React Native 2

CS571: Building User Interfaces

Cole Nelson & Yuhang Zhao

Today's Warmup

- Set back and relax! :)
- We'll try our in-class activity through "Expo Snacks" today instead of downloading starter code.

Badger IDs

Okay to hardcode them!

Otherwise, set an environment variable `EXPO_PUBLIC_CS571_BADGER_ID` to be your Badger ID.

```
import CS571 from '@cs571/mobile-client'

// ...

CS571.getBadgerId() // returns ID
```

May require a restart!

What will we learn today?

- What are other core concepts in React Native?
  - How can we do dynamic display?
  - What are Switch, ScrollView, Card, and Pressable?
- How can we perform animations?
- How can we perform navigation?

What is React Native?

A JS framework for building native, cross-platform mobile applications using React, developed by Facebook in 2015.

Unlike ReactJS, which was a library, React Native is a framework that includes everything* that we will need to build mobile applications.

React Native supports iOS and Android development.

Hello World!

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

function MyApp() {
  return (
    <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
      <Text>
        Try editing me! 🎉
      </Text>
    </View>
  );
}

export default MyApp;
```

Expo Snack

Issues with Cross-Platform

How can differentiate based on platform?

- e.g. iOS vs Android

How can we adjust to the size of the device?

- e.g. iPhone vs iPhone SE vs iPhone XL

Cross-Platform: By Platform

React Native provides a number of components that utilize platform capabilities that may not be available in other platforms, thus for cross-platform development, we need to utilize multiple platform-specific components.

e.g. `TouchableNativeFeedback` only works on Android; a *similar* effect can be achieved using `TouchableHighlight` on iOS.

Cross-Platform: By Platform

```
if (Platform.OS === 'android') {
  return (
    <TouchableNativeFeedback> ... </TouchableNativeFeedback>
  );
} else {
  return (
    <TouchableHighlight> ... </TouchableHighlight>
  );
}
```

Optionally, create two components e.g. `MyButton.ios.js` and `MyButton.android.js`.

Snack Solution

Cross-Platform: By Size

Mobile devices vary significantly in screen size, and we open need to obtain screen dimensions of the device using the `Dimensions` class in `react-native`.

```
const getScreenSize = () => {
 const screenWidth = Math.round(Dimensions.get('window').width);
 const screenHeight = Math.round(Dimensions.get('window').height);
 return { screenWidth: screenWidth, screenHeight: screenHeight };
}
```

Snack Solution

React Native Components

What are other common components we will be using?

- Switch
- ScrollView
- Pressable
- Card (not actually in React Native!)

Switch

A `Switch` is on or off.

- `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

ScrollView

Like a `View`, but scrollable! Make sure that it is in a view that is flex-ible.

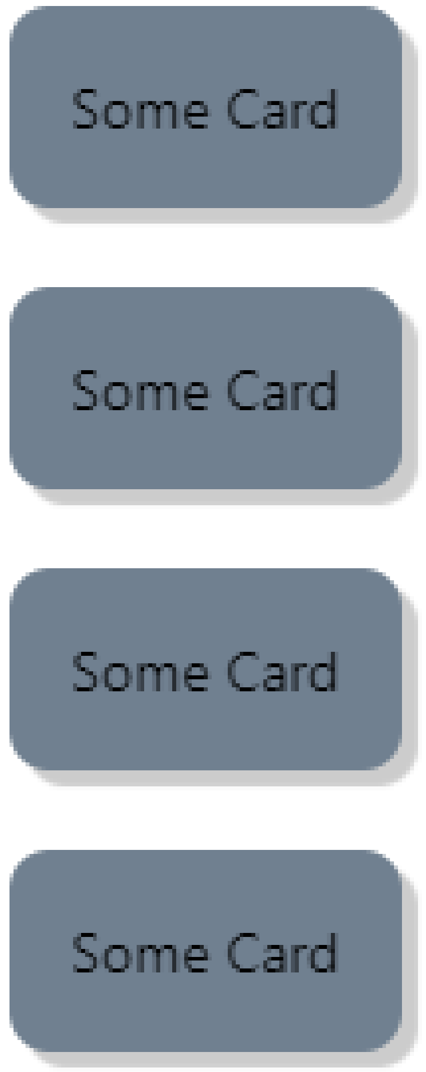
```
<View style={{flex: 1}}>
  <ScrollView>
    { /* A bunch of content here! */ }
  </ScrollView>
</View>
```

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

Animations

Providing *feedback* in a *visually aesthetic* way.

Animations using `Animated`

Providing *feedback* in a *visually aesthetic* way.

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

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

Animated Docs

Animated

`Animated` provides animations for...

- `View`
- `Text`
- `Image`
- `ScrollView`
- `FlatList` (similar to a ScrollView)
- `SectionList` (similar to a ScrollView)

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

Animated

These are animated using...

- `Animated.timing` ,
- `Animated.spring`
- `Animated.decay`

... which manipulate an `Animated.Value` , e.g.

```
Animated.timing(opVal, {
  toValue: 1,
  duration: 10000, // in ms
  useNativeDriver: true // must include
})
```

Animated

`Animated.Value` is used in combination with useRef.

```
const opVal = useRef(new Animated.Value(0)).current
```

To run an animation on page load...

```
useEffect(() => {
  Animated.timing(opVal, {
    toValue: 1,
    duration: 10000,
    useNativeDriver: true
  }).start() // don't forget this!
}, [])
```

```
export default function FadeInView(props) {
  const opVal = useRef(new Animated.Value(0)).current;
  useEffect(() => {
    Animated.timing(opVal, {
      toValue: 1,
      duration: 5000,
      useNativeDriver: true,
    }).start();
  }, []);
  return (
    <View>
      <Animated.View
        style={{
          height: 100, width: 100, opacity: opVal
          backgroundColor: "cyan",
        }}>
      </Animated.View>
    </View>
  );
}
```

Expo Snack

Animated

Can control many animations using...

- `Animated.parallel`
- `Animated.sequence`
- `Animated.loop`

`start()` and `stop()` apply to the set of animations

In parallel…

```
useEffect(() => {
  Animated.parallel([
    Animated.timing(height, {
      toValue: 800,
      duration: 10000,
      useNativeDriver: false, // cannot use native driver for height/width!
    }),
    Animated.timing(width, {
      toValue: 500,
      duration: 10000,
      useNativeDriver: false,
    })
  ]).start()
}, []);
```

In sequence...

```
useEffect(() => {
  Animated.sequence([
    Animated.timing(sizeVal, {
      toValue: 500,
      duration: 10000,
      useNativeDriver: false,
    }),
    Animated.timing(sizeVal, {
      toValue: 0,
      duration: 10000,
      useNativeDriver: false,
    })
  ]).start()
}, []);
```

In loop...

```
useEffect(() => {
  Animated.loop( // not an array!
    Animated.sequence([
      Animated.timing(sizeVal, {
        toValue: 500,
        duration: 10000,
        useNativeDriver: false,
      }),
      Animated.timing(sizeVal, {
        toValue: 0,
        duration: 10000,
        useNativeDriver: false,
      })
    ])
  ).start()
}, []);
```

Animated Demo

Expo Snack

Animated

You cannot *directly* add/subtract/multiply/divide `Animated.value`. Instead, you must use...

- `Animated.add(v1, v2)`
- `Animated.subtract(v1, v2)`
- `Animated.multiply(v1, v2)`
- `Animated.divide(v1, v2)`

Animated

e.g. start at 50 and grow from there.

```
<Animated.View
  style={{
    backgroundColor: "blue",
    height: Animated.add(height, 50),
    width: Animated.add(width, 50)
  }}>
</Animated.View>
```

Your turn!

Take this snack and make it so that the Badgers fade in as they are added.

Hint: Only change the `Badger` component.

Expo Solution

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 = createDrawerNavigator();

<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

What did we learn today?

- What are other core concepts in React Native?
  - How can we do dynamic display?
  - What are Switch, ScrollView, Card, and Pressable?
- How can we perform animations?
- How can we perform navigation?

Questions?