**CS624 Full Stack Development – Mobile App**

**HOS06A: React Native Styling II**

Revised by Sam Chung on April 26, 2023

Revised by Clark Ngo January 30, 2025

Reviewed by Naveena Moddu May 06, 2025

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Introduction:**

To create an engaging mobile application using Reactive Native, understanding the fundamentals of component styling is crucial. In this hands-on session (HOS), we will briefly introduce concepts with a few examples and then walk through the process of styling a real object - a Profile Card.

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

#### **Readings and Examples:**

* Visit the [CS624 Repository for Examples.](https://github.com/cityuseattle/cs624-examples)
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.
* Dabit, N. (2019). [React Native in Action](https://learning.oreilly.com/library/view/react-native-in/9781617294051/). Manning Publications. (ISBN 9781617294051)
  + Chapter 5: *Styling In Depth*

**Learning Outcomes**

* Section 1: Accessing GitHub Codespaces
* Section 2: Creating a mobile app
* Section 3: Adding drop shadows to the Profile Card
* Section 4: Scaling Profile Card from full size to thumbnail
* Section 5: Pushing your work to GitHub

**Section 1: Accessing GitHub Codespaces**

As we learned in the previous modules, access the GitHub Codespaces from the repository that you have created from Module05.

**Section 2: Creating a mobile app**

The styling of React Native components was covered in Module05. It demonstrated how to style View and Text components, which you will likely use daily, and which determine how a component looks. As the discussion continues, this Module delves deeper into platform-specific styles, drop shadows, modifying components with operations like translation, rotation, scaling, and skewing, and dynamically arranging components with flexbox.

You'll continue adding additional features to the “ProfileCards” example from the previous chapter by utilizing some of your newly learned styling strategies.

Follow the steps outlined in the previous modules and Setup React Native Environment with Expo Go. Copy the application “Styling” from the previous module **PE04 Part-1** to **Module06** folder.

In HOS05, you practiced the following source code.

* Listing 4.7 Initial framework for the Profile Card component
* Listing 4.10 Incorporating border properties into the Profile Card
* Listing 4.14 Modifying Profile Card styles to fix the layout

In PE04-ProfileCards/part01, you used the following source codes.

* Listing 4.15 Adding text to the Profile Card
* Listing 4.17 Setting font styles for Text elements in the Profile Card
* Listing 4.20 Completed Profile Card example

Launch a new terminal window in the Codespaces and then navigate to Styling folder using below command

>> cd Styling

Follow the steps mentioned at the end of this document to launch the app in your mobile device using “Expo Go.” If everything works, you will see the application working as shown below.

A blue card with a person in a circle

Description automatically generated

**Section 3: Adding drop shadows to components**

**Platform-specific sizes and styles**

The beauty of React Native is that you can use JavaScript to make apps that work on iOS and Android simultaneously. You already know how to select fonts that are only available on iOS or Android by using the utility function “Platform.select.” When defining styles in React Native, you use the logical concept of a pixel, a point on iOS, and a DP on Android.

**Creating drop shadows with** **“ShadowPropTypesIOS” and Elevation**

Please go through section 5.1.2 in the textbook to understand how “ShadowPropTypesIOS” and Elevation work on IOS and Android in depth.

Returning to the Profile Card example from the last module, let's add some drop shadows that will look fantastic on iOS but not so fantastic on Android. Both the circular image container and the full Profile Card container will receive a drop shadow.

Update the code in “App.js” to match the following.

Check Listing 5.1 Adding drop shadows to the Profile Card.

A picture containing calendar

Description automatically generated

Update “cardContainer” and “cardImageContainer” objects under styles as shown below,

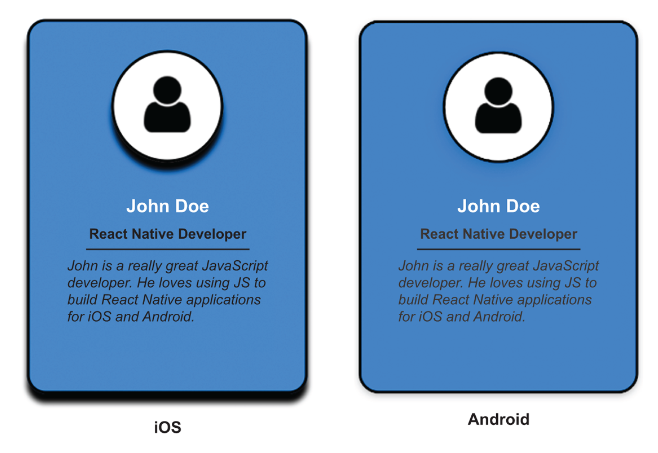
Text

Description automatically generated Timeline

Description automatically generated with medium confidence

Capture the screenshot of the complete simulator and save it in the Module06 folder.

The Profile card in the simulator while using iOS, Android, and Web should show something like the one below



A blue card with a person in a circle

Description automatically generated

**Web**

**Section 4: Scaling a Profile Card from full size to thumbnail**

Install the “immutability-helper” package using the following command in the terminal. The immutability helper function update lets you update a specific piece of the component’s state.

**$npm install****immutability-helper**

From module 6 of your example repository, copy the source code in the “UpdatedLisiting0503-ProfileCard” file.

This code displays a profile card, which includes an image, name, occupation, and description. The app uses the “TouchableHighlight” component to make the profile card interactive, and the “PropTypes” library to ensure that the correct data types are used for the card's properties. The update from “immutability-helper” library is used to modify the app's state, which allows the “handleProfileCardPress” function to toggle the visibility of a thumbnail image for profile card.

\*\*You may have to change the path of the user image on line 22.



Save the changes and see the output in the simulator. You should see a similar screen on your web platform. When you click or tap on the profile card, it collapses, as shown below.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Capture all the screenshots of the simulator and save them in the Module06 folder.

**Steps to verify the changes using the Simulator:**

1. Open the terminal in the GitHub Codespaces environment.
2. Navigate to the app folder using the following command   
   >>cd Styling
3. Type “**npx expo start --tunnel”** and press Enter to start the expo development server.
4. Wait for the development server to load and show the QR code.
5. Open the "Expo Go" app from your mobile device.
6. Scan the QR code shown in the terminal with the "Expo Go" or Camera (iOS) app.
7. Wait for the app to load on the mobile device.
8. If you are connected and want to reload, press r in the terminal.

**Section 5: Pushing your work to GitHub**

* 1. Go to Source Control on your GitHub Codespaces and observe the pending changes.

Graphical user interface, text, application

Description automatically generated

* 1. Type the Message for your changes in the Message box on the top. For example,” **Submission for Module06 – Your Name**.”
  2. Click on the dropdown beside the commit button and select **Commit & Push** to update the changes to your repository main branch.
  3. Select **Yes** when prompted.

Graphical user interface, application

Description automatically generated