Who I am

Leading a WebKit team at Samsung

WebKit reviewer since 2009

laszlo.gombos@gmail.com

@laszlogombos
Agenda

• The History of WebKit
• How to get involved with WebKit
• Architecture of WebKit
• Future challenges
Tizen & WebKit

- **Tizen** is an open source operating system designed to run applications from the **web** ecosystem.

- The **Web engine** responsible for executing web application in Tizen 2.1 is based on WebKit (browser + web runtime)

- **WebKit** is an open source project. It is a layout engine designed to allow web browsers to render web pages.
The history of WebKit
Speed of development

• Lifetime of the project (12 years)
  • ~150,000 commits
  • ~120,000 bugs

• Last year
  • ~35,000 commits, ~100 commits a day, 1 commit in every ~15 minutes
  • ~30,000 bugs

• 4 GB size of the repository
  • 3.2 GB (80%) test and test expectations – test driven development
  • ~35,000 tests, 1.7M lines of code

• No official releases of WebKit, ports have releases
The history of WebKit (1/2)

- 1998 – KHTML as part of KDE project on Linux (Qt)
- 2003 – Apple Safari based on KHTML on Mac (WebCore, JSC)
- 2005 – WebKit.org
- 2006 – Nokia S60 mobile browser on Symbian
- 2007 – Apple iPhone on iOS
- 2007 – Android browser
- 2007 – QtWebKit

(http://www.youtube.com/watch?v=Tldf1rT0Rn0)
The history of WebKit (2/2)

- 2008 – Google Chrome (Windows)
- 2010 – Samsung Dolfin browser
- 2010 – Blackberry 6
- 2010 – Apple announces WebKit2
- 2011 – Nokia N9 (based on WebKit2)
- 2012 – Google upstream android support
- 2013 – Opera to adopt Chrome port of WebKit
- 2013 – Apple started to upstream iOS port
- 2013 – Google split (Blink)
WebKit ports

- Apple Safari – MacOS (iOS), Windows (Apple)
- EFL – Linux/Tizen (Intel, Samsung)
- BlackBerry – QNX (BlackBerry)
- Qt - Linux, Windows, MacOS (Digia)
- Gtk – Linux, Windows, Mac (Igalia)
- WinCE - WinCE,
- WinCairo - Win
- (Nix) – Linux
- Chromium

http://paulirish.com/2013/webkit-for-developers/
Blink impact on WebKit

- **WebKit housekeeping**
  - Removed android, skia, V8 support that were only used by the chromium port
  - Test expectations for chromium
  - About 2GB data has been removed, mostly test expectations
  - ~170k lines of code removed (10%)

- **Key patches are cross-posted/merged between WebKit and Blink**
  - Same or different authors
Social layers

• **Committer**
  • 10-20 patches
  • Support of 3 reviewers
  • Good understanding project policies and good collaboration skills
  • ~270 committers (that are not yet reviewers)

• **Reviewer**
  • 80+ patches
  • Support of 4 reviewers from several ports
  • Unofficial reviews
  • ~130 reviewer

http://trac.webkit.org/browser/trunk/Tools/Scripts/webkitpy/common/config/contributors.json
http://www.webkit.org/coding/commit-review-policy.html
Distribution of reviewed commits last year

[http://blog.bitergia.com/2013/03/01/reviewers-and-companies-in-webkit-project/](http://blog.bitergia.com/2013/03/01/reviewers-and-companies-in-webkit-project/)
Get involved!
Tests, Tests, Tests

- **W3C**
  - https://github.com/w3c/web-platform-tests/

- **WebKit regression test-suite**
  - http://trac.webkit.org/browser/trunk/LayoutTests

- **You can help**
  - Upstream LayoutTests to W3C
  - Remove duplicated tests (after imported from W3C), WebKit bug #111513
  - Convert tests to reftests - http://trac.webkit.org/wiki/Writing%20Reftests
File bugs

Know *where* and *how* to file them

1. bugs.tizen.org – for Tizen
2. bugs.webkit.org – for WebKit
3. w3.org/Bugs/Public – for W3C

http://ejohn.org/blog/a-web-developers-responsibility/
http://fantasai.inkedblade.net/style/talks/filing-good-bugs/
Existing WebKit Bugs

• ~17,000 open bugs on bugs.webkit.org
• Bugs are still relevant and active back from 2005. Bug #15553 from 2007 just fixed on Feb-2013 (Opera).

You can help

• Categorize, prioritize, reproduce, add info, clarify, find a developer, find duplicates, close (check with reporter).
Contribute code

- Test driven development
  - Make you changes
  - Built and test (module, LayoutTests) locally
  - Upload your patch and check ews (early warning system) – bugs.webkit.org
  - Iterate with the community and get an r+ – irc (#webkit on freenode), webkit-dev
  - Check build bot after it lands and watch for regressions - build.webkit.org
  - http://trac.webkit.org/wiki/CommitterTips
Contribute code

• **Do your homework**
  • Code history in revision control
  • W3C specification,
  • Other engines behavior
  • Add yourself to watchlists

• **You can help**
  • Fix bugs
  • Gardening [http://trac.webkit.org/wiki/Keeping%20the%20Tree%20Green](http://trac.webkit.org/wiki/Keeping%20the%20Tree%20Green)
  • Code maintenance, remove dead code, refactor code, find FIXME, [http://trac.webkit.org/wiki/CommitterTips](http://trac.webkit.org/wiki/CommitterTips)
WebKit Architecture
**Major Components**

- **WebKit and WebKit2**  
  *(Embedding API)*

- **Bindings**  
  *(JavaScript API, Objective-C API)*

- **WebCore**  
  *(HTML, CSS, DOM, etc, etc)*

- **Platform**  
  *(Network, Storage, Graphics)*

- **JavaScriptCore**  
  *(JavaScript Virtual Machine)*

- **WTF**  
  *(Data structures, Threading primitives)*
Graphics backends

- GraphicsContext
  - Safari port
    - CoreGraphics
  - EFL port
    - Cairo
  - Gtk port
  - Qt port
    - QPainter

Rendering quality in graphics stack

Accelerated rendering
WebKit2

UI Process

Application

WebKit (UI Process)

WebKit (Web Process)

WebCore

JS Engine

Web Process
Lifecycle of a page

Network → Loader → HTML Parser → DOM → Script

CSS → Render Tree → Graphics Context
Split between WebKit & WebCore

- WebCore/loader
- WebCore/platform/network
- FrameLoaderClient - does the network request

2 code paths - Frames (FrameLoader) vs. Resources (DocLoader)
Loading

FrameLoader and DocLoader are both "front ends" to the loading system.

FrameLoader has a policy, provisional, and committed DocumentLoader.

Main resources are not currently cached (except in the PageCache, which is for constructed DOMs).

Cache stores resources in memory to avoid round-trips to the platform layer.

Each type of subresource has its own subclass of CachedResource.

SubresourceLoader is owned by DocumentLoader.

ResourceHandle is the "back end" of the loader and the interface to the platform layer.

https://www.webkit.org/blog/1188/how-webkit-loads-a-web-page/
Loading states for a frame

Policy phase (allow vs. deny)
• block popups
• start process for cross process navigation

Provisional phase (download vs. commit)
• Pass download to download manager

Committed phase (content rendered from server to render)
• start parsing
Caches

- HTTP disk cache (Port specific implementation)
- Memory cache (e.g. decoded images in WebCore)
- Page cache (back/forward navigation in WebCore)
HTML parser

Bytes

Characters

Tokens

Nodes

DOM

3C 62 6F 64 79 3E 48 65 6C 6F 2C 20 3C 73 70 61 6E 3E 77 6F 72 64 21 3C 2F 73 70 61 6E 3C 2F 62 6F 64 79 3E

Tokenizer

StartTag: body
Hello,
StartTag: span
world!
EndTag: span

TreeBuilder

body
Hello,
span
world!

body
Hello,
span
world!
DOM + CSS \(\rightarrow\) RenderTree

```html
#footer { position: fixed; bottom: 0; left: 0 }
body > span { font-weight: bold; }
```

---

Left Side:

- html
  - head
    - title
      - Greeting
  - body
    - Hello,
    - span
      - world!
    - img

Right Side:

- Render Block
  - Render Inline
    - Render Text
    - Render Text
    - Render Image
      - bold
      - fixed

---

**Layout**
RenderLayer

- Render Block
  - Render Block
    - Render Text
    - Render Inline
      - Render Text
      - Render Image
Future challenges
What runs in a process?

- One **application** process - initially one process for the whole browser application

- **Renderer** processes (per tab/origin/site instance) + plugin process + browser process

- **Network** process
  [https://docs.google.com/document/d/1ihpwbIG_EDirnLibkkglEtyFoEEcf7t9XNAn8JD4fQY/edit?pli=1](https://docs.google.com/document/d/1ihpwbIG_EDirnLibkkglEtyFoEEcf7t9XNAn8JD4fQY/edit?pli=1)

- **GPU** process

- **iFrame** process
Trade-offs for the process model

- HW capabilities (multicore CPU or GPU)
- Responsiveness (offload main UI thread)
- Security (process isolation)
- Robustness (software crash)
- Memory management (shared vs. cloned data)
- Process vs. thread
- Configurability (change model dynamically, reuse process)
API design for the Web

• What is the right level of abstraction?
  • Expose the service capability (pick a profile pic)
  • Expose the HW capability (camera api, gallery/file api)

• What level to expose to?
  • OS, browser chrome, renderer, web

• Examples of challenging APIs
  • Network characteristics, contact API, NFC

http://www.w3.org/Mobile/mobile-web-app-state/
API Security/Execution model

• When to allow access to an API
  • Only installed things (web apps, extensions, etc)?
  • Separate trust levels
• When and how to prompt the user
  • Installation time vs. runtime when needed
  • All permissions at once or one by one
• Separate versions of the API
  • different security requirements (high level vs. low level)
Conclusion

• Hacking on WebKit is exiting and there are ways to get involved at various commitment and technical levels.
• The best way to influence the web is directly contribute to upstream projects.