Wayland Support in Open Source Browsers

Xavier Castaño García

xcastanho at igalia dot com

GENIVI®

igalia

Myself, Igalia and Web Browsers

- Co-founder of Igalia in 2001. 65 engineers. Global
- Open Source consultancy: web browsers, multimedia, graphics, compilers, networking
- Igalia among the top contributors to upstream web browsers WebKit/JSC, Chromium/V8, Firefox/Servo/SpiderMonkey
- Working with the industry: automotive, tablets, phones, smart tv, set-top-boxes and several other embedded devices manufacturers

Outline

- Part I: Brief review on Wayland support on Open Source Web Browsers
- Part II: Wayland support in Chromium
- Part III: WebKit and WPE
- Part IV: Conclusions

Part I: Brief review on Wayland support on Open Source Web Browsers

Motivation

- Wayland is a mature solution
- Demand from different industries
  - Automotive
  - Mobile
  - Desktop
- Current alternatives on the Open Source web browsers arena:
  - Mozilla: Firefox(Gecko/Servo) / SpiderMonkey
  - Chromium / Blink / V8
  - WebKit / JSC

Mozilla Gecko

- Powers the Firefox browser
- Embedding not officially supported. Monolithic architecture
- Several open source browsers moved away from Gecko to WebKit about 10 years ago
- Red Hat is working in Wayland support for Gecko. Basic functionality

Mozilla Servo

- Next generation engine
- Designed for memory-safety, parallelism, embedding
- New set of tools and technologies: Rust
- Currently under heavy development. Too soon
- Preliminary Wayland support by Samsung Open Source Group

Chromium

- Vertical solution, from low-level graphics to UX
- Very powerful and feature complete
- Engineered to power Chrome and Chrome OS
  - Embedding, portability use cases are secondary. Fork is needed
- Designed to minimize external dependencies
  - External deps are managed by the project build system
  - Versions pinned, included in the build process
  - In general, not designed to exchange subsystems

Chromium & Wayland

- Two different efforts on having a native Wayland support:
  - Legacy Ozone-Wayland project (01.org)
  - New Wayland backend by Igalia

Chromium ecosystem

- External projects filling the gaps
  - CEF: Chromium Embedded Framework
    - Embed web content (WebView) in native applications
    - Hybrid web/native applications
    - Downstream Wayland support based on new Igalia's Wayland backend
  - QtWebEngine
    - Embed web content in Qt applications
    - Wayland support since Qt 5.10
    - Slower upgrade pace, linked to Qt releases
    - Commercial and GPLv3 license

Chromium ecosystem

- External projects filling the gaps (cont.)
  - Electron or NW.js (node-webkit)
    - Write apps with JS and HTML integrated with Node to access low level system from web pages
    - Pack Chromium and Node.js to build desktop apps with web technology
    - Lack of Wayland support

WebKit

- Powerful and complete
- Very flexible architecture (ports)
- Each port is an engine implementation with a stable API and a specific set of technologies (network, graphics, multimedia)
- Many ports:
  - Upstream: iOS/OSX, GTK+, WPE
  - Downstream: EFL, Qt, Sony,...

WebKit ports

- WebKitGTK+
  - Stable and also lightweight
  - Active development
  - WebKitGTK+ support Wayland
- QtWebKit
  - Officially abandoned in favor of Chromium-based QtWebEngine
  - Unofficial, volunteer-driven maintenance. Upgraded to latest Qt versions
  - Wayland support provided by Qt toolkit

WebKit ports

- WPE
  - Very lightweight, low hardware requirements
  - Strong multimedia capabilities
  - Backends enable Wayland support.

Part II: Wayland support in Chromium

Legacy Ozone-Wayland project

- Legacy, in-production Wayland implementation
- Developed mainly by Intel (01.org)
  - https://github.com/01org/ozone-wayland
- Currently in maintenance mode
  - Good community support
  - No more active development
    - No new features, no implementation of existing gaps
  - Latest supported version by Intel was 53

Legacy Ozone-Wayland project

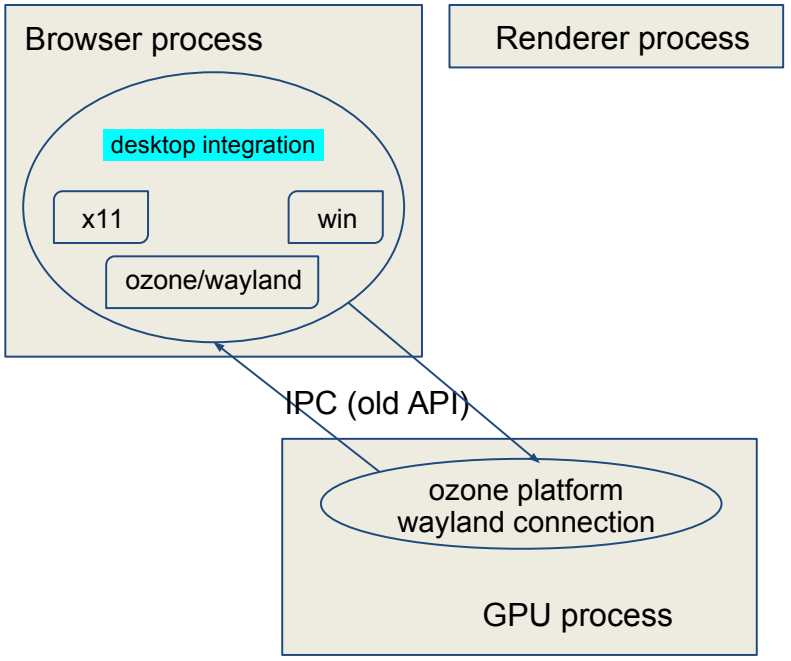
- Later maintained by LGe
  - https://github.com/lgsvl/chromium-src
  - LGe has been updating it until 64 so far
- Current Chromium stable is 65 (66 in beta)

Legacy Ozone-Wayland project

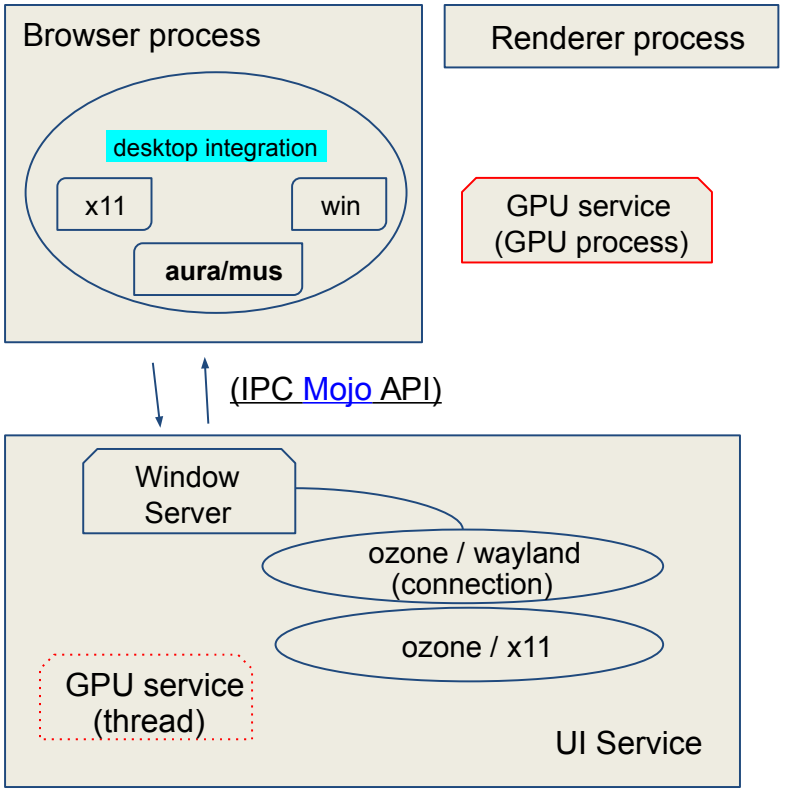
- This is the project currently used at GENIVI Development Platform
  - Not recommended for new products, plan to phase it out
  - Current release provides Chromium 64
- Why not merge Intel's backend upstream?
  - Blocker: architecture differences
    - Intel's code doesn't align with Chromium mid-term architecture plans

Chromium architecture now

Linux desktop integration (01.org)



Mus Linux desktop integration



New Wayland backend by Igalia

- New project hosted at:
  - https://github.com/Igalia/chromium
- More than one year of development so far
  - Developed by Igalia
  - Supported by Renesas
- Wide array of features currently implemented
  - XDG v6, keyboard, mouse & touch input, common window management, menus & tooltips, clipboard...
  - Main gaps: drag & drop, multi-screen, performance improvements.

New Wayland backend by Igalia

- Development process
  - Start from scratch, follow modern Chromium conventions and architecture
  - Buildbot running existing tests
  - Peer review
  - Track Chromium master
    - weekly rebases
    - continuous history clean up

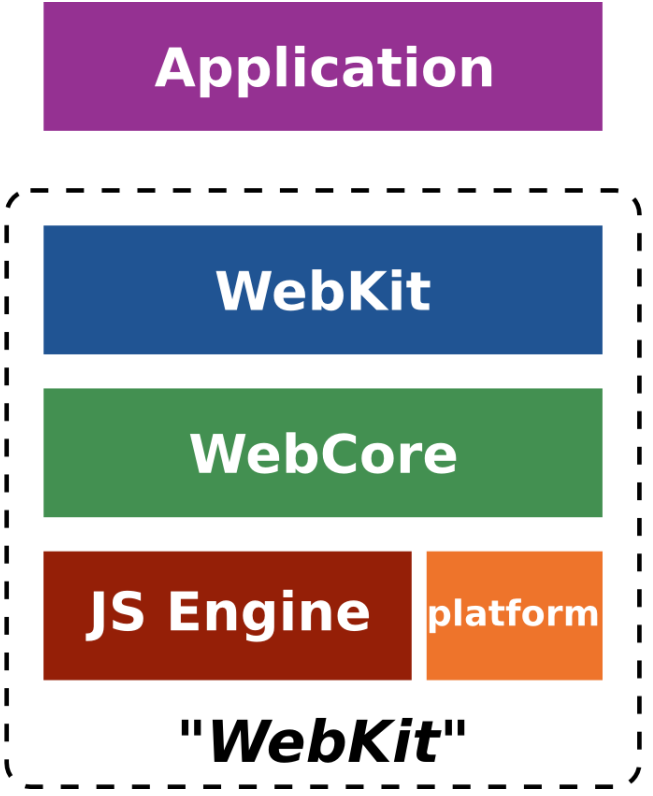
New Wayland backend by Igalia

- Ongoing upstreaming process
  - Periodic sync-up with Google
  - Shared design document. Live and dynamic document.
- First step: undo ChromeOS assumptions from new architecture
  - New architecture only officially used on ChromeOS → Google developers assumed ChromeOS use cases
  - Specifically: ChromeOS has one big container window
- Discussion and next steps in BlinkOn (currently happening!)

Part III: WebKit and WPE

WebKit. What is a port?

- From a simplified point of view, WebKit is structured this way:
  - WebKit: thin layer to link against from the applications
  - WebCore: rendering, layout, network access, multimedia, accessibility support...
  - JS Engine: the JavaScript engine. JavaScriptCore by default.
  - platform: platform-specific hooks to implement generic algorithms



WPE

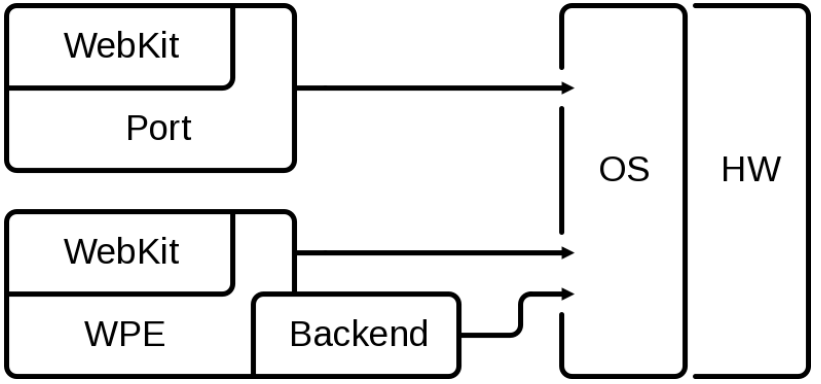
- Main use case: full-screen web content.
- Derives from WebKitGTK+
- Designed for simplicity and performance
- Toolkit and platform agnostic
- Gstreamer for media and JSC as JavaScript engine
- Reduces dependencies to a few common libraries:
  - Glib, FreeType, HarfBuzz, GnuTLS, pixman, cairo, libsoup
- GLES 2.0 for hardware accelerated rendering

WPE use cases

- Great performance in low-end hardware
  - Raspberry Pi 1/zero
- Very low memory footprint
  - A functional Raspberry Pi image can be about 40Mb
  - Possible to limit memory consumption (100Mb for a standard setup)
- Supports Wayland and also other backends
- Strong multimedia capabilities
- Well received in set-top-box market. Official part of RDK stack

WPE backends

- Main goal: efficient cross-process GPU buffer sharing
- Backends use platform-specific libraries to implement drawing and window management
- Can be independently developed
- Vulkan support down the line



Available WPE backends

- Libgbm: Intel, AMD, open source NVidia drivers for embedded devices (i.e. Jetson) – specific to Mesa driver
- Wayland-egl: uses Wayland as the protocol internally, can be used by Mesa as well as ARM Mali drivers
- LibWPEBackend-rdk covers 4-5 different stacks (Rpi, IntelCE, bcm-nexus via the native API, bcm-nexus via Wayland, westeros – RDK oriented compositor -)
- Working on an experimental libWPEBackend-android

WPE present and future

- Heavily developed during 2015-2017
  - Sponsored by Metrological
- Upstream since May 2017
- Stable Igalia team working on it
- Since 2017:
  - RDK consortium adopted the technology (>10M STB)
  - Different kinds of embedded devices companies adopting WPE
  - Automotive companies already considering it

WPE present and future

- Releases
- QA infastructure
- Documentation
- New graphics architecture
- Networking & Security
- JSC improvements on 32 bits
- More web standards (WebDriver, WebGL2, WebVR...)

WPE repositories

- Upstream
  - https://webkit.org/getting-the-code
- Downstream
  - https://github.com/WebPlatformForEmbedded

IV. Conclusions

Conclusions

- Wayland is a mature solution
  - Support in major Linux distros
  - Automotive industry
- Browsers are highly-demanding software
  - XWayland hurts performance
  - Native support is required

Conclusions

- Native Wayland support in major Linux browsers
  - Chromium: work by Igalia
    - Quite complete, upstream process ongoing
    - Projects in Chromium ecosystem are waiting for upstream support
  - WebKit: work by Igalia
    - Support is complete and published upstream
    - WPE for embedded, WebKitGTK+ for desktop and embedded
  - Firefox/Gecko: work by Red Hat
    - Work in progress, available in developer Nightly previews

GENIVI®