WebKit and Blink: Bridging the Gap Between the Kernel and the HTML5 Revolution

Juan J. Sánchez

LinuxCon Japan 2014, Tokyo

igalia

Myself, Igalia and WebKit

- Co-founder, member of the WebKit/Blink/Browsers team
- Igalia is an open source consultancy founded in 2001
- Igalia is Top 5 contributor to upstream WebKit/Blink
- Working with many industry actors: tablets, phones, smart tv, set-top boxes, IVI and home automation.

Outline

1. Why this all matters
2. 2004-2013: WebKit, a historical perspective
   - 2.1. The technology: goals, features, architecture, ports, webkit2, code, licenses
   - 2.2. The community: kinds of contributors and contributions, tools, events
3. April 2013. The creation of Blink: history, motivations for the fork, differences and impact in the WebKit community
4. 2013-2014: Current status of both projects, future perspectives and conclusions

PART 1: Why this all matters

Why this all matters

- Long time trying to use Web technologies to replace native totally or partially
- Challenge enabled by new HTML5 features and improved performance
- Open Source is key for innovation in the field
- Mozilla focusing on the browser
- WebKit and now Blink are **key projects** for those building platforms and/or browsers

PART 2: 2004-2013
WebKit, a historical perspective

PART 2.1
WebKit: the technology

The WebKit project

- **Web rendering engine** (*HTML*, *JavaScript*, *CSS*...)
- The engine is the product
- Started as a fork of KHTML and KJS in 2001
- *Open Source* since 2005
- Among other things, it's **useful for**:
  - Web browsers
  - Using web technologies for UI development

Goals of the project

- **Web Content Engine**: HTML, CSS, JavaScript, DOM
- **Open Source**: BSD-style and LGPL licenses
- **Compatibility**: regression testing
- **Standards Compliance**
- **Stability**
- **Performance**
- **Security**
- **Portability**: desktop, mobile, embedded...
- **Usability**
- **Hackability**

Goals of the project

NON-goals:

- "It's an engine, not a browser"
- "It's an engineering project not a science project"
- "It's not a bundle of maximally general and reusable code"
- "It's not the solution to every problem"

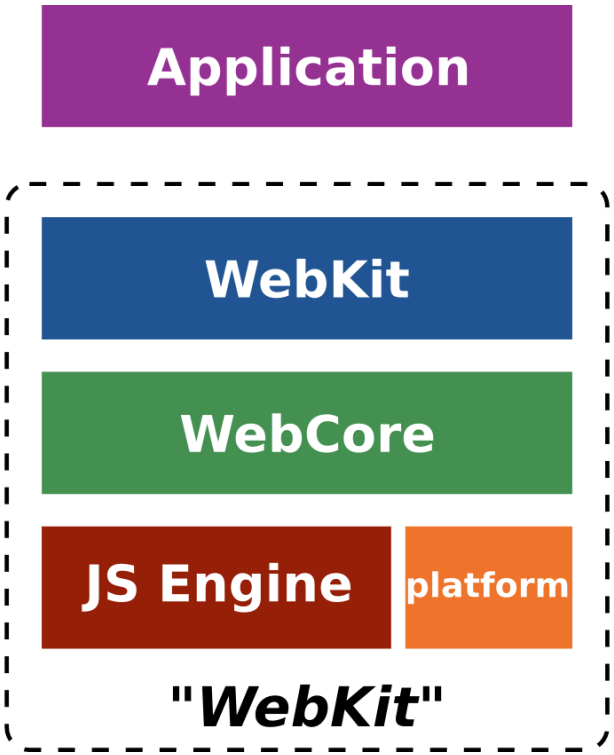
`http://www.webkit.org/projects/goals.html`

WebKit features

- HTML and XML support
- JavaScript support (ECMAScript 5.1)
- CSS 2.1, CSS 3 support
- SVG support
- Support for Plugins (NPAPI, WebKit Plugins)
- HTML5 support: multimedia, 3D graphics, advanced CSS animations and transformations, drag'n'drop, offline & local storage, connectivity...
- Accessibility support
- Q&A infrastructure: review process, continuous integration, 30.000 regression tests, API tests...
- Passing ACID3 with 100/100 tests since March 2008

WebKit Architecture

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

igalia

What is a WebKit port?

iOS

igalia

How many WebKit ports are there?

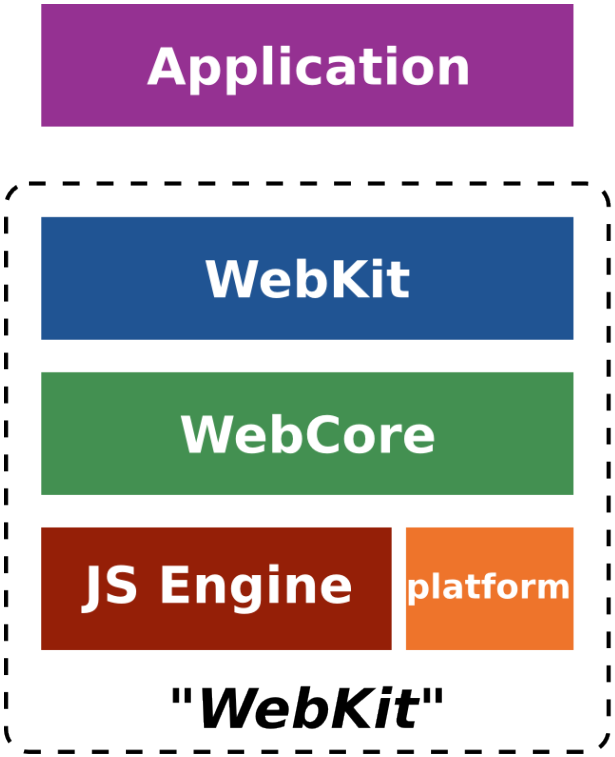
WebKit is available for different platforms:

- Main upstream ports in 2012/2013:
  - Mac OS X, iOS
  - GTK+ based platforms (GNOME)
  - Qt based platforms (KDE)
  - Enlightenment Foundation Libraries (EFL, Tizen)
  - Google Chromium / Chrome
  - WebKitNIX
- Other ports: wxWidgets, Brew MP, Symbian devices (S60), Win32, BlackBerry, Adobe Integrated Runtime (Adobe AIR)

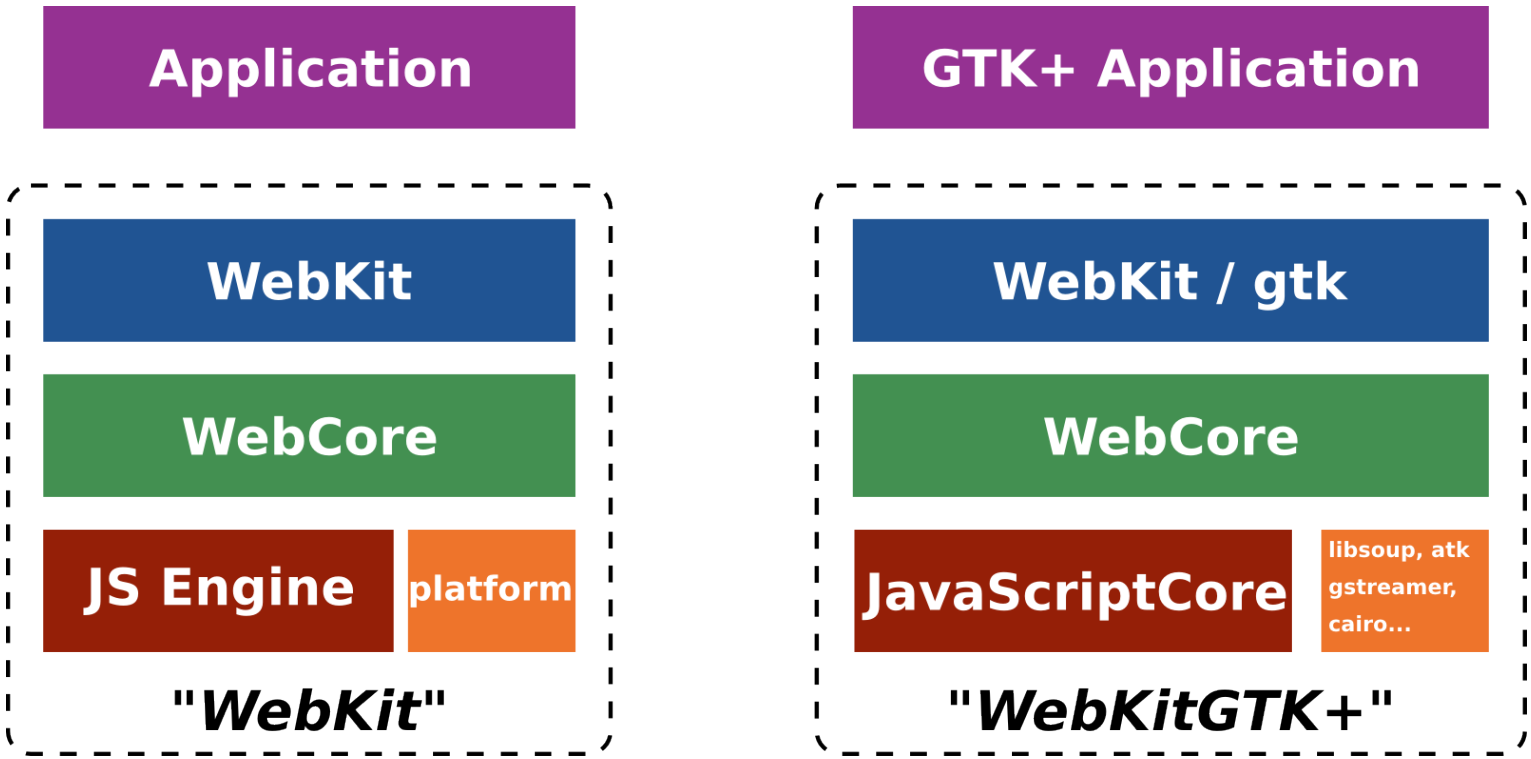
Some WebKit based browsers in 2013

- Amazon Kindle
- Arora
- BOLT browser
- Epiphany browser
- Google Chrome
- iCab (version >= 4)
- Iris Browser
- Konqueror
- Midori
- Nintendo 3DS
- OWB
- OmniWeb
- PS3 web browser
- RockMelt
- Safari
- SRWare Iron
- Shiira
- Sputnik for MorphOS
- Stainless
- Steel for Android
- TeaShark
- Uzbl
- Web Browser for S60 (Nokia)
- WebOS Browser

Architecture of a WebKit port



Architecture of a WebKit port



igalia

How do we use a WebKit port?

- **The WebView *widget***:
  A platform-specific widget that renders web content.

  It's the **main component** and it's useful for:
  - Loading URIs or data buffers pointing to HTML content
  - Go fullscreen, text/text+image zooming...
  - Navigate back and forward through history...

- **Events handling**:
  Allows embedders to get notified when something important happens or when some input is needed.

  Some examples of these events:
  - Getting notified when a load finished or failed
  - Asking permission for navigating to an URI
  - Requesting authorization for something..

A minibrowser written in Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import gtk
import webkit

def entry_activated_cb(entry, embed):
    embed.load_uri(entry.get_text())

# Widgets and signals
window = gtk.Window()
window.set_default_size(800, 600)
window.set_title("Mini browser written in Python")
embed = webkit.WebView(); # WebKit embed
entry = gtk.Entry()
entry.connect('activate', entry_activated_cb, embed)
scroller = gtk.ScrolledWindow()
scroller.add(embed)

# Pack everything up and show
vbox = gtk.VBox(False, 5)
vbox.pack_start(entry, False, False)
vbox.pack_start(scroller)
window.add(vbox)
window.show_all()

# Load a default URI and run
embed.load_uri("http://www.webkit.org")
gtk.main()
```

A minibrowser written in Python

igalia

A minibrowser written in C

```
#include <webkit/webkit.h>
static void entry_activated (GtkEntry *entry, WebKitWebView *embed)
{
  webkit_web_view_load_uri (embed, gtk_entry_get_text (entry));
}
int main (int argc, char** argv)
{
  gtk_init (&argc, &argv);

  /* Widgets and signals */
  GtkWidget *window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size (GTK_WINDOW (window), 800, 600);
  gtk_window_set_title (GTK_WINDOW (window), "Mini browser written in C");
  GtkWidget *embed = webkit_web_view_new();
  GtkWidget *entry = gtk_entry_new();
  g_signal_connect (entry, "activate", G_CALLBACK (entry_activated), embed);
  GtkWidget *scroller = gtk_scrolled_window_new(NULL, NULL);
  gtk_container_add (GTK_CONTAINER(scroller), embed);

  /* Pack everything and show */
  GtkWidget *vbox = gtk_box_new (GTK_ORIENTATION_VERTICAL, 0);
  gtk_box_pack_start (GTK_BOX(vbox), entry, FALSE, FALSE, 0);
  gtk_box_pack_start (GTK_BOX(vbox), scroller, TRUE, TRUE, 0);
  gtk_container_add (GTK_CONTAINER(window), vbox);
  gtk_widget_show_all (window);

  /* Load a default URI and run */
  webkit_web_view_load_uri (WEBKIT_WEB_VIEW (embed), "http://www.webkit.org");
  gtk_main();
  return 0;
}
```

A minibrowser written in C

igalia

What is WebKit2?

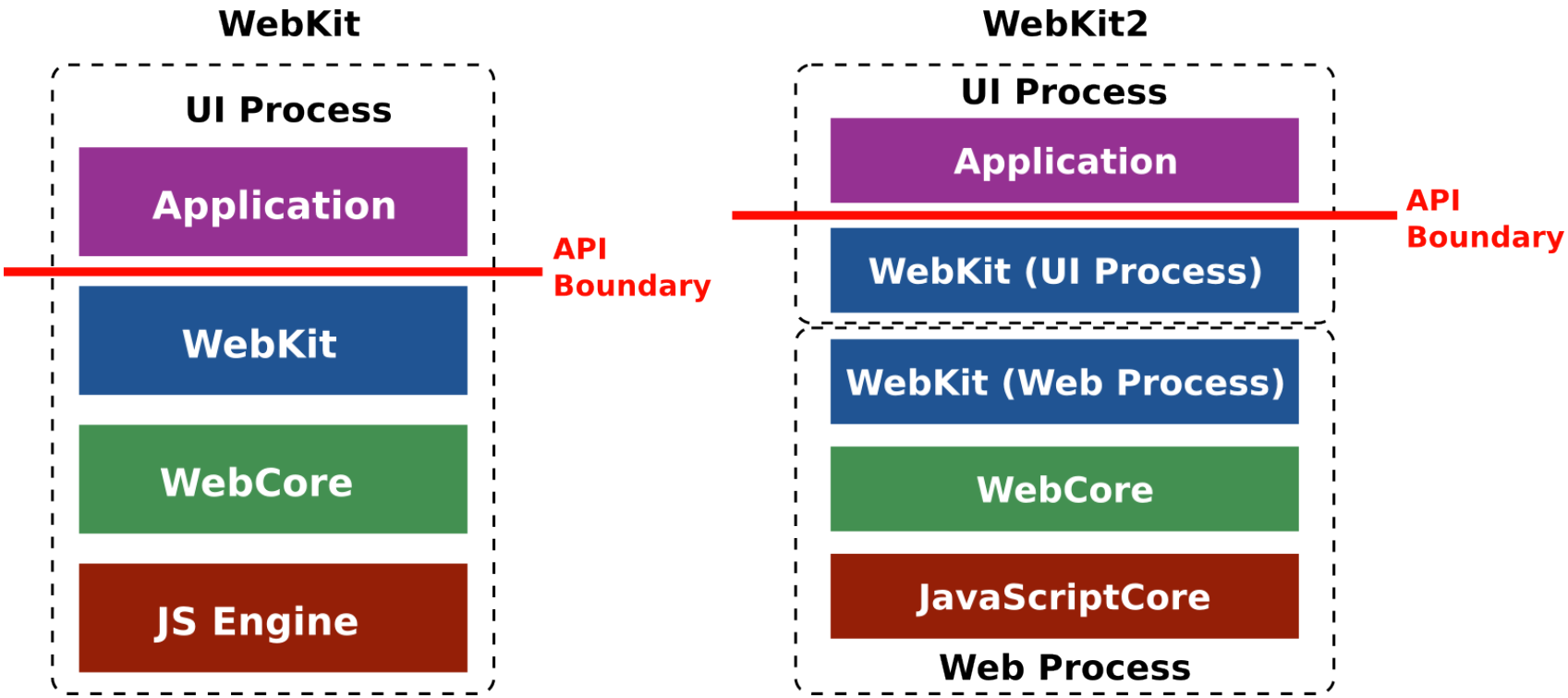
- New API layer designed to support a split process model (First release by Apple on April 8th, 2010[1]).
- Different to Chromium's multi-process implementation

  It's bundled in the framework (reusable)

- Different processes take care of different tasks:
  - **UI process**: the WebView *widget*, application UI
  - **Web process**: loading, parsing, rendering, layout...
  - **Plugin process**: each plugin type in a process
- It comes with Inter-Process Communication (IPC) mechanisms to communicate those processes bundled-in

`http://trac.webkit.org/wiki/WebKit2`

WebKit VS WebKit2



WebKit2 vs WebKit1

Pros:

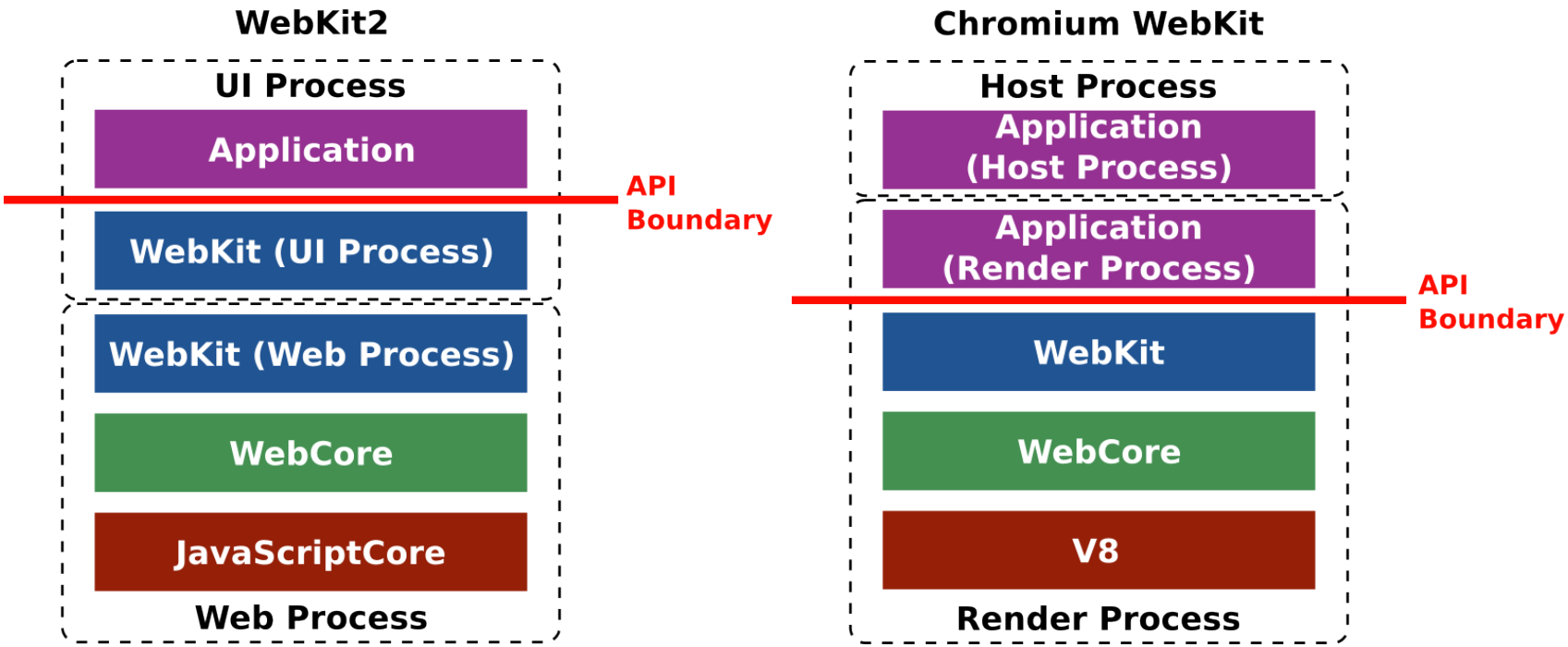
- Isolation
- Security (sandboxing, per-process privileges)
- Performance if processes are run in parallel
- Stability (a crash in the WebProcess does no kill the application)

WebKit2 vs WebKit1

Cons:

- Higher resource requirements
  - Multiple processes instead of just one
  - At the same time easier to release resources
  - Higher latency interaction with page content due to IPC
- Complexity
  - Coding (more complications interacting directly with page content)
  - Debugging
- WebKit1 and WebKit2 maintenance

WebKit2 VS Chromium



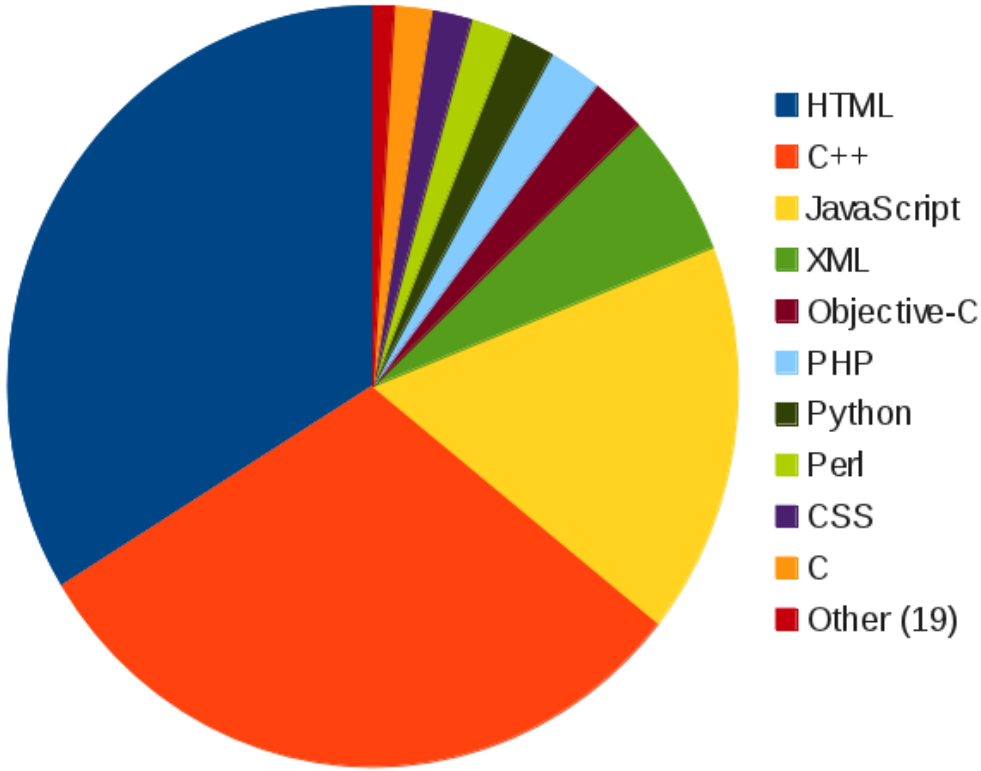
WebKit2: current status

- The main ports released WebKit2 browsers by 2013
- WebKit1 moving to maintenance mode or removed
- Cross-platform and non-blocking C API available
- Most challenges of the split process model solved
- Lots of new architectural changes about to come (e.g. Network Process)

The Source Code in numbers

According to Ohloh on May 17th, **lines of code per language**, without considering blank lines nor comments:

Language	LoC	%
HTML	1,955,561	32.4 %
C++	1,308,667	27.5 %
JavaScript	962,086	20.8 %
Objective-C	175,669	3.4 %
XML	158,555	2.5 %
C	121,951	3.0 %
PHP	100,345	2.3 %
CSS	93,247	1.6 %
Python	78,348	1.9 %
Perl	76,491	1.7 %
OpenGL Shad	52,234	1.8 %
Other (16)	50,000	1.1 %
Total	4,132,955	



`https://www.ohloh.net/p/WebKit/analyses/latest/language_summary`

Just considering C++, Objective-C and C >1.6M LoC!

Licenses: BSD 3-clause and LGPL

igalia

PART 2.2
WebKit: The community

A bit of history

Source: `http://ariya.ofilabs.com/2011/11/one-hundred-thousand-and-counting.html`

The WebKit Project in numbers

Commits per month till 2013:

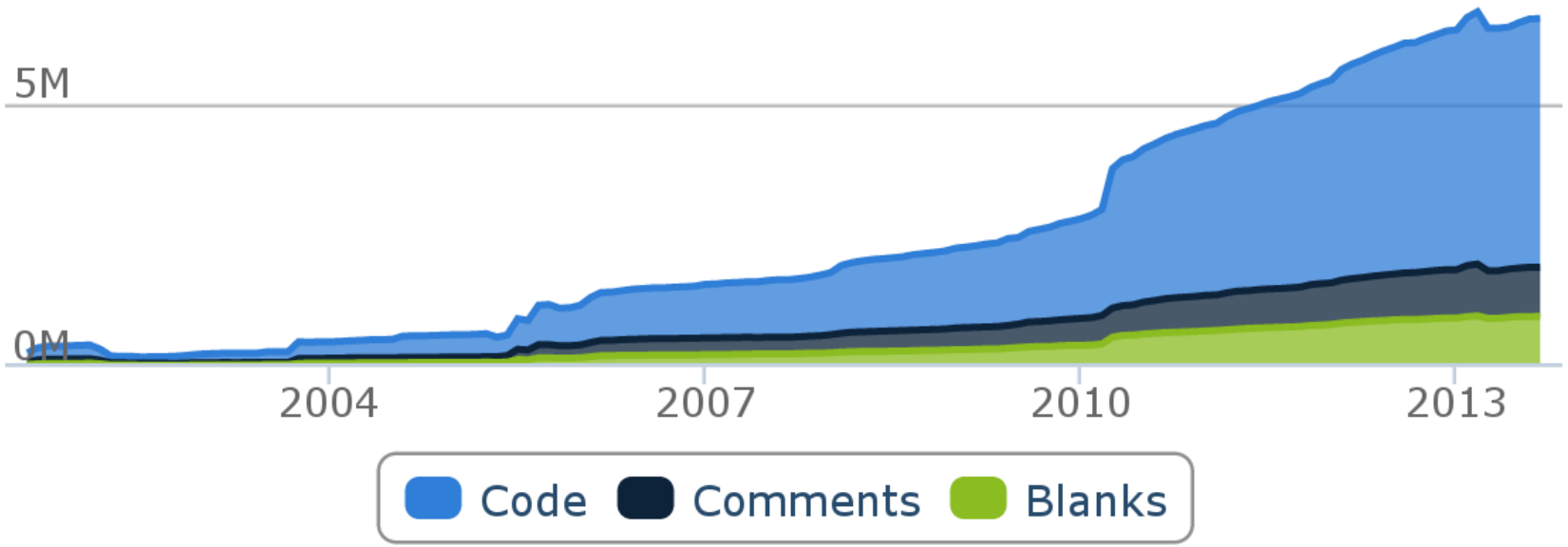
igalia

The WebKit Project in numbers

Contributors per month::

The WebKit Project in numbers

Evolution in the number of lines of code



Tracking ongoing work in WebKit

- Webkit is a big beast and a lot of organizations with different agendas are working in different parts:
  - Implementing new standards (like the CSS shaders from Adobe, or CSS3 GCPM from Apple)
  - Improvements in architecture, performance and internal code (WebKit2)
- On top of this there is the maintenance work (testing, continuous integration, bugfixing)
- No single centralized place to follow all the information: blogs, mailing lists, IRC, etc.

igalia

Activity of Companies by 2013

- Based on Bitergia's report[2]
- Based on reviewed commits
- "Gardening" commits filtered out
- From the beginning of the project till beginning of 2013

[2] http://blog.bitergia.com/2013/02/06/report-on-the-activity-of-companies-in-the-webkit-project/

Activity of Companies by 2013

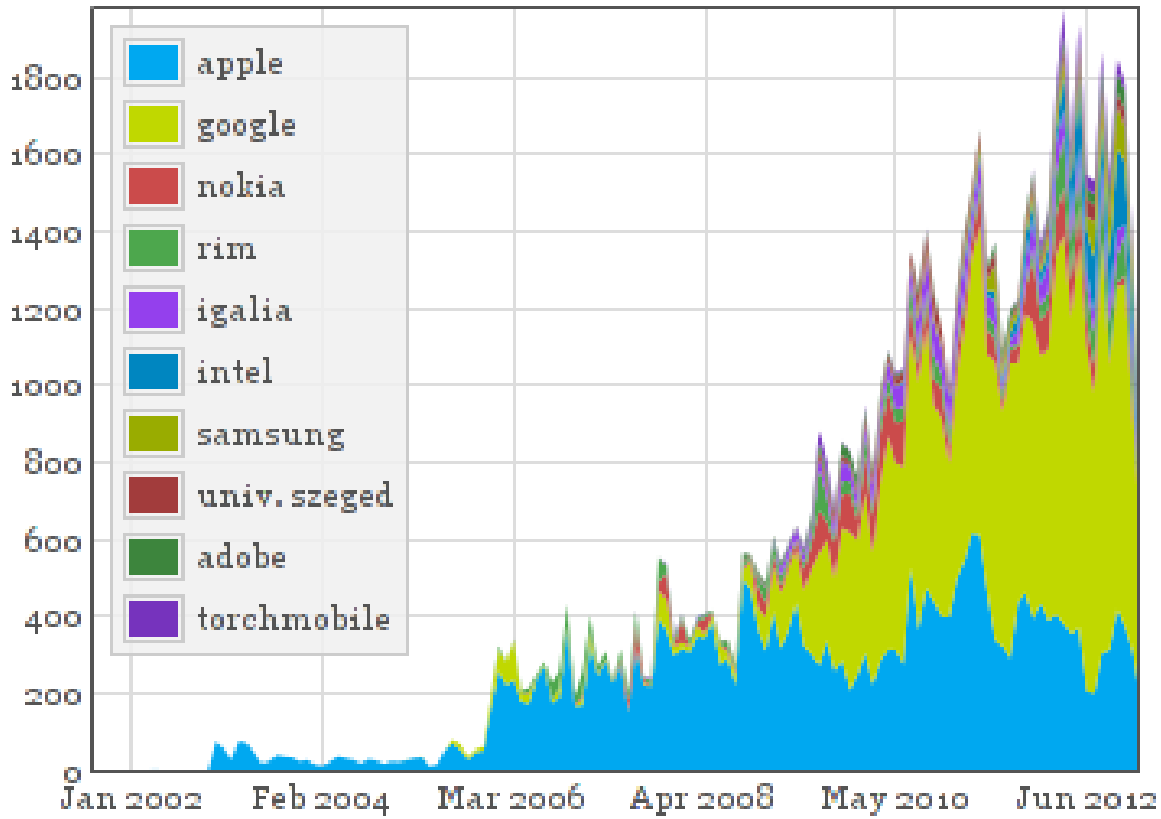


Figura : Commits per company (monthly)

Activity of Companies by 2013

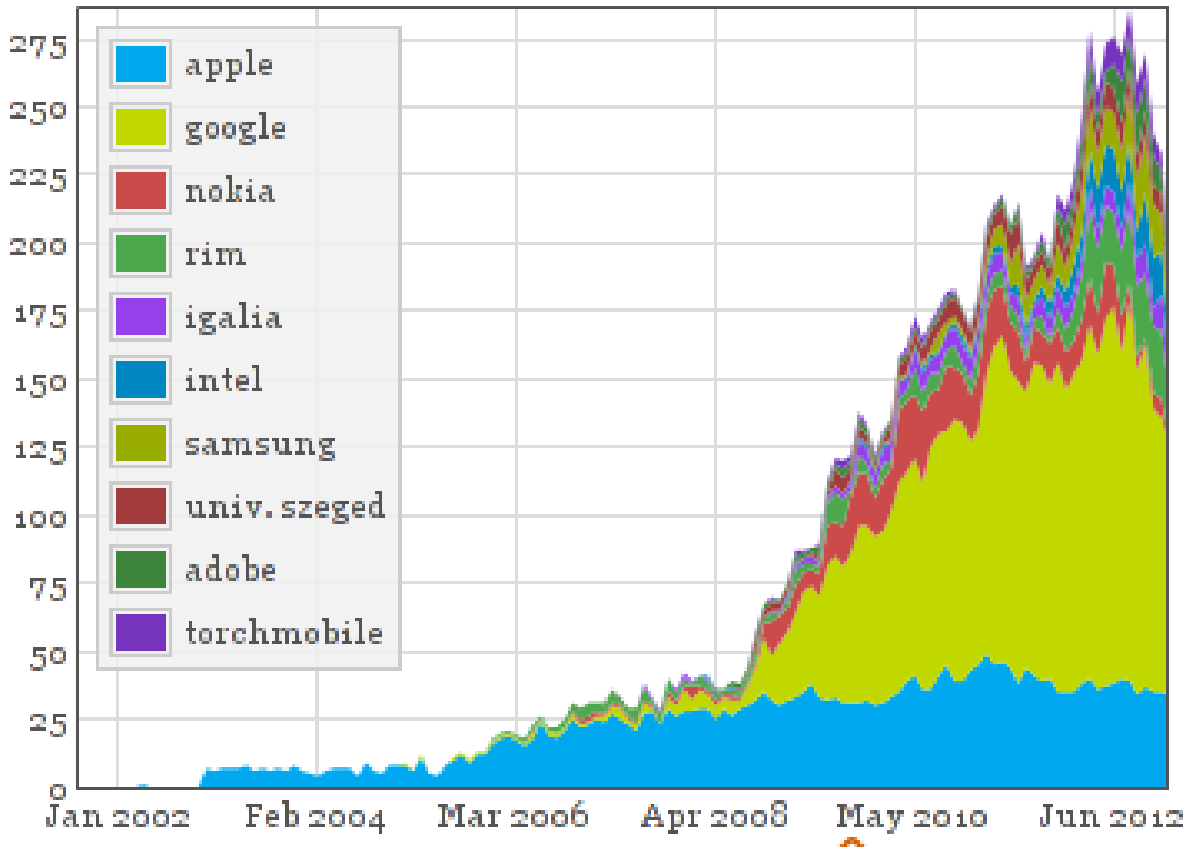


Figura : Active authors per company (monthly)

Activity of Companies by 2013

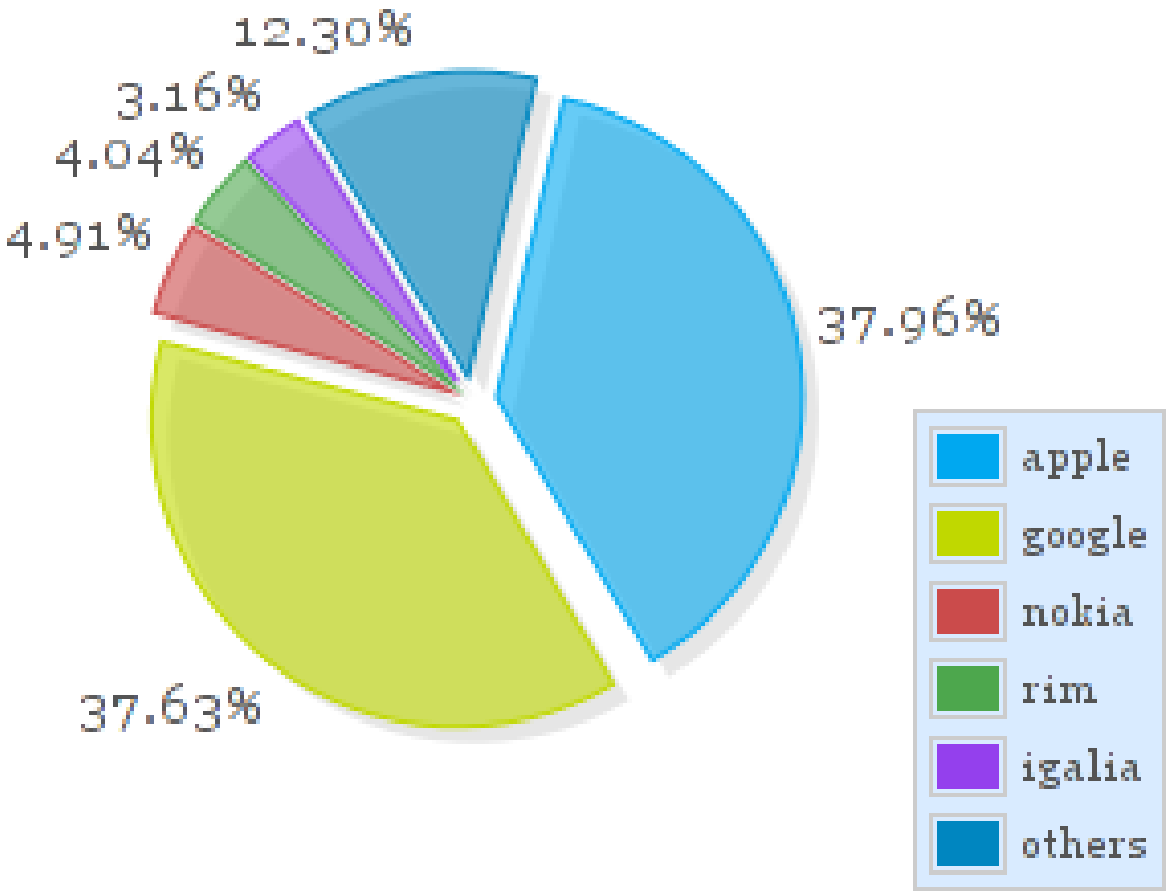


Figura : Commits per company

Activity of Companies by 2013

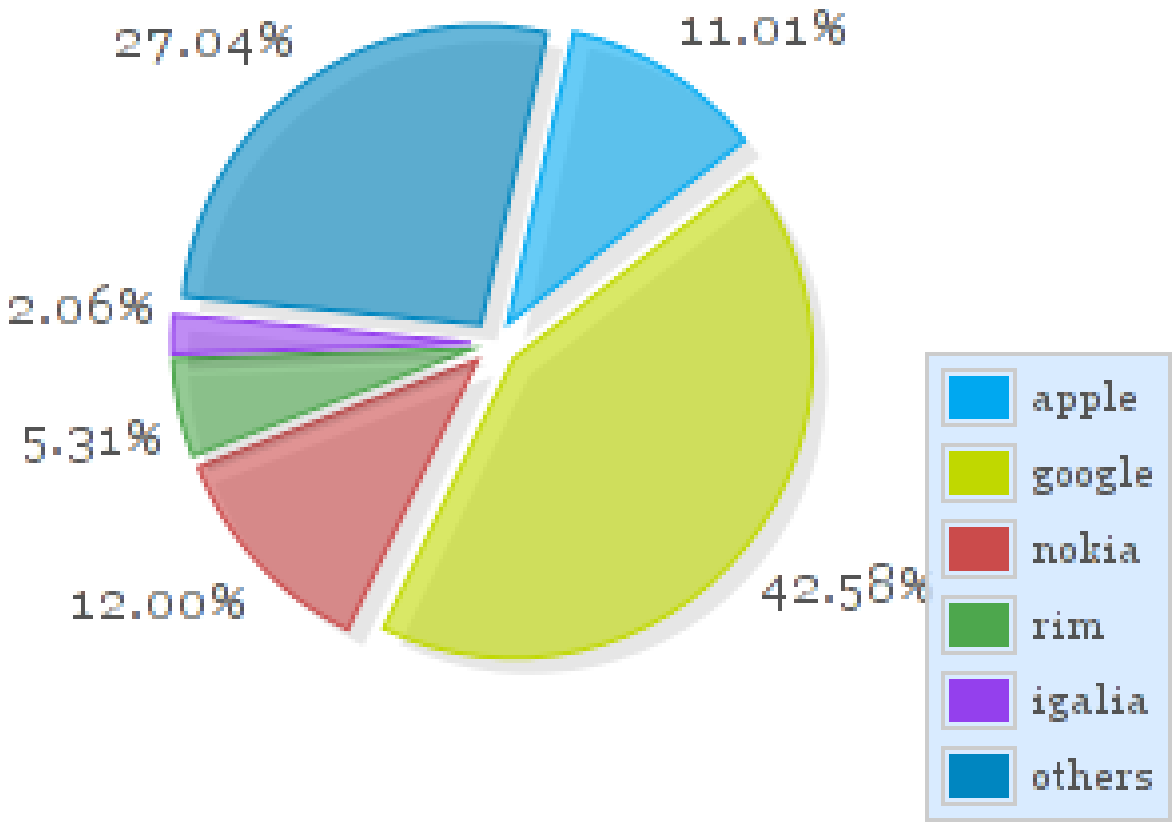


Figura : Active authors per company

Activity of Companies by 2013

Some conclusions from the authors:

- Google and Apple leading the project
- The diversity of the project has been increasing
- Contributions from other parties >25 % and growing
- > 20 companies actively contributing to WebKit
- 1500-2000 commits per month
- 387 committers from 29 different institutions

Committers and Reviewers

WebKit Committer

A WebKit Committer should be a person we can trust to follow and understand the project policies about checkins and other matters.

Has commit rights to the public SVN repository.

WebKit Reviewer

A WebKit Reviewer should be a person who has shown particularly good judgment, understanding of project policies, collaboration skills, and understanding of the code.

A WebKit Committer who can review other's patches.

Copyright for contributions

- There is no copyright transfer for the contributions
- Committers sign some papers where they commit to good behaviour

Releases

- There are no releases of WebKit itself
- Each port manages the release cycle, typically aligned with the target platform schedule

Types of contributions

- Bugfixing and new features in:
  - An existent port
  - The core components: webcore and JSC/V8
- Creation and maintenance of a new port

Guidelines for contributing patches to WebKit

1. Get and build the code from the SVN repository
2. Choose or create a bug report to work on
3. Code your changes and make sure you include new regression or unit tests if needed
4. Create a patch for your changes and submit it asking for review over it to appropriate reviewers
5. Update and change your patch as many times as needed
6. Once approved, land your patch or ask a committer/reviewer to do it
7. Watch for any regressions it might have caused

igalia

Creating a port: what needs to be done

- High level API (WebKit1 and WebKit2)
- Low level backend specific implementation
  - Web Template Framework (WTF): memory management, threading, data structures (vectors, hash tables, bloom filters, ...) numerical support, etc.
  - JSC vs V8
  - Networking: HTTP, DNS, cookies, etc.
  - Graphics: 2D/3D rendering, compositing, theming, fonts
  - Multimedia: media player for audio and video tags
  - DOM bindings
  - Accessibility
  - Smaller tasks: clipboard, popup and context menus, cursors, etc.
- Other things: favicons, plugins, downloads, geolocation, settings, navigation policies, etc.

igalia

Creating a port: the social side

- Difficult without full-time reviewers
- Reuse from other ports as much as possible
- Try to work upstream from the very beginning
- The risk of forking is big, the project moves fast
- Focus on testing and continuous integration
- Port-specific events and communication tools

igalia

Coordination and communication tools

- Website: `http://www.webkit.org/`

  Port specific Websites (e.g. `http://webkitgtk.org/`)

- Wiki: `http://trac.webkit.org/wiki`
- Blogs: `http://planet.webkit.org/`
- Source Code:
  - SVN: `http://svn.webkit.org/repository/webkit`
  - Git mirror: `git://git.webkit.org/WebKit.git`
- Bugzilla: `https://bugs.webkit.org/`
- Buildbots: `http://build.webkit.org/`
- Mailing lists: `http://lists.webkit.org/mailman/listinfo.cgi`
- IRC (*irc.freenode.net*): *#webkit* and *#webkitgtk+*

igalia

The WebKit Contributors Meeting

- Meeting for contributors to the WebKit project
- Organized in an “unconference”-like format
- Extremely useful to advance on some topics: Implementation of new APIs, WebKit2, accelerated compositing, helper tools, QA infrastructure...
- Yearly held in Cupertino, California. Hosted by Apple

Part 3

The creation of Blink (April 2013)

Google's Departure. Blink

- Google announced on April 3rd that they would be forking WebKit and creating 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: NetworkProcess
  - Code governance: Owners need to approve some core changes
- Big shock within the WebKit community

Differences between WebKit and Blink

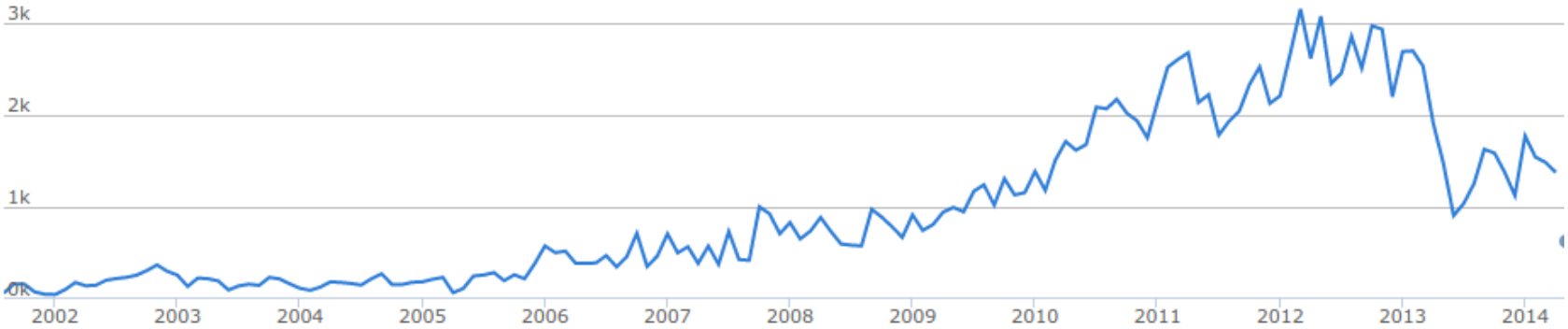
- Removes the concept of 'port' as it was defined in WebKit (deep platform integration): Skia, V8 and other libraries cannot be replaced
- Still possible to use Blink in other platforms, but now integration happens at Content level
- Only the rendering engine. Multi-process architecture is still in Chromium
- WebKit has committers, reviewers and owners (control some core areas). Blink only committers and owners (similar to WebKit reviewers)
- Peer review process a bit more relaxed in Blink
- Many architecture changes are started to be implemented

Early consequences of Blink for WebKit

- Google was the main contributor by # of commits
- Opera joined WebKit then moved to Blink. Other companies and communities started thinking what to do.
- Apple's position more dominant. Relaxed a bit the Owners policy
- Several WebCore modules left orphan. Other hackers assuming WebCore modules maintainership
- WebKit developers porting patches from/to Blink
- Many hacks to accomodate Chromium removed. Engines quickly starting to diverge at faster pace

Impact of Blink in numbers

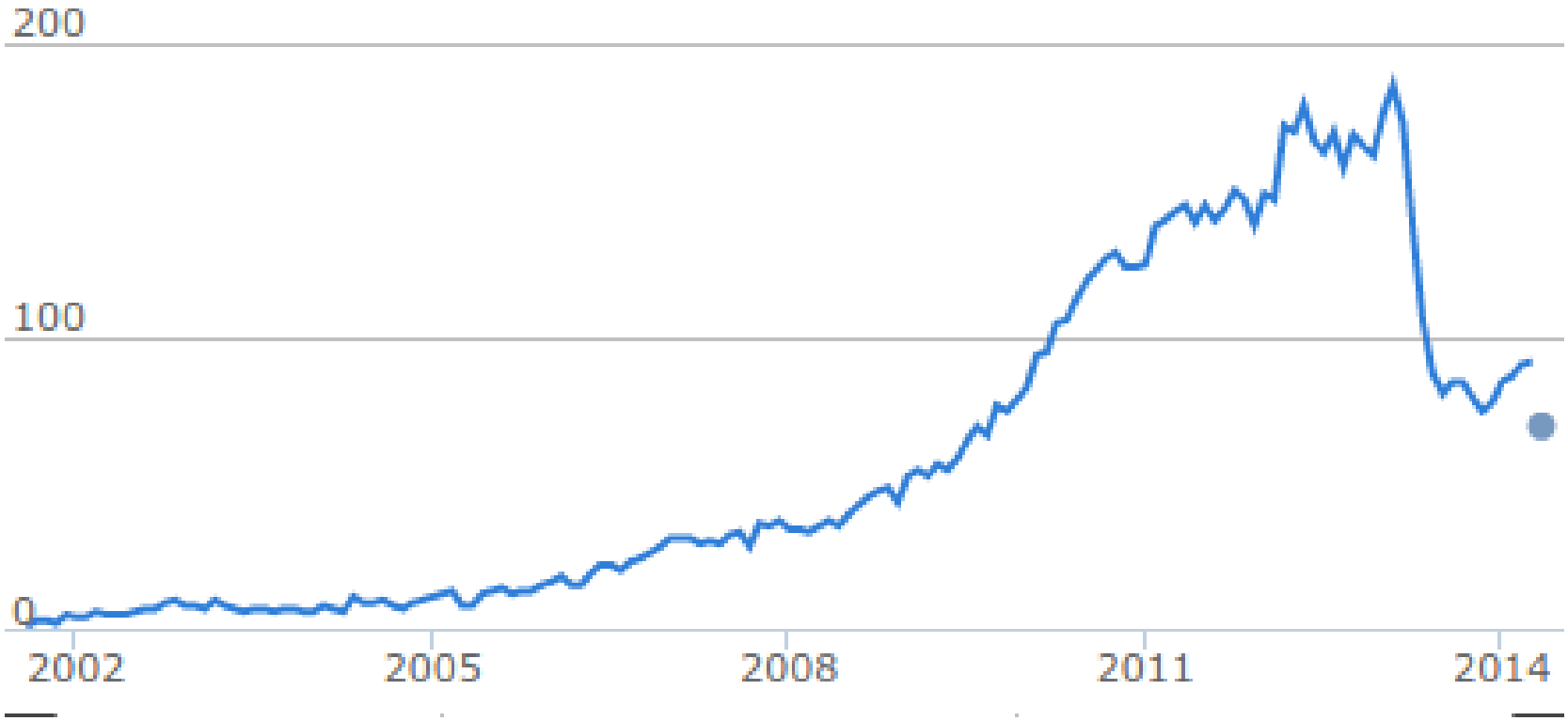
Commits per month in WebKit, including last months:



igalia

Impact of Blink in numbers

Contributors per month in WebKit, including last months:

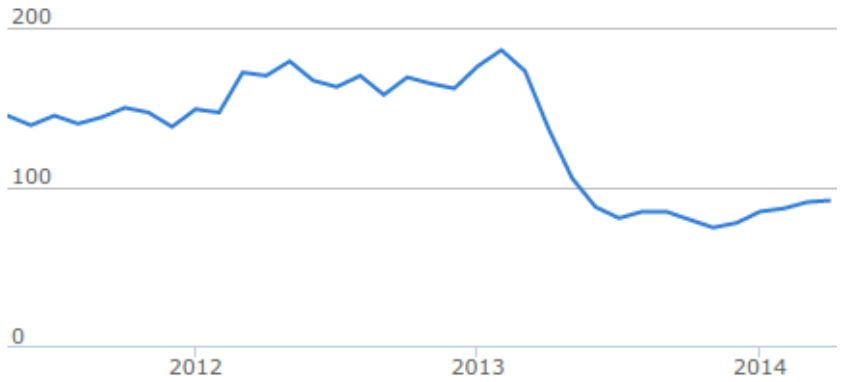


Impact of Blink in numbers

Contributors per month in 2013-2014:

Blink:

WebKit:



Impact of Blink in numbers

Commits per month in 2013-2014, Blink::

Commits per month in 2013-2014, WebKit::

igalia

The Blink community is born

- Dynamics of the community still being defined
- Opera early contributor and user of Blink
- Qt recently announces that they are moving to Blink (Sep 2013)
- Tizen to use Crosswalk (Blink based)
- Still unclear what all the downstreams are doing, but there is a tendency to at least try Blink/Chromium
- BlinkOn event twice per year, Fall in California, Spring in Europe/Asia/Australia

Main WebKit ongoing efforts and plans (I)

- The community is a bit smaller and stable after the fork.
- Some companies gaining visibility due to this fact: Adobe
- Little public discussion about core features and roadmap
- Most active port beside the ones Apple maintains is GTK+

Main WebKit ongoing efforts and plans (II)

- Some recent developments:
  - CSS Shapes
  - CSS Regions + new multicolumn implementation
  - CSS JIT Compiler (performance improvement)
  - Porting the code to C++11
  - JavaScriptCore's FTL JIT (performance improvement)

igalia

Main Blink ongoing efforts and plans (I)

- The community is growing progressively after the fork. Opera getting visibility
- Some external actors just consuming Chromium/Blink, not contributing or integrating upstream
- Open discussion in blink-dev with the 'intents' (to ship, to develop, to deprecate, to remove). 3 Owners need to support them
- Public roadmap with main focus for 2014 in mobile performance
- Google still keeping quite a lot of control, but already 9 owners outside Google

Main Blink ongoing efforts and plans (II)

- Ongoing work and roadmap:
  - CSS Shapes
  - CSS Regions not implemented, but support for multicolumn
  - Oilpan (tries to avoid memory leaks using a garbage collector instead of smart pointers)
  - Repaint after layout (performance optimization for rendering)
  - Web animations
  - Blink repository will be integrated into Chromium's

Open questions about Blink vs WebKit

- How open the dynamics of each community will be?
- Which project will innovate faster and keep higher quality and performance standards?
- For how long will it be possible to port patches from one project to the other?
- Will there be specialized areas where one of the projects will perform better? (e.g. WebKit and the memory consumption)
- Will those adopting Chromium/Blink be able to integrate it even when Google is not making things easy? Will any of them move back to WebKit?

Conclusions

- WebKit and Blink are key open source projects. They are at the very heart of the HTML5 revolution
- By 2013: it was the only true Open Source alternative for embedders for many years. Good community and architecture. Complex project
- Blink splitted community and efforts. Several WebKit ports and some downstreams decided to migrate. There are technical and community chanllenges in this migration that are still to be understood
- Both projects remain healthy for now. Still many open questions about both projects

igalia

Thank you!

Juan J. Sánchez
jjsanchez@igalia.com