OBJECT
COMPUTING

WEBINAR

React Native Animation

mark@objectcomputing.com

ANIMATED API ...

- Provided by React Native
- Four steps to use
  - import
  - create an animatable value
  - use a style to associate the animatable value with a component
  - write and call a function to trigger

... ANIMATED API ...

OBJECT COMPUTING

- Animatable builtin components include
  - **`View`**, **`ScrollView`**, **`Text`**, and **`Image`**
- Can create custom components that are animatable
  - ex. to animate marginTop like a CSS transition

```
import {Animated} from 'react-native';
const animate = () => {
  Animated.timing(
    new Animated.Value(30); // starting value
    {duration: 1000, toValue: 300} // ending value
  ).start();
};
```

could use current value

could make this an argument to **`animate`** function

```
// In some component ...
  render() {
    return (
      <View style={...}>
        ...
        <Button title="..." onPress={animate} />
        ...
      </View>
    );
  }
```

Are there any CSS properties that cannot be animated?

... ANIMATED API ...

- Can animate expanding a **`TextInput`** when it gains focus
  - using its **`onFocus`** and **`onBlur`** event handling props
- Can create continuous animations with **`Animated.loop`** function
  - one use is activity indicators
  - resets at end of each animation
  - can cancel

... ANIMATED API ...

OBJECT COMPUTING

- What does interpolation do?
  - `Animated.interpolate({inputRange: originalValues, outputRange: goalValue});`
- Easing functions
  - control progression of changes from input to output values

```
const animatedValue = new Animated.Value(value);
const interpolatedValue = animatedValue.interpolate({
  inputRange: [0, 1],
  outputRange: ['0deg', '360deg']
});

// Show this only when loading some resource.
<AnimatedImage
  source={____}
  style={{
    transform: [{rotate: interpolatedValue}]
  }}
/>
```

... ANIMATED API ...

OBJECT COMPUTING

- To run animations of multiple components in parallel
  - `Animated.parallel(values)` returns object with `start` method that must called to start animations
  - values is an array of values returned by `Animated.timing` calls
- To run multiple animations in series
  - `Animated.sequence(values)` returns object with `start` method that must called to start animations
- To run multiple animations whose start times are staggered
  - `Animated.stagger(values)` returns object with `start` method that must called to start animations

specifies start of each by some # of milliseconds more than previous

... ANIMATED API ...

- To reset an animated value so it can be used again
  - `animatedValue.setValue(someValue)`
- To run code when animations complete
  - pass a function to `start` method

... ANIMATED API ...

- Animations are performed in JavaScript thread by default
  - What does this mean since the app is compiled to a native app?
  - Where is the JS?
- Some animations can run in a native UI thread
  - perhaps only animations of non-layout properties
  - to do this, add `useNativeDriver: true` to object passed to `Animated.timing`

... ANIMATED API

- To create a custom animatable component
  - `const AnimatableName = Animated.createAnimatableComponent(ComponentName);`
  - for example, the component could be **`TouchableHighlight`**
  - **TRY THIS!**