From 'WebKit for Wayland' to WPE

Žan Doberšek
Igalia

e-mail	zdobersek@igalia.com
twitter	@falconsigh
www	blogs.igalia.com/zdobersek

Defining WPE

A new WebKit port, started in 2014.

Not tied to any toolkit or Linux environment/platform.

Result is some additional freedom, some additional troubles.

Defining WPE

Origins of the port are in WebKitGTK+.

The two ports still share most of non-GNOME dependencies.

Libglib, Libsoup, GStreamer, FreeType, Harfbuzz, Cairo, ...

Origin – WebKit for Wayland

Accurate name for a cheap first approach.

UIProcess is loaded as a shared library into the Wayland compositor.

WebProcess acts as the only fullscreen EGL client of that compositor.

It ... works.

Origin – WebKit for Wayland

Bad design.

UIProcess shouldn't exist in the same process as the system-wide compositor.

Can't show a second Web view.

Developing the idea

Let's provide:

- a platform-generic solution for processing Web content,
- simple adaptive approach to displaying processed graphical output.

Developing the idea

WebProcess uses EGL to render the scene.

The graphics buffer is shared with the UIProcess.

UIProcess takes care of displaying the content in platform-specific, use-case-specific way.

WPE

'WPE' - three-letter acronym used to define the port in WebKit.

Rather ambiguous meaning:

- **W**eb **P**latform **E**ngine
- **W**ebKit **P**ort **E**xperiment
- **W**ebKit **P**ara **E**mbebidos
- **W**ebKit **P**ure **E**mbedded

libWPE

A small C library defining platform-agnostic interfaces.

Minimal facility used for loading shared libraries providing implementations for those interfaces.

libWPE

wpe_renderer_host - per UIProcess.

wpe_view_backend - per view/page, in the UIProcess.

wpe_renderer_backend_egl - per WebProcess.

wpe_renderer_backend_egl_target - per view/page, in the WebProcess.

The related interfaces are able to coordinate between them.

libWPE

Input interfaces – used to pipe input events to the relevant view/page.

Pasteboard interfaces – for pasteboard facilities.

More? As required.

libWPE–mesa

Reference implementation.

Depends on the general modern Linux graphics stack.

Linux + libdrm + Mesa.

libWPE–mesa

Uses rendering nodes, libgbm to manage graphics buffers.

Nested compositor implementation pending review.

View backends support Wayland, DRM.

libWPE-mesa

Exportable view backend - hands off graphics buffers to the user.

EGLImage would be delivered to the user, can be used in other scenes.

Enables headless mode - snapshotting graphical output without visually presenting it.

API

WebKit2 C API.

A few additions specific to the WPE port (i.e. WKView).

No radical changes expected in the long-term.

Relations with upstream

The ultimate goal is to upstream the port, stay close to upstream until then.

Weekly releases until this summer, now rotating to mothly releases.

Pulls from upstream would still be done more frequently.

Roadmap

A bunch of work awaits!

Most (all?) of this applies to the GTK+ port as well.

Roadmap

Graphics:

- Display lists
- Additional composition optimizations
- Separate-thread image decoding
- GLES3 support
- Experiment with Vulkan
- Rewrite the graphics stack?

Roadmap

Media:

- MSE support
- WebM improvements (for MSE usage)

Roadmap

JavaScriptCore:

- Not much to add to a state-of-the-art engine
- ... except improving MIPS support

Roadmap

Standards:

- Area where WebKit likely lacks the most, but is improving
- A lot of low-hanging fruit
- Existing WebCore implementations with missing platform-specific parts

Roadmap

Usability:

- A small Web browser for testing purposes on desktop
- Suppport for customizable theming (via CSS)

Roadmap

Networking:

- HTTP2
- A bunch of security features
- Libsoup improvements

Roadmap

WPE, WPE-mesa currently reside in the WebKit repository, under Source/ThirdParty/.

They should be moved out into standalone repositories.

A precondition for upstreaming.

Roadmap

Address the WebKitGTK+ origin:

- Investigate the possibilities of basing the GTK+ port on WPE
- Possible to apply the relevant GNOME dependencies on WPE?
- Would the result of that be a WebKitGTK+ equivalent in all regards?

The goal would be to support two ports for the cost of one.

Questions?

And thank you!