WebAPKs on Android

By Pete LePage

Pete is a Developer Advocate

Add to Home Screen on Android does more than just add the Progressive Web App to the users Home Screen. Chrome automatically generates and installs a special APK of your app. We sometimes refer to this as a **WebAPK**. Being installed via an APK makes it possible for your app to show up in the app launcher, in Android's app settings and to register a set of intent filters.

To generate the WebAPK Chrome looks at the web app manifest, and other meta-data. Whenever the manifest changes, Chrome will need to generate a new APK.

Note: Since the WebAPK is generated each time the manifest changes, we recommend changing it only when necessary. Don't use it to store user specific identifiers, or other other data that will frequently change. Frequently changing the manifest will increase install time because the WebAPK will need to be re-generated with every change.

Android intent filters

When a Progressive Web App is installed on Android, it will register a set of intent filters for all URLs within the scope of the app. When a user clicks on a link that is within the scope of the app, the app will be opened, rather than opening within a browser tab.

Consider the following partial **`manifest.json`**, when launched from the app launcher, it would launch **`https://example.com/`** as a standalone app, without any browser chrome.

```
"start_url": "/",
"display": "standalone",
```

The WebAPK would include the following intent filters:

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data
```

```
    android:scheme="https"
    android:host="example.com"
    android:pathPrefix="/" />
</intent-filter>
```

If the user clicks on a link within an installed app to `https://example.com/read`, it would be caught by the intent and opened in the Progressive Web App.

Note: Navigating directly to `https://example.com/app/` from the address bar will work exactly as the same as it does for native apps that have an intent filter. Chrome assumes the user **intended** to visit the site and will open this site.

Using `scope` to restrict intent filters

If you don't want your Progressive Web App to handle all URLs within your site, you can add the scope property to your web app manifest. The `scope` property tells Android to only open your web app if the URL matches the `origin` + `scope`, and limits which URLs will be handled by your app and which should be opened in the browser. This is helpful when you have your app and other non-app content on the same domain.

Consider the following partial `manifest.json`, when launched from the app launcher, it would launch `https://example.com/app/` as a standalone app, without any browser chrome.

```
"scope": "/app/",
"start_url": "/",
"display": "standalone",
```

Like before, the generated WebAPK would include an intent filter but would modify the `android:pathPrefix` attribute in the APK's `AndroidManifest.xml`:

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data
    android:scheme="https"
    android:host="example.com"
    android:pathPrefix="/app/" />
</intent-filter>
```

Let's take a look at a few examples:

✓ `https://example.com/app/` - within `/app/`
✓ `https://example.com/app/read/book` - within `/app/`
🚫 `https://example.com/help/` - not in `/app/`
🚫 `https://example.com/about/` - not in `/app/`

See scope for more information about `scope`, what happens when you don't set it, and how you can use it to define the scope of your app.

Managing permissions

Permissions work in the same way as other web apps and cannot be requested at install time, instead they must be requested at run time, ideally only when you really need them. For example, don't ask for camera permission on first load, but instead wait until the user attempts to take a picture.

Note: Android normally grants immediate permission to show notifications for installed apps, but apps installed via WebAPKs are not granted this at install time, you must request it at runtime within your app.

Managing storage and app state

Even though the progressive web app is installed via an APK, Chrome uses the current profile to store any data, and it will not be segregated away. This allows a shared experience between the browser and the installed app. Cookies are shared an active, any client side storage is accessible and the service worker is installed and ready to go.

Though, this can be an issue if the user clears their Chrome profile, or chooses to delete site data.

Feedback

Was this page helpful?

YES NO

Great! Thank you for the feedback.

Sorry to hear that. Please open an issue and tell us how we can improve.

Frequently asked questions

What happens if the user has already installed the native app for the site?

Like add to home screen today, users will be able to add a site independent of any native apps. If you expect users to potentially install both, we recommend differentiating the icon or name of your site from your native app.

When a user opens a site installed via improved add to Home screen, will Chrome be running?

Yes, once the site is opened from the home screen the primary activity is still Chrome. Cookies, permissions, and all other browser state will be shared.

Will my installed site's storage be cleared if the user clears Chrome's cache?

Yes.

Will my app be re-installed when I get a new device?

Not at this time, but we think it is an important area and we are investigating ways to make it work.

Will I be able to register to handle custom URL schemes and protocols?

No.

How are permissions handled? Will I see the Chrome prompt or Android's?

Permissions will still be managed through Chrome. Users will see the Chrome prompts to grant permissions and will be able to edit them in Chrome settings.

What versions of Android will this work on?

Progressive web apps can be installed on all versions of Android that run Chrome for Android, specifically Jelly Bean and above.

Does this use the WebView?

No, the site opens in the version of Chrome the user added the site from.

Can we upload the APKs that are created to the Play Store?

> No. There is no key signing information supplied to enable you to create your own PWA that can be in the store.

Are these listed in the Play Store?

> No.

I am developer of another browser on Android, can I have this seamless install process?

> We are working on it. We are committed to making this available to all browsers on Android and we will have more details soon.
