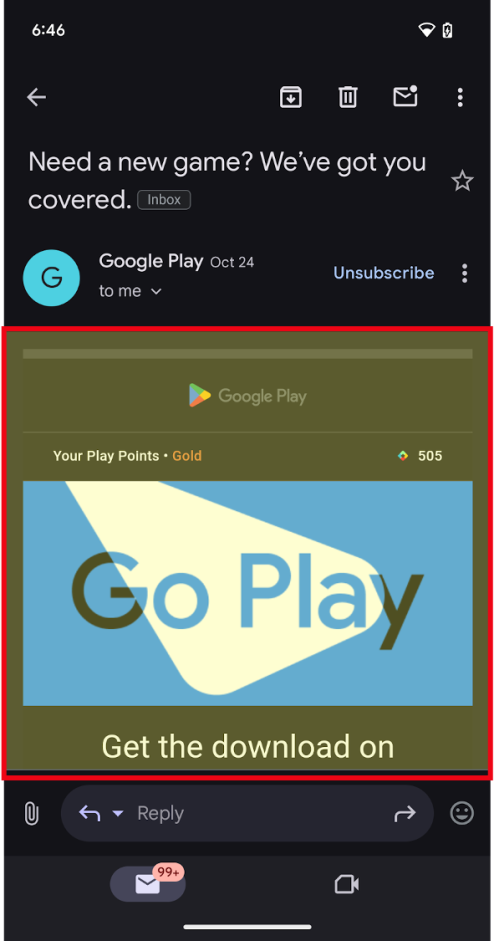
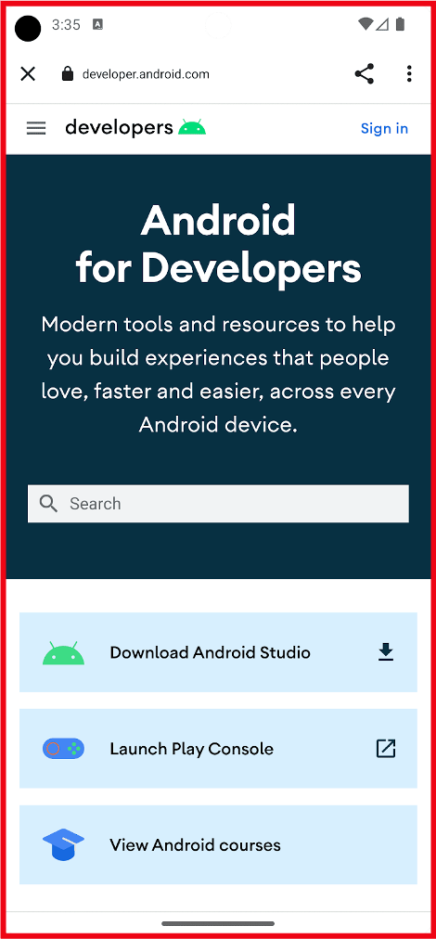
* **web content within your Android app**

Android allows you as a developer to build on the power of the web within your apps, so you can benefit from the flexibility and efficiency of being able to display certain types of content.

This lets you seamlessly integrate existing web content into your native Android application, such as to display a news feed, show interactive tutorials, display ads, or even host a mini-game without building everything from scratch. Think of it as a window to the internet, from within your app. There are two ways to embed web content into your app:

* [WebView](https://developer.android.com/reference/android/webkit/WebView): It displays web content you control inline where you want a high degree of flexibility in customizing or updating the UI.
* [Custom Tabs](https://developer.android.com/develop/ui/views/layout/webapps/overview-of-android-custom-tabs): A full in-app browsing experience powered by the user's default browser ([see browser support](https://developer.chrome.com/docs/android/custom-tabs/browser-support)) for when users click a link and you want to keep them in the app, instead of leaving to an external browser, with much of the browsing experience out-of-the-box.

**Figure 1.** WebView (left) and Custom Tab (right) outlined in red.

**Embed web content**

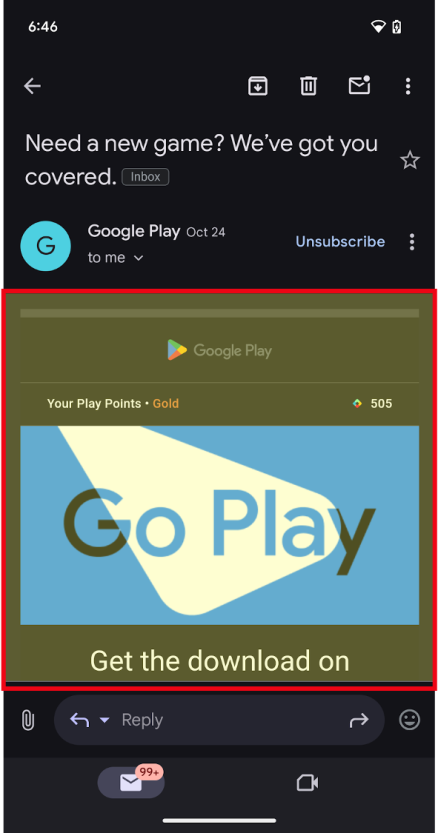
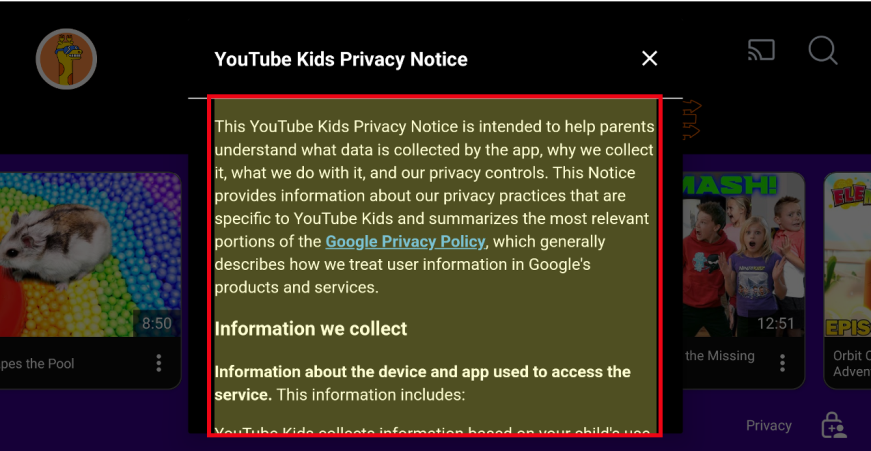
* Efficiency: Reuse existing code from your website. Build on existing web technologies and content.
* Integration: Leverage external content from 3P providers, such as Media, Ads, within your app.
* Flexibility: Update content dynamically without being constrained to predefined UIs, or without releasing app updates.

**use web content**

There are three primary uses cases for using the Web in your Android app:

**1.**[**Embedding web content into your app as primary or supporting content**](https://developer.android.com/develop/ui/views/layout/webapps/embed-web-content-in-app)**: Use WebView**

* Display your own web content inline as a primary experience where you want a high degree of flexibility in customizing or updating the UI.
* Display other content such as ads, legal terms and regulations, or other third-party content inline, or as a window within your app experience.

**Figure 2.** Web content embedded within the app with WebViews as primary (left) and supporting content (right).

**2.**[**In-app browsing**](https://developer.android.com/develop/ui/views/layout/webapps/in-app-browsing-embedded-web?)**using Custom Tabs, or WebView for more advanced use cases**

* Have a full in-app browsing experience for when users click a link and you want to keep them in the app, instead of leaving to an external browser.
  + Note: For large screen devices such as tablets and foldables, there are additional options to help apps take advantage of additional space:
  + Apps can open weblinks in split screen using [launch an adjacent multi-window experience](https://developer.android.com/develop/ui/compose/layouts/adaptive/support-multi-window-mode#launch_adjacent). This enables users to multitask between your app and a browser at the same time. OR
  + Custom Tabs have a side panel option that can open in the same task, but next to your existing app content.
* The Custom Tab is powered by the user's default browser, for browsers which support Custom Tabs.
  + While it's possible to use a WebView and provide a highly customizable in-app browsing experience, we recommend Custom tabs for an out-of-the-box browser experience and seamless transition for when a user wants to open a web link in the browser.

Web page with in-app link in red box on left, and an in-app browser
  shown on right.**Figure 3.** Clicking on an in-app link (left) and opening an in-app browser (right).

**3. Login or Authentication flows within your app**

Android's suggested approach is to build your login or authentication flows using [Credential Manager](https://developer.android.com/identity/sign-in/credential-manager). If you find you still need to use Embedded Web for these experiences, use the following guidance:

* Some apps use WebViews to provide sign-in flows for their users, including using a username and passkey (or password) specific to your app. This enables developers to unify the authentication flows across platforms.
* When linking out to a third-party identity provider or login experience, such as "Sign in with…", Custom Tabs are the way to go. Launching Custom Tabs ensures the user's credential remains protected and isolated to the 3rd party site.