WPE WebKit

HTML5 user interfaces for embedded devices

Juan José Sánchez Penas

Embedded Linux Conference

Prague, October 2017

jjsanchez@igalia.com | www.igalia.com

Myself, Igalia and Web Browsers

- Co-founder of Igalia in 2001. 60 engineers. Global
- Open source consultancy: 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: tablets, phones, smart tv, set-top-boxes, automotive and several other embedded devices manufacturers

igalia

Outline of the talk

- Part I: Why WPE? Problem and proposed solution
- Part II: What exactly is WPE? Architecture and features
- Part III: Where is WPE going? Current status and future

igalia

PART 1

Why WPE?

Problem and proposed solution

The problem: HTML5 and embedded devices

- Embedded devices are getting sophisticated
- Many have (or will have) GNU/Linux with a touch screen and will run apps
- The web is powerful, flexible and a comfort zone for many application developers
- Frequent use case: full-screen (kiosk mode web apps)
- Still low-end or mid-end hardware (limited resources, optimizations needed)

The solution: browser for embedded devices

- Focus on lightweight
- No need to solve all the use cases and needs
- Open source options we can base it on:
  - Firefox (Servo) / SpiderMonkey
  - Chromium / Blink / V8
  - WebKit / JSC

igalia

Firefox (Servo) / SpiderMonkey ?

- Years ago Mozilla decided not to target other browser developers
- Several open source browsers moved away from Gecko to WebKit about 10 years ago
- Firefox/Gecko has a quite monolithic architecture today
- Things might get better with Servo, but it is too soon

Chromium / Blink / V8 ?

- Very powerful and feature complete
- Not a flexible architecture
- No stable API provided for derived browsers (fork needed)
- Some interesting solutions:
    - Chromium Embedded Framework (CEF)
    - QtWebEngine
- Not particularly optimized for low-end hardware, Wayland support not ready in Linux yet, licensing issues,...

WebKit, then ?

- Also powerful and complete (Safari for OSX and iOS)
- Very flexible architecture (ports)
- Ports provide a stable API and can be part of upstream
- Available ports not ideal for our use case:
    - Upstream: iOS, OSX, GTK+
    - Downstream: EFL, Qt, Sony,...
- **Solution: creating a WebKit port for embedded devices**

igalia

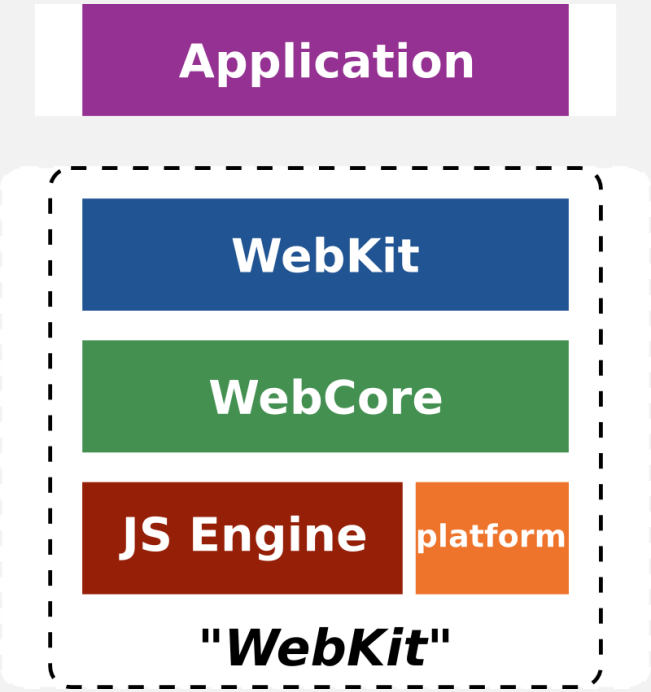
PART 2

What is WPE?

Architecture and features

WebKit

- Web rendering engine. The engine is the product
- Open source (and open development) since 2005
- Fexible architecture:



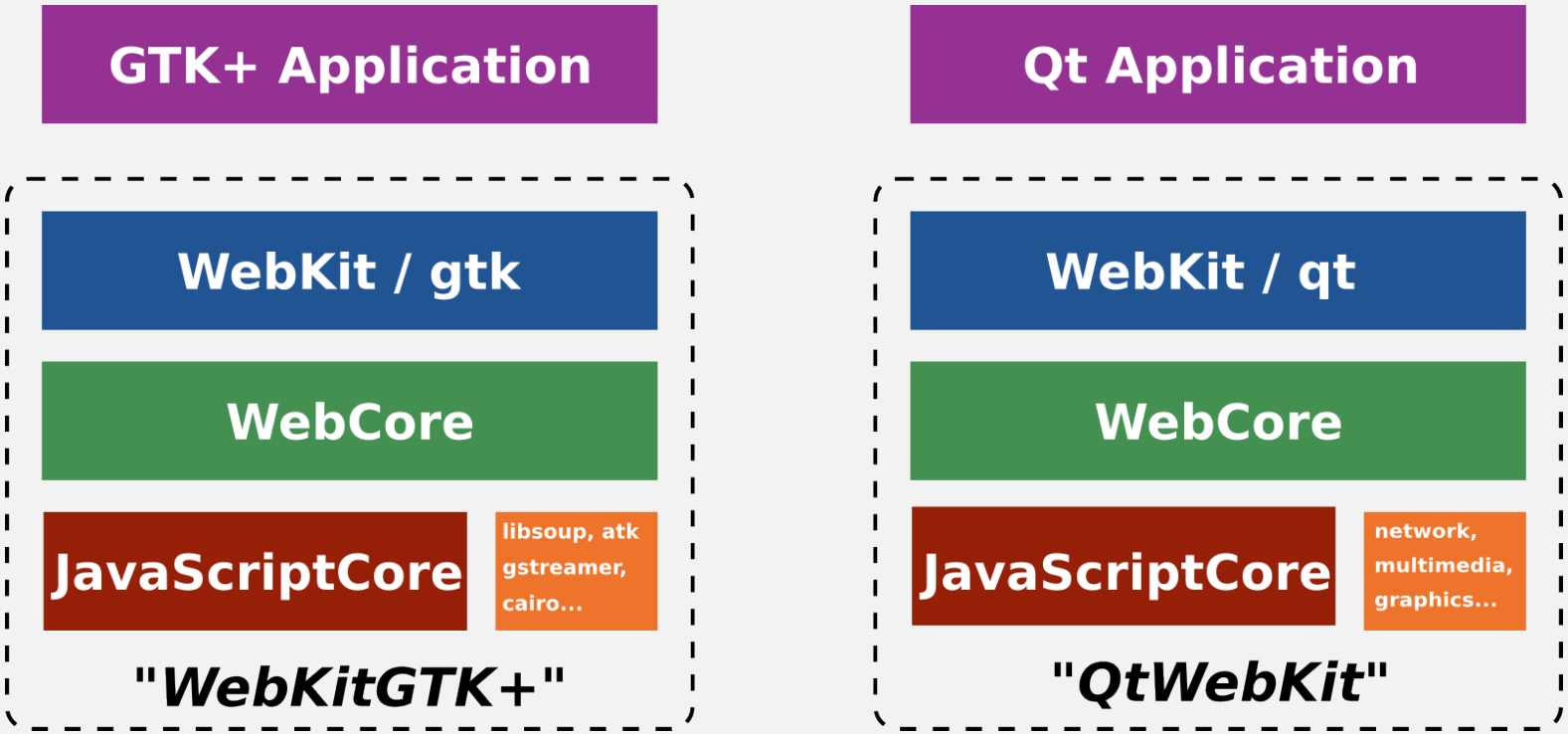
WebKit: thin layer to link against from the applications

WebCore: rendering, layout, network access, multimedia, accessibility support...

JS Engine: the JavaScript engine. JavaScriptCore by default.

Platform: platform-specific hooks to implement generic algorithms

WebKit ports: examples



A new WebKit port: WPE – Key requirements

- Main use case: full-screen content
- Fast and lightweight, minimal set of dependencies

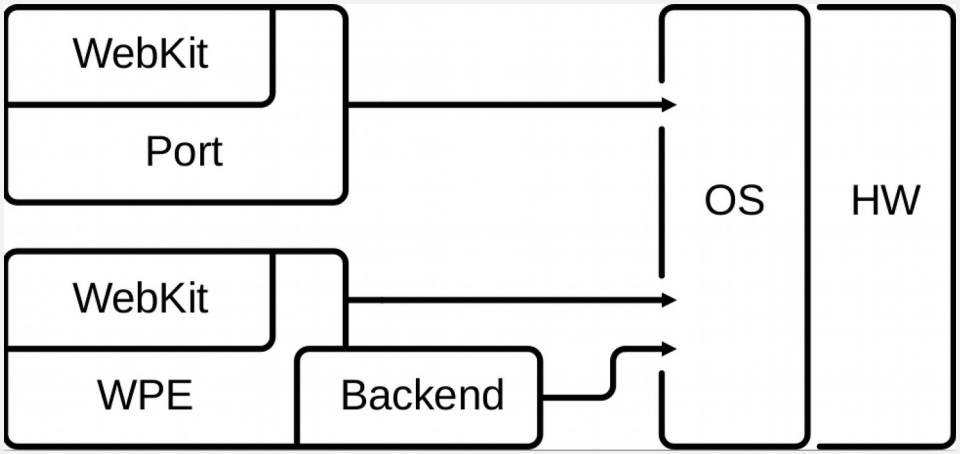
- Most HTML5 features need to be supported
- WebGL
- Accelerated canvas
- Hardware accelerated CSS 3D transformations
- Hardware accelerated video playback

A new WebKit port: WPE – Architecture

- Derives from WebKitGTK+. Part of the codebase shared
- Toolkit and platform agnostic
- GStreamer for media. JSC as JavaScript engine
- Reduces the dependencies to a few common libraries:
  - Glib, FreeType, HarfBuzz, GnuTLS, pixman, cairo, libsoup
- GLES 2.0 for hardware accelerated rendering
- Multiprocess: UI, Web, Network and Storage in different processes
- Intensive threading in composition, image decoding, media playback

A new WebKit port: WPE – Backends

- Main goal: efficient cross-process GPU buffer sharing
- Wayland, libgbm and other native implementations
- Necessary to glue the backend facilities with the provided EGL platform
- Renderer backend provides rendering target
- View backend provides a way to display the rendered buffer on screen
- Vulkan support down the line



A new WebKit port: WPE – Current backends

- Libgbm: Intel, AMD, open-source NVidia drivers for embedded devices (i.e. Jetson) -- specific to the Mesa library
- 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

A new WebKit port: WPE – Lightweight

- Reference hardware: Rpi 0-3 (desktop used for development too)
- A functional Raspberri Pi image can be about 40MB
- Low memory footprint: possible to define limits to consumption <100MB for a standard configuration
- Able to play YoutubeTV on a Rpi 0-1:
  - Using textured video
  - Raspberry Pi 0/1 is ~1000 DMIPS

WPE – strong focus on media (I)

- Hardware accelerated decoding where GStreamer plugins are available (Raspberry Pi and Broadcom Nexus)
- Hardware accelerated video rendering using GLES (allows CSS 3D transformations on the video)
- External rendering (hole punch) when required

WPE – strong focus on media (II)

- Media Source Extensions (**MSE**) support
  - MP4 done, WebM in progress (VP8 and VP9)
  - Youtube conformance 2016 passed, 2018 in progress
- Encrypted Media Extensions (**EME**) support
  - 0.1b (V1) done, with ClearKey and PlayReady
  - Proposed candidate (V3) under development:
    - Object oriented and promise based
    - Clearkey (W3C compliance and testing purposes)
    - PlayReady and Widevine (using OpenCDM)
- **WebRTC** support
  - Prototype done with OpenWebRTC (limitations)
  - Now adding libwebrtc in collaboration with Apple

igalia

PART 3

Where is WPE going?

Current status and future plans

WPE current status: upstream

- Port started in 2014 as an experiment
- Heavily developed during 2015-2017
- Integrated in WebKit upstream since May 2017
- Stable Igalia team working on it
- Community growing

- Functionally it is quite complete today!

WPE current status: adoption

- Media&entertainment industry:
  - Initially sponsored by Metrological
  - RDK consortium adopted the technology
  - Used by Comcast, Liberty Global and others
  - >10M set-top-boxes with WPE. Number growing
- Since 2017: several new use cases, various kinds of embedded devices adopting industrially WPE **==> *More hardware supported***

WPE: ongoing and future efforts (I)

- Stable release cycles:
  - Every 6 months (sync with WebKitGTK+)
  - They will be preview releases for now
- Improved QA infrastructure
  - More tests, more architectures, more target devices
- More documentation (project website)

WPE: ongoing and future efforts (II)

- New graphics architecture
- Further work on multimedia standards
- Networking & security
- Other Web Platform standards (WebDriver, WebGL2, WebVR,...)
- JSC improvements on 32bits (MIPS, ARMv6, ARMv7)
- Android prototype

WPE current status: repositories

- **Upstream WPE:**
  https://webkit.org/getting-the-code/
- **Downstream WPE:**
  https://github.com/WebPlatformForEmbedded
  (Includes some set-top-box related bits, and ad hoc solutions for specific target hardware in the context of RDK. Works as a playground for unstable or testing features which do not have a room in upstream yet)

Collaboration is welcome!

Thanks!

jjsanchez@igalia.com