React Native - Basic Components

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 & Google Flutter**
  + 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

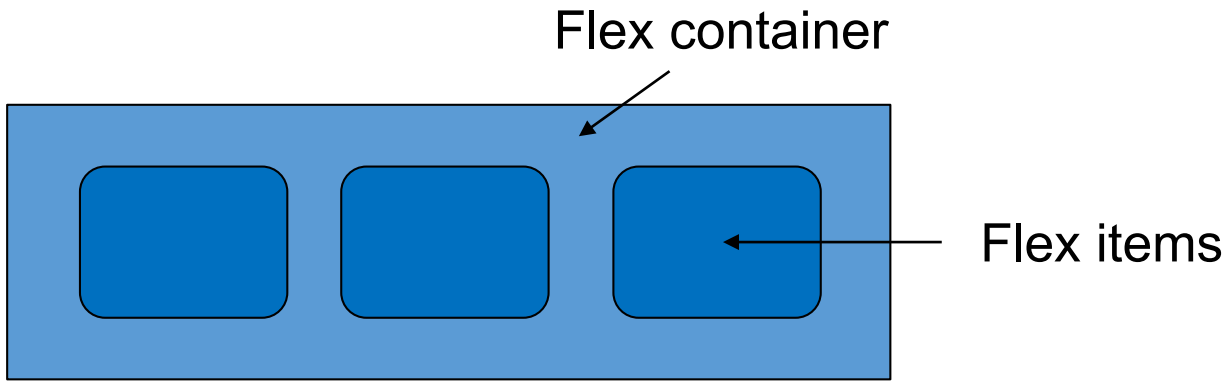
  <Button> - basic button component

  <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.



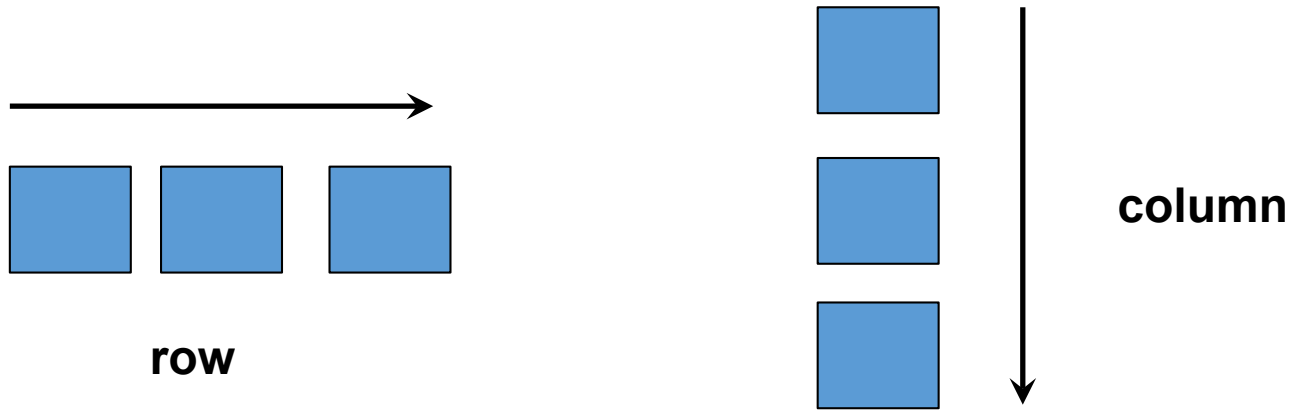
Haaga-Helia

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**



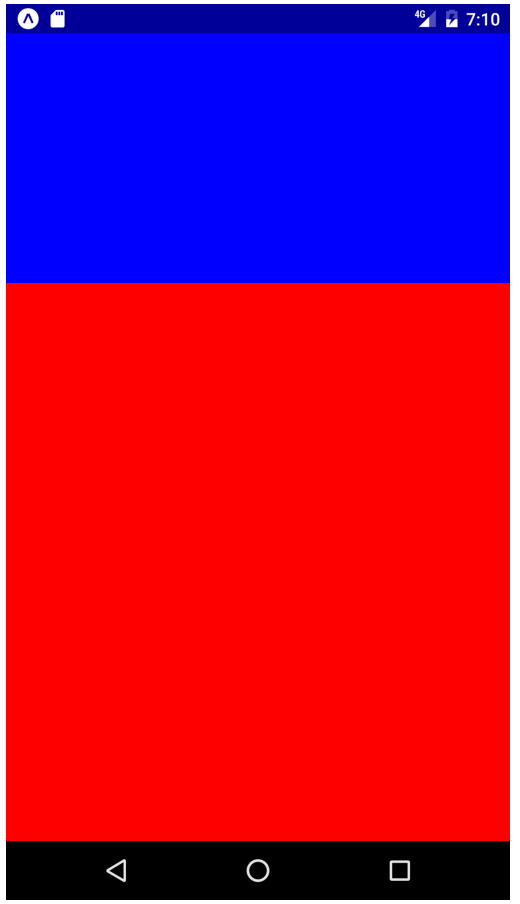
Haaga-Helia

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>
```

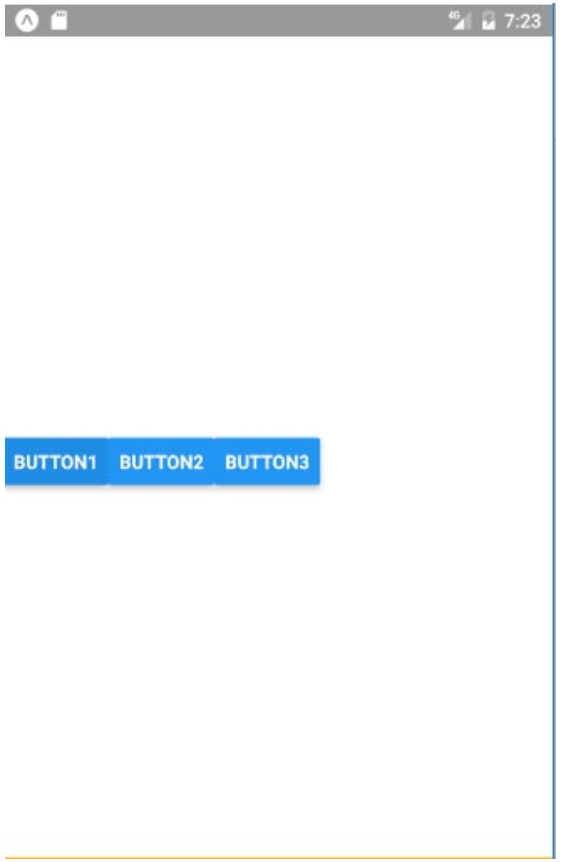


React Native: Flexbox

`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>
```



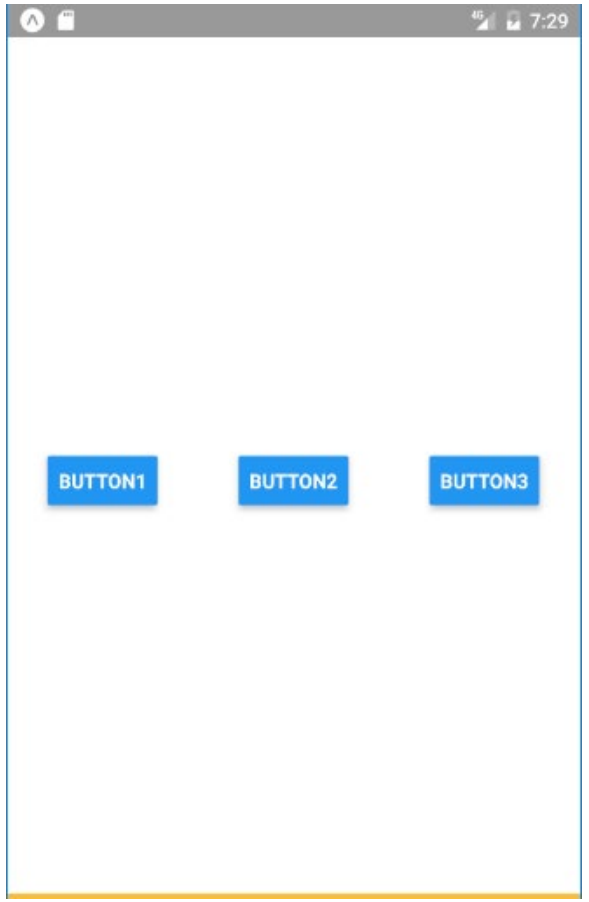
Haaga-Helia

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
```

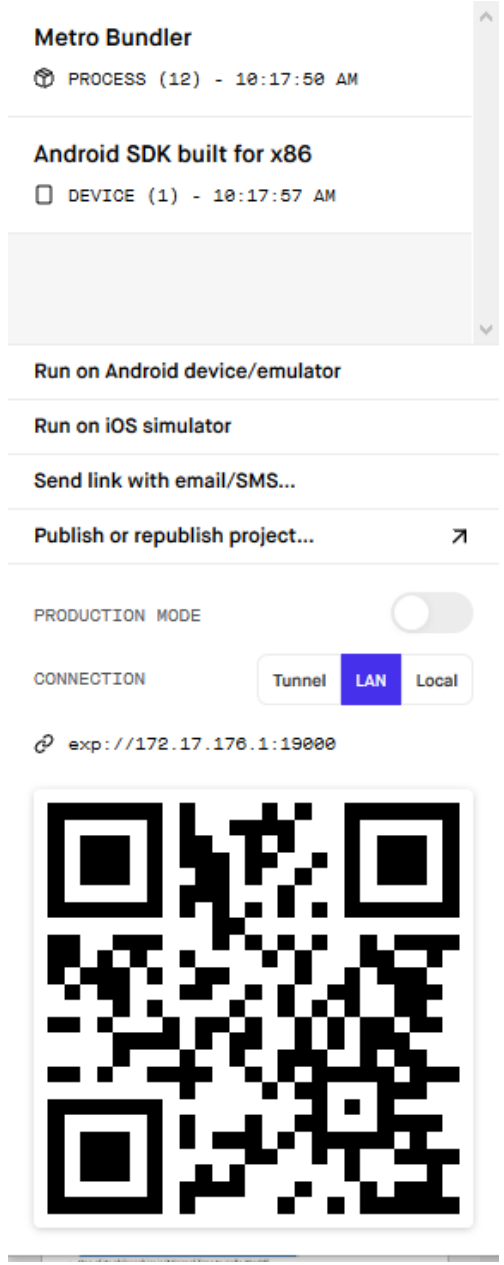
- Create your first app and run it

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

Note! macOS users have to install Watchman. See more from https://docs.expo.io/get-started/installation/

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

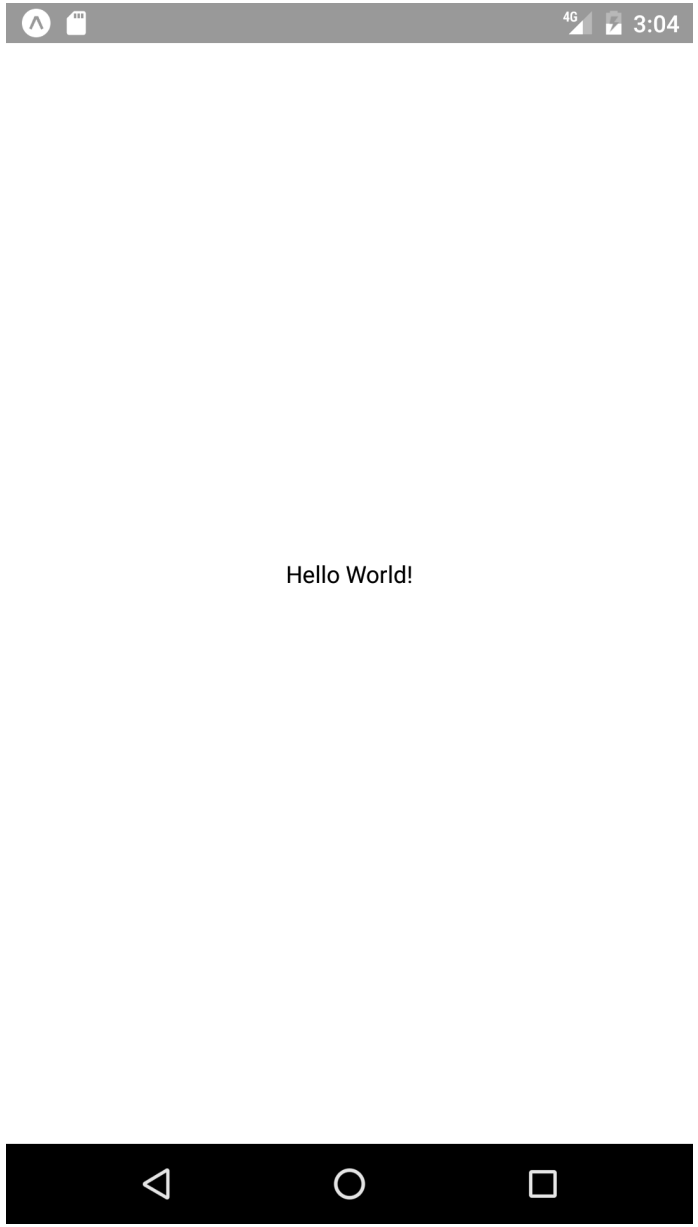
React Native: Hello World

- 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

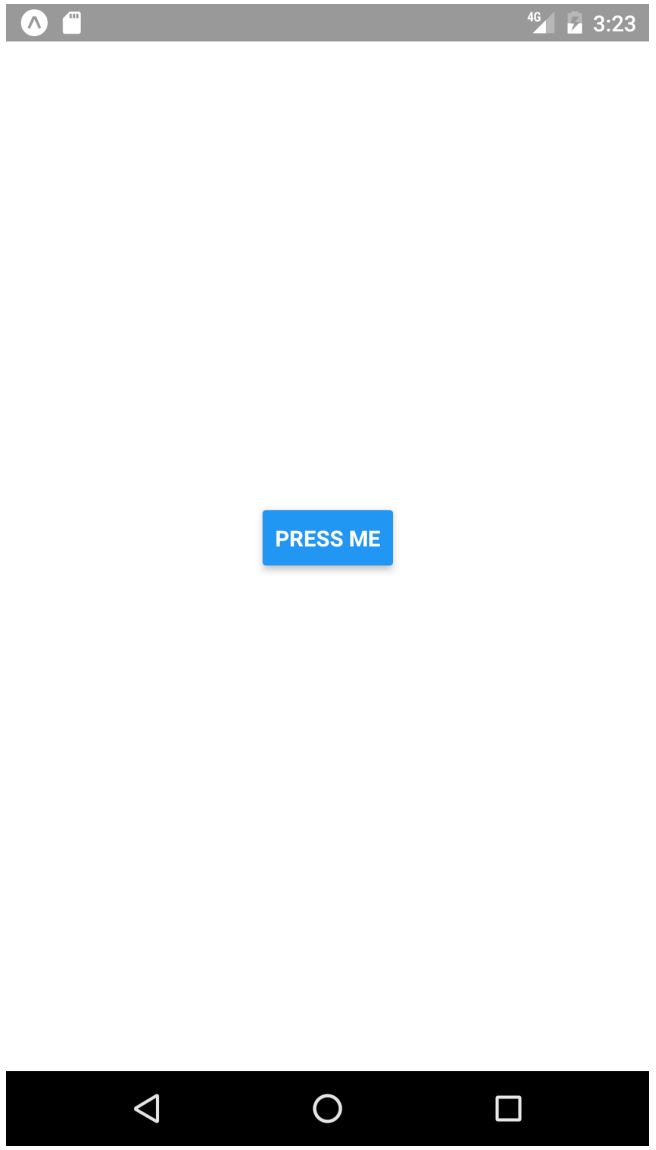
```
return (
  <View style={styles.container}>
    <Button onPress={buttonPressed} title="Press me" />
  </View>
);
```

- 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);
}
```

Haaga-Helia

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'
 },
});
```

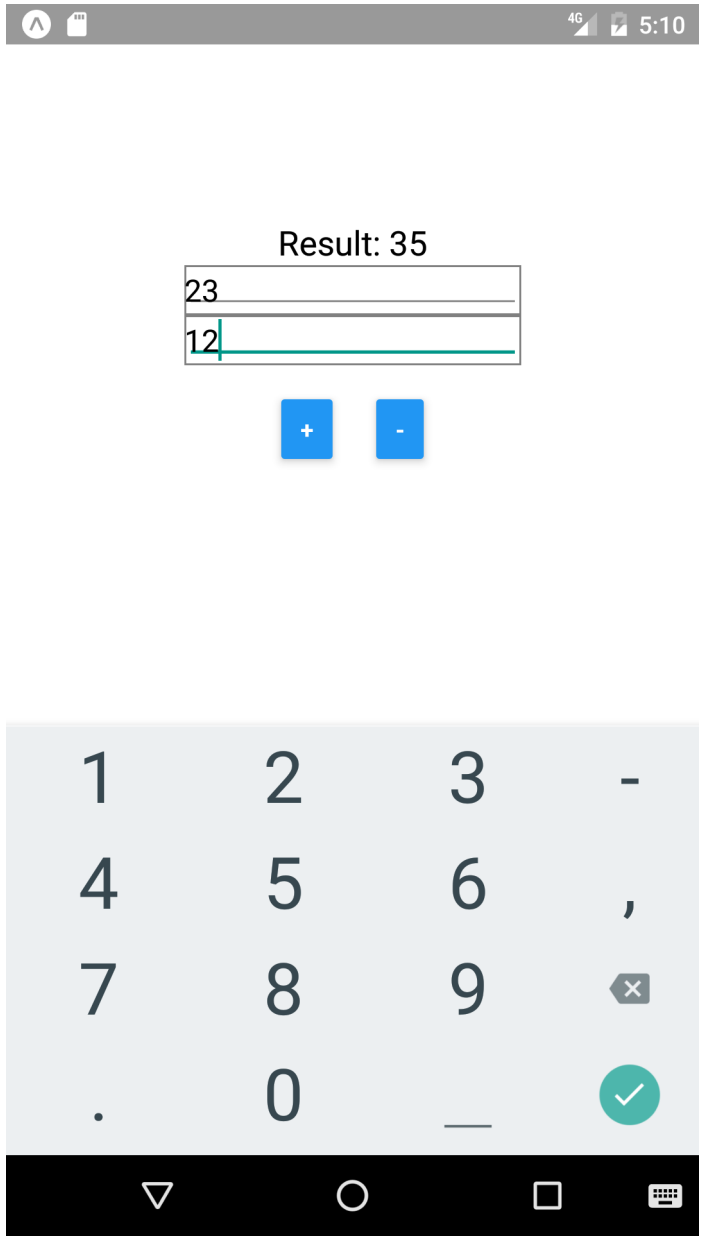
- Use Stylesheet

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

Exercise: Calculator

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



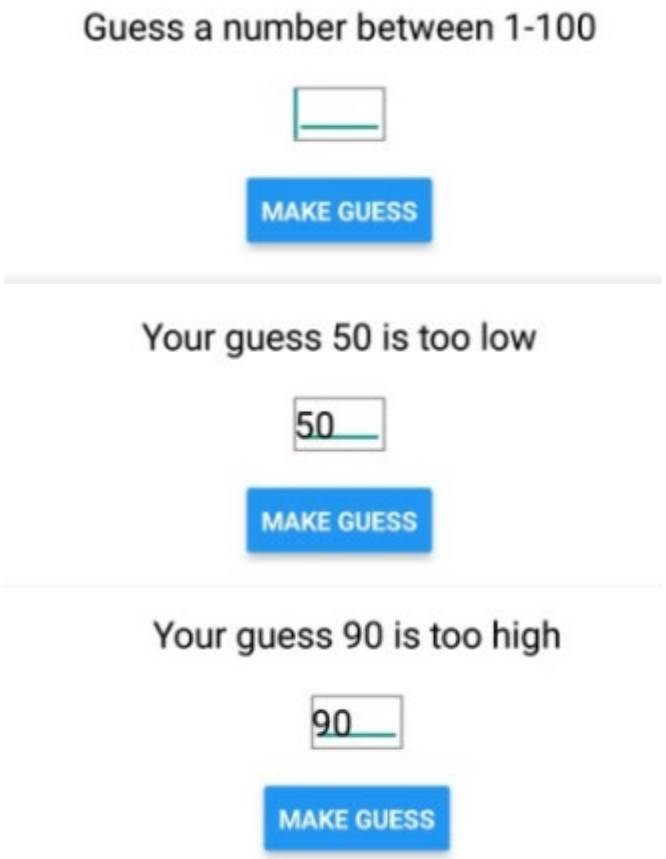
Haaga-Helia

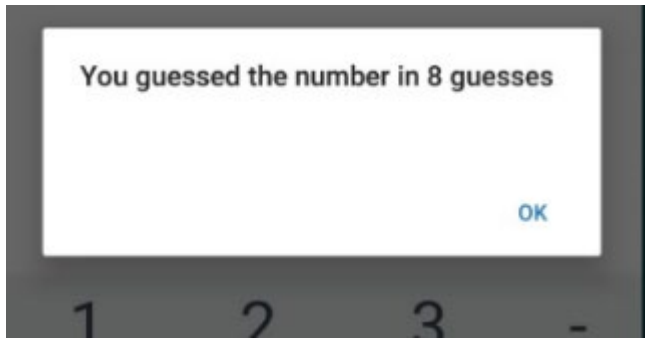
Exercise: Number Guessing Game

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

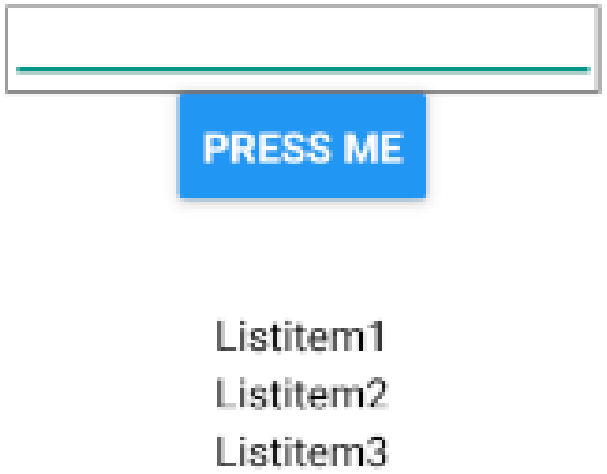
```
<FlatList
  data={data}
  renderItem={({item}) =>
        <Text>{item.key}</Text>}
/>
```

- 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

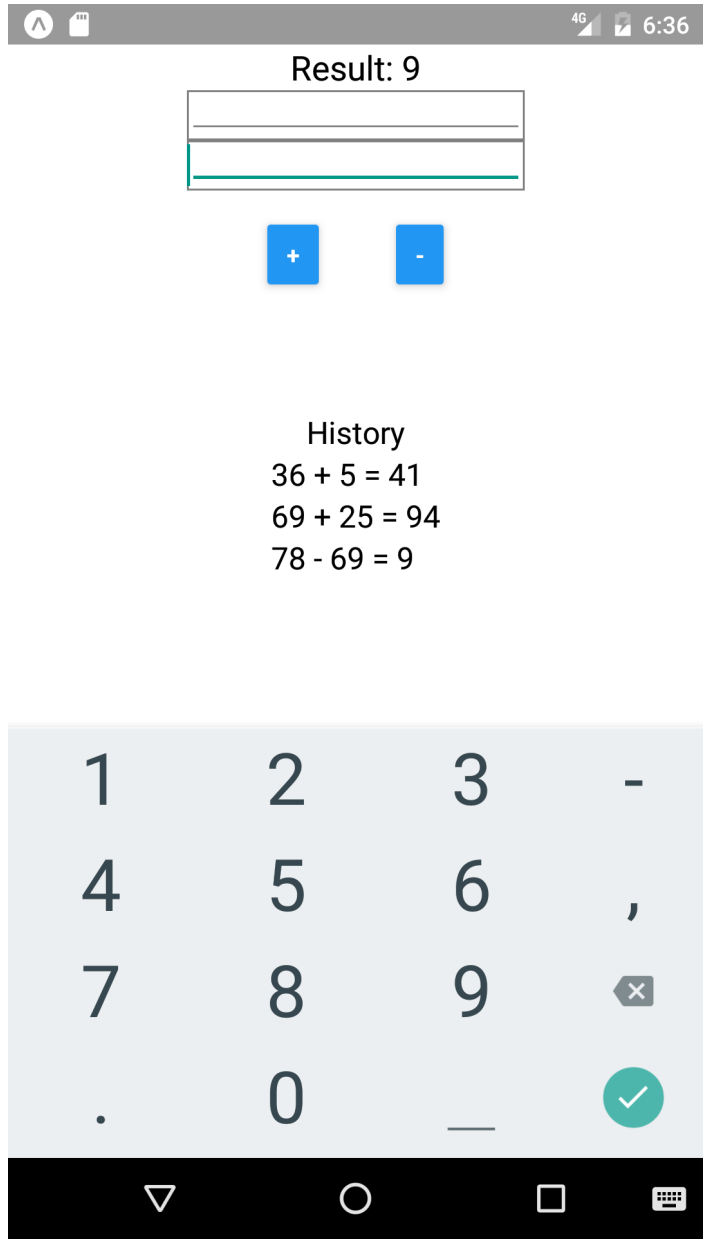
Example:

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

Exercise: Calculator with history

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.



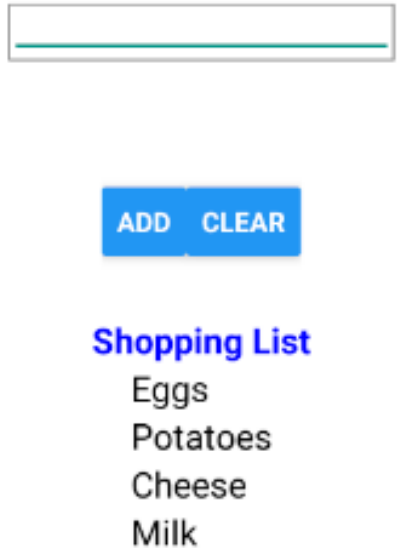
Haaga-Helia

Exercise: Shopping list

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.



More information

juha.hinkula@haaga-helia.fi

Haaga-Helia