Lesson 1.4: Learn to help yourself

Introduction

What you should already know

You should be able to:

- Understand the basic workflow of Android Studio.
- Create an app from scratch using the Empty Activity template.
- Use the layout editor.

What you'll learn

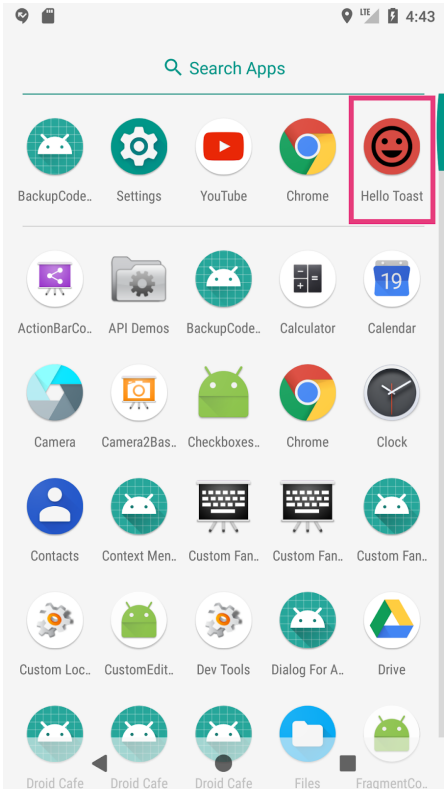
- Where to find developer information and resources.
- How to add a launcher icon to your app.
- How to look for help when you're developing your Android apps.

What you'll do

- Explore some of the many resources available to Android developers of all levels.
- Add a launcher icon for your app.

App overview

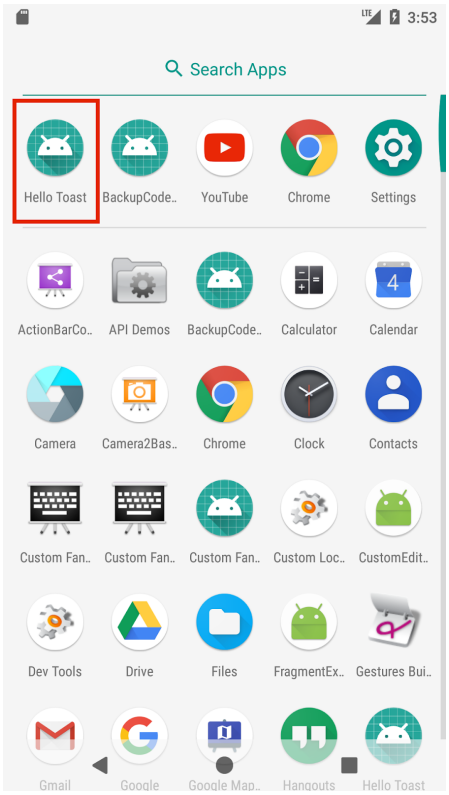
You will add a launcher icon to the HelloToast app you created previously or to a new app.



Task 1: Change the launcher icon

Each new app you create with Android Studio starts with a standard launcher icon that represents the app. The launcher icon appears in the Google Play store listing. When users search the Google Play store, the icon for your app appears in the search results.

When a user has installed the app, the launcher icon appears on the device in various places including the home screen and Search Apps screen. For example, the HelloToast app appears in the Search Apps screen of the emulator with the standard icon for new app projects, as shown below.



Changing the launcher icon is a simple step-by-step process that introduces you to Android Studio's image asset features. In this task you also learn more about accessing the official Android documentation.

1.1 Explore the official Android documentation

You can find the official Android developer documentation at developer.android.com.

This documentation contains a wealth of information that is kept current by Google.

16. Go to developer.android.com/design/.

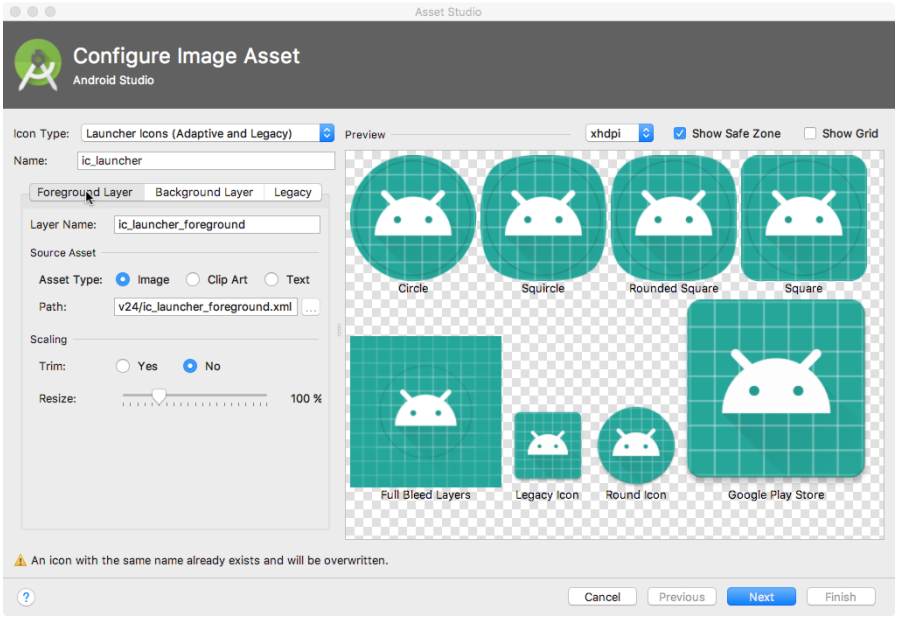
This section is about Material Design, which is a conceptual design philosophy that outlines how apps should look and work on mobile devices. Navigate the links to learn more about Material Design. For example, visit the Style section to learn more about the use of color and other topics.

17. Go to developer.android.com/docs/ to find API information, reference documentation, tutorials, tool guides, and code samples.
18. Go to developer.android.com/distribute/ to find information about putting an app on Google Play, Google's digital distribution system for apps developed with the Android SDK. Use the Google Play Console to grow your user base and start earning money.

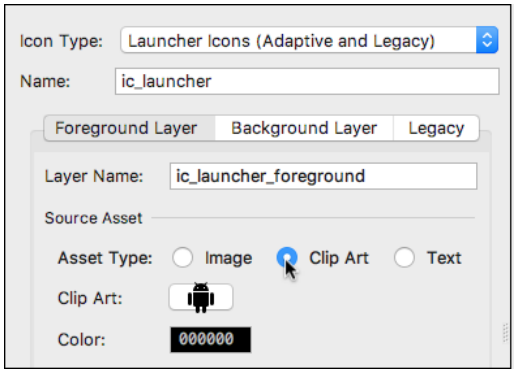
1.2 Add an image asset for the launcher icon

To add a clip-art image as the launcher icon, follow these steps:

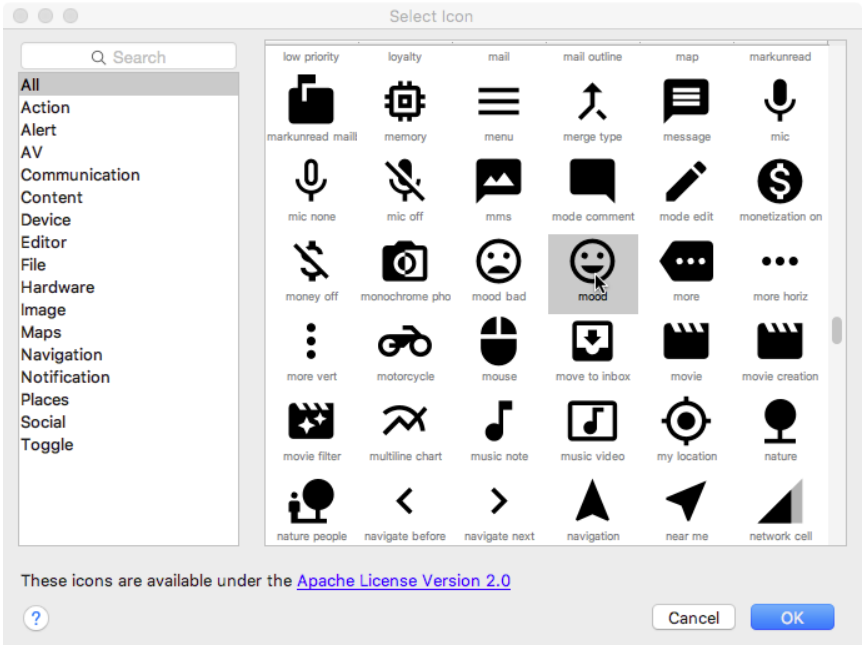
1. Open the HelloToast app project from the previous lesson on using the layout editor, or create a new app project.
2. In the **Project > Android** pane, **right-click** (or **Control-click**) the **res** folder and select **New > Image Asset**. The Configure Image Asset window appears.



3. In the **Icon Type** field, select **Launcher Icons (Adaptive & Legacy)** if it's not already selected.
4. Click the **Foreground Layer** tab, select **Clip Art** for the **Asset Type**.

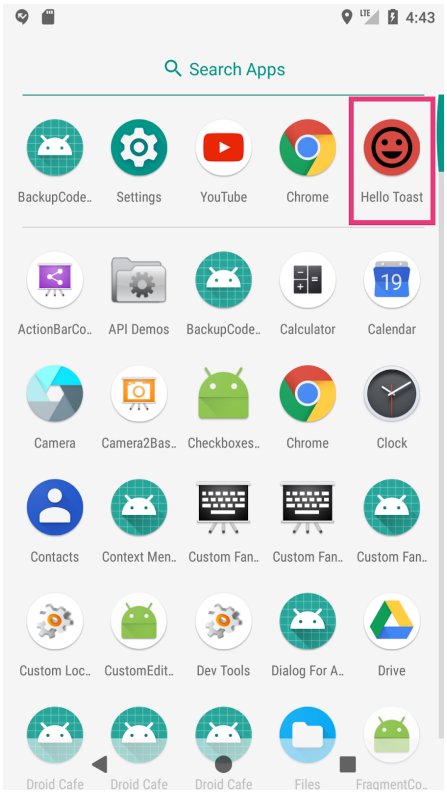


5. Click the icon in the **Clip Art** field. Icons appear from the material design icon set.
6. Browse the Select Icon window, choose an appropriate icon (such as the mood icon to suggest a good mood), and then click **OK**.



7. Click the **Background Layer** tab, choose **Color** as the **Asset Type**, and then click the color chip to select a color to use as the background layer.
8. Click the **Legacy** tab and review the default settings. Confirm that you want to generate legacy, round, and Google Play Store icons. Click **Next** when finished.
9. Run the app.

Android Studio automatically adds the launcher images to the **mipmap** directories for the different densities. As a result, the app launch icon changes to the new icon after you run the app, as shown below.



Tip: See Launcher Icons to learn more about how to design effective launcher icons.

Task 2: Use project templates

Android Studio provides templates for common and recommended app and activity designs. Using built-in templates saves time, and helps you follow design best practices.

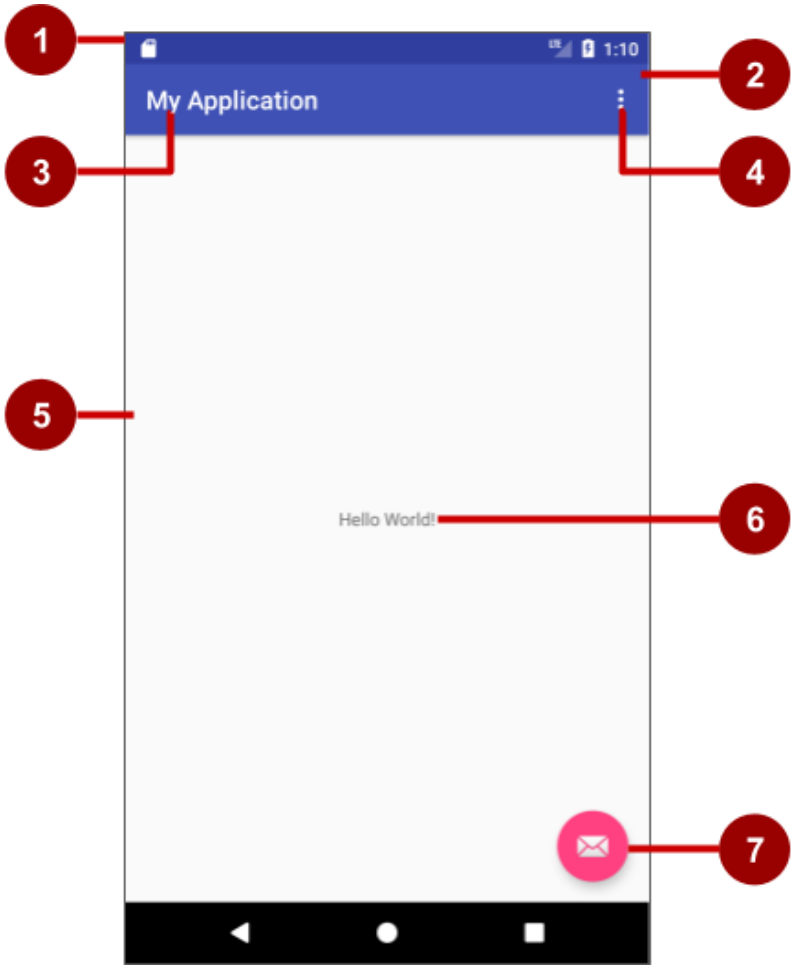
Each template incorporates a skeleton activity and user interface. You've already used the Empty Activity template. The Basic Activity template has more features and incorporates recommended app features, such as the options menu that appears in the app bar.

2.1 Explore the Basic Activity architecture

The Basic Activity template is a versatile template provided by Android Studio to assist you in jump-starting your app development.

1. In Android Studio, create a new project with the Basic Activity template.
2. Build and run the app.
3. Identify the labeled parts in the figure and table below. Find their equivalents on your device or emulator screen. Inspect the corresponding Java code and XML files described in the table.

   Being familiar with the Java source code and XML files will help you extend and customize this template for your own needs.



Architecture of the Basic Activity template

#	UI Description	Code reference
1	Status bar The Android system provides and controls the status bar.	Not visible in the template code. It's possible to access it from your activity. For example, you can hide the status bar, if necessary.
2	AppBarLayout > Toolbar The app bar (also called the action bar) provides visual structure, standardized visual elements, and navigation. For backwards	In `activity_main.xml`, look for `android.support.v7.widget.Toolbar`

	compatibility, the `AppBarLayout` in the template embeds a `Toolbar` with the same functionality as an `ActionBar`.	inside `android.support.design.widget.AppBarLayout`. Change the toolbar to change the appearance of its parent, the app bar. For an example, see the App Bar Tutorial.
3	Application name This is derived from your package name, but can be anything you choose.	In `AndroidManifest.xml`: `android:label="@string/app_name"`
4	Options menu overflow button Menu items for the activity, as well as global options, such as **Search** and **Settings** for the app. Your app menu items go into this menu.	In `MainActivity.java`: `onOptionsItemSelected()` implements what happens when a menu item is selected. **res > menu > menu_main.xml** Resource that specifies the menu items for the options menu.
5	Layout `ViewGroup` The `CoordinatorLayout` `ViewGroup` is a feature-rich layout that provides mechanisms for `View` (UI) elements to interact. Your app's user interface goes inside the `content_main.xml` file included within this `ViewGroup`.	In `activity_main.xml`: There are no views specified in this layout; rather, it includes another layout with an `include layout` instruction to include `@layout/content_main` where the views are specified. This separates system views from the views unique to your app.
6	`TextView` In the example, used to display "Hello World". Replace this with the UI elements for your app.	In `content_main.xml`: All your app's UI elements are defined in this file.
7	Floating action button (FAB)	In `activity_main.xml` as a UI element using a clip-art icon. `MainActivity.java` includes a stub in `onCreate()` that sets an `onClick()` listener for the FAB.

2.2 Customizing the app produced by the template

Change the appearance of the app produced by the Basic Activity template. For example, you can change the color of the app bar to match the status bar (which on some devices is a darker shade of

the same primary color). You may also want to remove the floating action button if you are not going to use it.

1. Change the color of the app bar (`Toolbar`) in `activity_main.xml` by changing the android:background to `"?attr/colorPrimaryDark"`, which sets the app bar color to a darker primary color that matches the status bar:

```
android:background="?attr/colorPrimaryDark"
```

2. To remove the floating action button, start by removing the stub code in `onCreate()` that sets an `onClick()` listener for the button. Open **MainActivity** and delete the following block of code:

```
FloatingActionButton fab = (FloatingActionButton)
                                            findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action",
                                        Snackbar.LENGTH_LONG)
                            .setAction("Action", null).show();
    }
});
```

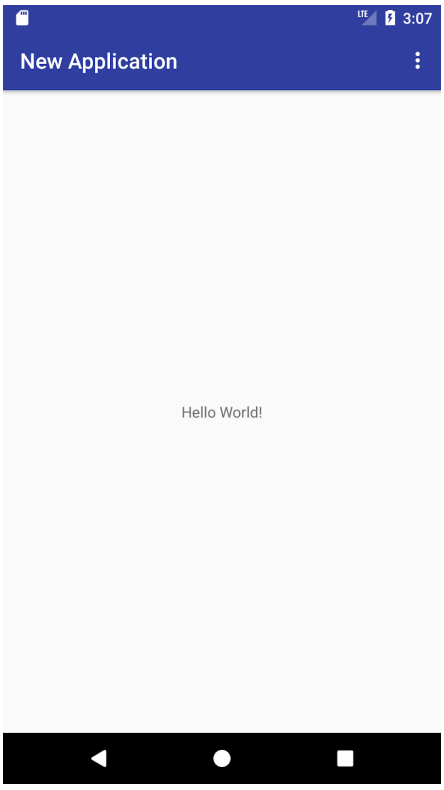
3. To remove the floating action button from the layout, delete the following block of XML code from `activity_main.xml`:

```
<android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@android:drawable/ic_dialog_email" />
```

4. Change the name of the app that is displayed in the app bar by changing the app_name string resource in strings.xml to the following:

```
<string name="app_name">New Application</string>
```

5. Run the app. The floating action button no longer appears, the name has changed, and the app bar background color has changed.



Tip: See Accessing Resources for details on the XML syntax for accessing resources.

2.3 Explore how to add activities using templates

For the practicals so far, you've used the Empty Activity and Basic Activity templates. In later lessons, the templates you use vary, depending on the task.

These activity templates are also available from inside your project, so that you can add more activities to your app after the initial project setup. (You learn more about the `Activity` class in another chapter.)

1. Create a new app project or choose an existing project.
2. In the **Project > Android** pane, **right-click** the **java** folder.
3. Choose **New > Activity > Gallery**.
4. Add an `Activity`. For example, click **Navigation Drawer Activity** to add an `Activity` with a navigation drawer to your app.
5. Double-click the layout files for the `Activity` to display them in the layout editor.

Task 3: Learn from example code

Android Studio and the Android documentation provide many code samples that you can study, copy, and incorporate with your projects.

3.1 Android code samples

You can explore hundreds of code samples directly from within Android Studio.

1. In Android Studio, choose **File > New > Import Sample**.
2. Browse the samples.
3. Choose a sample and click **Next**.
4. Accept the defaults and click **Finish**.

Note: The samples contained here are meant as a starting point for further development. We encourage you to design and build your own ideas into them.

3.2 Use the SDK Manager to install offline documentation

Installing Android Studio also installs essentials of the Android SDK (Software Development Kit). However, additional libraries and documentation are available, and you can install them using the SDK Manager.

1. Choose **Tools > Android > SDK Manager**.
2. In the left column, click **Android SDK**.
3. Select and copy the path for the Android SDK Location at the top of the screen, as you will need it to locate the documentation on your computer:

4. Click the **SDK Platforms** tab. You can install additional versions of the Android system from here.
5. Click the **SDK Update Sites** tab. Android Studio checks the listed and selected sites regularly for updates.
6. Click the **SDK Tools** tab. You can install additional SDK Tools that are not installed by default, as well as an offline version of the Android developer documentation.
7. Select the checkbox for "Documentation for Android SDK" if it is not already installed, and click **Apply**.
8. When the installation finishes, click **Finish**.
9. Navigate to the **sdk** directory you copied above, and open the **docs** directory.
10. Find **index.html** and open it.

Task 4: Many more resources

- The Android Developer YouTube channel is a great source of tutorials and tips.
- The Android team posts news and tips in the official Android blog.
- Stack Overflow is a community of programmers helping each other. If you run into a problem, chances are high that someone has already posted an answer. Try posting a question such as "How do I set up and use ADB over WiFi?" or "What are the most common memory leaks in Android development?"
- And last but not least, type your questions into Google search, and the Google search engine will collect relevant results from all of these resources. For example, "What is the most popular Android OS version in India?"

4.1 Search on Stack Overflow using tags

Go to Stack Overflow and type `[android]` in the search box. The `[]` brackets indicate that you want to search for posts that have been tagged as being about Android.

You can combine tags and search terms to make your search more specific. Try these searches:

- `[android] and [layout]`
- `[android] "hello world"`

To learn more about the many ways in which you can search on Stack Overflow, see the Stack Overflow help center.

Summary

- Official Android Developer Documentation: developer.android.com
- Material Design is a conceptual design philosophy that outlines how apps should look and work on mobile devices.
- The Google Play store is Google's digital distribution system for apps developed with the Android SDK.

- Android Studio provides templates for common and recommended app and activity designs. These templates offer working code for common use cases.
- When you create a project, you can choose a template for your first activity.
- While you are further developing your app, activities and other app components can be created from built-in templates.
- Android Studio contains many code samples that you can study, copy, and incorporate with your projects.

Related concept

The related concept documentation is in 1.4: Resources to help you learn.

Learn more

Android Studio documentation:

- Meet Android Studio
- Developer workflow basics

Android developer documentation:

- Android developer site
- Google Developers Training
- Layouts
- App resources overview
- Layouts
- Menus
- `TextView`
- String resources
- App Manifest

Code samples:

- Source code for exercises on GitHub
- Android code samples for developers

Videos:

- Android Developer YouTube channel
- Udacity online courses

Other:

- Official Android blog
- Android Developers blog
- Google Developers Codelabs
- Stack Overflow
- Android vocabulary