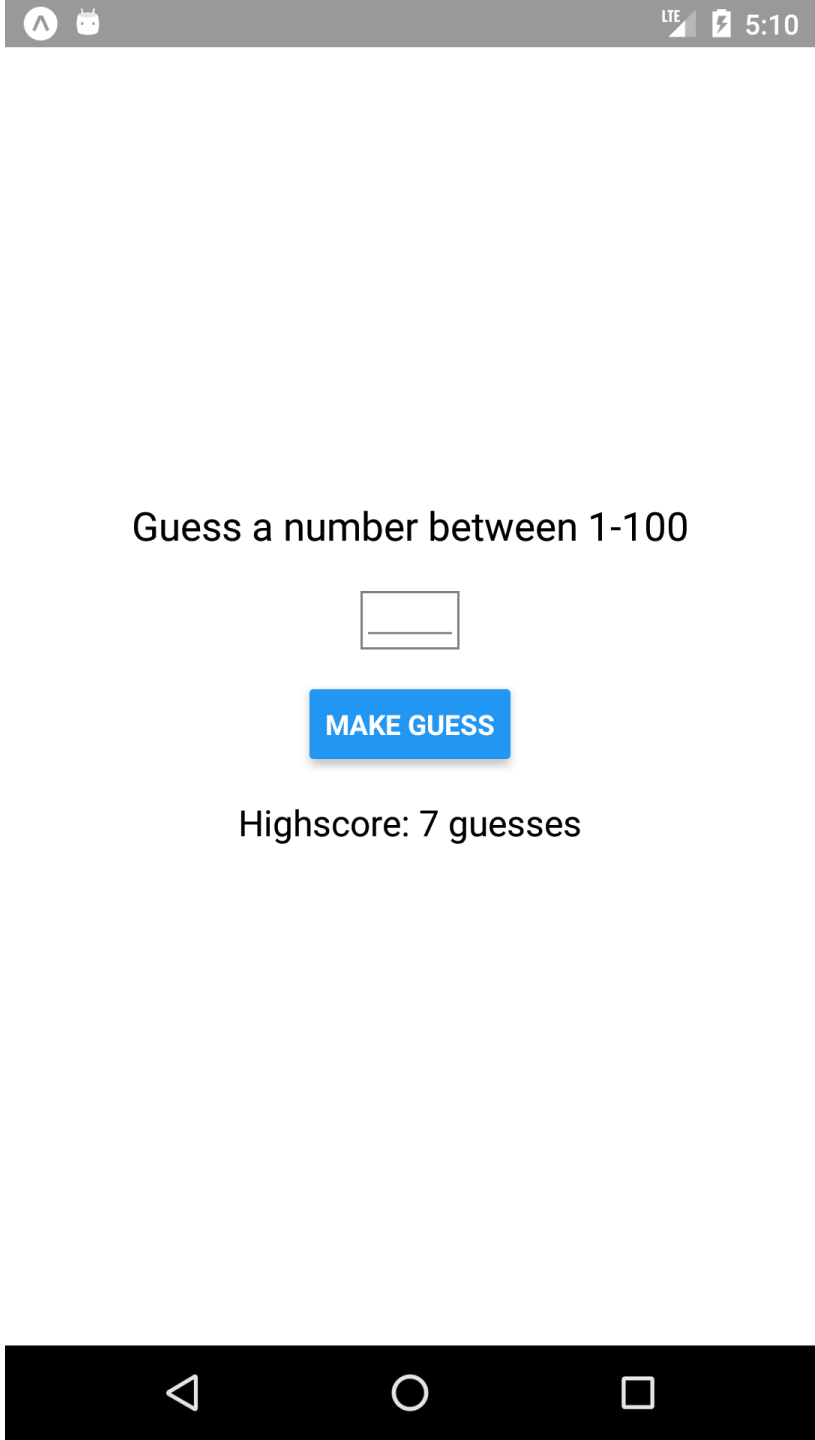
SQLite

- Usage of SQLite database will be covered later in Expo API section

Exercise 6

Number Guessing Game with Highscore

- Extend your game (Exercise 2) by adding highscore functionality
- Highscore is saved to AsyncStorage



Haaga-Helia

React Native: Networking

- Networking with React native is similar to React
- Following example use Github Jobs API to find IT related jobs by description and location (https://jobs.github.com/api)
- We need a state for the jobs that we get from the response and for description and location that we use for filter the query result.

```
const [desc, setDesc] = useState('');
const [location, setLocation] = useState('');
const [jobs, setJobs] = useState([]);
```

- Following URL is used to get jobs by location and description. Response contains an array of jobs and we will show the title and company from each jobs

  http://jobs.github.com/positions.json?description=java&location=london

React Native: Networking

- Render contains two input fields and a button. Button is used to execute fetch request.

```
<TextInput
  style={{fontSize: 18, width: 200}} value={desc}
  placeholder="Description" onChangeText={(desc) => setDesc(desc)}
/>
<TextInput
  style={{fontSize: 18, width: 200}} value={location}
  placeholder="Location" onChangeText={(location) => setLocation(location)}
/>
```

- Render contains also flatlist component to show results

```
<FlatList
  style={{marginLeft : "5%"}}
  keyExtractor={item => item.id}
  renderItem={({item}) => <Text>{item.title}, {item.company}</Text>}
  data={jobs}
/>
```

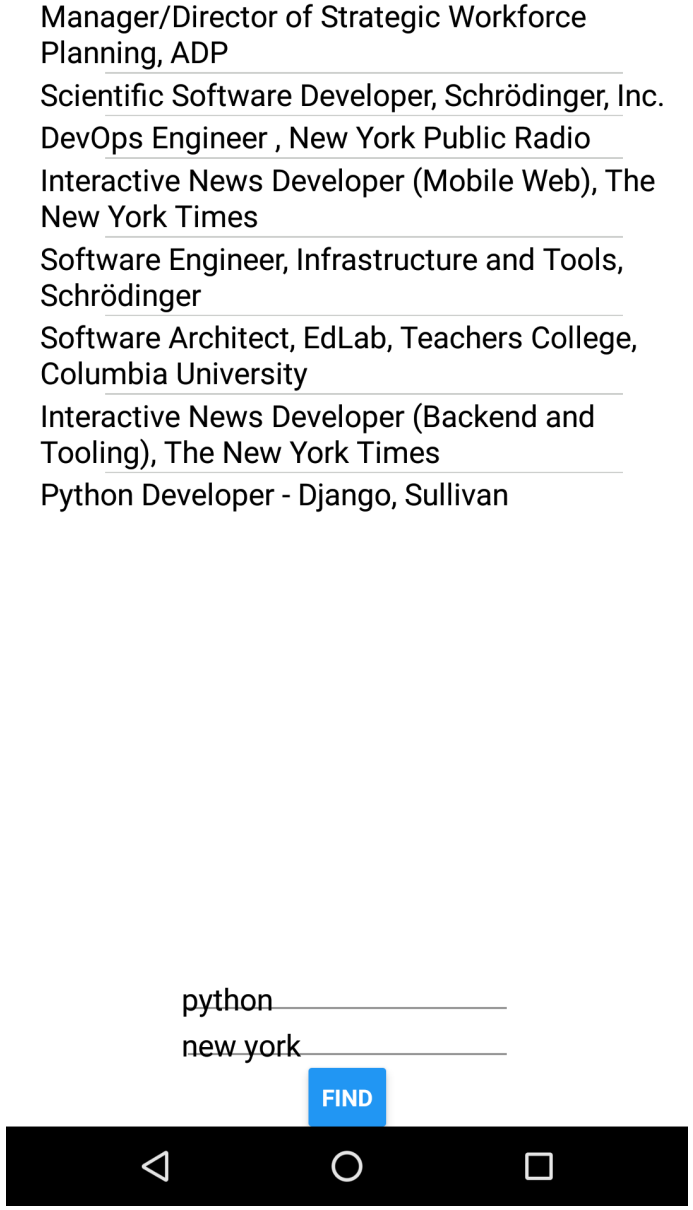
React Native: Networking

- `getJobs` function executes the request and get filter parameter values from the states. Result array is saved to state from the response.

```
getJobs = () => {
  const url = 'https://jobs.github.com/positions.json?description='+
              desc + '&location=' + location;
  fetch(url)
  .then((response) => response.json())
  .then((responseJson) => {
     setJobs(responseJson);
  })
  .catch((error) => {
    Alert.alert('Error', error);
  });
}
```

React Native: Networking

- You can find the App.js code from http://bit.ly/2zSqJIm

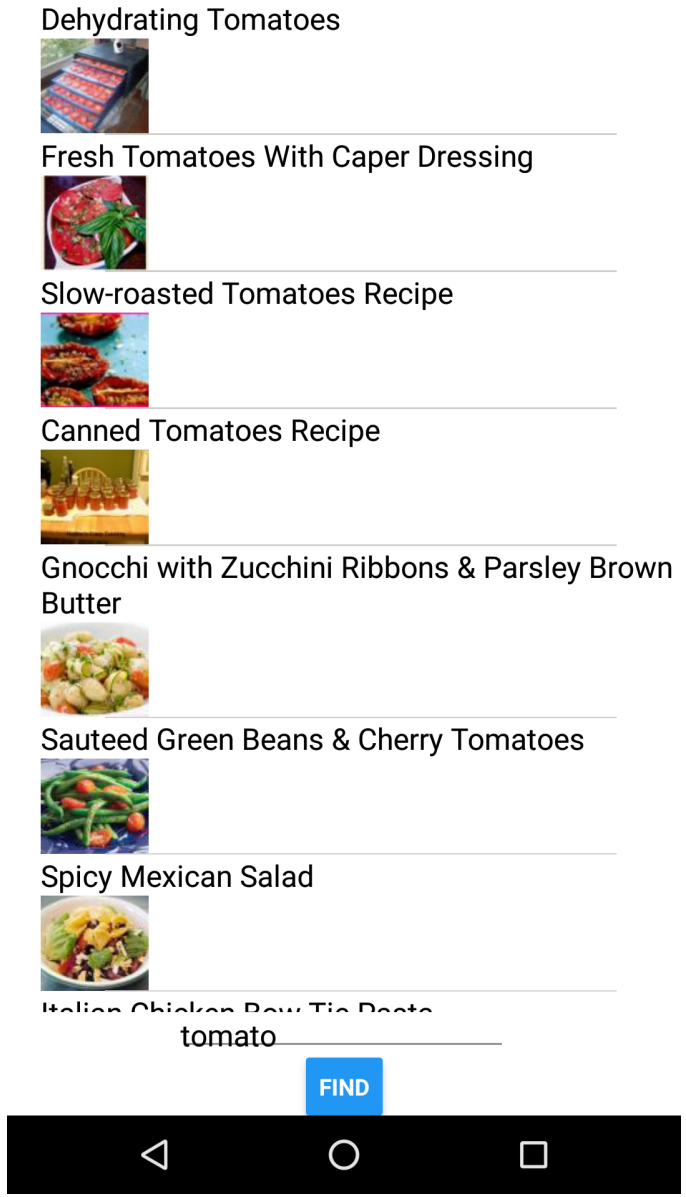


Haaga-Helia

Exercise 7

Recipe Finder

- Create an App that fetch food recipies by ingredients from Recipe Puppy API http://www.recipepuppy.com/about/api/
- App will show title and thumbnail of the recipes in the FlatList
- Example call: http://www.recipepuppy.com/api/?i=tomatoes



Haaga-Helia

Exercise 8

Euro Converter

- Create Euro Converter app
- Currency conversion rates are fetched from Fixer.io API. Register for free and get your API key (https://fixer.io)
  http://data.fixer.io/api/latest?access_key=YOUR_KEY
- Use Picker component to show currencies (Populate Picker values from the fetch response)

280.88 €

250 GBP

CONVERT

Hints!
How to get currency codes to picker?
Object.keys() method returns an array of given object's property names.
See the example from
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/keys

Haaga-Helia

React Native: Expo SDK

- Expo SDK provides access to different device and system functionalities (like contacts, camera, location etc.)
- Expo SDK is included when you use create-react-native-app boilerplate.
- Following slides cover some of these features
- See the documentation https://docs.expo.io/versions/latest/sdk/index.html

React Native: Expo SDK

- **MapView** is the component that shows Google Map (Android) or Apple Map (iOS).
- Following example shows Haaga-Helia Pasila campus in map
- Install react-native-maps component

  `expo install react-native-maps`
- Import MapView and Marker from expo sdk to your Component

```
import MapView, { Marker } from 'react-native-maps';
```

- Add MapView to return statement

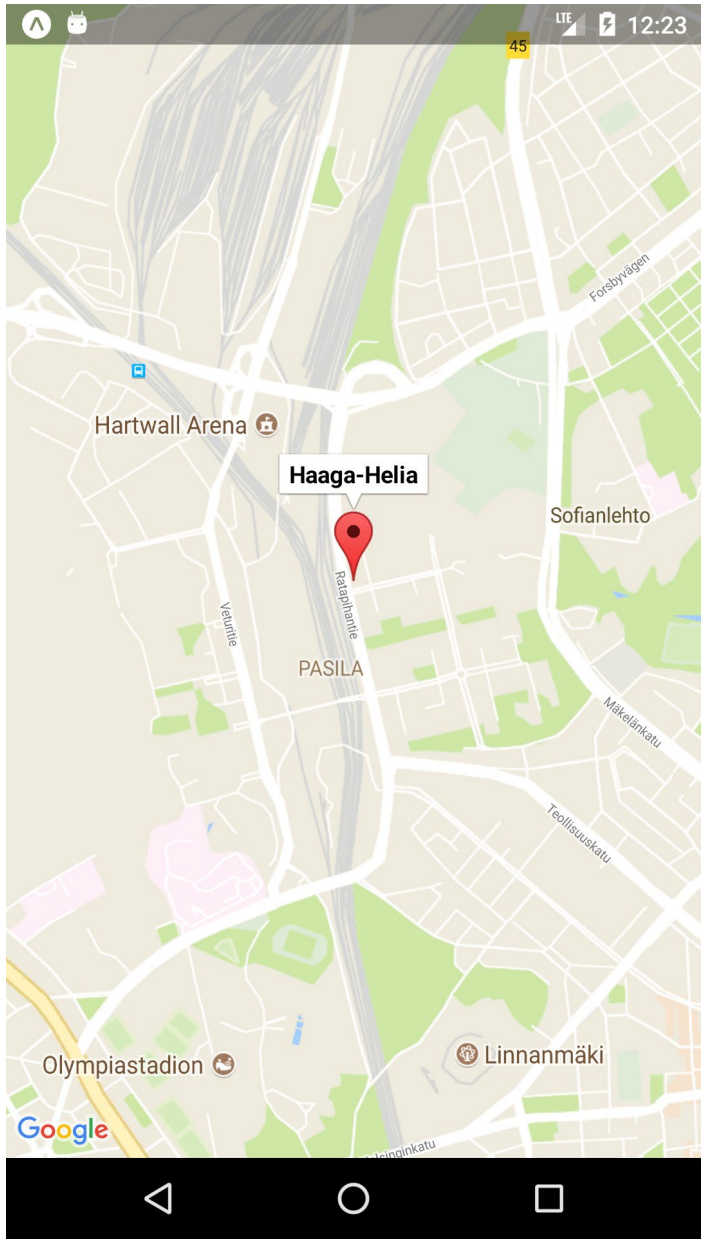
```
<MapView
  style={{ flex: 1 }}
  initialRegion={{
  latitude: 60.200692,
  longitude: 24.934302,
  latitudeDelta: 0.0322,
  longitudeDelta: 0.0221,
 }} />
```

Haaga-Helia

React Native: Expo SDK

- Add marker inside MapView

```
<MapView
 style={{ flex: 1 }}
 initialRegion={{
  latitude: 60.200692,
  longitude: 24.934302,
  latitudeDelta: 0.0322,
  longitudeDelta: 0.0221,
 }}>
<Marker
  coordinate={{
   latitude: 60.201373,
   longitude: 24.934041}}
  title='Haaga-Helia’ />
</MapView>
```



Haaga-Helia

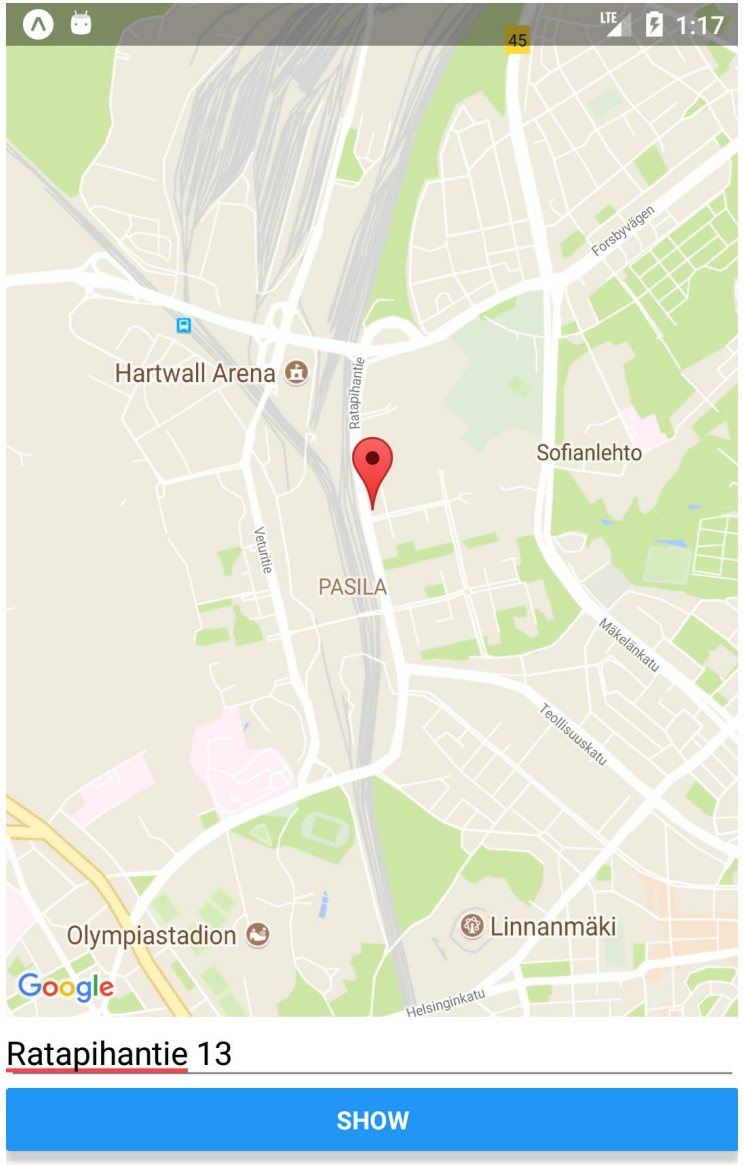
Exercise 9

Find the Address

- Create app where you can type an address and show it on the map
- Hints:

  Use MapQuest Api to transfer address to coordinates. It is free (15000 request/day)
  https://developer.mapquest.com/documentation/geocoding-api/
  OR
  Use Google geocoding Api to transfer address to coordinates (Needs registration and billing info)
  https://developers.google.com/maps/documentation/geocoding/intro

  See react-native-map documentation
  https://github.com/airbnb/react-native-maps

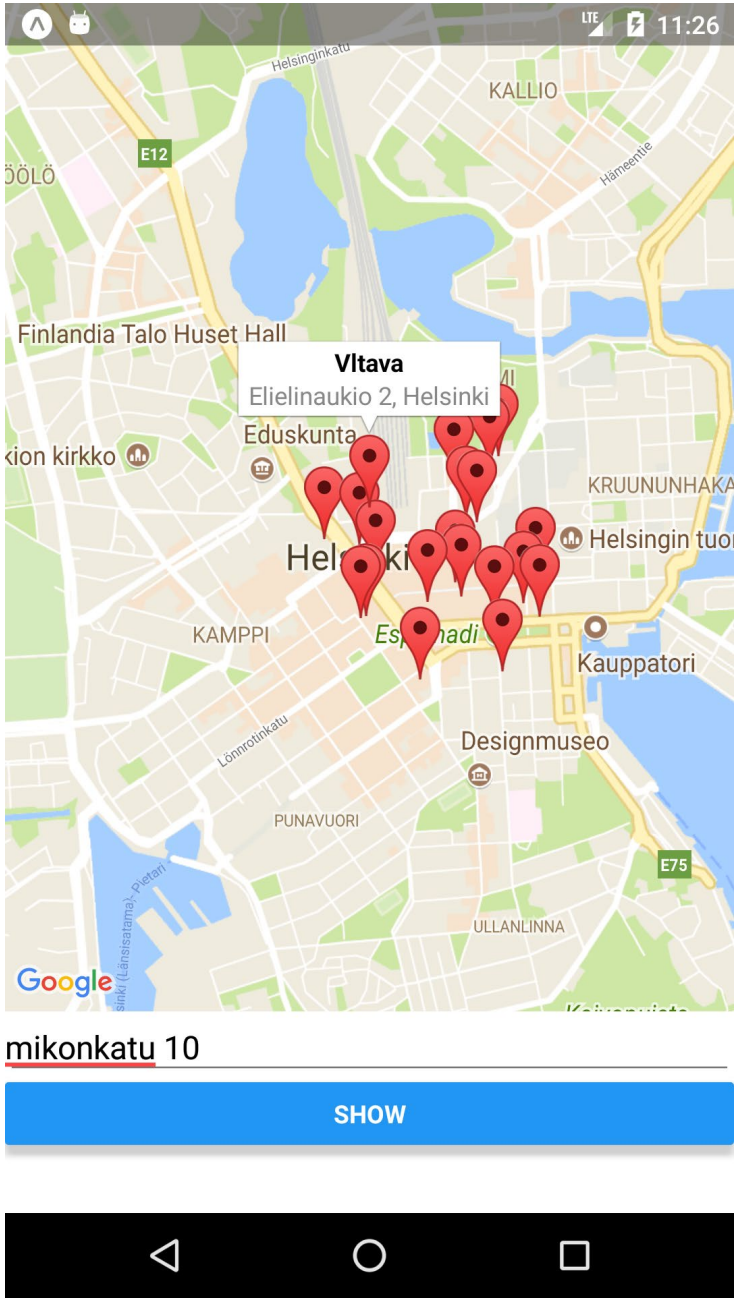


Haaga-Helia

Exercise 10 (Optional)

Restaurant Finder

- Extend previous exercise to show nearby restaurants from the typed address (Show restaurant name in the marker title and address in the marker description)
- Hints:
  - Use Google places API to get restaurants.
    https://developers.google.com/places/web-service/search
  - How to add list of markers
    https://github.com/airbnb/react-native-maps



Haaga-Helia

React Native: Expo SDK

- **Location** allows to read geolocation data from the device
- Install expo-location and permissions

`expo install expo-location AND expo install expo-permissions`

Usage example:

- Import Location and Permission from expo sdk to your Component. Permissions is needed to check device permissions (access to geolocation, contacts etc.).

```
import * as Location from 'expo-location';
import * as Permissions from 'expo-permissions';

export default function App() {
 const [location, setLocation] = useState(null);

 useEffect(() => {
   this._getLocationAsync();
 }, []);

 … continues
```

Haaga-Helia

React Native: Expo SDK

- getLocation() function – Check permission first

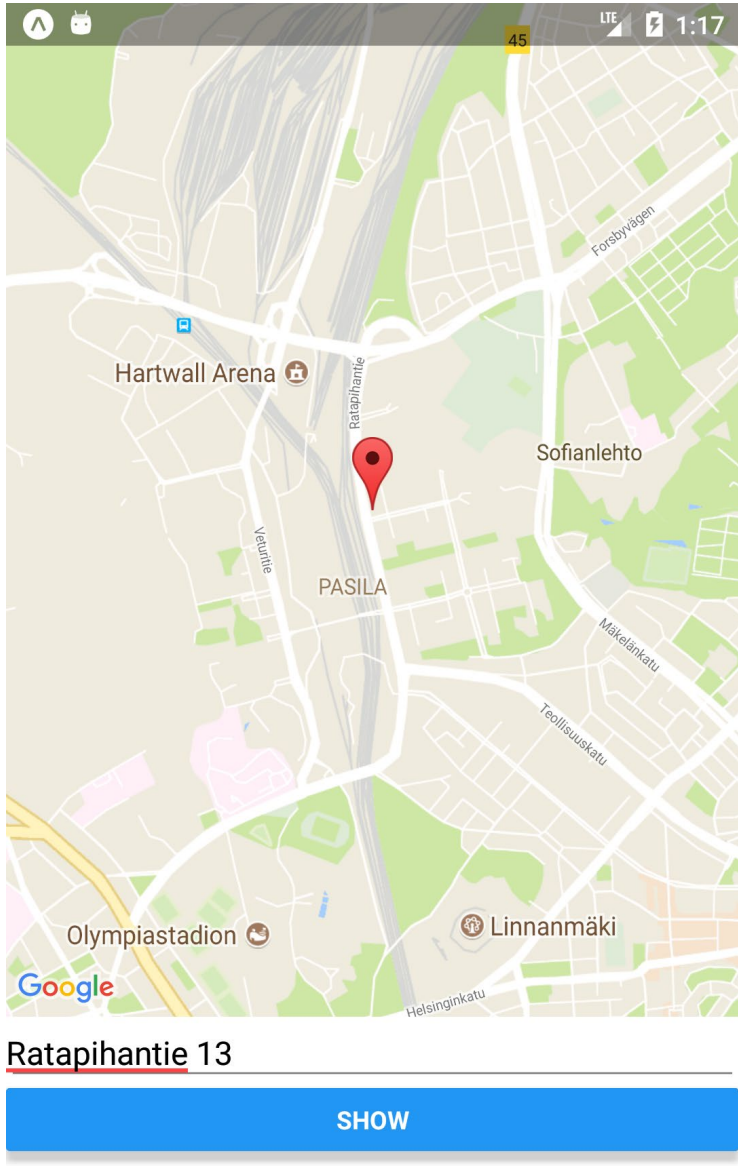
```
getLocation = async () => {
  //Check permission
  let { status } = await Permissions.askAsync(Permissions.LOCATION);
  if (status !== 'granted') {
    Alert.alert('No permission to access location');
  }
  else {
    let location = await Location.getCurrentPositionAsync({});
    setLocation(location);
  }
};
```

- getCurrentPositionAsync returns an object that contains attributes: latitude, longitude, altitude, speed etc.

Exercise 11

Modify exercise 9

- When the app is opened the marker is shown in the device's current position.



React Native: Expo SDK

- **SQLite** is the component that gives an access to SQLite database

Usage example: Courselist app where user types course title and credits. The course is saved to a database when save button is pressed. All courses are shown in the flatlist component.

- Install expo-sqlite

```
npm install expo-sqlite OR yarn add expo-sqlite
```

- Import SQLite from expo-sqlite to your Component

```
import { SQLite } from 'expo-sqlite';
```

React Native: Expo SDK

- States are needed for title and credit input fields and all courses fetched from the database.

```
const [credit, setCredit] = useState('');
const [title, setTitle] = useState('');
const [courses, setCourses] = useState([]);
```

- Open database (Returns database object)

```
const db = SQLite.openDatabase('coursedb.db');
```

React Native: Expo SDK

- Opening database returns database object. The object has method transaction which can be used for database operations. Method has three parameters: The first is one is used to execute sql statement. The second one is executed if errors happen. The third one is executed when transaction is completed successfully.

```
db.transaction(callback, error, success)
```

- Database is created in `componentDidMount` method by using `tx.executeSql` method

```
componentDidMount() {
  db.transaction(tx => {
    tx.executeSql('create table if not exists course (id integer primary key not
       null, credits int, title text);');
  });
}
```

Haaga-Helia

React Native: Expo SDK

- The goal is to create form with two input fields (title and credits) and button which saves item to database when pressed. Button will execute `saveItem` method which handles insert operation to database.

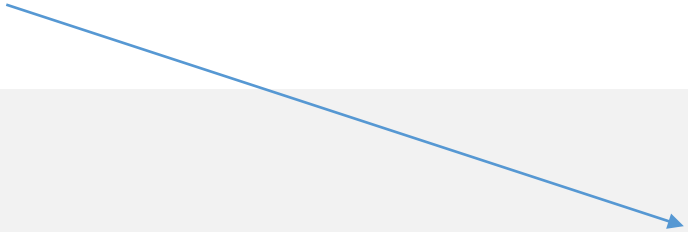
```
// Inside render method
<TextInput placeholder='Title' onChangeText={(title) => setTitle(title)}
value={title}/>
<TextInput placeholder='Credits' keyboardType="numeric" onChangeText={(credit) => setCredit(credit)}
value={credit}/>
<Button onPress={saveItem} title="Save" />
```

- `saveItem` method uses `executeSql` method to insert item in the course table. `updateList` method is executed after succesful insert. Query argument array is the second parameter in executeSql method (substitutes ?-marks in SQL statement)

```
saveItem = () => {
  db.transaction(tx => {
    tx.executeSql('insert into course (credits, title) values (?, ?);',
       [parseInt(credit), title]);
  }, null, updateList)
}
```

React Native: Expo SDK

- `updateList` method fetch all items from the course table and save data to courses state (-> rerender). The third parameter of `executeSql` method is success function which takes resultset object as a second argument. Resultset object contains `rows._array` which is the array of rows returned by query.



```
updateList = () => {
  db.transaction(tx => {
    tx.executeSql('select * from course;', [], (_, { rows }) =>
      setCourses(rows._array)
    );
  });
}
```

React Native: Expo SDK

- Flatlist show title and credits of the courses. Each rows contains also Text component that executes `deleteItem` function when it is pressed.

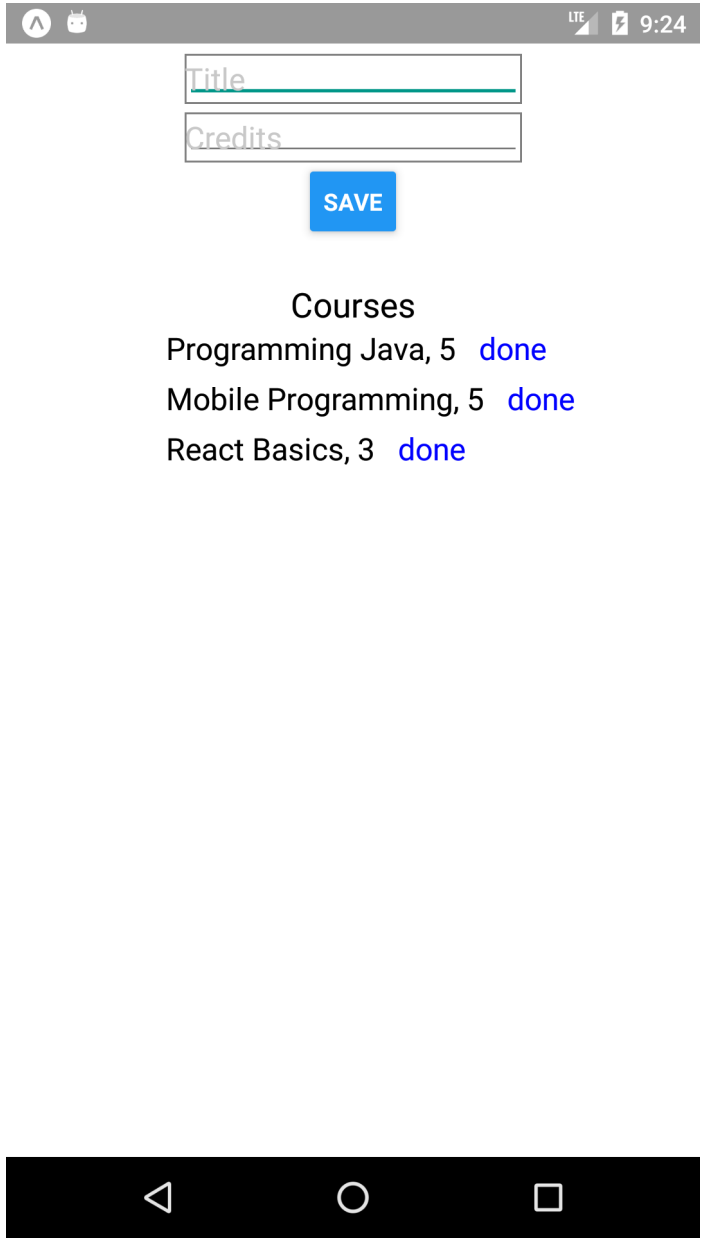
```
<FlatList
  style={{marginLeft : "5%"}}
  keyExtractor={item => item.id.toString()}
  renderItem={({item}) =>
    <View style={styles.listcontainer}><Text>{item.title},{item.credits} </Text>
      <Text style={{color: '#0000ff'}} onPress={() => deleteItem(item.id)}>done</Text>
    </View>}
  data={courses}
/>
```

- deleteItem function deletes item from the course table and updates flatlist after deletion.

```
deleteItem = (id) => {
  db.transaction(
  tx => { tx.executeSql(`delete from course where id = ?;`, [id]);}, null, updateList)
}
```

React Native: Expo SDK

- See the App.js source code from http://bit.ly/2AGCToQ



Haaga-Helia

Exercise 12

Shopping List with DB

- Extend your Shopping List (Exercise 4) by saving items to a SQLite database.
- Shopping item contains following columns: id (unique int), product (text), amount (text)
- Items can be deleted by pressing ‘bought’-link in item’s row in the list

Haaga-Helia

React Native: Firebase

- How to use Firebase realtime database (NoSQL)
- Install firebase to your app

```
npm install firebase
 OR
yarn add firebase
```

- Import firebase to your app

```
import * as firebase from 'firebase';
```

- Sign up to Firebase (https://firebase.google.com/) and create a new project
- Define your database rules for development purposes: Database → Rules tab

```
"rules": {
  ".read": true,
  ".write": true
}
```

Haaga-Helia

React Native: Firebase

- Get your app configuration parameters: Project Overview → Add app
- Initialize Firebase database using your own config parameters

```
// Initialize Firebase with your own config parameters
const firebaseConfig = {
  apiKey: "AIzaSyDj8VKxFj9kYKng0Gfy12345",
  authDomain: "shoppinglist.firebaseapp.com",
  databaseURL: "https://shoppinglist.firebaseio.com",
  projectId: "shoppinglist",
  storageBucket: "shoppinglist.appspot.com",
  messagingSenderId: "54449XX3822XX"
};

firebase.initializeApp(firebaseConfig);
```

React Native: Firebase

- Database reference can be used to read and write data

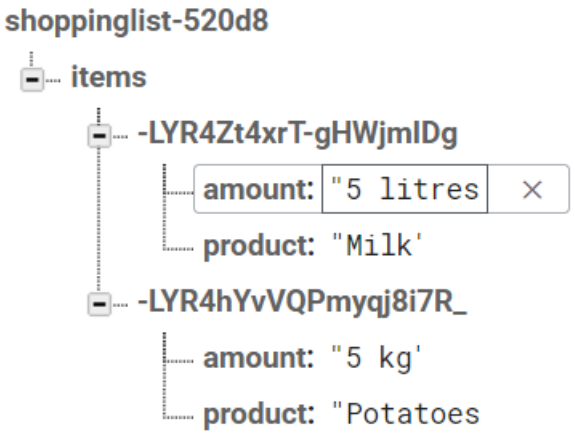
```
firebase.database().ref('items/')
```

- Introduce states for the product and items for listing

```
const [amount, setAmount] = useState('');
const [product, setProduct] = useState('');
const [items, setItems] = useState([]);
```

- Data can be saved using **`push`** method and passing saved value to it. Push method generates unique id to items in Firebase realtime database.

```
saveItem = () => {
  firebase.database().ref('items/').push(
   {'product': product, 'amount': amount}
  );
}
```



Haaga-Helia

React Native: Firebase

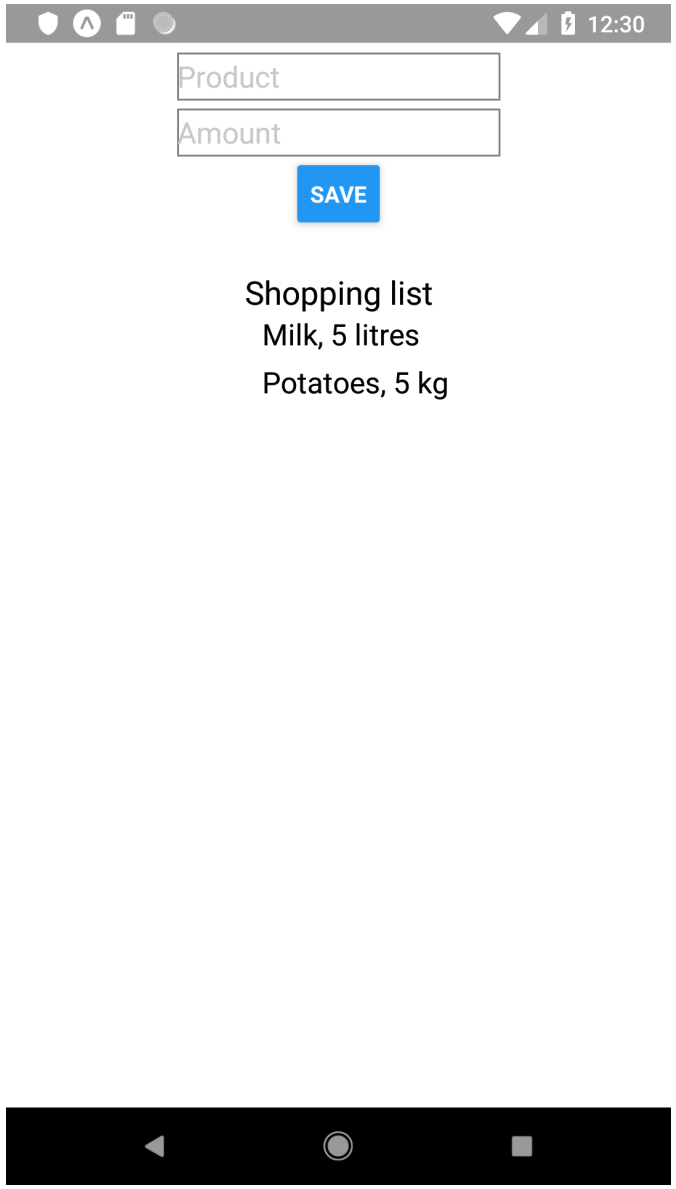
- Data can be read using **on** that listens for data changes in Firebase realtime database. Event type is 'value' that triggers callback each time the data changes.

```
useEffect(() => {
  firebase.database().ref('items/').on('value', snapshot => {
    const data = snapshot.val();
    const prods = Object.values(data);
    setItems(prods);
  });
}, []);
```

Exercise (Optional)

Shopping List with Firebase

- Create shopping list app using Firebase realtime database



React Native: Expo SDK

- **Camera** is the component that renders a preview of the device's camera.
- To use camera install it to your app

```
expo install expo-camera
```

Example: User can take a photo which is then shown in the app.

- Import Camera and Permission from expo sdk to your Component

```
import * as Permissions from 'expo-permissions';
import { Camera } from 'expo-camera';
```

React Native: Expo SDK

- States are needed for the image and permission. We also have to create a **ref** to camera component. By using **ref** we can access to camera component.

```
const [hasCameraPermission, setPermission] = useState(null);
const [photoName, setPhotoName] = useState('');
const [photoBase64, setPhotoBase64] = useState('');

let camera = React.createRef();
```

React Native: Expo SDK

- We check the permission to use camera using useEffect hooks.

```
useEffect(() => {
 askCameraPermission();
}, []);

askCameraPermission = async () => {
  let { status } = await Permissions.askAsync(Permissions.CAMERA);
  setPermission('granted' );
}
```

React Native: Expo SDK

- In the return statement we use conditional rendering. If the app has no permission to use camera we show a text otherwise camera, photo previews and button are rendered.

```
// Note! conditional rendering
if (hasCameraPermission == null) {
  return <View />;
} else if (hasCameraPermission == false) {
  return <Text>No access to camera</Text>;
} else {
  return (
    <View style={{ flex: 1 }}>
      <Camera style={{ flex: 4 }}
      ref={camera}/>
    <View>
      <Button title="Take Photo" onPress={snap} />
    </View>
    <View style={{flex: 4}}>
      <Image style={{flex: 1}}
      source={{uri: photoName}} />
      <Image style={{flex: 1}}
      source={{uri: `${photoBase64}`}} />
    </View>
   </View>
  );
}
```

React Native: Expo SDK

- Button component in the render() method invokes function that takes a photo.

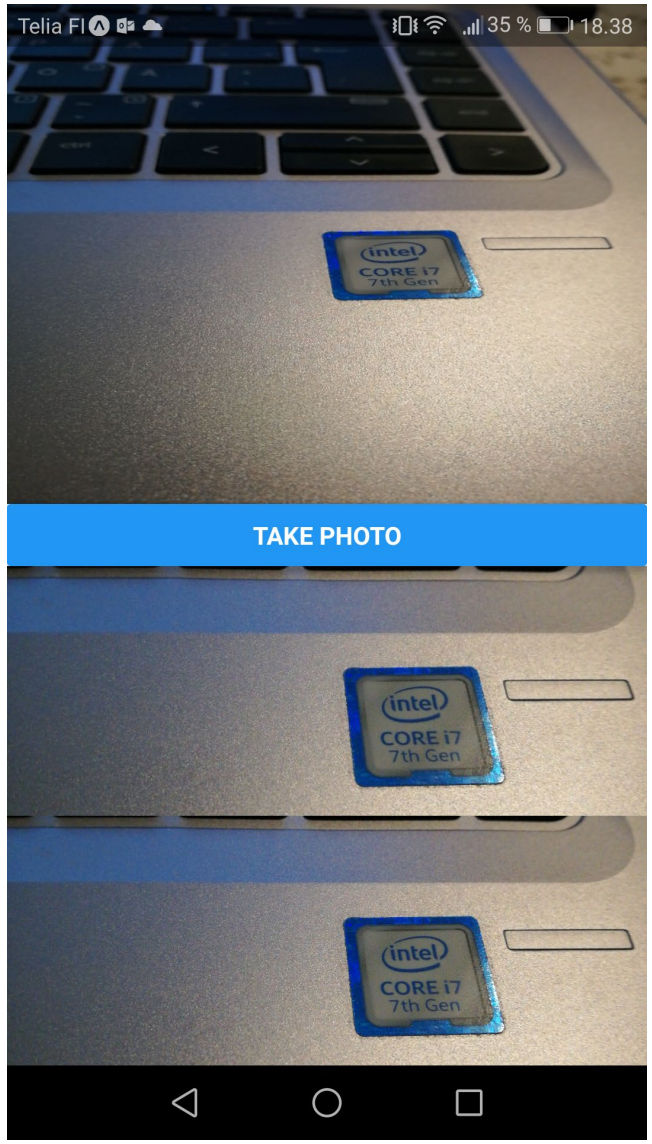
```
<View>
  <Button title="Take Photo" onPress={snap} />
</View>
```

- Below is the source code of the snap function. Camera's **takePictureAsync** method returns an object { uri, base64, width, height, exif }. Base64 is string that contains JPEG data of the image (base64 encoded).

```
snap = async () => {
  if (camera) {
    let photo = await camera.current.takePictureAsync({base64: true});
    setPhotoName(photo.uri);
    setPhotoBase64(photo.base64);
  }
};
```

Haaga-Helia

React Native: Expo SDK



React Native: Component libraries

- There is many different component libraries available for React Native. One example is React Native Elements UI Toolkit
- Link: https://react-native-training.github.io/react-native-elements/

How to get started:

- Install UI library to your project

```
npm install react-native-elements
```

Some examples of the components available

- Input component

```
import { Input } from 'react-native-elements';
```

Use components (note! Input inherits all props from native TextInput component)

```
<Input placeholder='Type your firstname' label='FIRSTNAME'
  onChangeText={(firstname) => this.setState({firstname})}
  value={this.state. firstname}/>
```

Haaga-Helia

React Native: Component libraries

- Button

Import Button

```
import { Input, Button } from 'react-native-elements’;
```

Use button component

```
<Input placeholder='Type your firstname’ label=’FIRSTNAME’
onChangeText={(description) => this.setState({description})}
value={this.state.description}/>
<Button raised icon={{name: 'save'}} onPress={this.saveItem}
  title="SAVE" />
```

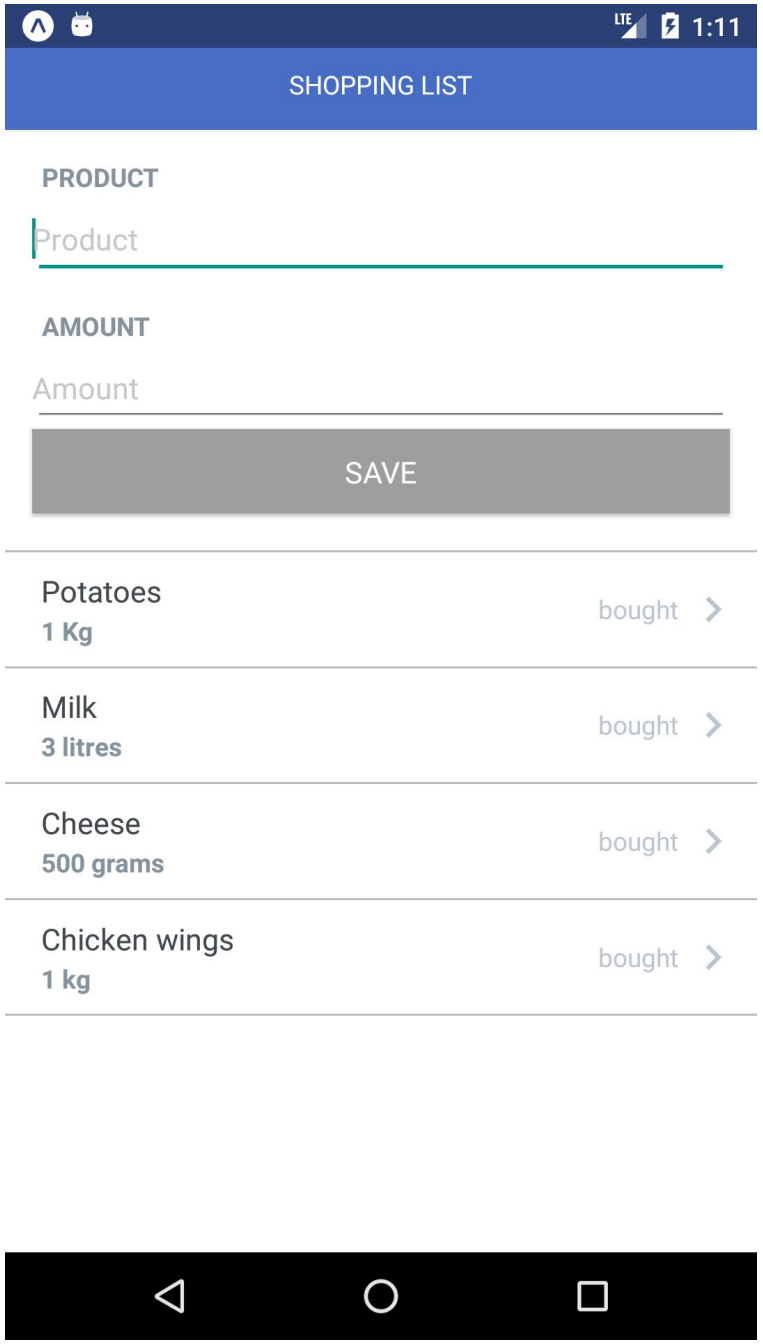
- There is a lot of other components available in the library. See the library documentation for more infromation

Exercise 13

Shopping List polished UI

- Use React Native Element UI Toolkit components in your Shopping List app

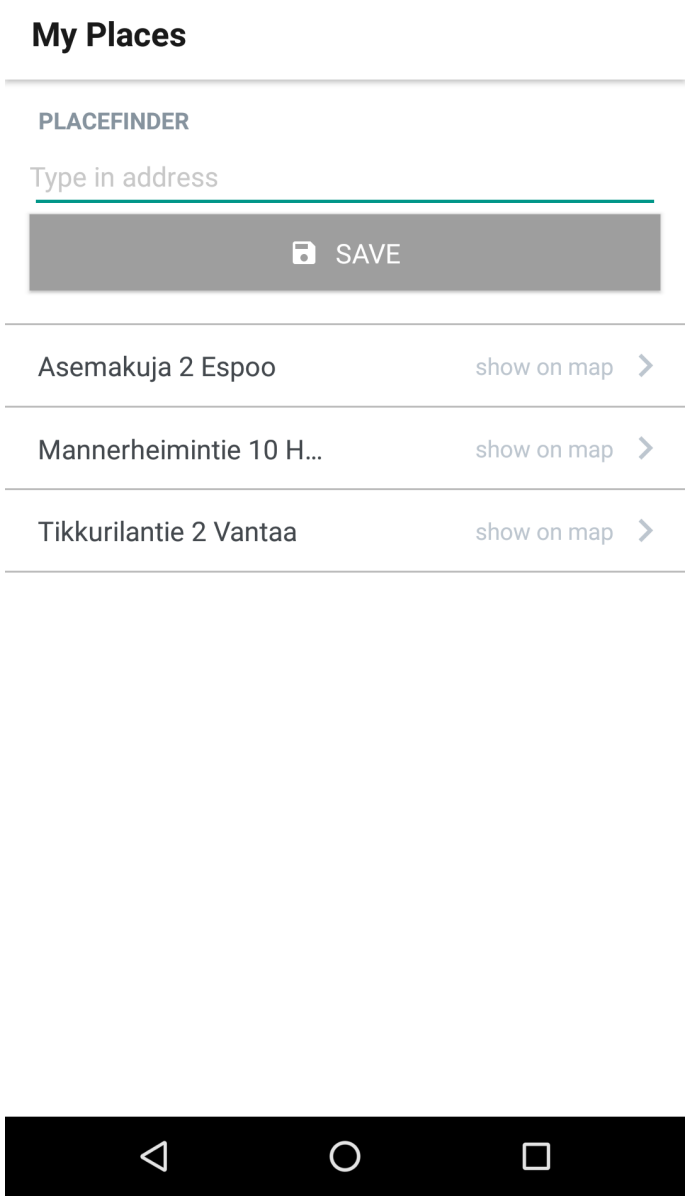
https://react-native-training.github.io/react-native-elements/

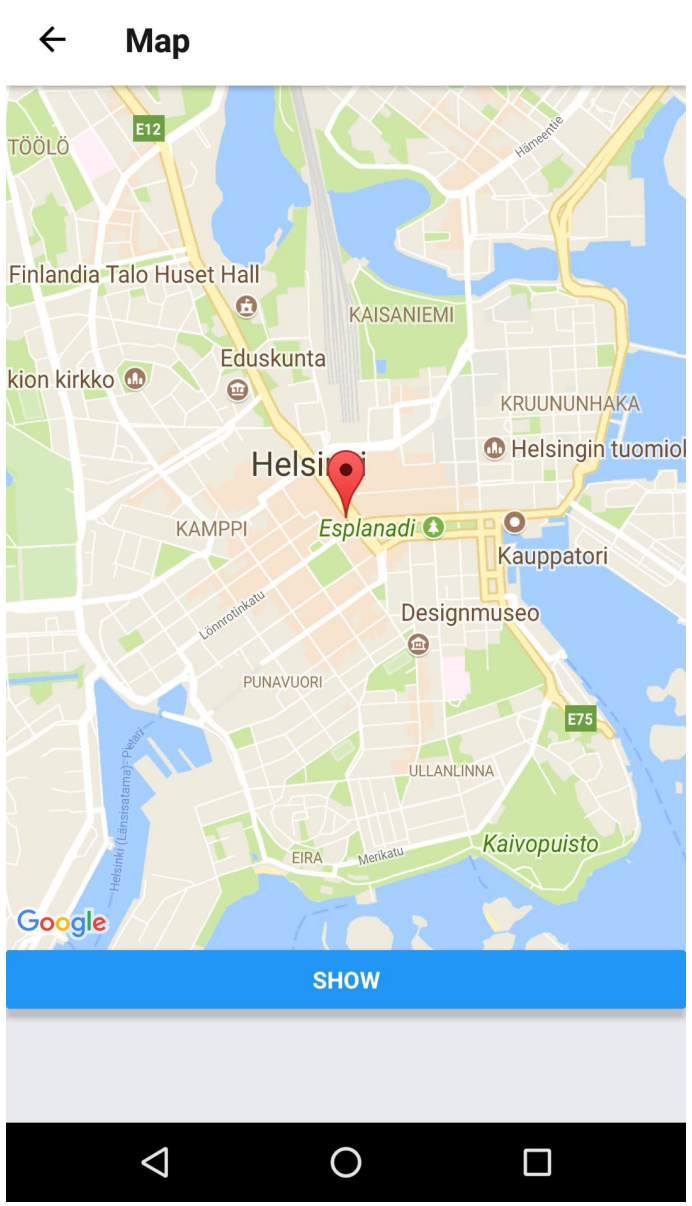


Exercise 14

My Places app

- Create an app where you can save addresses and show these on the map
- Show saved addresses in the React Elements UI list. If item is pressed it will be shown on the map. If item is long pressed it will be deleted.
- Use react navigation for page transitions





Haaga-Helia

React Native: Sources for 3rd party components & material

- js.coach - https://js.coach/
- Awesome React Native - https://github.com/jondot/awesome-react-native
- Native Base - https://nativebase.io/

More information

juha.hinkula@haaga-helia.fi