

Unit-3: Android Studio for Android Software Development

3

Unit Structure

- 3.0 Learning Objectives
- 3.1 Introduction
- 3.2 Features of Android Studio
- 3.3 App Workflow
- 3.4 Android Virtual Devices (AVD)
- 3.5 Using Hardware Device to test Application
- 3.6 Android Studio IDE Components
- 3.7 Android Studio Code Editor Customization
- 3.8 Coding Best Practices
- 3.9 Let us sum up
- 3.10 Check your Progress: Possible Answers
- 3.11 Further Reading
- 3.12 Assignment
- 3.13 Activities

3.0 Learning Objectives

After studying this unit student should be able to:

- List features of Android Studio
- Understand App Workflow
- Define Android Virtual Devices (AVD)
- Use Hardware Device to test an Application
- Use Android Studio IDE Components
- Customize the Android Studio Code Editor
- Learn Coding Best Practices

3.1 Introduction

Android Studio is the official Integrated Development Environment (IDE) for Android app development. Android Studio offers many features that enhance your productivity when building Android apps. In this unit we will explore in great details about Android Studio.

3.2 Features of Android Studio

Android Studio has following salient features for android application development:

Intelligent code editor: Android Studio provides an intelligent code editor capable of advanced code completion, refactoring, and code analysis. The powerful code editor helps you be a more productive Android app developer.

Project Wizard: New project wizards make it easier than ever to start a new project. Start projects using template code for patterns such as navigation drawer and view pagers, and even import Google code samples from GitHub.

Multi-screen app development: Build apps for Android phones, tablets, Android Wear, Android TV, Android Auto and Google Glass. With the new Android Project

View and module support in Android Studio, it's easier to manage app projects and resources.

Virtual devices for all shapes and sizes: Android Studio comes pre-configured with an optimized emulator image. The updated and streamlined Virtual Device Manager provides pre-defined device profiles for common Android devices.

Android builds evolved, with Gradle: Create multiple APKs for your Android app with different features using the same project.

To develop apps for Android, you use a set of tools that are included in Android Studio. In addition to using the tools from Android Studio, you can also access most of the SDK tools from the command line.

3.3 App Workflow

The basic steps for developing applications encompass four development phases, which include:

- **Environment Setup:** During this phase you install and set up your development environment. You also create Android Virtual Devices (AVDs) and connect hardware devices on which you can install your applications.
- **Project Setup and Development:** During this phase you set up and develop your Android Studio project and application modules, which contain all of the source code and resource files for your application.
- **Building, Debugging and Testing:** During this phase you build your project into a debuggable .apk package(s) that you can install and run on the emulator or an Android-powered device. Android Studio uses a build system based on Gradle that provides flexibility, customized build variants, dependency resolution, and much more. If you're using another IDE, you can build your project using Gradle and install it on a device using adb.

Next, with Android Studio you debug your application using the Android Debug Monitor and device log messages along with the IntelliJ IDEA intelligent coding features. You can also use a JDWP-compliant debugger along with the debugging and logging tools that are provided with the Android SDK.

Last, you test your application using various Android SDK testing tools.

- **Publishing:** During this phase you configure and build your application for release and distribute your application to users.

3.4 Android Virtual Devices (AVD)

An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device by defining hardware and software options to be emulated by the Android Emulator. An AVD consists of:

A hardware profile: Defines the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.

A mapping to a system image: You can define what version of the Android platform will run on the virtual device. You can choose a version of the standard Android platform or the system image packaged with an SDK add-on.

Other options: You can specify the emulator skin you want to use with the AVD, which lets you control the screen dimensions, appearance, and so on. You can also specify the emulated SD card to use with the AVD.

A dedicated storage area on your development machine: the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area.

The easiest way to create an AVD is to use the graphical AVD Manager. You can also start the AVD Manager from the command line by calling the android tool with the avd options, from the <sdk>/tools/ directory.

You can also create AVDs on the command line by passing the android tool options.

You can create as many AVDs as you need, based on the types of device you want to model. To thoroughly test your application, you should create an AVD for each general device configuration (for example, different screen sizes and platform versions) with which your application is compatible and test your application on each one. Keep these points in mind when you are selecting a system image target for your AVD:

- The API Level of the target is important, because your application will not be able to run on a system image whose API Level is less than that required by your application, as specified in the `minSdkVersion` attribute of the application's manifest file.
- You should create at least one AVD that uses a target whose API Level is greater than that required by your application, because it allows you to test the forward-compatibility of your application. Forward-compatibility testing ensures that, when users who have downloaded your application receive a system update, your application will continue to function normally.
- If your application declares a `uses-library` element in its manifest file, the application can only run on a system image in which that external library is present. If you want to run your application on an emulator, create an AVD that includes the required library. Usually, you must create such an AVD using an Add-on component for the AVD's platform.

Check Your Progress -1

- a) How many AVDs will you be able to create in Android Studio?
(A) One (B) Two (C) Three (D) As many as you want
- b) During _____ phase you build your project into a debuggable .apk package(s) that you can install and run on the emulator or an Android-powered device.
(A) Environment Setup (B) Development (C) Building (D) Publishing
- c) An AVD consists of
(A) Hardware Profile (B) A mapping to a System Image
(C) Dedicated Storage Area (D) All of these

3.5 Using Hardware Device to test Application

When building a mobile application, it's important that you always test your application on a real device before releasing it to users.

You can use any Android-powered device as an environment for running, debugging, and testing your applications. The tools included in the SDK make it easy to install and run your application on the device each time you compile. You can install your application on the device directly from Android Studio or from the command line with ADB.

3.6 Android Studio IDE Components

Android Studio has just created an application project and opened the main window as shown below.

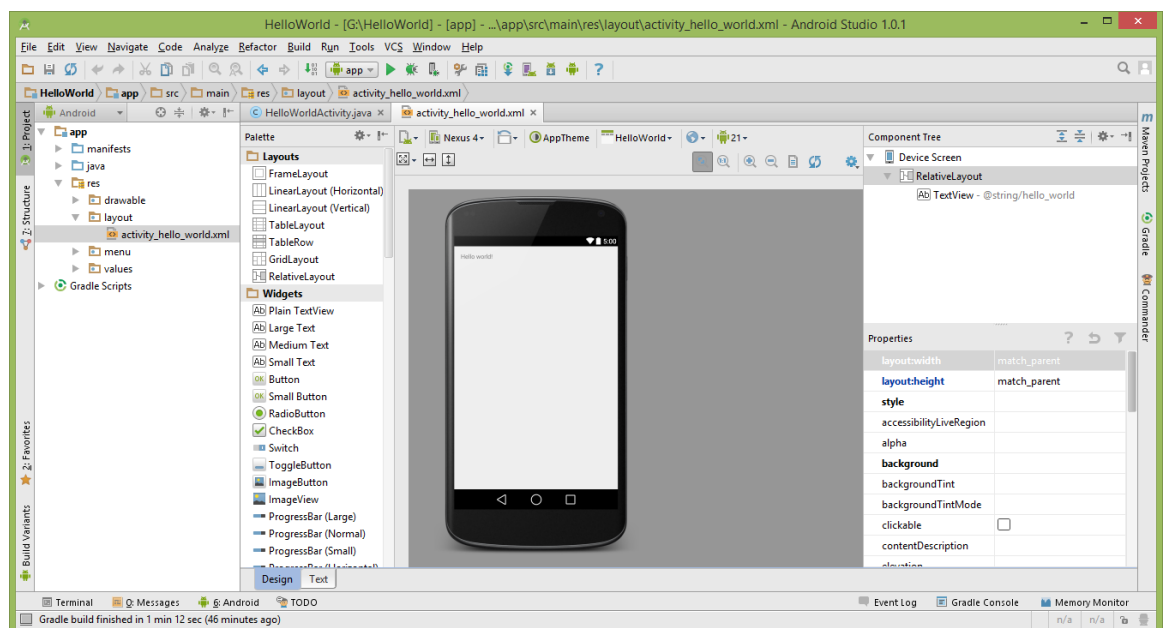


Figure-24: Android studio IDE

Project Tool Window

The newly created project and references to associated files are listed in the Project tool window located on the left hand side of the main project window. The user interface design for our activity is stored in a file named `activity_hello_world.xml` which can be located using the Project tool window as shown below in Figure

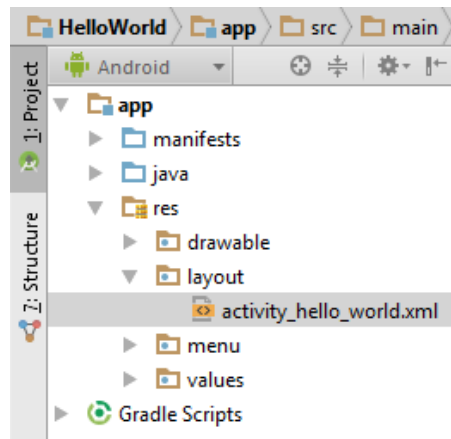



Figure-25: Android studio IDE

Double click on the file to load it into the User Interface Designer tool which will appear in the center panel of the Android Studio main window as shown below:

Designer Window

This is where you design your user interface. In the top of the Designer window is a menu set to Nexus 4 device which is shown in the Designer panel.

To change the orientation of the device representation between landscape and portrait simply use the drop down menu immediately to the right of the device selection menu showing the icon .

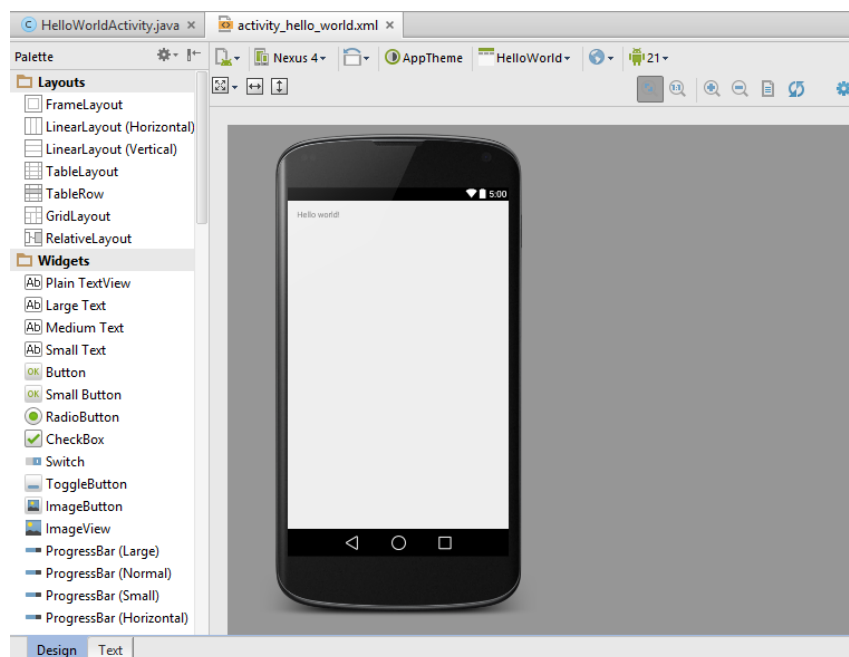


Figure-26: Designer Window

Pellet

On left hand side of the panel is a palette containing different categories of user interface components that may be used to construct a user interface, such as buttons, labels and text fields. Android supports a variety of different layouts that provide different levels of control over how visual user interface components are positioned and managed on the screen.

Component Tree Panel

Component Tree panel is by default located in the upper right hand corner of the Designer panel and is shown in Figure and shows layout used for user interface.

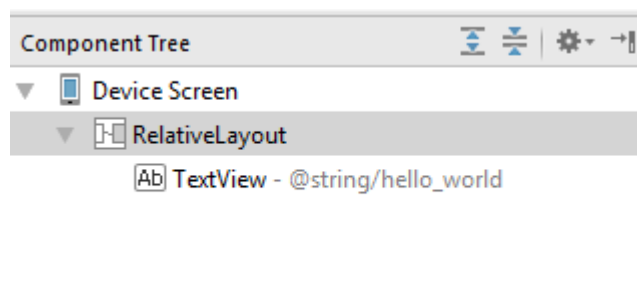
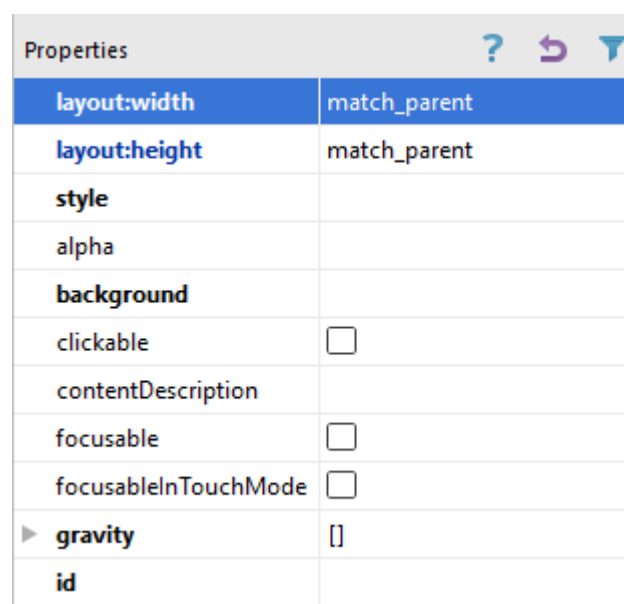


Figure-27: Component Tree Panel

Property Window

Property Window allows setting different properties of selected component.



Properties	
layout:width	match_parent
layout:height	match_parent
style	
alpha	
background	
clickable	<input type="checkbox"/>
contentDescription	
focusable	<input type="checkbox"/>
focusableInTouchMode	<input type="checkbox"/>
gravity	[]
id	

Figure-28: Property Window

XML Editing Panel

We can modify user interface by modifying the `activity_hello_world.xml` using UI Designer tool but we can also modify design by editing XML file also. At the bottom of the Designer panel are two tabs labeled Design and Text respectively. To switch to the XML view simply select the Text tab as shown in Figure. At the right hand side of the XML editing panel is the Preview panel and shows the current visual state of the layout.

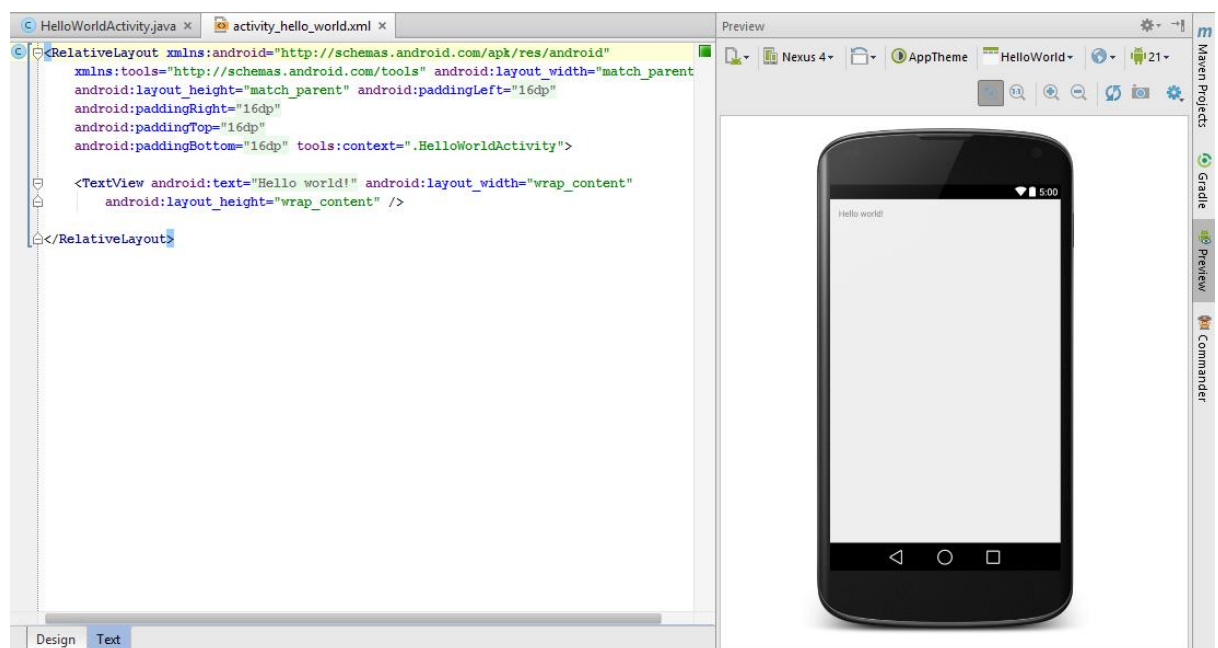


Figure-29: XML Editing Panel

Previewing the Layout

In above figure layout has been previewed of the Nexus 4 device. The layout can be tested for other devices by making selections from the device menu in the toolbar across the top edge of the Designer panel. We can also preview screen size for all currently configured device as shown in figure.

Check Your Progress – 2

- Component Tree panel is by default located in the upper left hand corner of the Designer panel (True/False)
- Property Window allows setting different properties of selected component. (True/False)

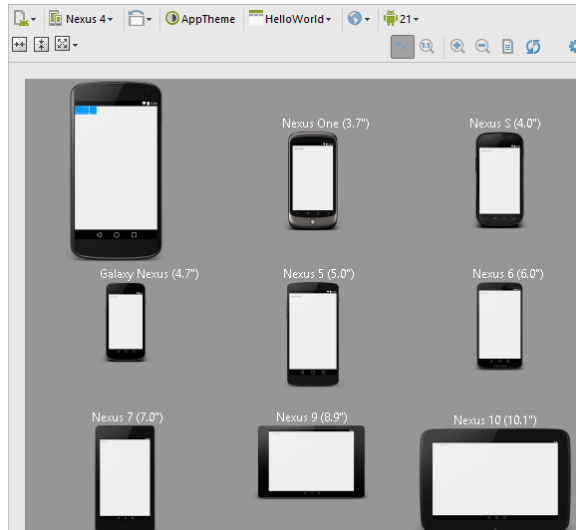


Figure-30: Previewing the Layout

3.7 Android Studio Code Editor Customization

We customize code editor for font, displaying quick help when mouse moves over code.

Font Customization:

From File menu select settings option following dialog will open.

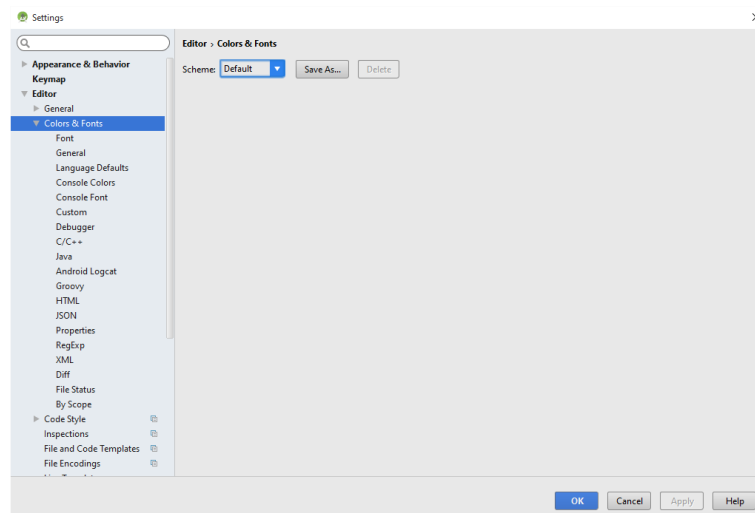


Figure-31

Select Colors & Fonts option. In scheme Default scheme is displayed. Click **“Save As...”** Button and give new name as **“My Settings”** to scheme. Now select font option as shown below and customize font as per your requirement.

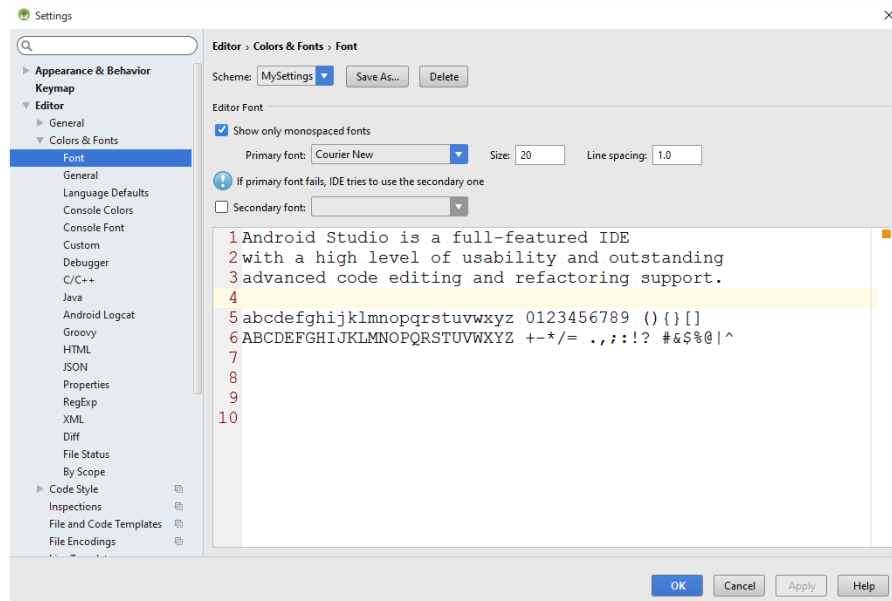


Figure-32

Show quick doc on mouse move

In android studio code editor, when you move your mouse over a method, class or interface, a documentation window would appear with a description of that programming element. This feature is by default disabled in android studio. This feature can be enabled in android studio as follows.

From file menu select settings option following dialog box appears. Select General option from list and scroll down to checkbox highlighted with red rectangle in below figure. Then press OK.

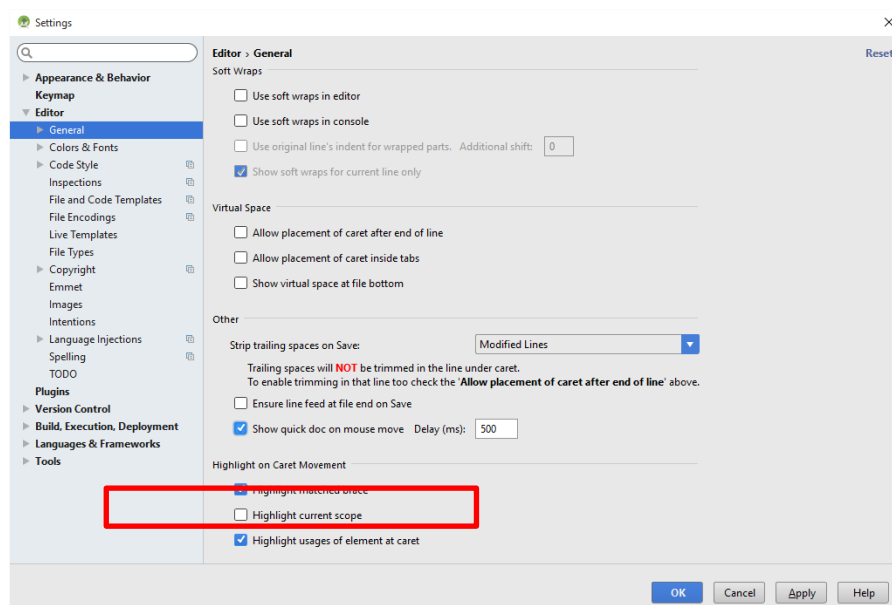


Figure-33

3.8 Coding Best Practices

Following are some of the code editing practices you should follow when creating Android Studio apps.

Code Practice	Description
Alt + Enter key	For quick fixes to coding errors such as missing imports, variable assignments, missing references, etc. You can use Alt + Enter key to fix errors for the most probable solution.
Ctrl + D	The Ctrl + D key are used for quickly duplicating code lines or fragments. Instead of copy and paste, you can select the desired line or fragment and enter this key.
Navigate menu	Use the Navigate menu to jump directly to the class of a method or field name without having to search through individual classes.
Code folding	This allows you to selectively hide and display sections of the code for readability. For example, resource expressions or code for a nested class can be folded or hidden in to one line to make the outer class structure easier to read. The inner class can be later expanded for updates.
Image and color preview	When referencing images and icons in your code, a preview of the image or icon appears (in actual size at different densities) in the code margin to help you verify the image or icon reference. Pressing F1 with the preview image or icon selected displays resource asset details, such as the dp settings.
Quick documentation	You can inspect theme attributes using View > Quick Documentation (Ctrl+Q), If you invoke View > Quick Documentation on the theme attribute ?android:textAppearanceLarge, you will see the theme inheritance hierarchy and resolved values for the various attributes that are pulled in.

Table-2

The following tables list keyboard shortcuts for common operations.

Action	Android Studio Key Command
Command look-up (autocomplete command name)	CTRL + SHIFT + A
Project quick fix	ALT + ENTER
Reformat code	CTRL + ALT + L (Win) OPTION + CMD + L (Mac)
Show docs for selected API	CTRL + Q (Win) F1 (Mac)
Show parameters for selected method	CTRL + P
Generate method	ALT + Insert (Win) CMD + N (Mac)
Jump to source	F4 (Win) CMD + down-arrow (Mac)
Delete line	CTRL + Y (Win) CMD + Backspace (Mac)
Search by symbol name	CTRL + ALT + SHIFT + N (Win) OPTION + CMD + O (Mac)

Table-3

Following table lists project and editor key commands:

Action	Android Studio Key Command
Build	CTRL + F9 (Win), CMD + F9 (Mac)
Build and run	SHIFT + F10 (Win), CTRL + R (Mac)
Toggle project visibility	ALT + 1 (Win), CMD + 1 (Mac)
Navigate open tabs	ALT + left-arrow; ALT + right-arrow (Win) CTRL + left-arrow; CTRL + right-arrow (Mac)

Table-4

You can change these shortcuts from file menu settings option as shown below.

Check your Progress – 3

- The short cut key to access quick documentation is F1 (True/False)
- The Ctrl+D are used for quickly duplicating code lines or fragments. (True/False)
- You can use Ctrl+Enter to fix errors for the most probable solution (True/False)

d) We can customize the code editor for font, displaying quick help when mouse moves over code. (True/False)

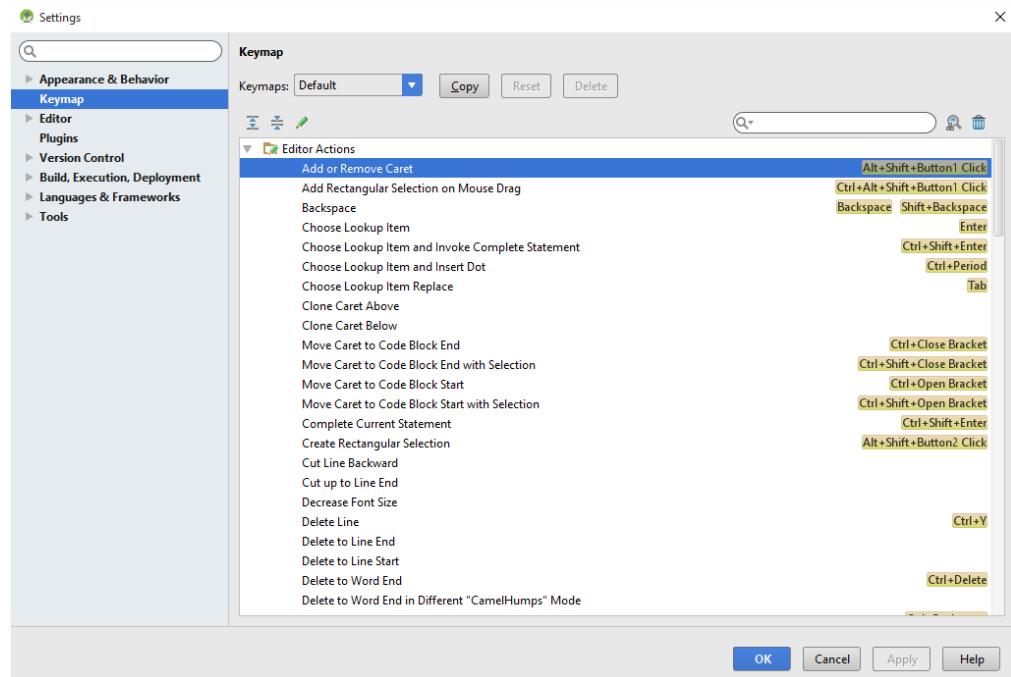


Figure-34

3.9 Let us sum up

Before starting the development of Android based applications, the major step is to set up a suitable development environment. This consists of the Java Development Kit (JDK), Android SDKs, and Android Studio IDE. In this chapter, we have covered the steps necessary to install these packages on Windows operating system and how to create Hello World application project using Android Studio. We have also discussed android studio IDE component and project structure and how to do code editor customization and discuss some of the best practices for coding.

3.10 Check your Progress: Possible Answers

- 1-a) D 1-b) C 1-c) D
2-a) False 2-b) True
3-a) False 3-b) True 3-c) False 3-d) True

3.11 Further Reading

- <https://developer.android.com/studio/intro>

3.12 Assignment

- Explain different components of Android Studio

3.13 Activities

- Create two AVDs with different configuration in Android Studio
- Customize Android Studio for font and show quick doc on mouse move