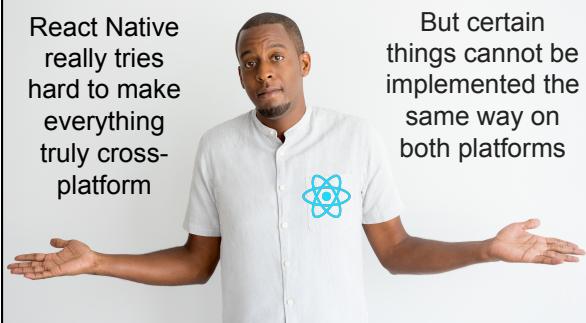


Platform-specific Development

tl;dr

- React Native tries to be 100% cross platform, but that is impossible in some situations (Like DatePicker, for example).
- Besides, we may want there to be differences
- When that happens, we can use
 1. Platform-specific extensions, serving a whole different file
 2. The Platform module which gives us
 - Platform.OS
 - Platform.select()

React Native
really tries
hard to make
everything
truly cross-
platform



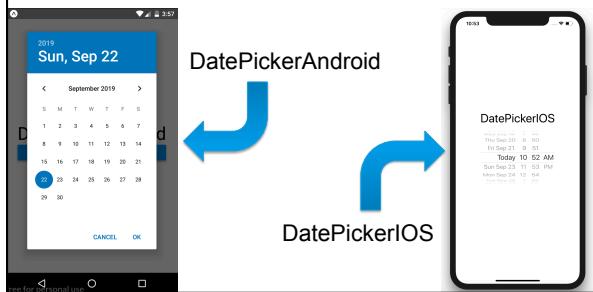
But certain
things cannot be
implemented the
same way on
both platforms

Truly cross-platform development gives us a few choices

-
- A Venn diagram with two overlapping circles. The left circle is orange and labeled "iOS". The right circle is green and labeled "Android". Four arrows point from the text descriptions below to the diagram:
1. Focus on iOS. Let Android suffer (points to the iOS circle)
 2. Focus on Android. Let iOS suffer (points to the Android circle)
 3. Use a limited set of features that are well-supported on both (points to the intersection of the two circles)
 4. Write a ton of code for each platform to strengthen the features on both (points to the intersection of the two circles)

DatePicker*

There are two DatePickers



To use either on the wrong platform
is an error

DatePickerAndroid on iOS



DatePickerIOS is not supported on this platform!

DatePickerIOS on Android

DatePickerIOS is a JSX component

```
<DatePickerIOS
  date* - The initial date
  onDateChange* - Event when they change a date.
    Returns the date chosen.
  maximumDate, minimumDate
  mode - 'date', 'time', or 'dateTime'
  timeZoneOffsetinMinutes - Force a timezone
    instead of looking at our location
/>
*Required
```

Simple DatePickerIOS example

```
Foo.js
function Foo() {
  return (
    <DatePickerIOS date={new Date()} 
      onDateChange={setDate} />
  )
}

function setDate(date) {
  console.log("The date: ", date);
}

}
```

DatePickerAndroid is a JS Object

`open(options)` - Opens the standard Android date picker dialog and returns a Promise

```
options = {
  date - The initial date (defaults to today)
  minDate, maxDate
  mode - 'calendar', 'spinner'
}
```

The Promise callback receives an object

- o action - dateSetAction or dismissedAction
- o year, month (0-11), day (1-31)

Simple DatePickerAndroid example

Foo.js

```
function Foo() {
  return (
    <Button onPress={getDate}
            onDateChange={setDate} />
  )
}

function getDate() {
  DatePickerAndroid.open({date:new Date()})
    .then( ({month,day,year}) =>
      setDate(new Date(year,month,day)));
}

function setDate(date) {
  console.log("The date: ",date);
}
```

Android-only

- DatePickerAndroid
- DrawerLayoutAndroid
- ProgressBarAndroid
- ToolbarAndroid
- ViewPagerAndroid

iOS-only

- DatePickerIOS
- MaskedViewIOS
- NavigatorIOS
- PickerIOS
- ProgressViewIOS
- SegmentedControlIOS
- SnapshotViewIOS
- TabBarIOS

How to handle Platform-specific code

Two main ways to implement Platform Specific code

- 1.Platform-specific file extensions
- 2.The Platform module

Platform Specific File Extensions

- If you ...


```
import MyButton from './MyButton';
```
- And there is no MyButton.js.
- But there are ...


```
MyButton.ios.js
```

```
MyButton.android.js
```
- Then the compiler will pick the proper one for each platform when compiling

Platform.OS is set to a string, 'ios' or 'android'

```
function CustomButton() {
  const MyButton = Platform.OS === 'ios' ?
    TouchableHighlight : TouchableWithoutFeedback;
  return (
    <MyButton onPress={console.log}>
      <Text>Press me</Text>
    </MyButton>
  );
}
```

Platform.select() returns a different object depending on the platform

```
function CustomButton() {
  const MyButton = Platform.select({
    ios: TouchableHighlight,
    android: TouchableWithoutFeedback,
  });
  return (
    <MyButton onPress={doIt}>
      <Text>Press me</Text>
    </MyButton>
  );
}
```

tl;dr

- React Native tries to be 100% cross platform, but that is impossible in some situations (Like DatePicker, for example).
- Besides, we may want there to be differences
- When that happens, we can use
 1. Platform-specific extensions, serving a whole different file
 2. The Platform module which gives us
 - Platform.OS
 - Platform.select()
