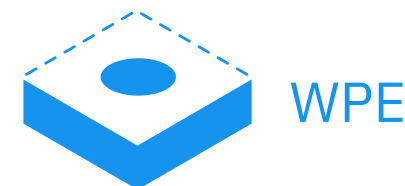


WPE Hands-On: Writing a Launcher for Embedded Devices with the New WPEPlatform API



Mario Sánchez-Prada
Prague, October 8th 2026

About Me

- **Software Engineer** and partner at **Igalia**
- **Open Source contributor**: GNOME, Chromium, WebKit
- **Other work**: Linux-based OSes (e.g. Endless OS), Flatpak, Samsung Smart TV, Maemo (Nokia)
- Currently **coordinating** Igalia's **WebKit team**

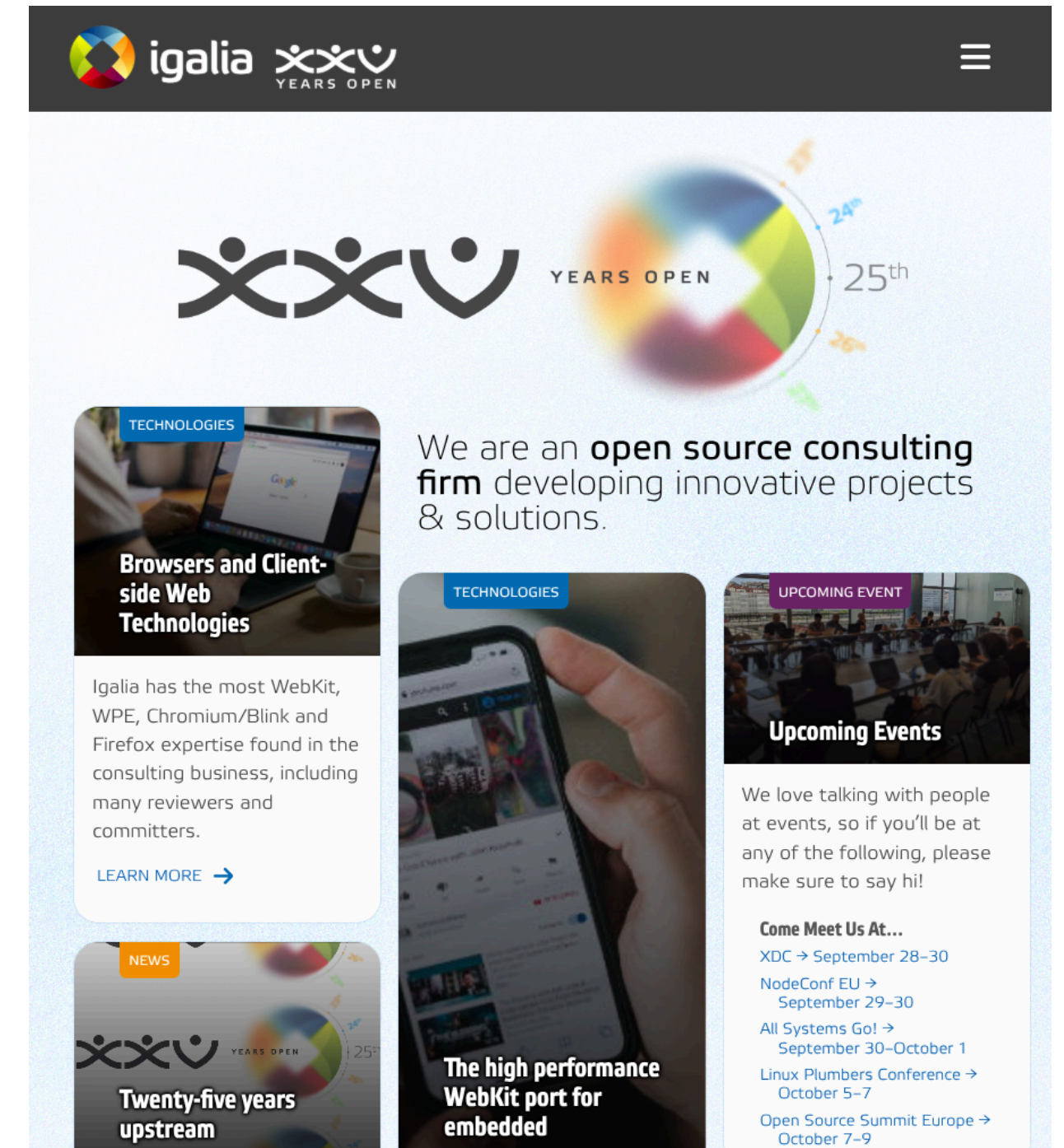
Contact me: mario@igalia.com



About Igalia

- **Specialized Open Source consultancy**
 - Founded in A Coruña (Spain) in 2001.
 - Fully remote and with a flat structure.
- **Top contributors to the main Web Engines**
 - Second-largest contributor to WebKit after Apple.
 - Maintainers of the two Linux WebKit *ports*.
- **Other OSS work:**
 - Graphics, multimedia, compilers, Linux kernel...
- **Members of different working groups:**
 - W3C, WHATWG, TC39, Test262, Khronos...

<https://www.igalia.com>



Slides in PDF:



Outline

1. What Is WPE WebKit?
2. The Old Model: libwpe and WPE Backends
3. WPEPlatform: The New Model
4. Hands-On: Writing a Launcher
5. Live Demo
6. Migrating Existing Integrations



What Is WPE WebKit?



What Is WPE WebKit?

Official port of the WebKit Web engine for embedded Linux devices.

- **Modern Web Platform** implementation, with a focus on:
 - Performance, security, low resource footprint, flexibility.
- **Backend-based I/O**: DRM, Wayland, headless or custom
- **Hardware acceleration** across many different SoCs
- **Ships in many commercial products**:
TVs, set-top boxes, home appliances, digital signage...



 <https://wpewebkit.org>



From Codebase to Products



- Runs in **very different products** and **millions of shipped devices**
 - e.g. set-top boxes, smart TVs, **appliances**, signage, POS, industrial HMI's...
- All of them needed **to embed/integrate WPE** into their platforms
- WPE's great flexibility had a cost: **integration was complex...** until now



What Is WPE WebKit?

[illegible]

WPE WebKit built on September 25th 2026

WPE Hands-On: Writing a Launcher for Embedded Devices with the New WPEPlatform API

Mario Sánchez-Prada, Open Source Summit Europe 2026



The Old Model: libwpe and WPE Backends



Embedding WPE with the (Now) Legacy API

At least **three upstream projects to integrate** into your product

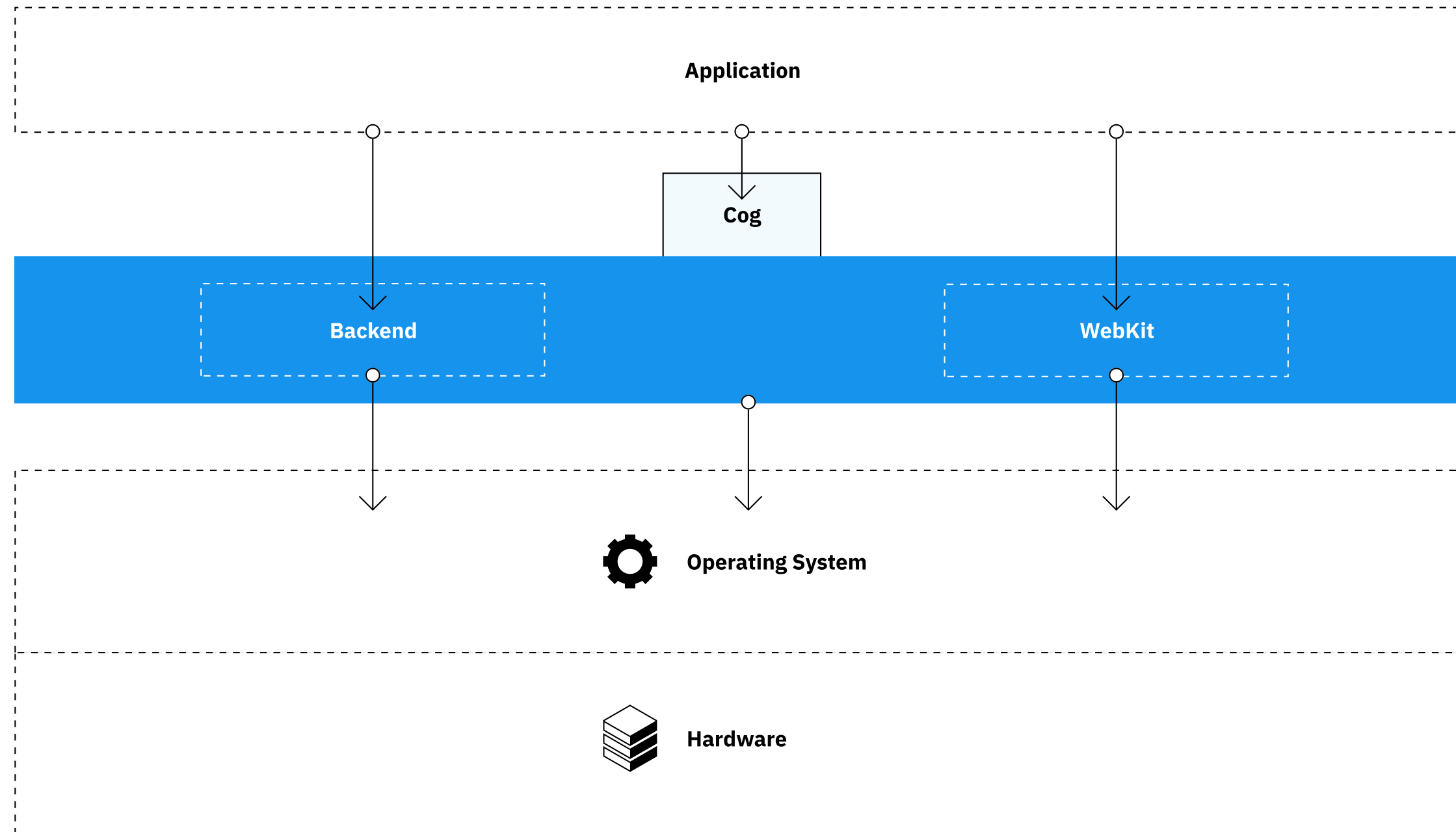
-  **WPE WebKit** (WebKit repository): renders the page into **buffers**, never to a screen.
-  **libwpe** (out of tree): **defines** the platform contract, implements nothing.
-  **A backend** (out of tree): **implements** that contract, loaded by **.so** name.
⇒ **WPEBackend-fdo** was the *de-facto* backend for most integrations.

...and then a fourth one: **Cog** (or your own launcher)

i.e. renders buffers from WPE to screen and handles input events



The Old Design



The **platform layer** was **scattered** across **libwpe**, the **backend** and the **launcher**

`libwpe` + `WPEBackend-fdo` + `Cog`: ~26,600 lines across 3 repos



Issues With The Old Model

- 🔗 **Multiple versions:** four release cadences, one support matrix
- 🧩 **Shared glue existed, but out of tree:** the **Cog** launcher
- 🚪 **Not using Cog?** Need to do **platform-specific work**
⇒ You could use `libcogcore`, still another out-of-tree dependency.
- 📦 **Out-of-tree code** changes independently from WebKit's internals
⇒ Bugs harder to detect (and fix!)



WPEPlatform: The New Model



Enter WPEPlatform

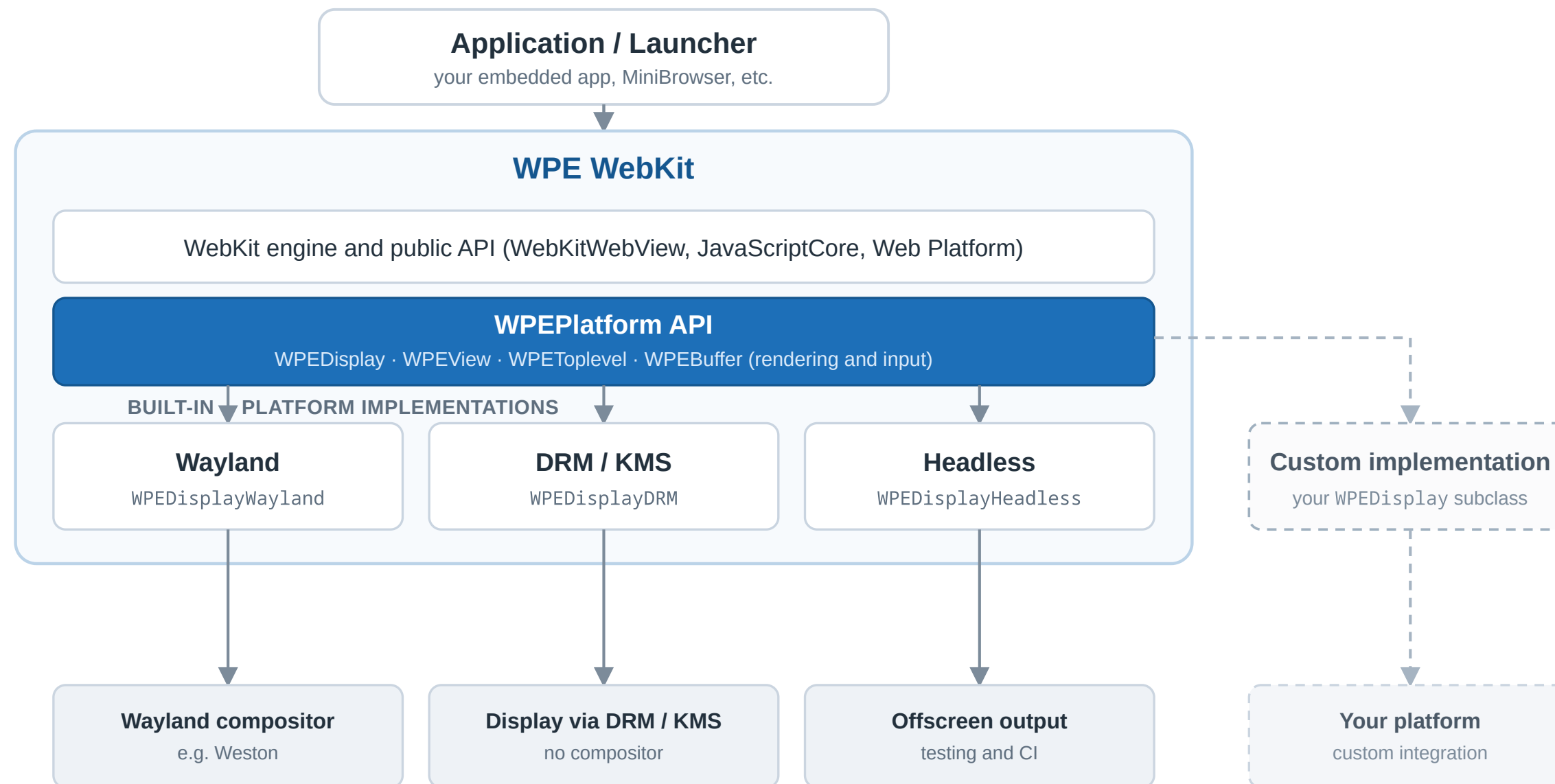
-  **In-tree:** lives inside WebKit, in [Source/WebKit/WPEPlatform/](#)
-  **Public GObject API**, versioned and released with WebKit itself
-  **Built-in support** for **Wayland**, **DRM/KMS** and **headless**
-  **Extensible:** write your own for other platforms (e.g. [WPEPlatformGTK](#))
-  **Stable/recommended API** as of WPE WebKit 2.54 ([announcement](#))

 <https://wpewebkit.org/reference/stable/wpe-platform-2.0>

Note: Cog, libwpe and its backends are still available for now, but will be deprecated



The New Design



The **platform layer** moved **inside** WPE WebKit, as part of the **WPEPlatform API**

All within one library: ~28,600 lines in WebKit itself



Four Main Classes to Learn

Class	Role
<code>WPEDisplay</code>	The connection to a platform. Root of everything else.
<code>WPEToplevel</code>	A native top-level window (or its equivalent).
<code>WPEView</code>	The rendering surface for one web view.
<code>WPEBuffer</code>	Pixel data produced by WebKit, handed to the view.

- **Platform implementers** subclass `WPEDisplay`, `WPEToplevel` and `WPEView`
 - `WPEBuffer` is **not** platform-specific (e.g. `WPEBufferDMABuf`, `WPEBufferSHM`)
- **Other:** `WPEScreen`, `WPESettings`, `WPEEvent`, `WPEKeymap`...



Three Built-In Platforms

Platform	<code>pkg-config</code> module	Typical use
Wayland	<code>wpe-platform-wayland-2.0</code>	Devices with a compositor
DRM/KMS	<code>wpe-platform-drm-2.0</code>	Devices owning the display (e.g. kiosks)
Headless	<code>wpe-platform-headless-2.0</code>	Tests, capture, render pipelines

- All compiled into `libWPEWebKit-2.0.so` (no separate per-backend library)
- **Optional** at build time, **enabled by default**, **automatically selected**
- **External platforms** are **not bundled in** (external modules or linked by the app)



Zero-Copy Buffer Sharing

- 🎬 A frame reaches the screen without ever being copied
- 🔓 Web process never touches the display: it renders offscreen
- 🤝 It reaches the UI process as a **GPU platform buffer**, not pixels
 - ⇒ **DMA-BUF** on Linux, **AHardwareBuffer** on Android.
- 💻 Handed straight to the platform: **wpe_view_render_buffer()**
- 🧮 **Format negotiation**: FourCC + **modifiers** pick the **GPU's own layout**
 - ⚠️ Needs **direct scanout**: guaranteed on **DRM**, the compositor's call on **Wayland**
 - 🐢 No GPU buffer sharing? **WPEBufferSHM** keeps it running, **but it copies**



Hands-On: Writing a Launcher



A Simple WPE Launcher

```
1 #include <wpe/webkit.h>
2
3 int
4 main (int argc, char *argv[])
5 {
6     const char *uri = (argc > 1) ? argv[1] : "https://wpewebkit.org";
7
8     g_autoptr(WebKitWebView) web_view =
9         WEBKIT_WEB_VIEW (g_object_new (WEBKIT_TYPE_WEB_VIEW, NULL));
10
11     g_autoptr(GMainLoop) loop = g_main_loop_new (NULL, FALSE);
12
13     webkit_web_view_load_uri (web_view, uri);
14     g_main_loop_run (loop);
15
16     return 0;
17 }
```



A (Slightly More) Complete Launcher

```
1 #include <wpe/wpe-platform.h>
2 #include <wpe/webkit.h>
3
4 static void
5 on_view_closed (WPEView *view, gpointer user_data)
6 {
7     g_main_loop_quit ((GMainLoop *) user_data);
8 }
9
10 int
11 main (int argc, char *argv[])
12 {
13     const char *uri = (argc > 1) ? argv[1] : "https://wpewebkit.org";
14
15     WPEDisplay *display = wpe_display_get_default ();
16     if (!display) {
17         g_printerr ("Could not connect to a WPE display.\n");
18         return 1;
19     }
20
21     g_autoptr(WebKitWebView) web_view =
22         WEBKIT_WEB_VIEW (g_object_new (WEBKIT_TYPE_WEB_VIEW,
23                                         "display", display, NULL));
24
25     g_autoptr(GMainLoop) loop = g_main_loop_new (NULL, FALSE);
26     WPEView *wpe_view = webkit_web_view_get_wpe_view (web_view);
27     g_signal_connect (wpe_view, "closed",
28                       G_CALLBACK (on_view_closed), loop);
29
30     webkit_web_view_load_uri (web_view, uri);
31     g_main_loop_run (loop);
32
33     return 0;
34 }
```

1. 🔍 Get a display

Checks each registered platform in priority order.

2. 🖼️ Create the web view

Creates `WPEView` with the given `WPEDisplay`.

3. 🔄 Create main loop and monitor `closed` event

⚠️ Wayland-only: no support on DRM/headless.

4. 🚀 Load the URI and run the main loop

Events will run on the default GLib main context.

The launcher will **keep running** until the "window" closes
and the `closed` signal is emitted (and handled)



Handling Input

One signal for everything: **WPEView::event**

```
static gboolean
on_view_event (WPEView *view, WPEEvent *event, gpointer user_data)
{
    if (wpe_event_get_event_type (event) != WPE_EVENT_KEYBOARD_KEY_DOWN)
        return FALSE;

    if ((wpe_event_get_modifiers (event) & WPE_MODIFIER_KEYBOARD_CONTROL)
        && wpe_event_keyboard_get_keyval (event) == WPE_KEY_q) {
        quit ();
        return TRUE;    /* consumed: the page never sees it */
    }
    return FALSE;
}

...

WPEView *view = webkit_web_view_get_wpe_view (web_view);
g_signal_connect (view, "event",
                  G_CALLBACK (on_view_event), NULL);
```

Keyboard, pointer, scroll and touch all arrive here, with **typed accessors**



Driving the "Window"

The view lives in a **WPEToplevel**, which acts as your window *handle*

```
WPEView *view = webkit_web_view_get_wpe_view (web_view);
WPEToplevel *toplevel = wpe_view_get_toplevel (view);

if (toplevel) {
    wpe_toplevel_set_title (toplevel, "Hello WPE");

    if (wpe_toplevel_get_state (toplevel) & WPE_TOPLEVEL_STATE_FULLSCREEN)
        wpe_toplevel_unfullscreen (toplevel);
    else
        wpe_toplevel_fullscreen (toplevel);
}
```

- Also available: `wpe_toplevel_maximize|minimize|resize()` (and variants)
- You can connect to `WPEView::toplevel-state-changed` to handle state changes
- **Support differs** across platforms: all on Wayland, size + fullscreen on headless
⇒ **No support** for state changes **on DRM** (you already own the whole screen).



Building It

- Two `pkg-config` modules: `wpe-webkit-2.0` and `wpe-platform-2.0`
⇒ No need to check for `wpe-platform-2.0` (required by `wpe-webkit-2.0`)

CMake

```
pkg_check_modules(WPEWebKit REQUIRED
  IMPORTED_TARGET wpe-webkit-2.0)

add_executable(my-launcher main.c)
target_link_libraries(my-launcher PRIVATE
  PkgConfig::WPEWebKit)
```

Meson

```
wpe_webkit_dep =
    dependency('wpe-webkit-2.0')

executable('my-launcher', 'main.c',
  dependencies: [wpe_webkit_dep])
```

No per-backend library: built-in backends part of `libWPEWebKit-2.0`



Choosing a Platform

Portable (automatic selection)

```
WPEDisplay *display =  
    wpe_display_get_default ();
```

Tries each platform in priority order
Same binary runs anywhere

Pinned to a backend (e.g. Wayland)

```
g_autoptr(WPEDisplay) display =  
    WPE_DISPLAY (wpe_display_wayland_new ());  
  
if (!wpe_display_connect (display, &error))  
    return 1;
```

Needs `wpe-platform-wayland-2.0`
and `<wpe/wayland/wpe-wayland.h>`

- **Important:** Pinned displays are **owned** by you and **pending connection** upon creation ⚠
- **Pick it also at runtime:** `WPE_PLATFORM=wayland|drm|headless` (no rebuild needed)



Implement Your Own Platform

- 🎯 Only if your target is **none of** Wayland, DRM/KMS or headless:
 - 🌳 **Subclass** `WPEDisplay`, `WPEView` and `WPETopLevel`.
 - 🔌 **Ship it as an auto-discovered module** or **link it directly** into your launcher.
 - 🧩 You can also **extend a built-in instead**, if needed (e.g. Wayland).
 - 📖 Reference implementation to read: [WPEPlatformGTK](#).
 - ⇒ Provides integration with the GTK toolkit.
- ⚙️ `WPESettings` covers the smaller knobs: default toplevel size, dark mode, key repeat...





Live Demo

~150 lines of code written in C, one `meson.build` file

Slightly bigger than the previous examples due to the callbacks
(convenient for the demo), but essentially the same thing

Source code: <https://github.com/mariospr/oss-eu-2026-demo>



Migrating Existing Integrations



Coming From Your Own Launcher: Mostly Deleting

Before (WPEBackend-fdo)

```
struct wpe_view_backend *wpe_backend =  
    /* ...fdo exportable backend... */;  
  
WebKitWebViewBackend *backend =  
    webkit_web_view_backend_new (wpe_backend,  
                                NULL, NULL);  
  
WebKitWebView *web_view =  
    g_object_new (WEBKIT_TYPE_WEB_VIEW,  
                 "backend", backend, NULL);
```

After (WPEPlatform)

```
WebKitWebView *web_view =  
    g_object_new (WEBKIT_TYPE_WEB_VIEW, NULL);
```

Your settings, network session, navigation and signal handlers carry over **unchanged**.



Gone with it: EGL export callbacks, buffer release, input dispatch

Migration mapping table: wpewebkit.org/reference/stable/wpe-platform-2.0/migration-mapping.html



If You Are Using Cog



Cog has been the standard WPE launcher for years, and **still works**

⇒ Still **built on the legacy API** (i.e. libwpe and its platform modules).

⇒ The migration path is **writing the launcher we wrote earlier**.



Ask what you **actually use Cog for**

⇒ Many products need only a URL, fullscreen and a few shortcuts.



Port the features you need, one at a time

⇒ They are now **small**.

If you **relied on Cog for platform work** ⇒ **WPEPlatform does it now** 🎉



How to Get Started

-  **WPE WebKit 2.54 or newer** for the stable WPEPlatform API
-  **Check your build has it:** `pkg-config --exists wpe-platform-2.0`
-  **Try Wayland first** if you have a compositor (any desktop or laptop)
 - ⇒ It's the most complete backend: full input and window control
-  **Also consider headless**, very useful to get it running in CI
 - ⇒ No display, compositor or device needed



Conclusion

Embedding WPE used to mean managing four different projects

- WPE WebKit, libwpe, an out-of-tree backend, and a launcher (e.g. Cog).

WPEPlatform is in-tree, stable since 2.54

- Support for Wayland, DRM/KMS and headless included.
- Four classes to learn to do platform-specific work.

A simple launcher is just a few lines and one file

- No backend, no buffer management, no input dispatch.
- Zero-copy buffer sharing across processes via GPU platform buffers.

Migrating your own launcher is mostly deleting code

- With Cog, just port only the product-specific bits: the platform work is done for you.

You can start today, with no hardware

- Check for `wpe-platform-2.0`, run on wayland or headless, then move to the device.



Further Resources

WPEPlatform:

API reference & tutorials: <https://wpewebkit.org/reference/stable/wpe-platform-2.0>

Check: *Migrating from libwpe*, and the symbol-by-symbol *Migration mapping table*

In-tree docs: <Source/WebKit/WPEPlatform/docs/>

WPE WebKit:

Website: <https://wpewebkit.org>

Mailing list: <https://lists.webkit.org/mailman3/lists/webkit-wpe.lists.webkit.org>

Matrix: [#wpe:matrix.org](#)

WebKit:

Website: <https://webkit.org>

Documentation: <https://docs.webkit.org>

Mailing list: <https://lists.webkit.org/mailman3/lists/webkit-dev.lists.webkit.org>



Slides in PDF:



Questions?

Mario Sánchez-Prada

mario@igalia.com



WPE



