React Native 3

CS571: Building User Interfaces

Cole Nelson

Announcements

See @325 | Snack Solution

Announcements

Scholarships available for *continuing CS students* until Thursday, April 13th!

- David Dewitt Undergraduate Scholarship
- Hina and Faisal Mushtaq Scholarship
- Paul D Salmon Memorial Scholarship
- Wai Ying Yick and Dr. Chiu Wai Yuen Endowment

scholarships.wisc.edu

What will we learn today?

- A review of animations and modals
- Making a custom component
- Using simple gestures
- Using switches
- Using React Navigation

Animations using `Animated`

Provides *visual aesthetics* and *feedback*.

```
import { Animated } from 'react-native'
```

May also consider using a third-party-library like react-native-reanimated or lottie.

Animated Docs

Animated

`Animated` provides animations for...

- `View`
- `Text`
- `Image`
- `ScrollView`
- `FlatList`
- `SectionList`

... e.g. `<Animated.View>{/* ... */}</Animated.View>`

Demos

In class, we built/went through several demos...

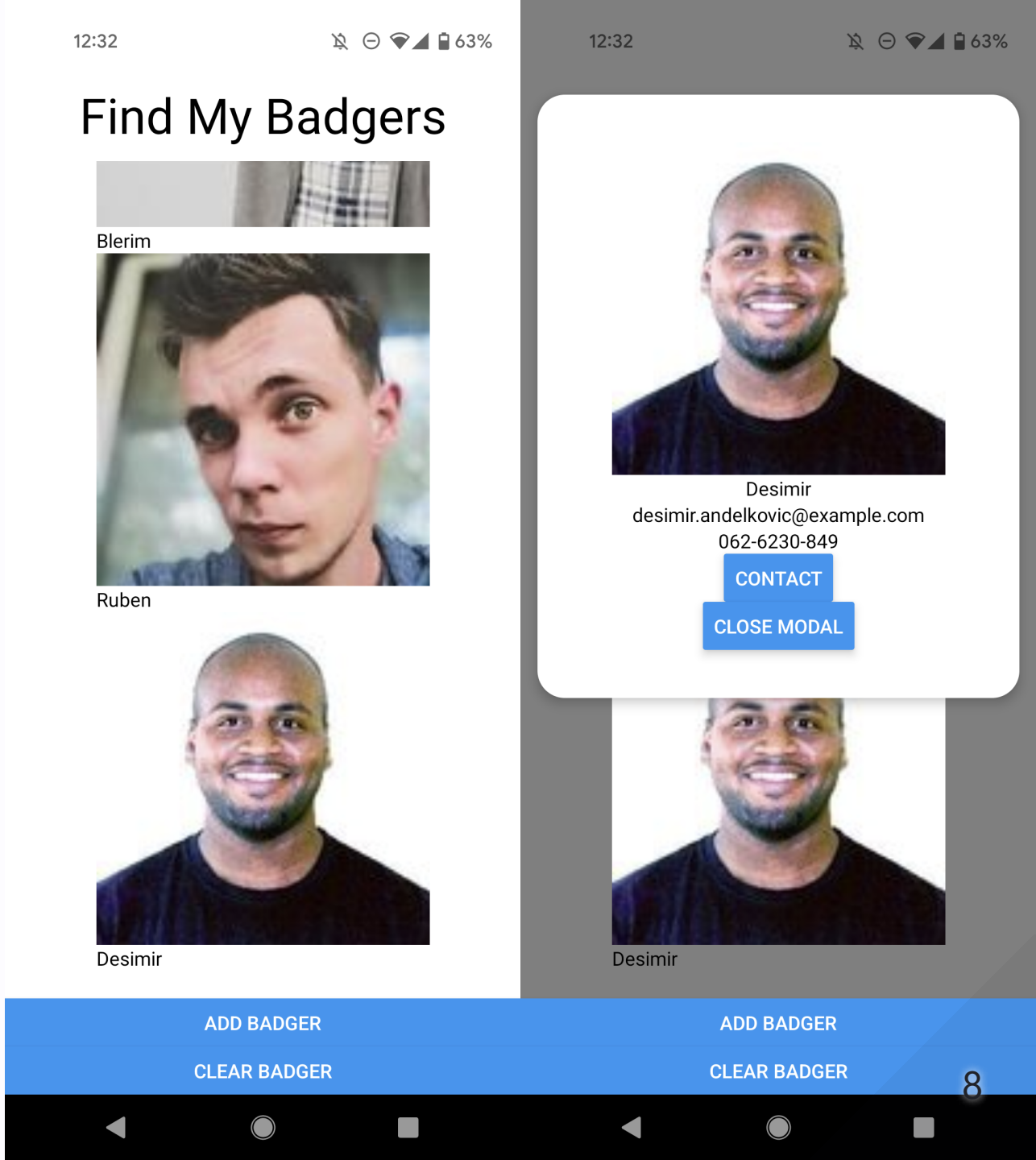
- Simple Animated Demo
- Complex Animated Demo
- Find My Badgers Animated Demo
- Find My Badgers Animated Demo w/ Modal

Modal

A secondary window.

```
import { Modal } from 'react-native'
```

Alternative: react-native-modal



Modal

What you need...

1. Something to open the modal
2. Some content inside the modal
3. Something to close the modal

We often manage whether the modal is open or closed using a state variable, e.g.

```
const [modalVisible, setModalVisible] = useState(false);
```

Modal Properties

- `animationType` : 'slide', 'fade', 'none'
- `onShow` : callback func
- `onRequestClose` : callback func
- `transparent` : true/false
- `visible` : true/false often handled by a state variable

Modal Docs | Modal Snack

FindMyBadgers Demo

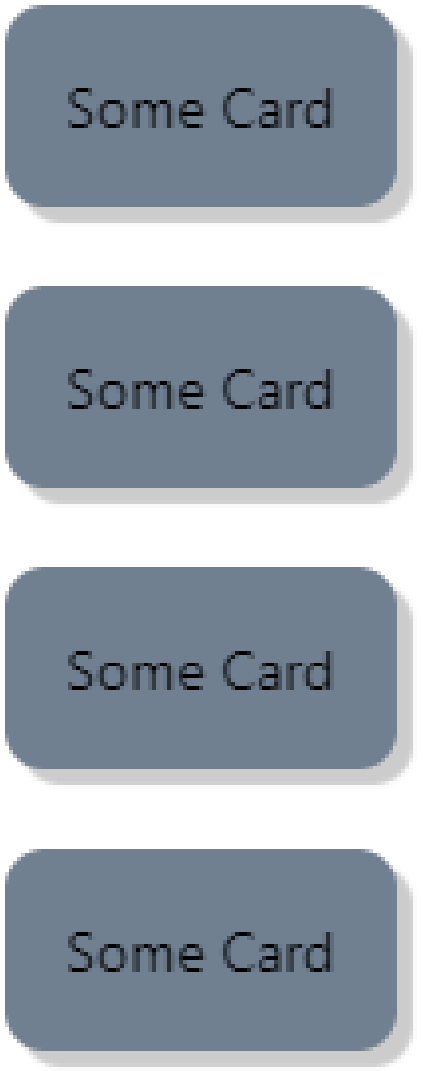
As a React Native app with animations and modals.

Snack Solution

Card

React Native does not have the concept of a "card"...

1. Use a third-party library like `react-native-paper`.
2. Create our own component!



Card

```
export default function BadgerCard(props) {
    return <Pressable onPress={props.onPress}>
        <View style={[styles.card, props.style]}>
            {props.children}
        </View>
    </Pressable>
}
```

Pressable | React Native Paper Card

Card

Adding in the `styles.card` ...

```
const styles = StyleSheet.create({
    card: {
        padding: 16,
        elevation: 5,
        borderRadius: 10,
        backgroundColor: 'slategray',
    }
})
```

Card

Using the BadgerCard...

```
function App() {
  return <View style={styles.main}>
    <BadgerCard
      onPress={() => Alert.alert("Hello", "World!")}
      style={{backgroundColor: "red"}}
    >
      <Text>Testing Custom Card</Text>
    </BadgerCard>
  </View>
};
```

Adding Gestures

```
export default function BadgerCard(props) {
    return <Pressable
      onPress={props.onPress}
      onLongPress={props.onLongPress}
    >
        <View style={[styles.card, props.style]}>
            {props.children}
        </View>
    </Pressable>
}
```

Snack Solution

Your turn!

Create an app that contains a card with only a few words, such as `lorem ipsum`.

When pressed, this card should open a modal that contains even more words such as `lorem ipsum dolor sit...` Allow the user to then close this modal.

Bonus: Can you animate the text?

Snack Solution

Switch

Oops! We forgot to cover an input component `Switch`.

- `value` boolean value of on/off
- `onValueChange` callback function

Switch

```
<Switch
  trackColor={{true: 'darksalmon', false: 'lightgrey'}}
  thumbColor={isOn ? 'crimson' : 'grey'}
  onValueChange={toggle} // callback function
  value={isOn} // boolean state variable
/>
```

Snack Solution

React Native

Things that differentiate mobile from the web...

- **Animations**
- **Gestures** < More Next Week!
- Navigation < Today!
- Sensing < Next Week!

Navigation in React Native

A more mobile-centric library.

React Navigation Alternatives

React Native is a framework* but still lacks support for things like navigation.

- React Router previously!
- React Navigation new!
- `return isHome ? <HomeScreen> : <SettingsScreen>`
- Other outdated libraries...

React Navigation Installation

Just a few dependencies...

```
npm install @react-navigation/native react-native-screens react-native-paper
react-native-safe-area-context react-native-gesture-handler
react-native-reanimated  @react-navigation/native-stack
@react-navigation/drawer @react-navigation/bottom-tabs
```

This is done for you on the homeworks.

Beware of your auto-imports!

React Navigation

We will use...

- Tab Navigation: `@react-navigation/bottom-tabs`
- Drawer Navigation: `@react-navigation/drawer`
- Stack Navigation: `@react-navigation/native-stack`

...others exist!

Navigation Basics

- Must be nested inside of a `NavigationContainer`
- Create navigators via a function `createNAVIGATOR()`
  e.g. `createBottomTabNavigator()`
- Navigators consist of a *navigator* and a set of *screens*

```
<NavigationContainer>
 <SomeNav.Navigator>
   <SomeNav.Screen name="Bookstore" component={BookstoreScreen}/>
   <SomeNav.Screen name="Book" component={BookScreen}/>
 </SomeNav.Navigator>
</NavigationContainer>
```

Navigation Basics

- `useNavigation` is a custom React hook that can be used to help us navigate
  - Supports `navigate`, `reset`, `goBack` among others
- Information can be passed from screen to screen via *route params* (see Native Stack Navigator example)
- Navigators can be styled
- Navigators can be nested

Tab Navigation

```
const SocialTabs = createBottomTabNavigator();

<NavigationContainer>
  <SocialTabs.Navigator>
    <SocialTabs.Screen name="NewsFeed" component={NewsFeedScreen}/>
    <SocialTabs.Screen name="Notifications" component={NotificationScreen}/>
    <SocialTabs.Screen name="AboutMe" component={AboutMeScreen} />
  </SocialTabs.Navigator>
</NavigationContainer>
```

Expo Snack Solution

Drawer Navigation

```
const SocialDrawer = createBottomTabNavigator();

<NavigationContainer>
  <SocialDrawer.Navigator>
    <SocialDrawer.Screen name="NewsFeed" component={NewsFeedScreen}/>
    <SocialDrawer.Screen name="Notifications" component={NotificationScreen}/>
    <SocialDrawer.Screen name="AboutMe" component={AboutMeScreen} />
  </SocialDrawer.Navigator>
</NavigationContainer>
```

Expo Snack Solution

Stack Navigation

```
const BookStack = createNativeStackNavigator();

<NavigationContainer>
  <BookStack.Navigator>
    <BookStack.Screen name="Bookstore" component={BookstoreScreen}/>
    <BookStack.Screen name="Book" component={BookScreen}/>
  </BookStack.Navigator>
</NavigationContainer>
```

Expo Snack Solution

Stack Navigation

Can push a screen onto the history stack via `navigation.push(screenName, params)`

- `screenName` is the name of the screen to navigate to, e.g. `Book`
- `params` is an optional object of parameters to pass to the receiving screen.
- `params` is recieved as `props.route.params`

Nested Navigation

- Navigators can be nested.
  - Stack in Tabs
  - Stack in Drawer
  - Stack in Tabs in Drawer (e.g. Example Below)
  - Stack in Stack in Tabs
  - Stack in Stack in Stack in Stack in Stack
- Make use of the `headerShown` option!

Expo Snack Solution

A Note on Expo

Expo is a library for quickly getting started with React Native projects. No need to...

- cocoa pods install ❌
- maven/gradle building ❌
- react native linking ❌

You may need to use specific expo libraries, such as @expo/vector-icons

What did we learn today?

- A review of animations and modals
- Making a custom component
- Using simple gestures
- Using switches
- Using React Navigation

Questions?