React Native 4

CS571: Building User Interfaces

Cole Nelson

What will we learn today?

- How to pass data using React Navigation?
- How to perform "switch" navigation?
- How to store secrets on mobile devices?
- How to do advanced gestures and animations?
- How to use sensors?
- How to deploy our apps?

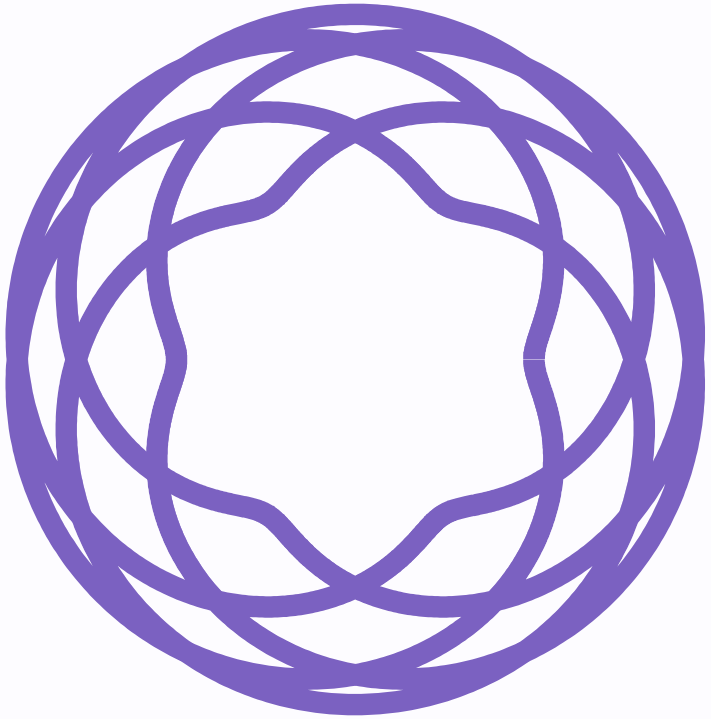
React Navigation

Last time we covered...

- Stack Navigation
- Tabs Navigation
- Drawer Navigation

How do we pass data down?

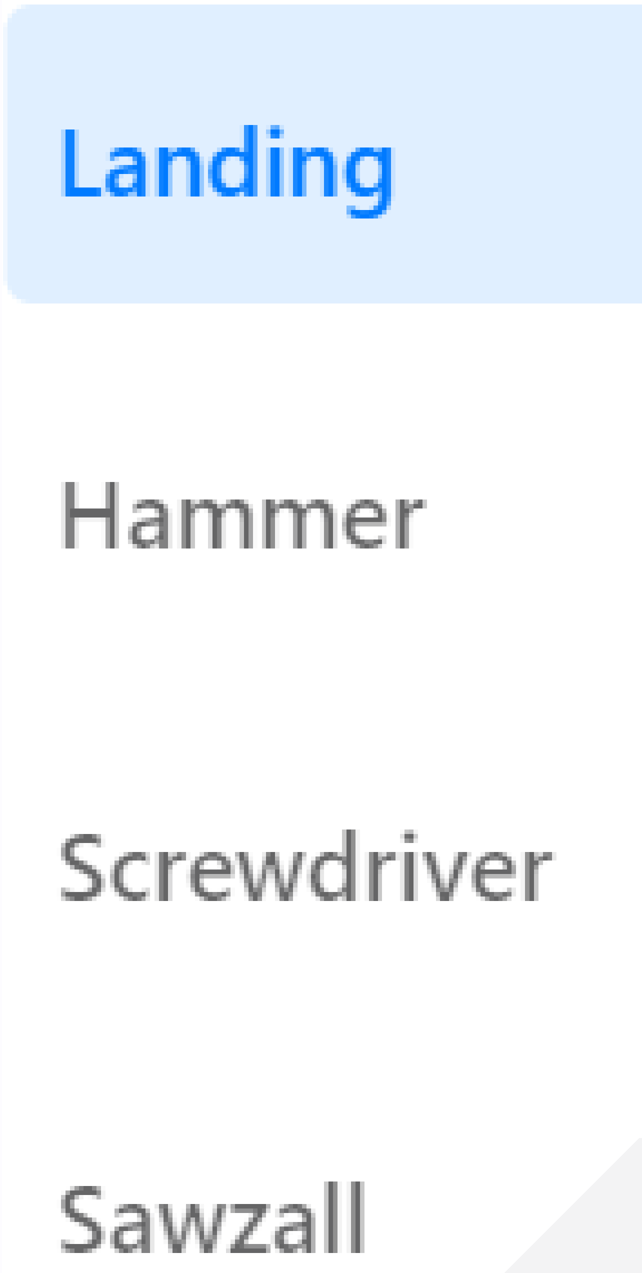
- Using context
- Using render callbacks



Passing Data Down

Problem: Create a drawer with many tools in it.

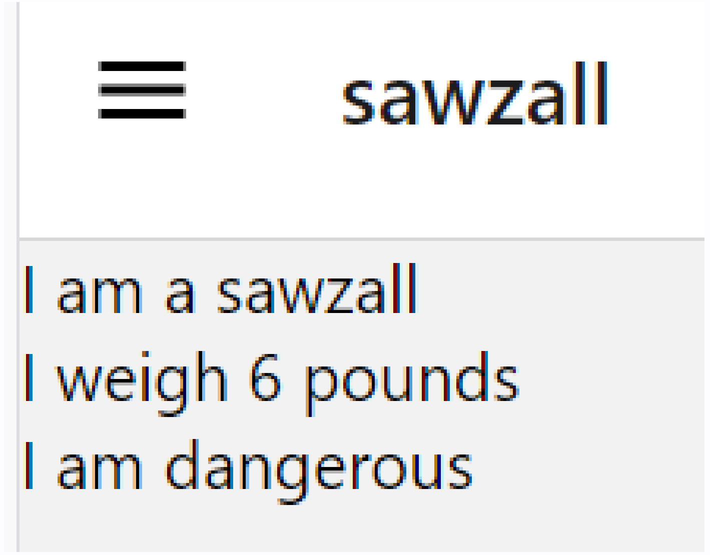
```
const [tools, setTools] = useState([
  { name: 'hammer', weight: 2, dangerous: false },
  { name: 'screwdriver', weight: 1, dangerous: false },
  { name: 'sawzall', weight: 6, dangerous: true }
]);
```



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>
}
```



I am a sawzall
I weigh 6 pounds
I am dangerous

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>
```

Similar Solution: BadgerChat

❌ 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} />
  </>
);
```

React Navigation AuthFlow

"Switch Navigation"

Premise: perform a conditional render.

If the user is signed in, show them their needs feed.

Else give the user the option to sign in or sign up.

Snack Solution

HW10 Demo

Switch navigation and data passing.

Secure Storage

Storing secrets (asynchronously!)

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`!

Authentication

Download HW10 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!)
- Cannot store emojis 😔
- 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!

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 DialogFlow!

Secure Storage

CS571-ifying the expo-secure-store example.

Use your phone, not the web!

Advanced Gestures

Making use of diverse mobile inputs ✌️ ☝️ ✋ 👌 👍

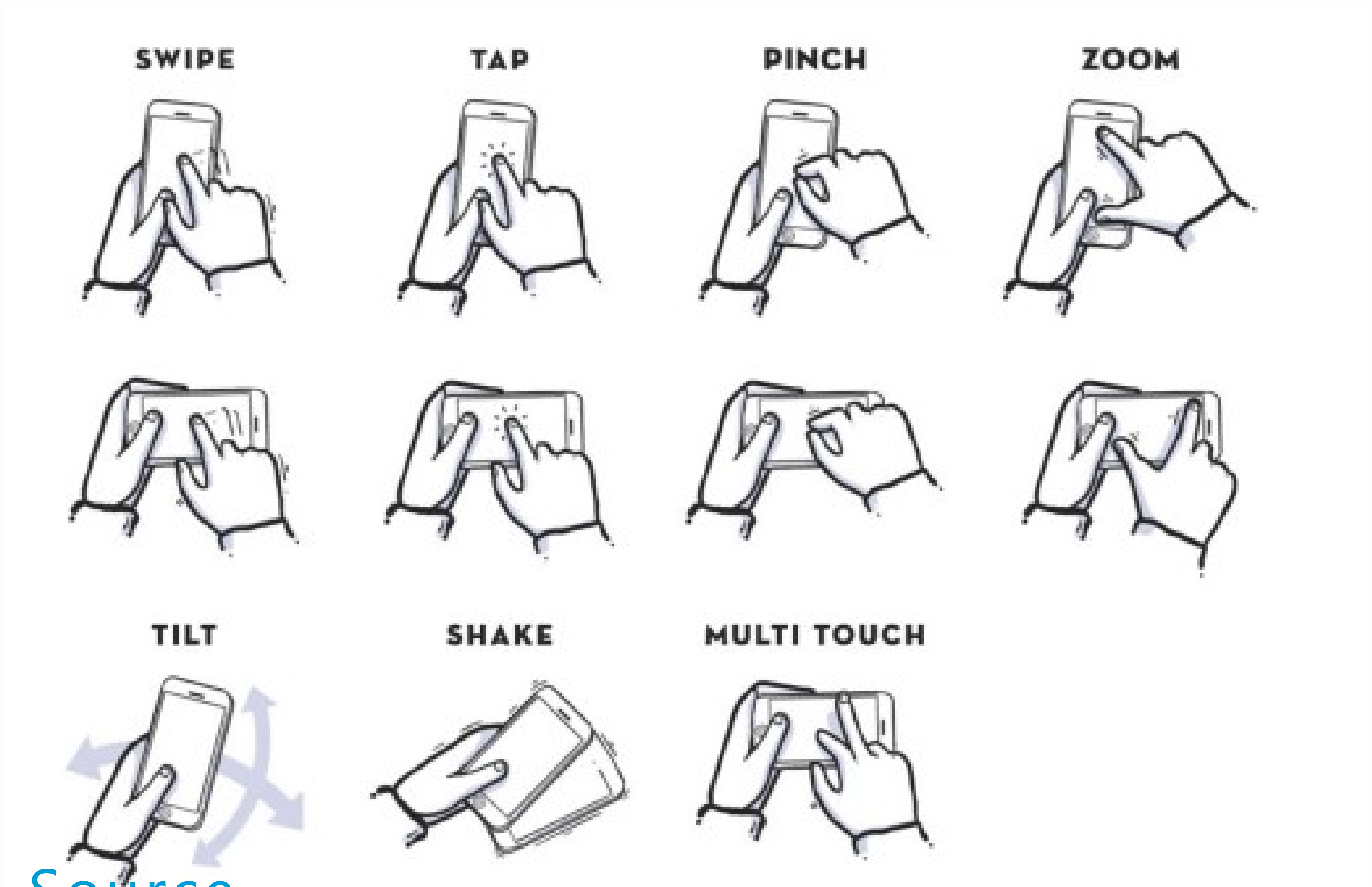


Image Source

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

PanResponder & Animated

See S22 Example

react-native-gesture-handler & react-native-reanimated

See Example

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 (ASO)

What did we learn today?

- How to pass data using React Navigation?
- How to perform "switch" navigation?
- How to store secrets on mobile devices?
- How to do advanced gestures and animations?
- How to use sensors?
- How to deploy our apps?

Questions?