HTML5/WRT: How competent is your code?

Guobing Chen & Raji Shunmuganathan
Intel Corp

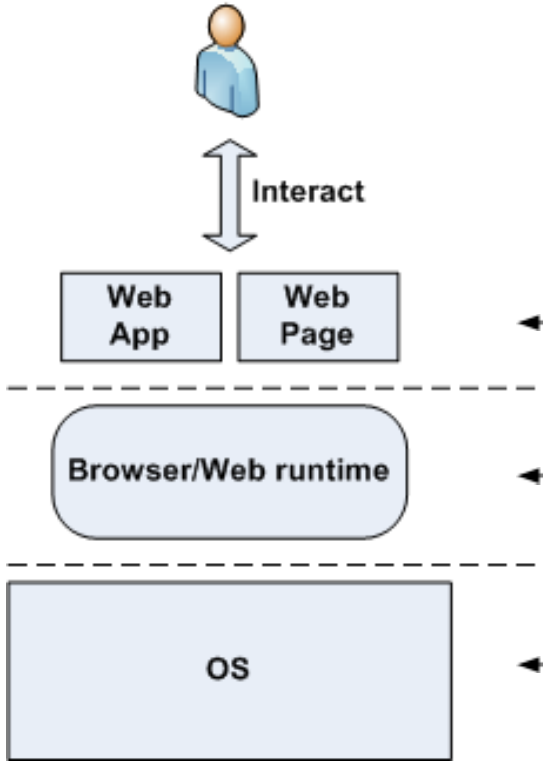
Agenda

- Introduction
- Web performance on two levels
  - Programming level: web app and page efficiency
  - Library level: library and platform efficiency
- Programming-level performance
- Library-level performance
- Summary
- Q & A

Introduction

- **Web platform** as the future for mobile devices
- **Performance** is **crucial** to make **Web platform** successful

Web performance on two levels



- End user interacts with web app/page
- web app/page is based on browser/WRT
- Browser/WRT is based on system

← ***Programming-level performance***

← ***Library-level performance***

← **System-level performance**

Programming-level performance

- Web app and page efficiency

What is the situation for web apps?

- **Native app**
  - lots of tools for performance tuning
    - ICC compiler to optimize C/C++ code
    - VTune for hotspot finding
- **Web app**
  - Few similar tools available in industry
  - What you can get
    - Google-closure: reducing the code size
    - Chrome-Profiles-tool: profiling CPU hotspot of JS
    - pieces of BKMs for web apps programming

Our solution: WAPP

- **WAPP**: **W**eb **A**pp **P**rogramming **P**erformance
- Methodology
  - A benchmark suite to measure the performance **difference** between **different web programming styles**
  - **knowledge about the best programming style** in certain situations (browser, OS/platform, usage scenario)
  - **Apply the knowledge** to optimize the web app
- Typical areas for optimization
  - Basic JS operations
  - Animation

Findings with WAPP

- Finding 1: ***Loop - for, while, each*** (**Unit:** Ops/sec)

Programming style	NetBook / Chromium	Notebook / Firefox	iPad2 / Safari	Result
for(var i=0;i<len;i++)	5292	10366	5668	
for(i=0;len=arr.length;i<len;i++)	3753	7185	4069	
for(i=0;i<arr.length;i++)	5515	10290	5756	
for(var in arr)	1626	3736	1679	Loser
for(i=arr.length;;i--)	6730	13726	7028	Winner
Array.foreach()	10395	11574	6366	Winner
$.each()	9666	10661	7732	Winner
While	5437	10578	5604	

- **RED**: The worst one
- **GREEN**: The best one

Findings with WAPP

- Finding 2: ***Array: Clone*** (**Unit:** Ops/sec)

Programming style	NetBook / Chromium	Notebook / Firefox	iPad2 / Safari	Result
Array.slice(0)	15889	34838	12613	Winner
Array.concat()	16915	34709	12053	Winner
Loop copy	5000	6250	4876	Loser

- **RED**: The worst one
- **GREEN**: The best one

Findings with WAPP

- Finding 3: ***Array: Clear*** (**Unit:** Ops/sec)

Programming style	NetBook / Chromium	Notebook / Firefox	iPad2 / Safari	Result
Array.slice	18170	35133	13894	Loser
Array.length=0	21709	41218	16467	Winner
Array=[]	20714	24755	16293	Loser

- **RED**: The worst one
- **GREEN**: The best one

Findings with WAPP

- Finding 4: ***Dom Operation: Append*** (**Unit:** Ops/sec)

Programming style	NetBook / Chromium	Notebook / Firefox	iPad2 / Safari	Result
appendChild()	1933	15854	2476	Winner
appendTo()	468	5505	621	
append()	709	6602	841	
$.html($.html()) (JQuery)	30	135	30	Loser

- **RED**: The worst one
- **GREEN**: The best one

Library-level performance

-Library and platform efficiency

Our solution: WRTBench

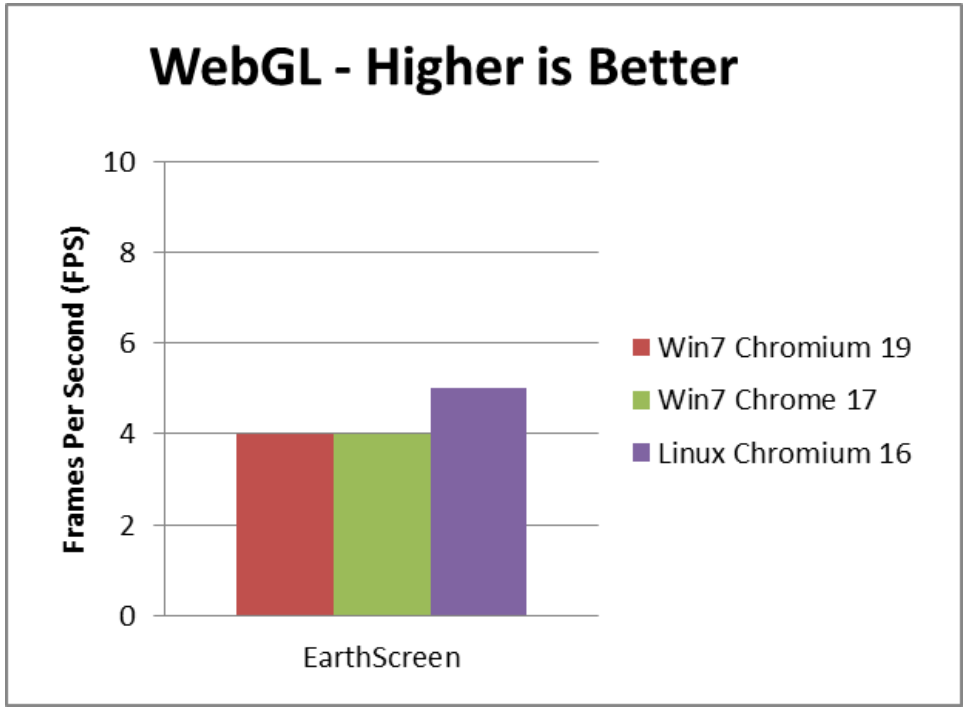
- **WRTBench**: samples the performance of Web platforms based on the common Web API feature list
- Major features
  - ***WebGL**** - Transformation, composite, texture, video
  - ***Canvas 2D***: Transformation, draw image, color, style
  - ***CSS3***: 2D & 3D Transformation, rotation, shadow
  - ***Photo processing***: Photo edit, resize, rotate, flip, color
  - ***Local storage***
  - ***File API***
  - ***JavaScript* rendering***

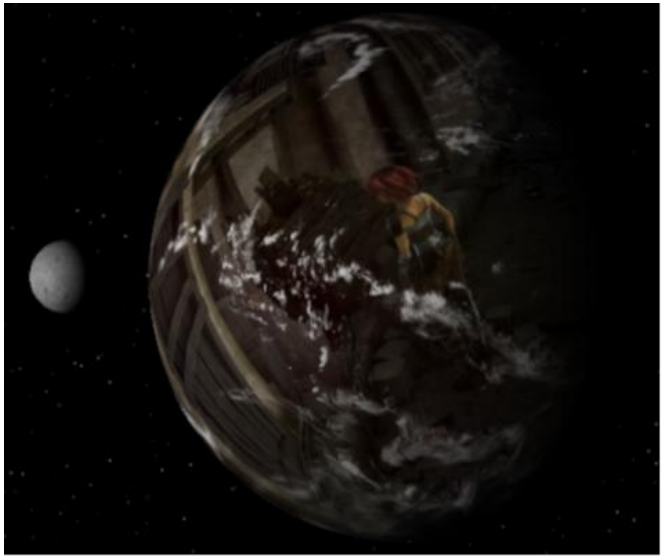
Combinations of Web platforms

OS/Browser Performance	Chromium	Chrome*	Firefox	Safari
Linux	X		X	
Win7	X	X		
iOS				X



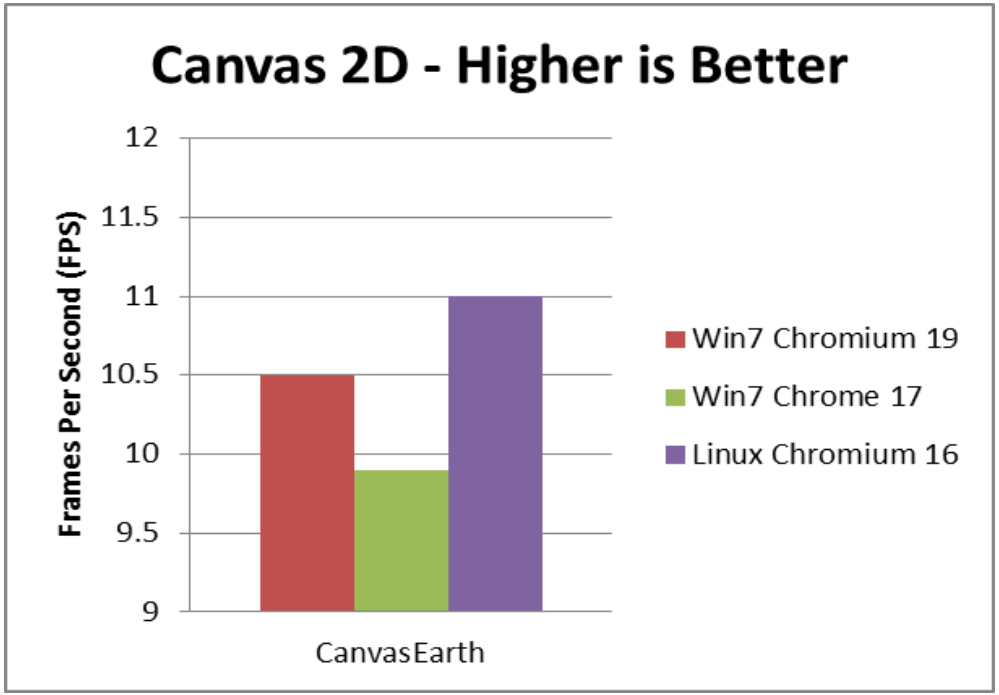

WebGL





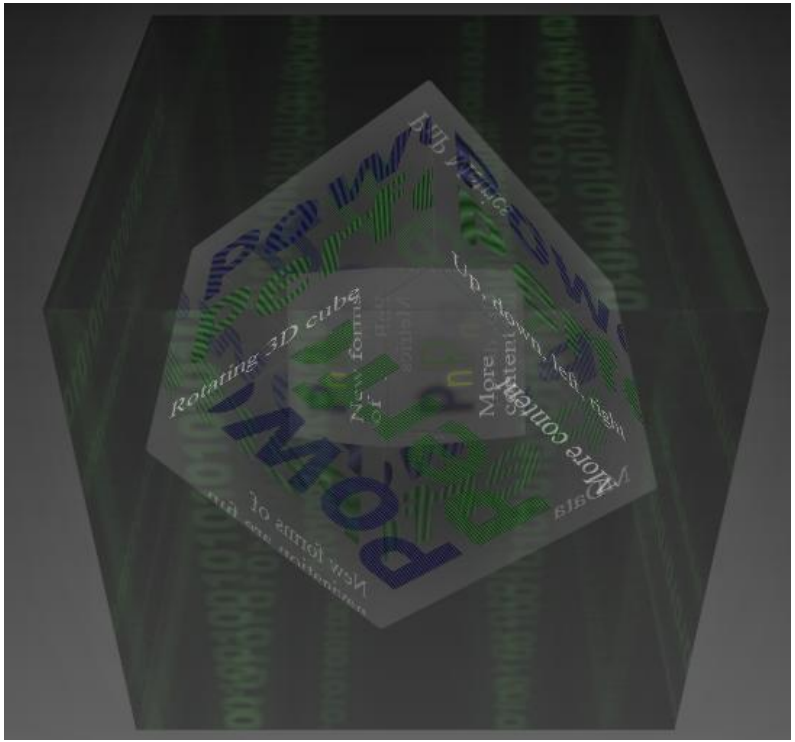
- **No WebGL support** on Safari
- Firefox still **ramping up** on Linux
- Chromium is a **winner** in this space, taking advantage of Linux

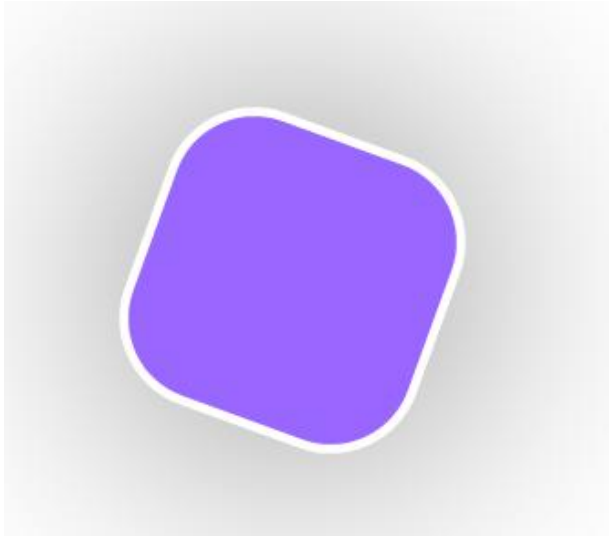
Canvas 2D Rendering



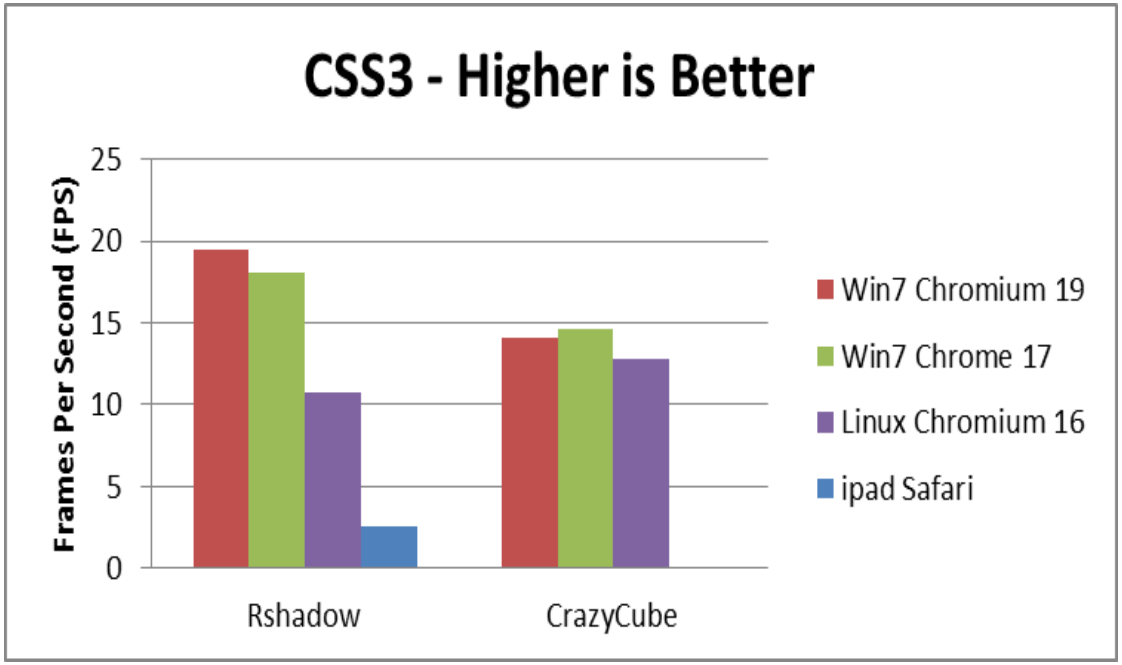
- Firefox and Safari **lack** the Canvas 2D aspect
- Chromium performs **much better** then Chrome
- Linux provides **good** performance

CSS3



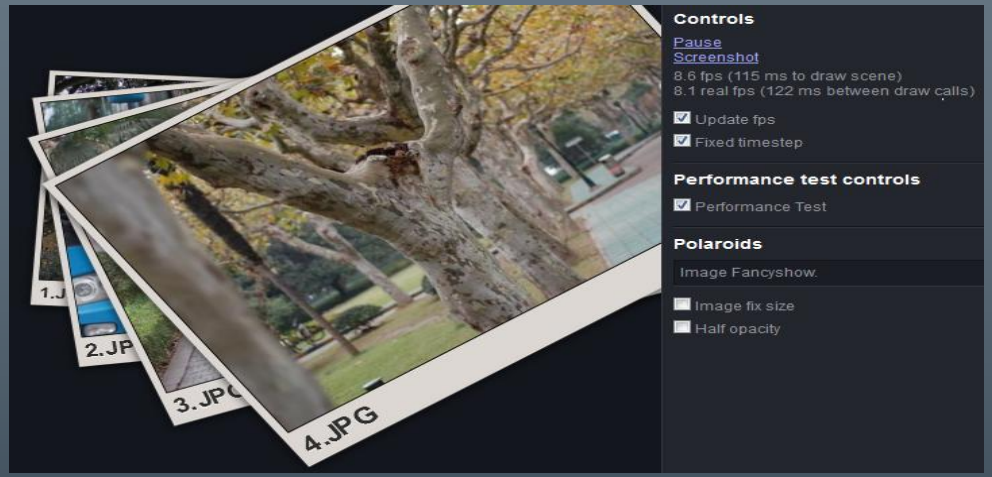


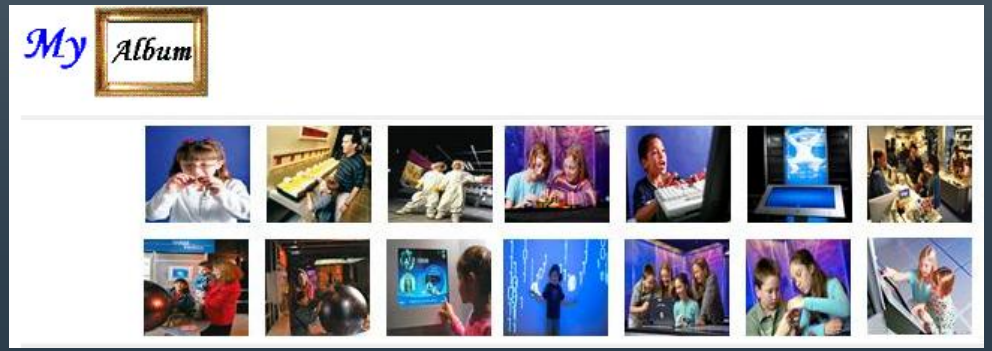
CSS3



- Again a winning space for Chromium & Chrome
- Firefox still ramping up on Linux
- Safari **lags far behind**
- Linux getting closer to Win7 in 3D model

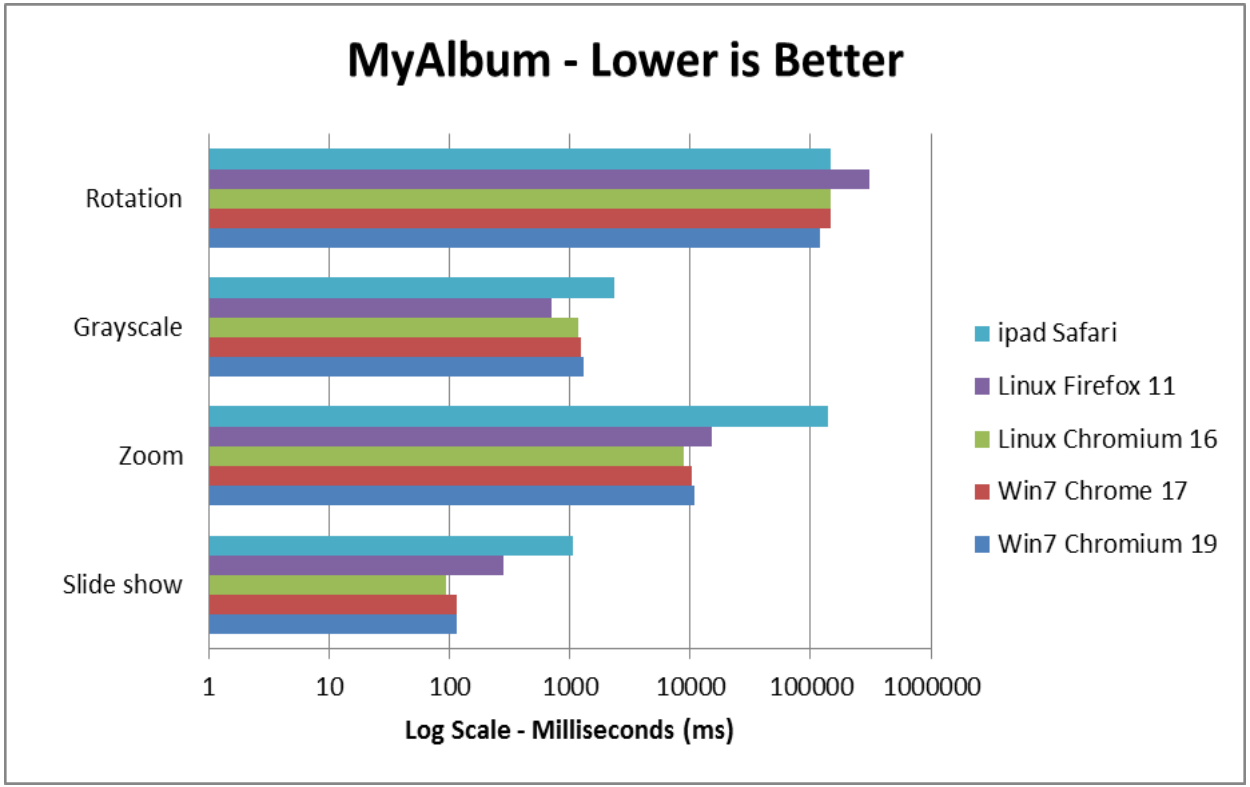
HTML5 Photo Processing





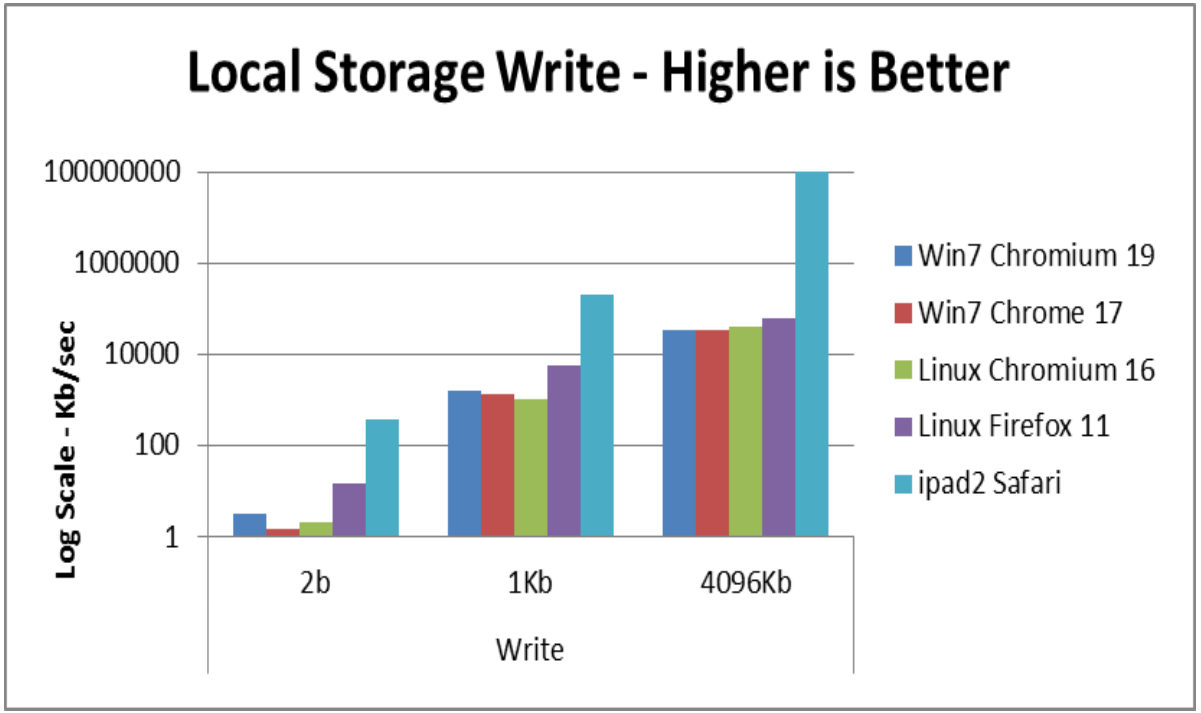
TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012

HTML5 Photo Processing



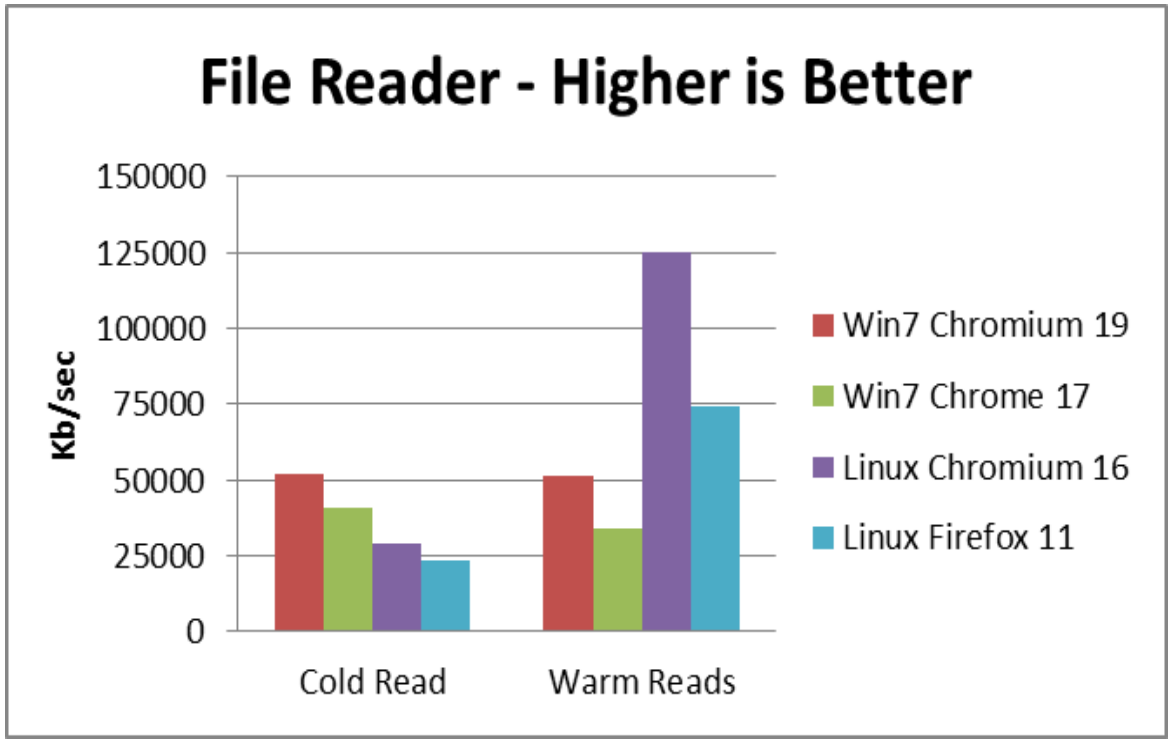
- No major differences in performance
- Safari gaps a little behind compared to the remaining browsers

HTML5 Local Storage



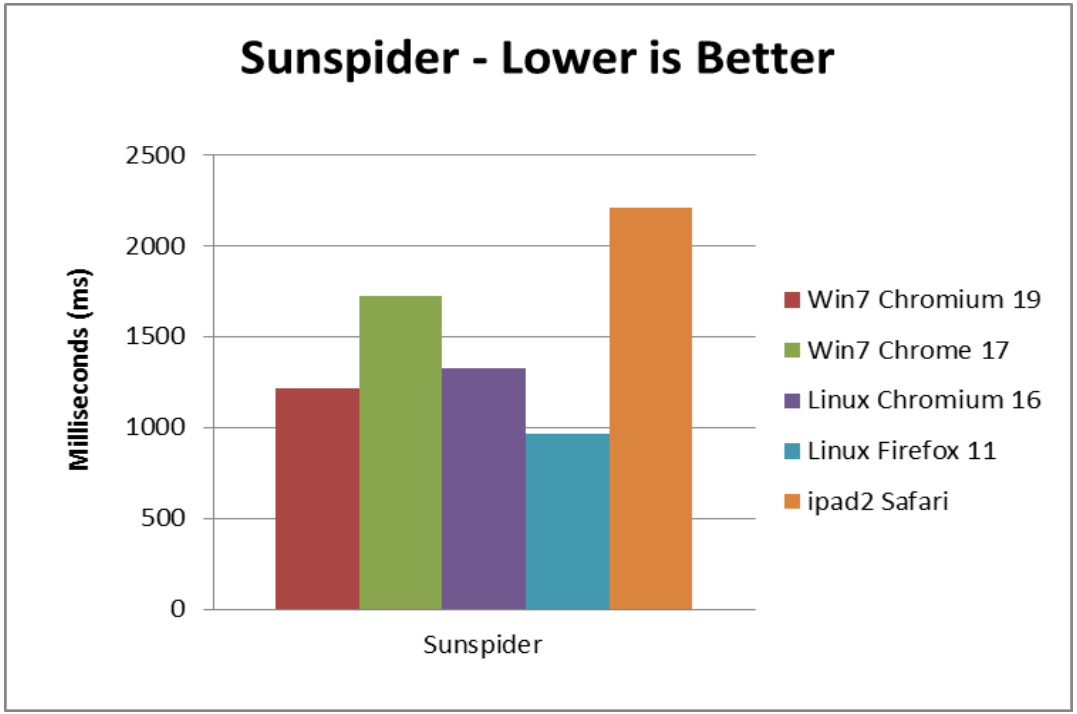
- **Trend is the same** across most browsers
- Safari outperforms rest

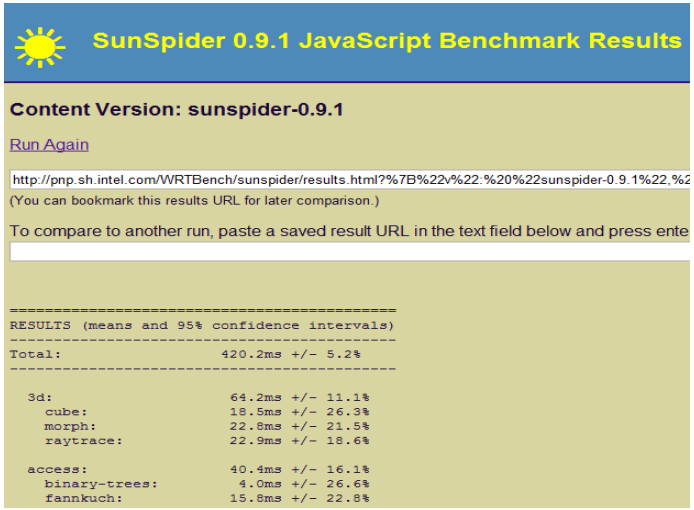
HTML5 File API



- Media file ~30 MB
- No support on Safari yet
- Linux takes good advantage of **caching**

JavaScript





- Firefox projects great performance with SpiderMonkey
- Linux almost **equates** Win7 performance wrt Chromium

Summary

- Web performance can be improved on two levels:
  - Program web app with the **right code**
    - Avoid using statements/APIs which lead to bad performance
    - Using native APIs instead of 3rd party libraries for DOM operation
  - Optimize web platform especially on the **major features**
    - Linux gives out the **good** performance in the areas of WebGL, Canvas 2D and Warm Reads
    - In **par** with most other browsers wrt Local Storage & Photo Processing
    - **Getting competitive** with Windows in CSS3 and JavaScript

Please contact us if you are interested: rajalakshmi.shunmuganathan@intel.com, guobing.chen@intel.com

Q & A

TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012