

Banner Ads



Banner ads are rectangular image or text ads that occupy a spot within an app's layout. They stay on screen while users are interacting with the app, and can refresh automatically after a certain period of time. If you're new to mobile advertising, they're a great place to start.

This guide shows you how to integrate banner ads from AdMob into an Android app. In addition to code snippets and instructions, it also includes information about sizing banners properly and links to additional resources.

Prerequisites

- Import the Google Mobile Ads SDK, either by itself (<https://developers.google.com/admob/android/quick-start>) or as part of Firebase (<https://firebase.google.com/docs/admob/android/quick-start>).

Add AdView to the layout

The first step toward displaying a banner is to place **AdView** (<https://developers.google.com/android/reference/com/google/android/gms/ads/AdView>) in the layout for the **Activity** or **Fragment** in which you'd like to display it. The easiest way to do this is to add one to the corresponding XML layout file. Here's an example that shows **AdView** at the bottom of an **Activity**:

main_activity.xml

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.google.android.gms.ads.AdView
        xmlns:ads="http://schemas.android.com/apk/res-auto"
        android:id="@+id/adView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        ads:adSize="BANNER"
        ads:adUnitId="ca-app-pub-3940256099942544/6300978111">
    </com.google.android.gms.ads.AdView>

</RelativeLayout>

```

Note the following required attributes:

- **ads:adSize** - Set this to the ad size you'd like to use (see the [banner size section](#) (#banner_sizes) below for details).
- **ads:adUnitId** - Set this to the unique identifier given to the [ad unit](#) ([//support.google.com/admob/answer/7356431](http://support.google.com/admob/answer/7356431)) in your app where ads are to be displayed. If you show banner ads in different activities, each would require an ad unit.

You can alternatively create AdView programmatically:

```

AdView adView = new AdView(this);
adView.setAdSize(AdSize.BANNER);
adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
// TODO: Add adView to your view hierarchy.

```

Warning: Make sure you set the ad size and ad unit ID in the same manner (i.e. set both in XML or both programmatically).

Always test with test ads

The sample code above contains an ad unit ID and you're free to request ads with it. It's been specially configured to return test ads rather than production ads for every request, which makes it safe to use.

However, once you register an app in the AdMob UI and create your own ad unit IDs for use in your app, you'll need to explicitly [configure your device as a test device](https://developers.google.com/admob/android/test-ads#enable_test_devices) (https://developers.google.com/admob/android/test-ads#enable_test_devices) when you're developing. This is **extremely important**.

Testing with real ads (even if you never tap on them) is against AdMob policy and can cause your account to be suspended. See [Test Ads](https://developers.google.com/admob/android/test-ads) (<https://developers.google.com/admob/android/test-ads>) for information on how you can make sure you always get test ads when developing.

Load an ad

Once the AdView is in place, the next step is to load an ad. That's done with the [loadAd\(\)](https://developers.google.com/android/reference/com/google/android/gms/ads/AdView#loadAd(com.google.android.gms.ads.AdRequest)) ([https://developers.google.com/android/reference/com/google/android/gms/ads/AdView#loadAd\(com.google.android.gms.ads.AdRequest\)](https://developers.google.com/android/reference/com/google/android/gms/ads/AdView#loadAd(com.google.android.gms.ads.AdRequest))) method in the AdView class. It takes an [AdRequest](https://developers.google.com/android/reference/com/google/android/gms/ads/AdRequest) (<https://developers.google.com/android/reference/com/google/android/gms/ads/AdRequest>) parameter, which holds runtime information (such as targeting info) about a single ad request.

Here's an example that shows how to load an ad in the `onCreate()` method of an Activity:

MainActivity.java (excerpt)

```
package ...  
  
import ...  
import com.google.android.gms.ads.AdRequest;
```

```
import com.google.android.gms.ads.AdView;

public class MainActivity extends AppCompatActivity {
    private AdView mAdView;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        MobileAds.initialize(getApplicationContext(),
            "ca-app-pub-3940256099942544~3347511713");

        mAdView = (AdView) findViewById(R.id.adView);
        AdRequest adRequest = new AdRequest.Builder().build();
        mAdView.loadAd(adRequest);
    }
    ...
}
```

That's it! Your app is now ready to display banner ads.

Ad events

To further customize the behavior of your ad, you can hook onto a number of events in the ad's lifecycle: loading, opening, closing, and so on. You can listen for these events through the

AdListener

(<https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener>) class.

To use an AdListener

(<https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener>) with

AdView (<https://developers.google.com/android/reference/com/google/android/gms/ads/AdView>), simply call the setAdListener()

([https://developers.google.com/android/reference/com/google/android/gms/ads/AdView#setAdListener\(com.google.android.gms.ads.AdListener\)](https://developers.google.com/android/reference/com/google/android/gms/ads/AdView#setAdListener(com.google.android.gms.ads.AdListener)))

method:

```
mAdView.setAdListener(new AdListener() {
    @Override
    public void onAdLoaded() {
        // Code to be executed when an ad finishes loading.
        Log.i("Ads", "onAdLoaded");
    }
});
```

```

    }

    @Override
    public void onAdFailedToLoad(int errorCode) {
        // Code to be executed when an ad request fails.
        Log.i("Ads", "onAdFailedToLoad");
    }

    @Override
    public void onAdOpened() {
        // Code to be executed when an ad opens an overlay that
        // covers the screen.
        Log.i("Ads", "onAdOpened");
    }

    @Override
    public void onAdLeftApplication() {
        // Code to be executed when the user has left the app.
        Log.i("Ads", "onAdLeftApplication");
    }

    @Override
    public void onAdClosed() {
        // Code to be executed when when the user is about to return
        // to the app after tapping on an ad.
        Log.i("Ads", "onAdClosed");
    }
});

```

Each of the overridable methods in [AdListener](#)

(<https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener>)

corresponds to an event in the lifecycle of an ad.

Overridable methods

onAdLoaded()	The <u>onAdLoaded()</u> (https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener#onAdLoaded) is executed when an ad has finished loading. If you want to delay adding the <u>AdView</u> (https://developers.google.com/android/reference/com/google/android/gms/ads/AdView) will be loaded, for example, you can do so here. If you're using a third-party analytics library, place the call to record them.
onAdFailedToLoad()	The <u>onAdFailedToLoad()</u> (https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener#onAdFailedToLoad)

that includes a parameter. The `errorCode` parameter indicates what type of failure the [AdRequest](https://developers.google.com/android/reference/com/google) (<https://developers.google.com/android/reference/com/google>

- `ERROR_CODE_INTERNAL_ERROR` - Something happened internally; for instance
- `ERROR_CODE_INVALID_REQUEST` - The ad request was invalid; for instance
- `ERROR_CODE_NETWORK_ERROR` - The ad request was unsuccessful due to network
- `ERROR_CODE_NO_FILL` - The ad request was successful, but no ad was returned

onAdOpened() This method is invoked when the user taps on an ad. If you're using an analytics service, you should call `onAdOpened()` to record the event.

onAdLeftApplication() This method is invoked after [onAdOpened\(\)](https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener) (<https://developers.google.com/android/reference/com/google/android/gms/ads/AdListener>) when the user leaves the app to open another app (such as the Google Play Store), backgrounding the current app.

onAdClosed() When a user returns to the app after viewing an ad's destination URL, this method is invoked. You should call `onAdClosed()` to perform any other work necessary to make the app ready for interaction. See the <https://github.com/googleads/googleads-mobile-android-examples/blob/master/advanced/APIDemo/app/src/main/java/com/google/android/gms/ads/example/AdvancedAPIDemoApp.java> for an implementation of the ad listener methods in the Android API Demo app.

Banner sizes

The following banner sizes are supported:

Size (WxH)	Description	Availability	AdSize constant
320x50	Standard Banner	Phones and Tablets	<code>BANNER</code>
320x100	Large Banner	Phones and Tablets	<code>LARGE_BANNER</code>
300x250	IAB Medium Rectangle	Phones and Tablets	<code>MEDIUM_RECTANGLE</code>
468x60	IAB Full-Size Banner	Tablets	<code>FULL_BANNER</code>
728x90	IAB Leaderboard	Tablets	<code>LEADERBOARD</code>
Screen width x 32 50 90	Smart Banner	Phones and Tablets	<code>SMART_BANNER</code>

Note: If an app tries to load a banner that's too big for its layout, the SDK won't display it. Instead, an error message will be written to the log.

Smart Banners

Smart Banners are ad units that render screen-width banner ads on any screen size across different devices in either orientation. Smart Banners help deal with increasing screen fragmentation across different devices by "smartly" detecting the width of the device in its current orientation and making the ad view that size.

Three ad heights are implemented in smart banners:

Ad height	Screen height
32dp	$\leq 400\text{dp}$
50dp	$> 400\text{dp}$ and $\leq 720\text{dp}$
90dp	$> 720\text{dp}$

Typically, Smart Banners on phones have a height of 50dp in portrait and 32dp in landscape. On tablets, height is normally 90dp in both orientations.

When an image ad isn't large enough to take up the entire allotted space, the image will be centered, and the space on either side will be filled in.



To use Smart Banners in XML, specify the constant `SMART_BANNER` for the ad size and set the width of the `AdView` to `match_parent`. For example:

```
<com.google.android.gms.ads.AdView
  xmlns:ads="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  ads:adSize="SMART_BANNER"
  ads:adUnitId="ca-app-pub-3940256099942544/6300978111">
</com.google.android.gms.ads.AdView>
```

To create a Smart Banner programmatically, use `AdSize.SMART_BANNER` as the ad size:

```
AdView adView = new AdView(this);  
adView.setAdSize(AdSize.SMART_BANNER);
```

Additional resources

Samples

- [Banner example](#)
([//github.com/googleads/googleads-mobile-android-examples/tree/master/admob/BannerExample](https://github.com/googleads/googleads-mobile-android-examples/tree/master/admob/BannerExample))
- minimal implementation of banner ads
- [API Demo](#)
([//github.com/googleads/googleads-mobile-android-examples/tree/master/advanced/APIDemo](https://github.com/googleads/googleads-mobile-android-examples/tree/master/advanced/APIDemo)) -
features advanced banner topics

Mobile Ads Garage video tutorials

- [Banner Implementation](#)
(<https://www.youtube.com/watch?v=h-FMndW2kHo&list=PLOU2XLYxmsIKX0pUJV3uqp6N3NeHwHh0c&index=2>)
- [Banner Best Practices](#)
(<https://www.youtube.com/watch?v=BE2K1IpXaSI&list=PLOU2XLYxmsIKX0pUJV3uqp6N3NeHwHh0c&index=3>)

Next steps

- If you haven't already, create your own app and banner ad unit in the [AdMob UI](#) ([//apps.admob.com](https://apps.admob.com)) and use your newly created app ID and ad unit ID in your code. Remember to configure your device with test ads.
- Learn about [ad targeting](#) (<https://developers.google.com/admob/android/targeting>) and [banner ad guidance](#) ([//support.google.com/admob/answer/6128877](https://support.google.com/admob/answer/6128877)).
- Try another ad format:
 - [Interstitial](#) (<https://developers.google.com/admob/android/interstitial>)
 - [Rewarded Video](#) (<https://developers.google.com/admob/android/rewarded-video>)

- **Native** (<https://developers.google.com/admob/android/native>)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/) (<http://creativecommons.org/licenses/by/3.0/>), and code samples are licensed under the [Apache 2.0 License](http://www.apache.org/licenses/LICENSE-2.0) (<http://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated July 14, 2017.