Browsers for the automotive: an introduction to WebKit for Wayland

Silvia Cho

mscho@igalia.com

igalia

Igalia and WebKit/Chromium

- Open source consultancy founded in 2001
- Top contributor to upstream WebKit and Chromium
- Working with many industry actors: tablets, phones, IVI, smart TV, set-top boxes, and smart home

Outline

- Browser requirements for the automotive
- A bit of history
- Selecting the best alternatives
  Introduction to WebKit for Wayland
- Conclusions

Browser requirements for the automotive

Requirements

- Browser as an application and as a runtime:
  - User Experience: specific standards and UI modification
  - Portability: support of specific hardware boards (performance optimization)
  - OTA updates
- Browser as an application:
  - Functionalities
- Browser as a run time:
  - Application manager integration

Available alternatives

1) Licensing a proprietary solution

2) Deriving a new browser from the main open source browser technologies:

- **Chromium**
- **WebKit**

(Firefox: Mozilla removed support in their engine for third party browser developers)

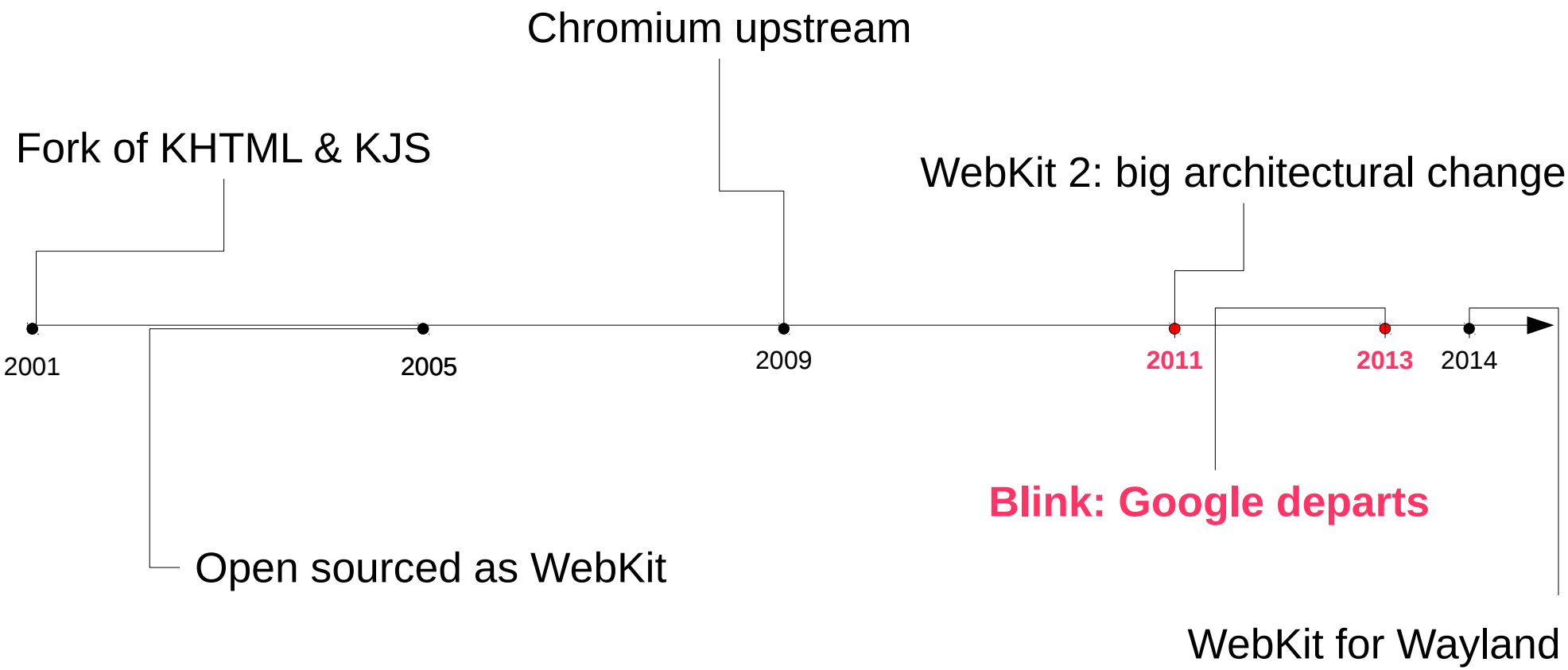
igalia

Understanding the main alternatives

- Decision between Chromium and WebKit
- Chromium and WebKit share a lot of history, design and code
- Learning the history of WebKit and Chromium improves the understanding of the pros and cons of each solution

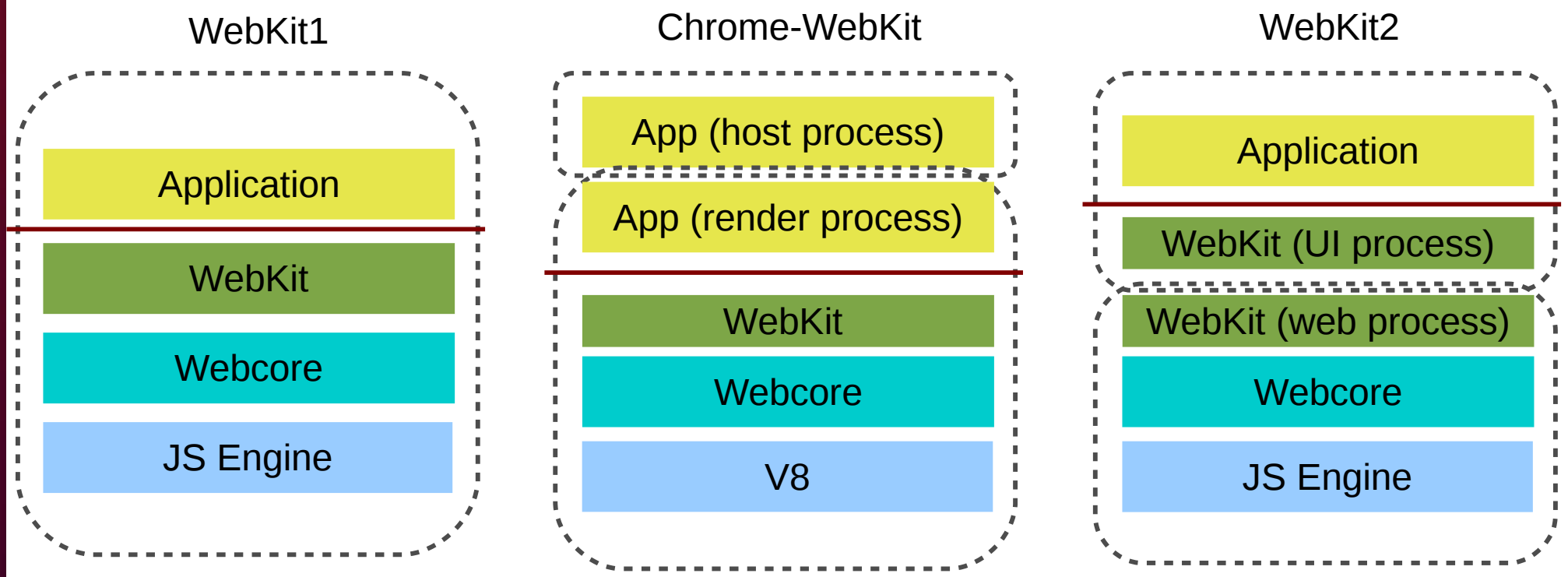
A bit of history

WebKit Project History



igalia

WebKit1 vs Chrome-WebKit vs WebKit2 (2012)



igalia

Google’s Departure and Blink

- Google announced on April 3rd, 2013 that they would fork WebKit and create Blink
- Motivations according to Google:
  - They were not using WebKit2 anyway
  - Easier to do ambitious architectural changes after the fork
  - Simplification of the codebase in Blink
- Tension between Apple and Google before the fork
  - Architectural decisions: Network Process
  - Code governance: Owners need to approve some core changes
- Big shock within the WebKit community

Current status in consumer industry

- Early consequences:
  - Many WebKit contributors chose to migrate their projects to Blink/Chromium and created Crosswalk, QtWebEngine, etc.
  - Some WebKit ports became deprecated
- Ports have been removed. Many libraries became default inside Chromium (SKIA, V8, FFMPEG, AURA, etc)
- Because of porting and maintenance challenges, the adoption of Chromium in consumer industry has been slower than expected.
- Recent trend towards more use of Chromium

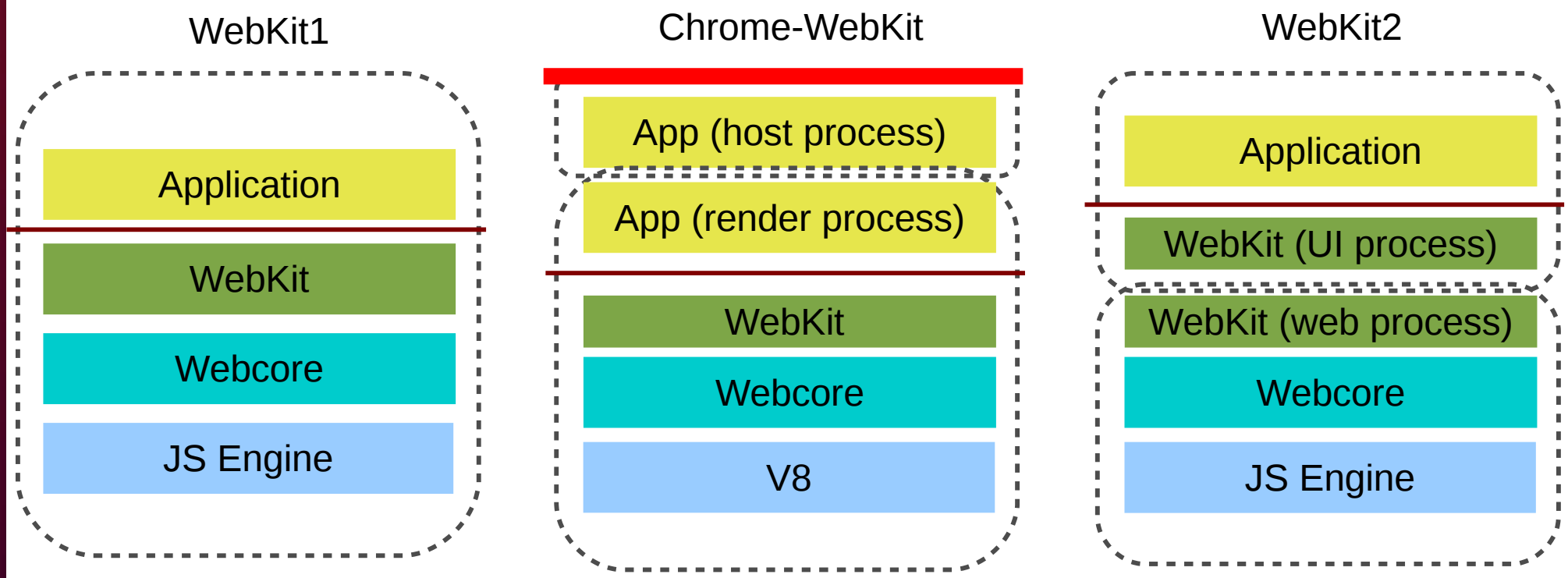
igalia

Selecting the best alternative: Introduction to WebKit for Wayland

Selecting the alternatives

- Chromium alternatives:
  - Chromium directly
  - Chromium Embedded Framework (CEF)
  - QtWebEngine
  - Crosswalk

WebKit1 vs Chrome-WebKit vs WebKit2 (2012)



igalia

Chromium

- Architecture designed primarily for browser application development

  - Latest HTML5 specs implementations
  - Content API changing constantly; embedding APIs is unstable
  - Wayland support is not finished
  - Ports have been removed and many libraries became default
  - FFMPEG is the default multimedia framework; AURA is the default graphics toolkit. If changes or replacements are needed, it will require big changes to the source base.

Selecting the alternatives

- Chromium alternatives:
  - Chromium directly
  - Chromium Embedded Framework (CEF)
  - QtWebEngine
  - Crosswalk

- WebKit alternatives:
  - QtWebKit (legacy, efforts to revive it)
  - WebKitEFL (maintenance mode)
  - WebKitGTK+ (active)
  - WebKit for Wayland (active)

WebKit for Wayland

- Designed for embedded systems
- It does not depend on a toolkit anymore; it doesn’t necessarily need Wayland
- Output can be embedded into a separate OpenGL scene or into a toolkit (e.g., Qt)
- Output can be displayed directly via the window or display manager
- Less memory footprint

WebKit for Wayland

- It is hardware-accelerated which relies on EGL and OpenGL ES
- It abstracts underlying rendering and presentation interfaces
- Flexible architecture
  - easy to plug in components (e.g. multimedia framework)

  - possible to modify the whole stack without diverging much from the latest upstream version

WebKit for Wayland

Things to do:

- Upstream remaining work (end of 2016)
- Optimize on specific hardware
- Implement custom UIs
- Implement latest HTML5 specs (if needed)

Conclusion:

- From technical POV in what regards to an embedded environment, WebKit for Wayland is a great choice

Chromium: things to consider

- Architecture designed primarily for browser application development
  - Latest HTML5 specs implementations
  - Content API changing constantly; embedding APIs is unstable
  - Wayland support is not finished
  - Ports have been removed and many libraries became default
  - FFMPEG is the default multimedia framework; AURA is the default graphics toolkit. If changes or replacements are needed, it will require big changes to the source base. Forking is one way to have control of the code for your needs.
  - Forks have a natural maintenance burden risk, given that upstream Chromium evolves very fast.

- **But it can still be used for the embedded environment if following issues are properly addressed:**

Chromium: things to do

- Create a stable API (or use CEF)
- Finish Ozone-Wayland support
- Integrate native multimedia framework
- Set up a proper branching strategy for a possible Chromium fork
- Optimize on specific hardware
- Implement custom UIs

Conclusions

Conclusions

- Both WebKit for Wayland and Chromium are active open source projects in terms of code and contributors
- Each solution has a different design purpose

  Apples vs. Oranges

  - Wrong question: which is better?
  - Correct question: what needs do I have?
- It is important to be aware of the implications of the pending issues and set up proper strategies to cope them

Thank you!

Silvia Cho (mscho@igalia.com)


igalia