Update on the open source browser space

Jacobo Aragunde Pérez
blogs.igalia.com/jaragunde
• Open Source experts and consultants
• 15 years of experience
• Important contributions to:
  • Client-side web technologies: WebKit, Blink/Chromium, Servo
  • Graphics & Multimedia: Mesa, GStreamer
  • Compilers: V8, JavaScriptCore, SpiderMonkey, Guile
  • Software-defined networking: Snabb
  • ...
Outline

- Open source browser technologies
- Wayland support in Chromium
- WPE: support for Wayland and other backends in WebKit
- Other options
Open source browser technologies
Open source web platforms

- Mozilla: Gecko and Servo
- WebKit family: OS X/iOS, WebKitGTK+, WPE
- Chromium and related projects
Mozilla

• Gecko engine
  – Powers the Firefox browser
  – Embedding not officially supported

• Servo: next generation engine
  – Designed for memory-safety, parallelism, embedding
  – New set of tools and technologies: Rust
  – Currently under heavy development
WebKit

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...
- platform: platform-specific hooks to implement generic algorithms
WebKit ports

• Each port is an engine implementation with a specific set of technologies
  • Platform bits: network, graphics, multimedia
  • Specific API
• Many ports have existed: OS X and iOS, WebKitGTK+, EWebKit (EFL), QtWebKit, Chrome/Chromium..
• Currently official ports: OS X/iOS, WebKitGTK+, WPE (in process)
Chromium

- Vertical solution, from low-level graphics to UX
- Engineered to power Chrome and Chrome OS
  - Embedding, portability use cases are secondary
- 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 ecosystem

• External projects filling the gaps
• **CEF**: Chromium Embedded Framework
  • Embed web content in applications
  • Hybrid applications
• **Electron**
  • Web application runtime
Wayland support in Chromium
Ozone-Wayland project

- Most complete Wayland implementation yet
- Developed mainly by Intel
- Downstream project at [github](https://github)
- Currently in maintenance mode
  - No more active development
  - Latest supported version is 53
Upstream Wayland implementation

- Preliminary state
- Following Chromium master
- Not high priority at Google → Igalia taking the lead of the implementation
- Framed in a bigger effort to re-architect Chromium
Upstream Wayland implementation

• Why not merge Intel’s code upstream?
  • Blocker: architecture differences
    – Intel’s code doesn’t align with Chromium mid-term architecture plans
  • Approach: implement basic bits following new architecture, then migrate features and code as possible
Chromium architecture now
Long-term plan: service-based
WPE: support for Wayland and other backends in WebKit
WPE

- **Web Platform for Embedded**
  - Previously known as *WebKit For Wayland*
- Designed for simplicity and performance
- Supports Wayland and also other backends
- Great performance in low-end hardware
- Currently in review process to become an official WebKit port
WPE use cases

- Strong multimedia capabilities
- Very lightweight, low hardware requirements
  - Raspberry Pi 1/zero
- Well received in set-top-box market
- Official part of RDK stack
WPE backends

- Backends use platform-specific libraries to implement drawing and window management
- Can be independently developed
Other options
Other options with Wayland support

- WebKitGTK+
  - Wayland through the GTK+ toolkit support
- QtWebEngine
  - Chromium-based
  - Wayland through the Qt toolkit support
Conclusions

- Chromium
  - Full-featured browser and fast-paced development
  - Increased cost of maintenance
- Intel’s Ozone-Wayland
  - Available for short-term goals
  - Transition to upstream Chromium implementation as soon as it’s ready
- QtWebEngine
  - Ideal to integrate with Qt applications
  - Slower upgrade pace, linked to Qt releases
Conclusions

- WPE
  - Lightweight
  - Customizable graphic backends
  - Stable APIs, designed for third-parties to build products upon
  - No browser features, it’s a web engine
- WebKitGTK+
  - Stable and also lightweight
  - Availability linked to the GTK+ toolkit