Mobile Dev 3

CS571: Building User Interfaces

Cole Nelson & Yaxin Hu

Today's Warmup

- Again, we'll fork an existing snack for our in-class exercise! :)
- Download and import today's Postman collection!
- Replace `ENTER_YOUR_BID` with your Badger ID.
- Toy around with it! What's different from the HW6 API? Also, read the docs.

What will we learn today?

1. Be able to provide for richer navigation, including data passing, switch navigation, modals, and dynamically-loaded lists.
2. Be able store and send secrets using mobile devices.
3. Be able to tap into advanced mobile features such as sensors, gestures, and animations.
4. Be able to build and deploy your application.

A Quick Note on Accessibility

1. Logically group elements with an `accessible` prop.
2. Provide `accessibilityLabel` (primary information) and `accessibilityHint` (auxiliary information) for each of these elements.
3. Make other considerations: color contrast, reliance on gestures and visual cues, among other W3 recommendations.
4. Test it yourself! Use VoiceOver or TalkBack.

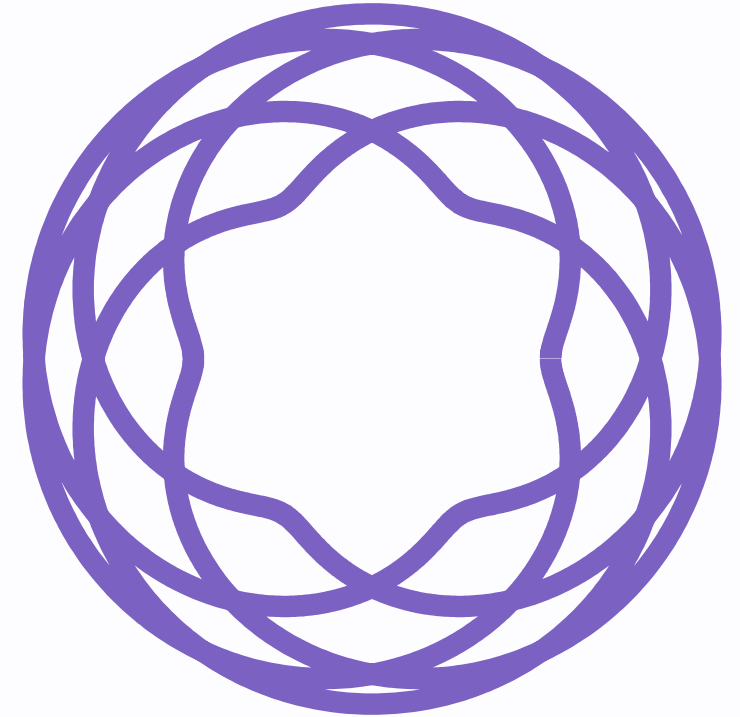Read the full documentation here.

Richer Navigation

React Navigation

Last time we covered…

- Stack Navigation
- Tabs Navigation
- Drawer Navigation

How do we pass data down?

- Using context
- Using render callbacks

Passing Data Down

Problem: Create a drawer with many tools in it.

```
const [tools, setTools] = useState([
  { name: 'Hammer', weight: 2, dangerous: false },
  { name: 'Screwdriver', weight: 1, dangerous: false },
  { name: 'Sawzall', weight: 6, dangerous: true }
]);
```

Hammer

Screwdriver

Sawzall

Passing Data Down

Displaying the tool data.

```
export default function ToolScreen(props) {
  return <View>
    <Text>I am a {props.name}</Text>
    <Text>I weigh {props.weight} pounds</Text>
    {
      props.dangerous ?
        <Text>I am dangerous</Text> :
        <Text>I am not dangerous</Text>
    }
  </View>
}
```

Creating Navigator

```
<NavigationContainer>
  <ToolDrawer.Navigator>
    <ToolDrawer.Screen name="Landing" component={LandingScreen}/>
    <ToolDrawer.Screen name="Hammer" component={ToolScreen}/>
    <ToolDrawer.Screen name="Screwdriver" component={ToolScreen}/>
    <ToolDrawer.Screen name="Sawzall" component={ToolScreen}/>
  </ToolDrawer.Navigator>
</NavigationContainer>
```

❌ hardcoding tool names
❌ not passing tool data

Snack Example

```
<NavigationContainer>
  <ToolDrawer.Navigator>
    <ToolDrawer.Screen name="Landing" component={LandingScreen}/>
    {
      tools.map(tool => {
        return <ToolDrawer.Screen
          key={tool.name}
          name={tool.name}
          component={ToolScreen}
        />
      })
    }
  </ToolDrawer.Navigator>
</NavigationContainer>
```

❌ not passing tool data

```
<NavigationContainer>
  <ToolDrawer.Navigator>
    <ToolDrawer.Screen name="Landing" component={LandingScreen}/>
    {
      tools.map(tool => {
        return <ToolDrawer.Screen key={tool.name} name={tool.name}>
          {(props) => <ToolScreen {...props} {...tool}/>}
        </ToolDrawer.Screen>
      })
    }
  </ToolDrawer.Navigator>
</NavigationContainer>
```

✅ correct way!

Snack Example

Switch Navigation

There is a fourth, "informal" navigation.

⚠️ Do not use the one from React Navigation! It is very out-of-date. It existed in React Navigation < 4.x.

"Switch" Navigation

```
isSignedIn ? (
  <>
    <Stack.Screen name="Home" component={HomeScreen} />
    <Stack.Screen name="Profile" component={ProfileScreen} />
    <Stack.Screen name="Settings" component={SettingsScreen} />
  </>
) : (
  <>
    <Stack.Screen name="SignIn" component={SignInScreen} />
    <Stack.Screen name="SignUp" component={SignUpScreen} />
  </>
);
```

Switch Navigation

Modal

A secondary window.

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

Modal

Reminder: A modal is a secondary overlay that *requires* user interaction before continuing.

This is a *controlled component*, e.g.

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

Modal Snack

Modal

A modal is nothing more than a secondary overlay that is `visible` or not! You still need to style it.

```
const styles = StyleSheet.create({
  modalView: {
    margin: 20,
    alignItems: 'center',
    shadowColor: '#000',
    shadowOpacity: 0.25,
    shadowRadius: 4,
    elevation: 5,
    // ...
```

Modal Docs | Find My Badgers Modal

FlatList

A ScrollView that incrementally loads.

For you to learn! Read the docs.

FlatList

Can you turn this ScollView into a FlatList using just the documentation?

Bonus: Can you implement pull-to-refresh?

Snack Solution

Managing Secrets

What's so secret?

A JSON Web Token (JWT)!

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MjgsInVzZXJuYW1lIjoidGVzdDEyN
DU2NyIsImlhdCI6MTY5OTkxNDQxMiwiZXhwIj
oxNjk5OTE4MDEyfQ.Z0WiskxiQUTzEhHocjzg
4y5QaFEOCgg4Zz8sLszSMB0

What is this? A signed access token good for a set period of time.

Authentication

Web Development

- Interface with the *user's browser.*
- Prefer `HTTP-Only` cookies.

Mobile Development

- Interface with the *user's operating system.*
- Prefer OS-level secure encrypted storage.

Authentication

Web: Specify *option* `include: 'credentials'`

```
fetch("https://example.com/api/submit", {
  method: "POST",
  credentials: "include",
  // ...
```

Authentication

Mobile: Specify…

- *header* `Authorization`
- *value* `Bearer <JWT>`

```
fetch("https://example.com/api/submit", {
  method: "POST",
  headers: {
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiIs..."
  // ...
```

If the request has a body, don't forget `Content-Type` !

HW9 Demo

Secrets and `Authorization`

Download and import today's Postman collection.

Secure Storage

Uh oh! We have to store a JWT?

- React has no built-in way to handle credentials!😮
- expo-secure-store
  - Only works on Android and iOS -- not web!
- Stores key/value pairs to *persistent* storage.
- Up to 2KB of data per value (small!)
- Additional options for passcode 🔒 and biometrics 👉

Secure Storage

Secure storage is asynchronous* in nature.

- `getItemAsync(key)`
- `setItemAsync(key, val)`
- `deleteItemAsync(key)`

Note: Key refers to a storage key (think HashMap) *not* an encryption key! Similar to `localStorage`

* See Slide 28

Secure Storage

`expo-secure-store` shows an example of `async` / `await` … What is this?…

Same thing as `then` and `catch` on a `Promise` (think `fetch`)… just a slightly different syntax!

We'll cover `async` / `await` in our next series!

Secure Storage

[CS571-ifying](#) the expo-secure-store example.

Use your phone, not the web!

*New in `expo-secure-store` are the methods `getItem` and `setItem` which happen *synchronously*! You can use these instead if you wish.

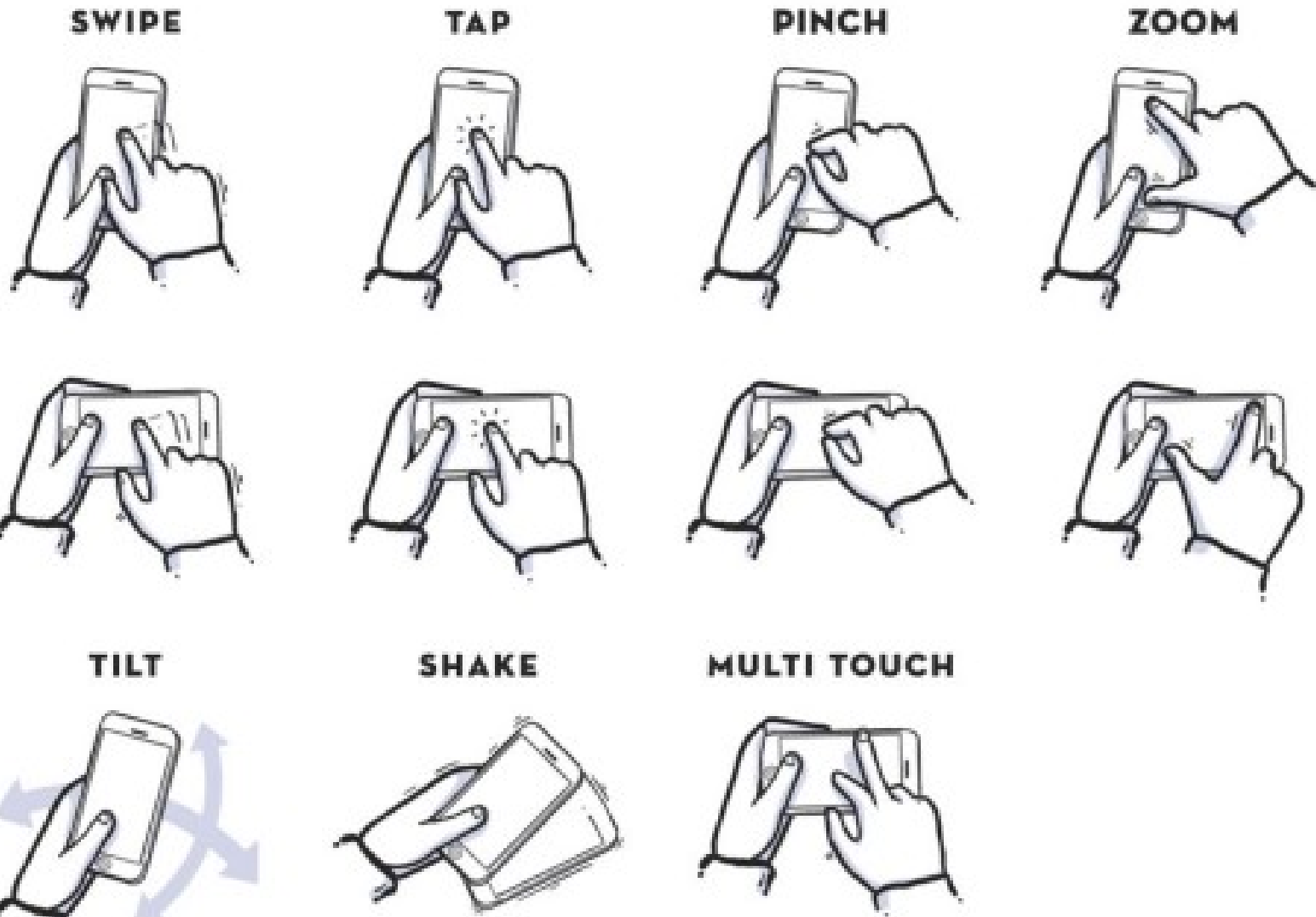# Advanced Gestures

Making use of diverse mobile inputs ✌️☝️✋👌👍

Image Source

Gestures

React Native provides methods to detect when and where the user's fingers move.

Higher-level gesture response libraries...

- react-native's PanResponder
- react-native-gesture-handler
- component libraries, e.g. react-native-paper

Component Library Gestures

react-native's Button `onPress`

react-native-paper's Card `onPress` and `onLongPress`

react-native-elements' Slider `onSliding`

react-native-maps' Marker `onDrag`

react-navigation's Drawer gesture

Sensors

Use `expo-sensors` instead of `react-native-sensors`

- Accelerometer
- Barometer
- Gyroscope
- LightSensor
- Magnetometer
- Pedometer

Not all devices have all sensors!

Other Sensors

- `expo-camera`
- `expo-battery`
- `expo-haptics`
- `expo-av`
- `expo-brightness`

Beware of permissions!

Deployment

Getting your app to a production environment.

Deployment

iOS vs Android Market Share

Region	iOS	Android	Other
USA	54%	45%	1%
North America	52%	47%	1%
Asia	16%	83%	1%
Worldwide	28%	71%	1%

Source: GlobalStatCounter

Deployment

Use Expo Application Services (EAS)

```
npm install -g eas-cli
```

```
eas build -p android
eas build -p ios
```

An `.apk` gets deployed to the Google Play Store

An `.ipa` gets deployed to the iOS App Store

Deployment

No web server for deployment! Hosted on Google Play Store or iOS App Store.

App goes through a review process.

- **Google Play Store:** Hours to Days
- **iOS App Store:** Days to Weeks

Deployment Considerations

- Reliability
- Performance
- Monitoring
- Business Value of Delivery
- App Store Optimization

Questions?