React Native

Platform specific code

Native Components

Methods

- React Native provides two ways to easily organize your code and separate it by platform:
  - Using the Platform module.
  - Using platform-specific file extensions.
- Certain components may have properties that work on one platform only.
  - All of these props are annotated with @platform and have a small badge next to them on the website.

Platform Module

- React Native provides a module that detects the platform in which the app is running.
- You can use the detection logic to implement platform-specific code.
- Use this option when only small parts of a component are platform-specific.

Platform Module

```
import {Platform, StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  height: Platform.OS === 'ios' ? 200 : 100,
});
```

Platform.OS will be ios when running on iOS and android when running on Android.

Platform Module

- There is also a Platform.select method available,
  - given an object containing Platform.OS as keys,
  - returns the value for the platform you are currently running on.

Platform Module

```
import {Platform, StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  container: {
    flex: 1,
    ...Platform.select({
      ios: {
        backgroundColor: 'red',
      },
      android: {
        backgroundColor: 'blue',
      },
    }),
  },
});
```

This will result in a container having flex: 1 on both platforms,
a red background color on iOS,
and a blue background color on Android.

The ellipses (…) are part of the React Native meta language. There is basically a preprocessor that will change this construct into either the line "backgorundColor:'red'" or "backgroundColor:'blue'" (depending on the platform) *before* the code is compiled into a native version.

Platform Module

- Since Platform.select accepts any value, you can also use it to return platform specific component

```
const Component = Platform.select({
   ios: () => require('ComponentIOS'),
   android: () => require('ComponentAndroid'),
})();

<Component />;
```

Platform Module

- On Android, the Platform module can also be used to detect the version of the Android Platform in which the app is running:

```
import {Platform} from 'react-native';

if (Platform.Version === 25) {
  console.log('Running on Nougat!');
}
```

Platform Module

- On iOS, the Version is a result of -[UIDevice systemVersion], which is a string with the current version of the operating system.
- An example of the system version is "10.3".
- For example, to detect the major version number on iOS:

```
import {Platform} from 'react-native';

const majorVersionIOS = parseInt(Platform.Version, 10);
if (majorVersionIOS <= 9) {
  console.log('Work around a change in behavior');
}
```

Example (complete)

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


export default class App extends React.Component {
  render() {
    return (
  <View style={styles.header}>
    <Text style={styles.text}>I am Header</Text>
  </View>
    );
  }
}
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
```

```
header: {
  height: Platform.OS === 'android' ? 76 : 100,
  marginTop: Platform.OS === 'ios' ? 0 : 24,
  ...Platform.select({
    ios: { backgroundColor: '#f00', paddingTop: 24},
    android: { backgroundColor: '#00f'}
  }),
  alignItems: 'center',
  justifyContent: 'center'
},
text: {
  color: '#fff',
  fontSize: 24
}
```

Platform-specific extensions

- When your platform-specific code is more complex, you should consider splitting the code out into separate files.
- React Native will detect when a file has a .ios. or .android. extension and load the relevant platform file when required from other components.

Platform-specific extensions

- For example, say you have the following files in your project:

```
BigButton.ios.js
BigButton.android.js
```

- You can then require the component as follows:

```
const BigButton = require('./BigButton');
```

- React Native will automatically pick up the right file based on the running platform.

Example: separate components

```
import React from 'react';
import { StyleSheet, Text, View, Platform } from 'react-native';
import DateComp from './dateComp';
export default class App extends React.Component {
  constructor(props) {
   super(props);
  }
 render() {
   return (
     <View style={styles.container}>
       <Text style={{fontSize:20}}>Date</Text>
       <DateComp />
     </View>
   );   }  }
```

dateComp.ios.js

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


class Home extends Component {
  constructor(props){
    super(props);
    this.state = { myState: this.props.origText, isOrig: true};
   }
```

dateComp.ios.js

```
updateState = () => {
    if (this.state.isOrig){
        this.setState({ myState: 'ask what you can do for your country', isOrig: false })
    }
    else{
        this.setState({ myState: this.props.origText, isOrig: true })
    };
};
```

dateComp.ios.js

```
render() {
  {/* Note that pic is rebound every time render() is called */}
  let pic = {
    uri: 'https://upload.wikimedia.org/wikipedia/commons/d/de/Bananavarieties.jpg'
  };
  return (
    <View style={{alignItems: 'center', marginTop: 100}}>
      <Text style={{color:"red", fontSize:20}} onPress = {this.updateState}>
        {this.state.myState}
      </Text>
      <Image source={pic} style={{width: 193, height: 110}}/>
    </View>
  );    }   }
```

dateComp.ios.js

```
export default class StateApp extends Component {
  render() {
    return (
       <Home origText='Ask not what your country can do for you'/>
    );
  }
}
```

dateComp.android.js

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


class Home extends Component {
  constructor(props){
    super(props);
    this.state = { myState: this.props.origText, isOrig: true};
  }
```

dateComp.android.js

```
updateState = () => {
    if (this.state.isOrig){
        this.setState({ myState: 'ask what you can do for your country', isOrig: false })
    }
    else{
        this.setState({ myState: this.props.origText, isOrig: true })
    };
};
```

dateComp.android.js

```
render() {
  {/* Note that pic is rebound every time render() is called */}
  let pic = {
    uri: 'https://upload.wikimedia.org/wikipedia/commons/d/de/Bananavarieties.jpg'
  };
  return (
    <View style={{alignItems: 'center', marginTop: 100}}>
      <Text style={{color:”blue", fontSize:20}} onPress = {this.updateState}>
        {this.state.myState}
      </Text>
      <Image source={pic} style={{width: 193, height: 110}}/>
    </View>
  );    }   }
```

dateComp.Android.js

```
export default class StateApp extends Component {
  render() {
    return (
       <Home origText='Ask not what your country can do for you'/>
    );
  }
}
```

What is a platform specific component?

- See

https://facebook.github.io/react-native/docs/components-and-apis#ios-components-and-apis

Example: iOS datePicker

```
import React, { Component } from 'react'
import {
  DatePickerIOS,
  View,
  StyleSheet,
} from 'react-native'
```

Example: iOS datePicker

```
export default class App extends Component {
  constructor(props) {
    super(props);
    this.state = { chosenDate: new Date() };


    this.setDate = this.setDate.bind(this);
  }


  setDate(newDate) {
    this.setState({chosenDate: newDate})
  }
```

Example: iOS datePicker

```
 render() {
   return (
     <View style={styles.container}>
       <DatePickerIOS
         date={this.state.chosenDate}
         onDateChange={this.setDate}
       />
     </View>
   )
 }
}
```

Example: iOS datePicker

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center'
  },
})
```