Episode Book

EPISODE #5939

Android Notifications: PWA vs Native, What You Actually Get

PUBLISHED	13 October 2026 at 18:15 UTC
DURATION	22:10
LISTEN	myweirdprompts.com/episode/android-notifications-pwa-vs-native-56628/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

Daniel asked us to look at Android notifications from a developer's seat, and the honest answer isn't "native wins." It's about where control actually sits in the stack — and it flips depending on whether you mean the developer's control or the user's. Native Android hands you a rich, opinionated channel architecture with importance levels, categories, and vibration patterns, then deliberately freezes it and gives the user final say. The web platform gives you one Notification interface and a Push API, one global permission per origin, and a scheduling primitive that was built and then withdrawn. We walk through the primitives on both sides, the haptics story down to the hardware, and the places where the loudest numbers come from people selling native wrappers.

TRANSCRIPT



Corn

There's a story the app-builder blogs like to tell about notifications on Android, and it goes like this. Native is the serious choice, PWAs are the cheap one, and if you care whether your alerts actually arrive you build native. That story has a commercial interest behind it. Keep that in mind.



Herman

And it's wrong in an interesting way, which is better than being wrong in a boring way.



Corn

Daniel sent us a prompt. He wants to look at notifications on Android from a developer's seat, and he wants to split the question in two shapes. The PWA, and the native Android build. What primitives do you actually get for triggering a notification, can haptics be driven deterministically or do they just inherit whatever the user has set, can you rely on notification triggers at all in a PWA, and if what you really want is for the user to have full control over how the app talks to them, does native give you something the web simply doesn't have.



Herman

Four threads, and they don't all pull the same direction.



Corn

That's why I want to do this properly. Most pieces on this treat it as a straight contest, and it isn't one.



Herman

Here's the shape of it. The question sounds like which is better, and it isn't that. It's about where control actually sits in the notification stack. And the answer flips depending on whether you mean the developer's control or the user's.



Corn

Which is the sort of sentence that sounds like a dodge until you see the mechanism.



Herman

To get concrete. A PWA is a web app running in a browser context with a service worker behind it. A native Android build is an APK with a manifest, runtime permissions, and direct access to the platform's notification APIs. Those are not two implementations of one thing. They're two different architectures that happen to produce a similar-looking notification.



Corn

And people assume they're the same because both put a banner on the lock screen.



Herman

Right, and the structural asymmetry is the whole episode. Native Android has a rich, opinionated notification architecture built around channels, and that architecture explicitly hands final control to the user. The web platform has a much thinner surface. One Notification interface and a Push API. There's no channel concept, no importance level, no per-category anything.



Corn

One global switch per origin.



Herman

Per origin, yes. Your whole site is one notification relationship, and the user either wants it or doesn't.



Corn

It's like being asked whether you like a restaurant, and the only answer is yes or no, and the menu is whatever they feel like.



Herman

Now, before we start on primitives, one thing I want to flag. When you go looking for how big the reliability gap between these two actually is, the sources disagree sharply. And the ones with the loudest numbers tend to be app-builder shops selling native wrappers.



Corn

Whose business is converting web apps into native shells. So the finding that native performs better is, let's say, convenient.



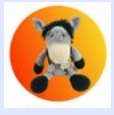
Herman

Convenient. One outfit puts native at about a thirty-six percent open rate against twelve for web push, roughly three to one. Another says the delivery infrastructure is the same and the differences are at the edges. Both hop through the same push services. Both cap payloads around four kilobytes.



Corn

Both numbers can't be the whole story.



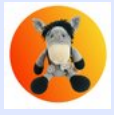
Herman

Which is itself a finding. Let's start with the part that's actually well documented, which is native.



Corn

Give me the plumbing.



Herman

Since Android eight point oh, API level twenty-six, every notification must be assigned to a notification channel. Post without one and the notification simply does not appear. The system logs an error and that's the end of it.



Corn

Silent failure. My favourite kind.



Herman

A channel is built from three things. A unique ID, a user-visible name, and an importance level. And importance runs from zero to four. Zero is none, four is high. High gets you sound and a heads-up banner. Default gets you sound. Low, no sound. Minimum and none don't even put an icon in the status bar.



Corn

So the developer picks the interruption level at the moment they create the channel.



Herman

And here's the constraint that shapes everything else in this episode. After you create a notification channel, you can't change the notification behaviours. That's Google's own wording. The user has complete control at that point.



Corn

Say that again, because I want to be sure I heard it. The developer can't change it.



Herman

The developer can only change the channel's name and description afterwards. Nothing behavioural. And if you try to recreate the channel with the original values, it's a no-op. The system ignores you.



Corn

So the design decision is frozen at first launch, permanently, on every device where the app has ever run.



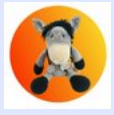
Herman

For that channel ID. Yes.



Corn

That's not a limitation. That's a landmine with a label on it.



Herman

And there's a second gate on top. Since Android thirteen, API thirty-three, apps have to declare the POST_NOTIFICATIONS runtime permission. So you ask the user, and then you have one shot at configuring what they've just agreed to receive.



Corn

What's in the toolkit once you're through that gate?



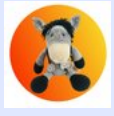
Herman

NotificationCompat.Builder for constructing them, NotificationManagerCompat for posting, NotificationChannelCompat for the channel itself. Channel groups, so you can organise them. And system-wide categories like CATEGORY_ALARM and CATEGORY_REMINDER that tell the platform what kind of thing this is, which affects how it's treated for things like Do Not Disturb.



Corn

Alarms get to cut through. A promotional nudge doesn't.



Herman

If you're honest about the category. And you can also read back what the user has done to your channel. There's getNotificationChannel, getNotificationChannels. You can query the vibration pattern, the sound, the importance. So you're not flying blind.



Corn

You can see that they've turned you down.



Herman

You can see it exactly. And you can deep-link them into the channel settings screen so they can adjust it themselves, which is the sanctioned move. What you cannot do is change it for them.



Corn

Now the web side.



Herman

Web Push is the only standard way for a server to reach a PWA user who has closed the app. You call subscribe on the push manager, you get back a subscription object, the browser wakes the service worker with a push event, and the worker calls show notification.



Corn

No service worker running, no notification.



Herman

And the Notification interface itself is not bad on options. Body, icon, image, badge, actions, tag, renotify, require interaction, silent, timestamp, vibrate.



Corn

That's a reasonable-looking list.



Herman

It is. And here's the first hard no in the research. The Notification Triggers API, which would have let a PWA schedule a notification for a future time without a network connection, was abandoned. Chrome's own documentation says development has ended. Their words, it wasn't clear that they could provide consistent and reliable experiences across platforms.



Corn

So that's gone.



Herman

That's gone. A PWA cannot reliably schedule a local notification for a future time without a server push at that moment.



Corn

That's the biggest concrete argument for native in this entire episode. If your app is a calendar, an alarm, a medication reminder, anything where the trigger is a clock rather than a server, the web platform doesn't have an answer.



Herman

Not a reliable one. You can approximate it with a periodic background sync or a server, but the clean primitive was designed, built and then withdrawn.



Corn

Why withdrawn?



Herman

Because scheduling a local notification means the browser process has to stay alive and trusted on every platform, and they couldn't promise that. On one platform it worked, on another the browser got frozen, and a reminder that fires sometimes is worse than no reminder, because people depend on it.



Corn

A calendar that misses Tuesday is not a calendar.



Herman

Take the two examples. A calendar app on native. You give it a channel with high importance, you tag the notification with CATEGORY_REMINDER, you schedule it locally, and at the appointed minute it fires a heads-up banner with sound. No server involved. Airplane mode, it still works.



Corn

And the same app as a PWA?



Herman

Your server has to wake up at the right minute, push to the subscription, and hope the subscription is still alive. If the server is down, or the push subscription was dropped because the user cleared browsing data three weeks ago, the reminder doesn't arrive. Nobody finds out. The notification just isn't there.



Corn

And you'd never know whether it was your server or their phone.



Herman

That's the honest position. It's a delivery chain with about four links, and any one of them can fail quietly.



Corn

What about a chat app? That's the one everybody actually experiences.



Herman

On native you get a Messages channel at high importance with a custom vibration pattern. A PWA gets one global notification permission per origin and a single Notification interface with no channel concept at all. You can't separate your marketing from your replies, because there's nowhere to put the distinction.



Corn

The user's only recourse is to mute the entire site.



Herman

Which they will do, and then they've muted the thing they wanted.



Corn

Now the permission asymmetry, because I think that one's underrated.



Herman

Chrome on Android only allows the notification permission prompt in response to a user gesture. You can't ambush somebody the moment the page loads. And a denial is close to permanent. The user has to go into site settings to undo it. There's no second ask.



Corn

So the PWA gets one gesture-triggered shot.



Herman

One shot, and the recovery path is buried three menus deep. Native has its own permission gate, but once you're through it you've got the channel system, which is the actual subject here.



Corn

Channels are a user-control mechanism that happens to be built by developers.



Herman

That's exactly what they are.



Corn

Let's do haptics, because this is where I expected a clean answer and I don't think there is one.



Herman

There isn't. Native gives developers more haptic primitives than it ever has. There's the old HapticFeedbackConstants surface from Android one point five. Predefined VibrationEffect from Android ten. VibrationEffect primitive composition from Android eleven. And VibrationEffect.Builder composition, which becomes the preferred API from Android sixteen.



Corn

A lot of surface area.



Herman

For notifications specifically, the pattern gets attached to the channel. There's setVibrationPattern on the channel builder, setVibrate on the notification compat builder, and a couple of older entry points. They all take a basic on-off waveform where the first number is the delay before the vibrator turns on.



Corn

So the developer writes the rhythm.



Herman

The developer writes it at channel creation. And then the channel locks, and the user has final say. There's a thing called VibrationAttributes that classifies the vibration by usage so the right user settings get applied rather than surprising anybody. And the user can switch off Android's Touch feedback setting entirely.



Corn

Globally.



Herman

Globally, one toggle, and your carefully designed double pulse is now nothing.



Corn

And then the hardware.



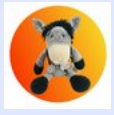
Herman

The hardware is the part people forget. There's hasAmplitudeControl on the vibrator. If a device doesn't have it, every amplitude value you supply collapses to on or off depending on whether it's zero. A gentle pulse and a hard buzz become the same thing.



Corn

Which is a different phone, not a different setting.



Herman

A different phone. There's also areEnvelopeEffectsSupported for envelope effects. And the composition API has no automatic fallback, so one unsupported primitive kills the whole effect. The newer builder API adds framework-level fallback, which is a real improvement, but it's new.



Corn

Now the web version of that.



Herman

Notification.vibrate takes a vibration pattern through the Vibration API. And MDN flags it as limited availability, not baseline, because it doesn't work in some of the most widely used browsers, and marks it experimental.



Corn

Experimental is a strong word for something that's been in browsers for years.



Herman

It's the honest word. And here's a gap I'd rather flag than paper over. Whether Notification.vibrate actually fires on Android Chrome is not confirmed by anything I could find. MDN marks it experimental and limited availability but doesn't state Android Chrome support explicitly either way. So I can't tell you it works.



Corn

And notification.silent is the other end of it. That suppresses sound or vibration regardless of what the device settings say.



Herman

Regardless. So the developer can ask for silence even if the user wants noise, which is the one place where the developer wins this argument.



Corn

So state the determinism answer plainly, because I think it's cleaner than it looks.



Herman

A developer can specify a vibration pattern for a notification channel. A developer cannot guarantee it plays. It depends on the user's channel settings, the global touch feedback setting, and whether the hardware can even render the pattern.



Corn

In both shapes.



Herman

Native gives you more expressive primitives and a pattern scoped to a channel. The PWA gives you a single vibrate array that may not fire at all on the platform you care about.



Corn

Nobody can guarantee a haptic. Which leads us to the question underneath Daniel's prompt. Not what the developer can specify, but who holds the controls once the notification is live.



Herman

And this is where I think the framing most people bring to it falls over. The common assumption is that native gives the developer more control. It doesn't. It gives the developer less, deliberately, and hands it to the user.



Corn

Explain the mechanism, because that's the piece that surprises people.



Herman

The channel system is a contract. You, the developer, declare what kinds of things you intend to send, at what intrusion level, and then the user is handed a control panel listing every one of those categories with an individual switch. They can mute the promotions channel and keep the messages channel on high. They can turn off vibration on one and keep the sound on another. The platform was designed to give them exactly that.



Corn

And the price of that design is that the developer writes the taxonomy once, at first launch, and then loses the pen.



Herman

Loses the pen permanently. Compare that with the PWA, which has essentially one global on or off per origin. So which one gives the user more control?



Corn

Native. Obviously.



Herman

Native, and by a wide margin. But that's only if you ask the question the way Daniel asked it, which is about the user. If you ask which shape gives the developer more control, the answer flips, and it's the PWA, because on the web the developer owns the notification behaviour entirely. There's nothing for the user to reconfigure. The only lever they have is the master switch.



Corn

That's a counterintuitive place to land.



Herman

It's the correct place to land. Native is better for the user at the cost of the developer. The web is better for the developer at the cost of the user.



Corn

Now the reliability numbers, because I don't want to leave that vague.



Herman

On Android, PWA push works well and has worked for years. A subscribed browser session is enough. No home-screen install required, which is where it differs from iOS, where push only arrives if the app is installed to the home screen.



Corn

So the Android web push story is not broken.



Herman

It isn't broken. The problems are at the edges and they're specific. Clearing browser data drops the subscription. And the OEM battery optimizers, Samsung's, Xiaomi's, Oppo's, OnePlus's, Vivo's, will kill the browser process, which deregisters the service worker. One source puts the effective drop at twenty to forty percent on those OEMs after forty-eight to seventy-two hours of inactivity.



Corn

And native survives that because the push goes to the app process, not the browser.



Herman

That's the argument. High-priority messages also bypass Doze and App Standby on the native side. And you avoid the via Chrome attribution that Android shows on web notifications, which matters more than people admit, because it's a small badge that tells the user this arrived through the browser.



Corn

On a Samsung after three days of not opening the site, the difference is real.



Herman

Real and hard to quantify. Which is why I'd flag those open-rate numbers rather than repeat them. The people publishing the most dramatic ones are selling the fix.



Corn

And the other camp says it's all the same pipes.



Herman

Both hop through a push service. Both cap payloads at roughly four kilobytes. At the infrastructure layer that's true. The difference is what happens on the handset, and on aggressive Android skins that difference is not nothing.



Corn

I'll take the middle. Native delivers more reliably on bad OEMs and gives the user a real control panel. The web is good enough on stock Android and everywhere the optimiser behaves.



Herman

And the web one is losing ground, if that source is right, because the battery optimisation arms race is getting more aggressive rather than less.



Corn

Which matters if notification delivery is load-bearing for your app. If a missed notification is an annoyance, ship the PWA. If a missed notification is a missed appointment, that's a different calculation.



Hilbert

I've got a channel on my own phone called Hilbert's Alerts. I set it up myself. High importance, custom vibration pattern, the one that means it's actually important.



Corn

Which app?



Hilbert

That's the thing. I'm not sure whether I set it up through an app or through the system settings. I don't think the distinction matters.



Herman

It does, a bit.



Hilbert

It doesn't. What matters is that the channel IDs developers pick are wrong. I can still recite one. com dot example dot urgent dash alerts dash v2.



Corn

You remember that one?



Hilbert

I remember it because it was the correct name and someone changed it. That's not the point. The point is you said the problem is that you can't change channel behaviour after creation. That's not the problem. The problem is that most developers create their channels on first app launch, before they've thought about what the channels should be. Then they're stuck with a bad taxonomy forever. I told them this.



Corn

You told them.



Hilbert

I told everyone this. Someone who knew someone asked me what I thought about the importance levels. I said they should go to eleven. They didn't listen. I still have the email where I suggested it. I'll forward it to you.



Herman

I'd like to see that.



Hilbert

I can't find it. It's in a folder called Sent, Important, and I have never been able to open that folder.



Corn

Hold on, the levels going to eleven.



Hilbert

Eleven. They went to four.



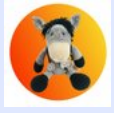
Corn

Herman, level four.



Hilbert

Wait, the second mic's peaking. Pull it back. Sorry. So they went to four, and now every app on the phone has a channel for promotions with the same name as a channel for something you actually need, and the user mutes the wrong one, and that's the whole story. Nobody asked me.



Herman

I've got one thing to take from that, and it's not the eleven.



Hilbert

What?



Herman

The ID. You can't rename it either. You can change the name the user sees, but the ID is the identity. Once you ship a channel called urgent alerts version two, that string is in the phone forever.



Hilbert

That's what I said. Nobody listens. I'll find that email.



Corn

It changes how I think about it, actually. It's not that the taxonomy is frozen. It's that the taxonomy is a public commitment from the day you ship.



Herman

Which is the lower-level version of the same point, and I'll take it.



Hilbert

Good. Because I'm not going to find that email.



Corn

The bit that didn't make it into the episode proper is the one line in the haptics docs I keep thinking about. There's a vibration attribute whose entire job is to stop the phone surprising the user.



Herman

The classification for the haptics. You tell the system what kind of vibration this is so the right user settings get applied.



Corn

The platform is so worried about buzzing somebody at the wrong moment that it built a whole category system just to prevent it.



Herman

And the web platform's answer to that problem is a single array of milliseconds.



Corn

There's the whole episode in two sentences. Maybe the web gets there eventually, a channel-equivalent model, and maybe the gap on Android is permanent rather than temporary.



Herman

And there's a hole nobody's filled. Whether Notification.vibrate actually fires on Android Chrome. That's a plain empirical question and I couldn't find a straight answer anywhere.



Corn

If it doesn't, the web haptics story is a spec sheet rather than a feature.



Herman

Which gets worse, not better, if the battery optimisers keep getting more aggressive, because then the developers who care most about delivery move to native and the web path stays exactly this thin.



Corn

If you're enjoying the show, a review helps other people find it. That's been My Weird Prompts. Hilbert Flumingtop produces it. Our site is my weird prompts dot com. And if you've got a question of your own, send us your own prompt on Telegram at t dot me slash MWP listener bot.



Herman

We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Android Developers, Create and manage notification channels, last updated 2026-10-01 [primary]**
   https://developer.android.com/develop/ui/compose/notifications/channels

2. **Android Developers, Android haptics API reference, last updated 2026-10-06 [primary]**
   https://developer.android.com/develop/ui/views/haptics/haptics-apis

3. **Android Developers, Notification runtime permission [primary]**
   https://developer.android.com/develop/ui/views/notifications/notification-permission

4. **Android Developers, Display time-sensitive notifications [primary]**
   https://developer.android.com/develop/ui/views/notifications/time-sensitive

5. **Chrome for Developers, Notification Triggers API, published 2019-10-24 [primary]**
   https://developer.chrome.com/docs/web-platform/notification-triggers

6. **web.dev, Push notifications FAQ [primary]**
   https://web.dev/articles/push-notifications-faq

7. **MDN, Notification interface**
   https://developer.mozilla.org/en-US/docs/Web/API/Notification

8. **MDN, Notification: vibrate property, last modified 2024-03-06**
   https://developer.mozilla.org/en-US/docs/Web/API/Notification/vibrate

9. **ROIBest, PWA Push Notifications on Android, published 2026-08-11**
   https://www.roibest.com/resources/blog/pwa-push-notifications-android/

10. **apkservice.mobi, Native Android Push vs Web Push (PWA): 2026 Open Rate Comparison, updated 2026-04-29**
    https://apkservice.mobi/native-android-push-vs-web-push/

11. **InstantPWA, Web push vs native push: what actually differs in 2026, published 2026-05-08**
    https://instantpwa.com/blog/web-push-vs-native-push

12. **MobiLoud, How to Set Up Push Notifications for Your PWA, updated 2026-08-12**
https://www.mobiloud.com/blog/pwa-push-notifications/

13. **OpenPWA, Web Push and PushManager.subscribe()**
https://openpwa.net/reference/notifications/web-push/