Pushing The Limits of Web Browsers

... or why speed matters

by Lars Bak, Google Inc

Background

- *26 years optimizing implementations of object-oriented programming languages*

Systems I Have Worked On

- *86-91 Beta runtime system*
- *91-94 Self virtual machine and IDE*
- *94-95 Strongtalk virtual machine*
- *95-00 Hotspot JVM*
- *00-02 Monty JVM*
- *02-06 OOVM Smalltalk*
- *06-11 V8 JavaScript engine*
- *11-12 Dart programming platform*

Agenda

- *Motivation for virtual machines*
- *The past: Self & Hotspot*
- *The present: V8 & Dart*
- *What I've learned building VMs*

Why Language Based VMs

- *Platform independent execution*
- *Sandboxing*
- *Optimizations can take place at runtime*
- *Debugging is possible in production*
- *Loading of third part code at runtime*

VM Side Effect

- *When the VM gets faster*
- *Existing programs run faster*
- *Programmers get room for software innovation*

Self

- *Research from Stanford University and Sun Microsystems Labs. Inc.*
- *Managed by David Ungar and Randall B. Smith*
- *Staged adaptive compilation technology*
  - *inline caching, inlining, and deoptimization*
- *Efficient memory management*
- *Outsiders suspected memory was traded for speed*

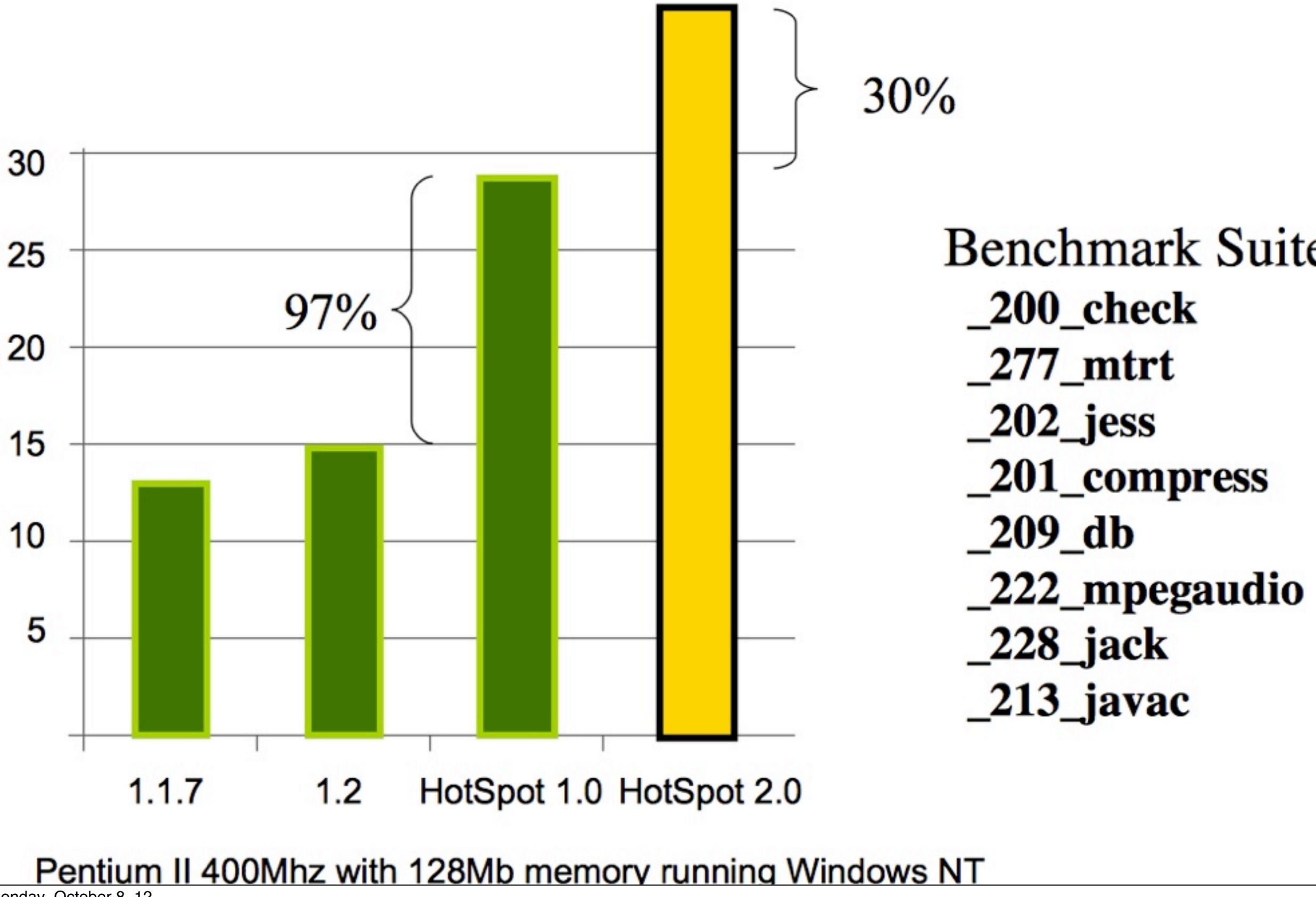
Strongtalk

- *Implemented a high performance Smalltalk system using adaptive compilation BUT without memory bloat*
- *Applied vm techniques to Java and called it Hotspot*
- *Team: David Griswold, Gilad Bracha, Urs Hoelzle, Robert Griesemer, Steffen Grarup, Srdjan Mitrovic, and me*
- *Startup acquired by Sun Microsystem 1997*

Hotspot

- *Technology*
  - *Based on the Strongtalk VM internals*
  - *Interpreter + simple JIT*
  - *Generational GC*
  - *Fast synchronization*
  - *First implemented with cooperative threads*

SPEC jvm98 www.spec.org/jvm98



Hotspot Still Going Strong

- *Rumors has it that Hotspot is still an active project after 17 years*
- *Caveat: I left the Java world 10 years ago*

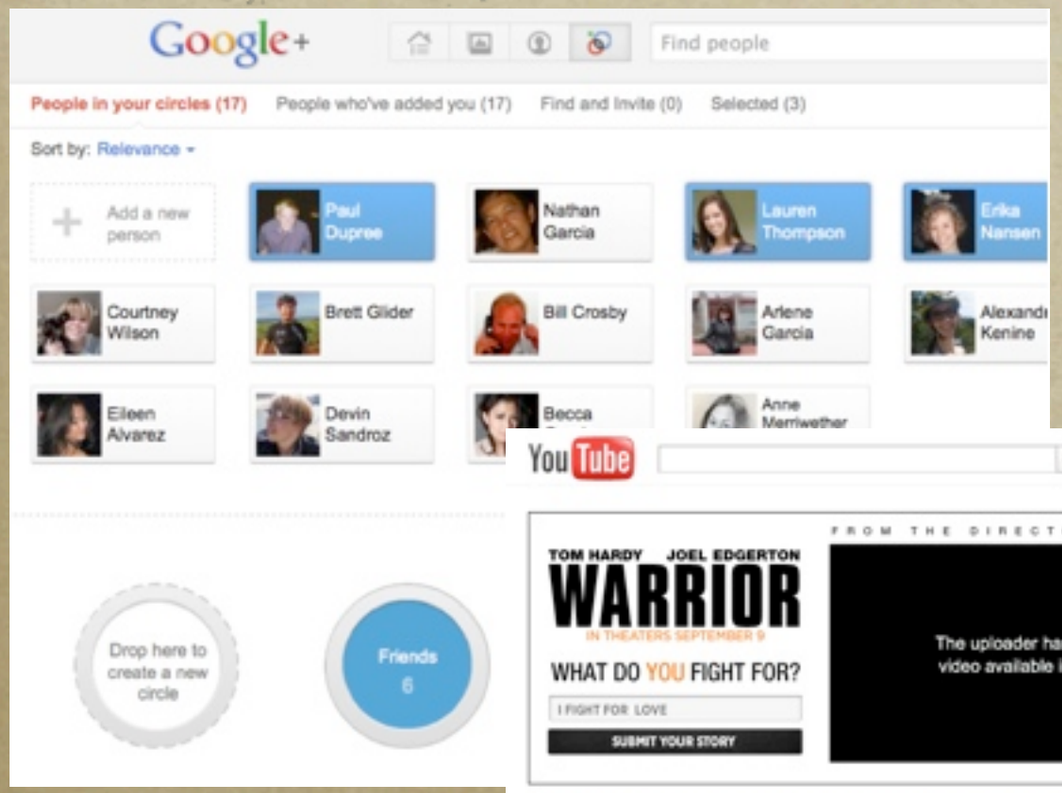
Hotspot Reflections

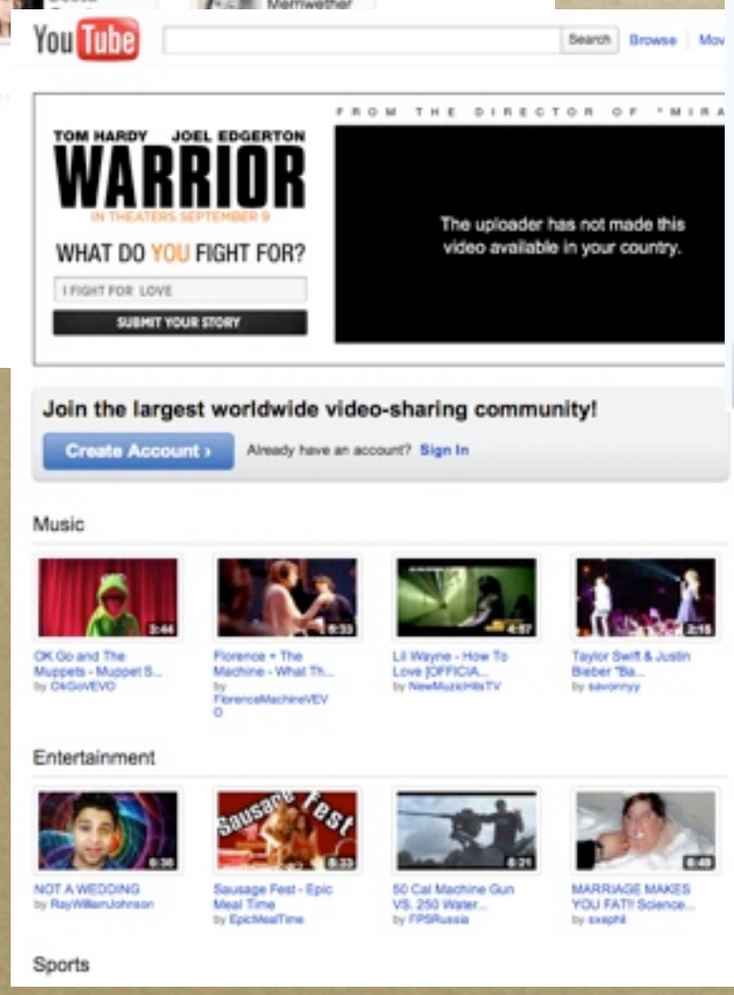
- *Conservative collectors hide problems*
  - *JSR bytecode caused int/pointer slots*
  - *Raw pointers are likely to leak*
- *Allowing virtual behavior on heap objects from C++ is problematic*
  - *'this' subject to GC*
- *Bug-tail for multi-threaded execution is long*

V8: A JavaScript Engine

- *Hired by Google in fall 2006 to improve performance of JavaScript*
- *Matched strategy for Chrome*
  - *Simple, secure, and fast*

Speed Enables Applications





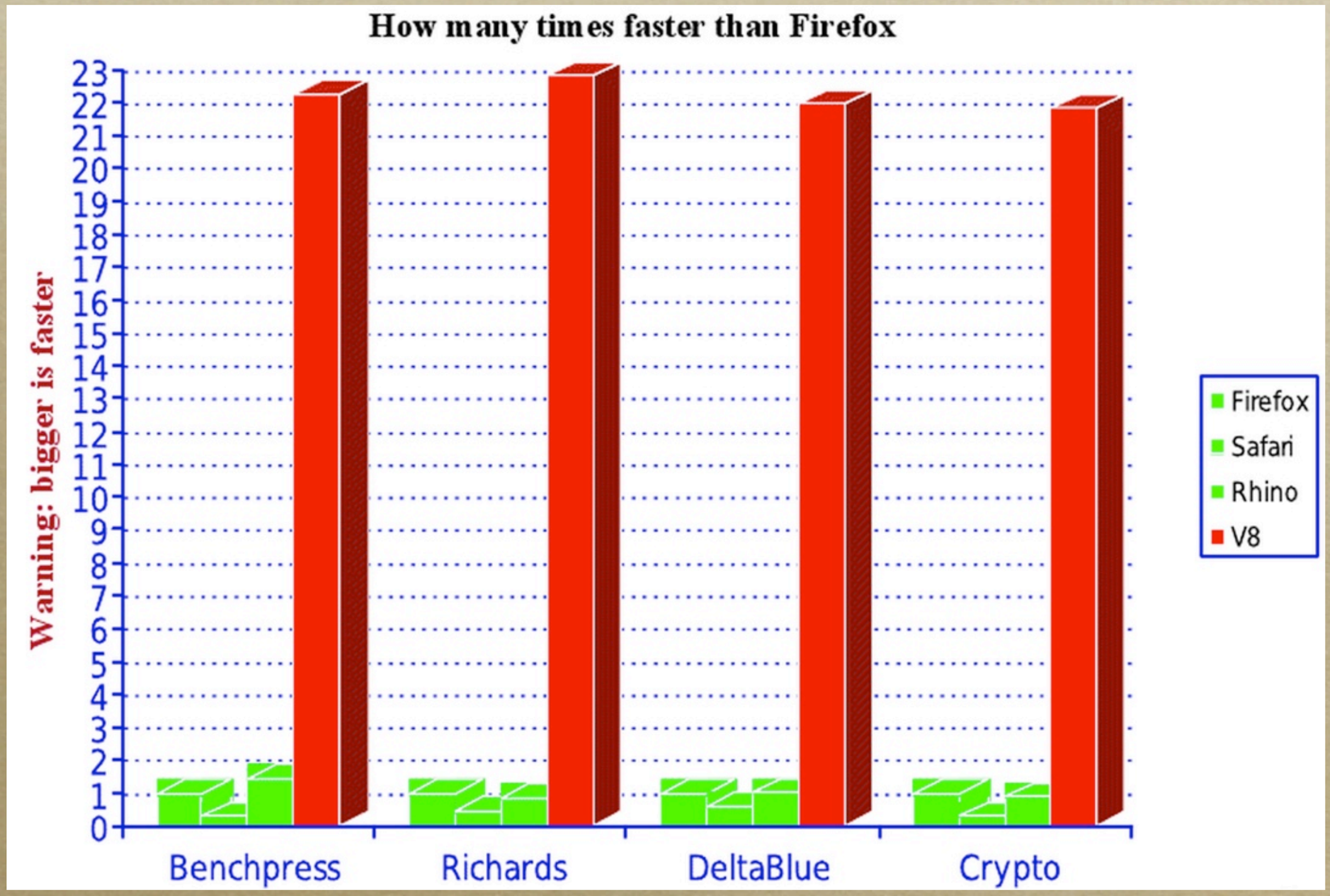
Original V8 Goals

- *Make JavaScript 10 times faster*
- *Enable JavaScript programs to scale*
- *Open source the project*
- *... and raise the industry performance bar*

Ideas Behind V8

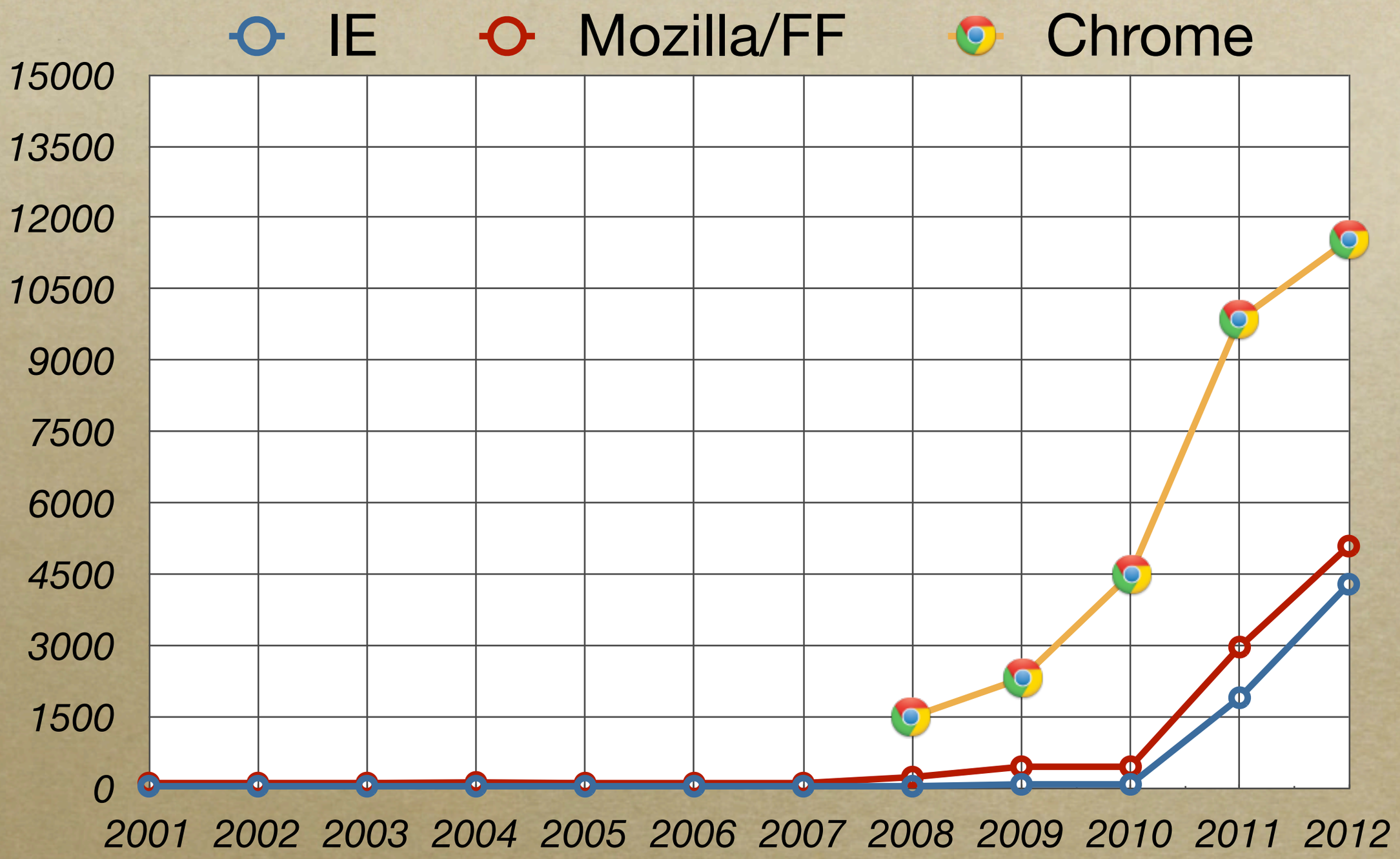
- *Map transitions*
  - *enables standard optimizations*
- *No bytecodes only machine code*
- *Efficient memory management*

Progress After 4 Months



JavaScript Performance Timeline

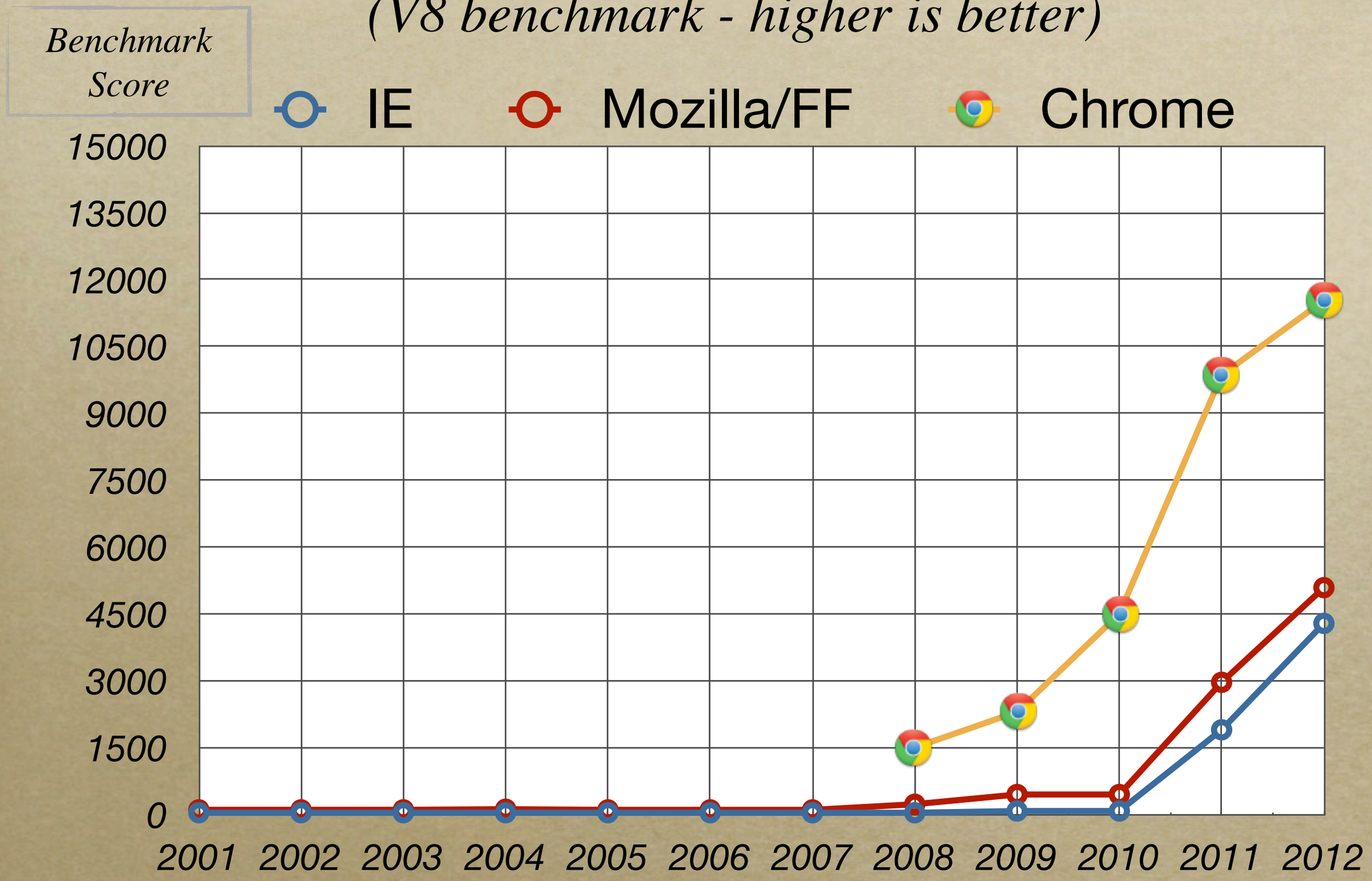
(V8 benchmark - higher is better)



by Jim Hugunin

JavaScript Performance Timeline

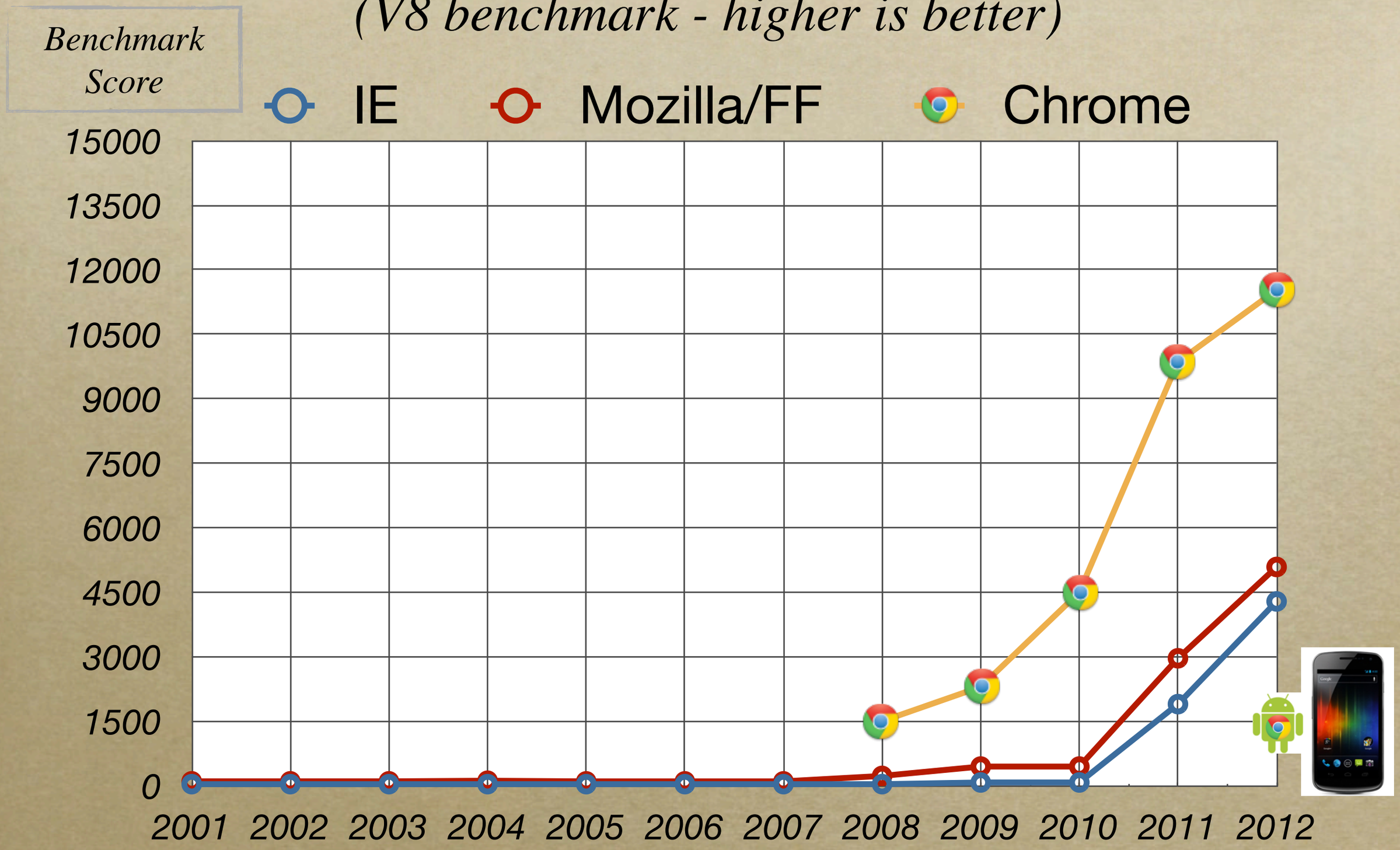
(V8 benchmark - higher is better)



by Jim Hugunin

JavaScript Performance Timeline

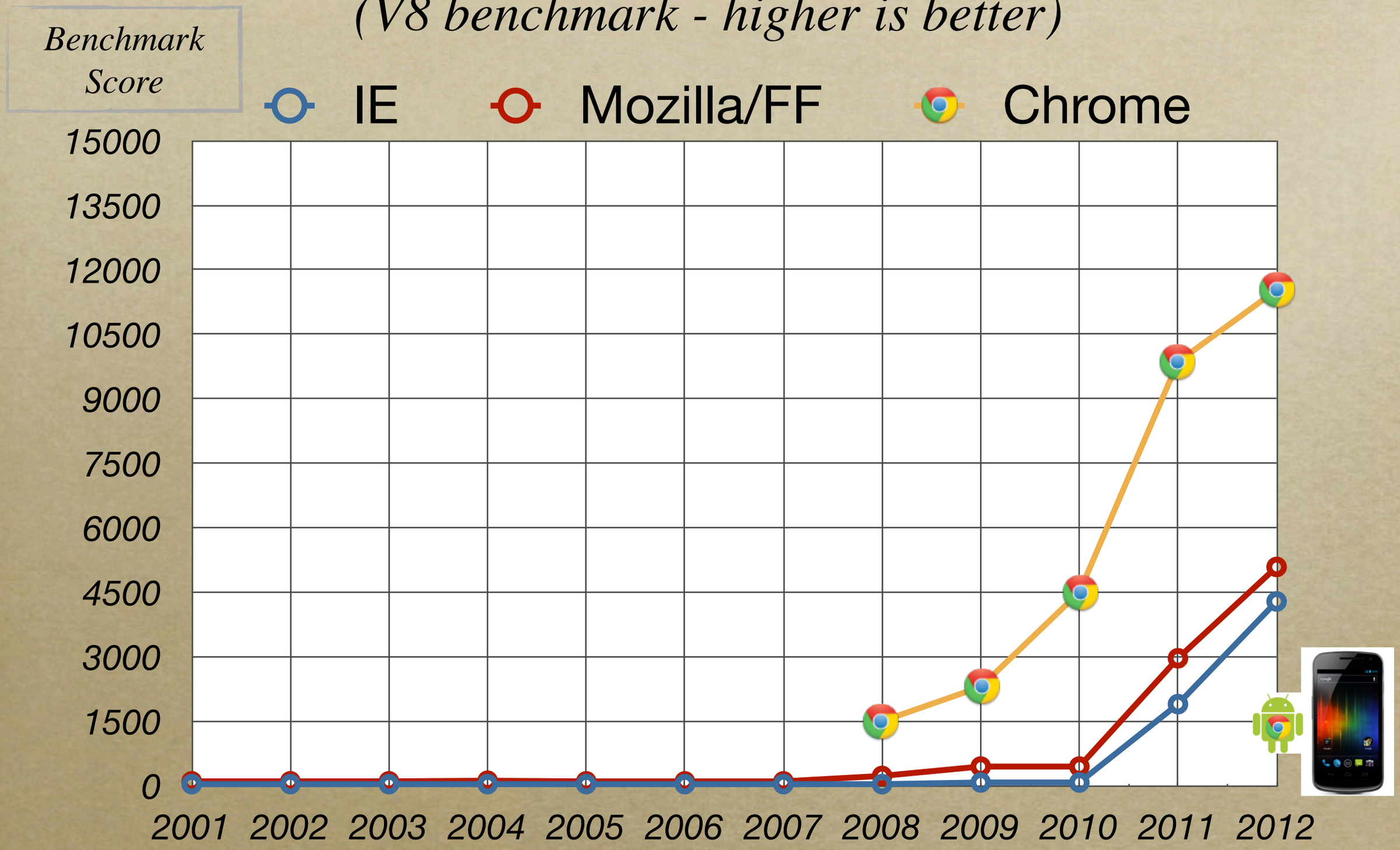
(V8 benchmark - higher is better)



by Jim Hugunin

JavaScript Performance Timeline

(V8 benchmark - higher is better)



Warning: 5 month old numbers

by Jim Hugunin

V8 Reflections

- *Map transitions worked much better that expected*
- *Exposing raw pointers in the C++ runtime system was a mistake*
- *Making V8 a separate project resulted in unexpected users*

V8 Users

- *Chrome web browser*
- *Android web browser*
- *node.js (server side JavaScript)*
- *Web OS*

V8 Users

- *Chrome web browser*
- *Android web browser*
- *node.js (server side JavaScript)*
- *Web OS*

Over 12500 revisions since being open sourced

JavaScript is Now Faster but ...

- *Promotes spaghetti style programming*
- *Object-oriented programming is hard*
- *Objects can changed on-the-fly*
- *No support for libraries*
- *Tool support is weak*
- *Slow application startup*
- *Runtime performance is unpredictable*

JavaScript Example

```
assert(2.0 == '2' == new Boolean(true) == '1');
```

The expression is true BUT all the implicit conversions will make your head explode

The Web is Great

- *Developing small applications is easy*
- *Platform independence*
- *No installation of applications*
- *Supports incremental development*

The Web is Great

- *Developing small applications is easy*
- *Platform independence*
- *No installation of applications*
- *Supports incremental development*

... but innovation is crucial for survival

Goals for A New Web Platform

- *Support for programming in the large*
- *Compared to V8 performance*
  - *Ultra-fast startup, 10 times faster*
  - *Predictable performance, 2 times faster*
- *Avoid fragmentation of the web*

The Dart Programming Language

- *A simple and unsurprising OO language*
  - *Class-based single inheritance*
  - *Interfaces with default implementation*
  - *Optional static types*
  - *Real lexical scoping*
  - *Single-threaded*

... and it is compact and readable

Inspiration to Dart

- *Object model inspired by* **Smalltalk**
- *Compilation strategy inspired by* **Self**
- *Optional types inspired by* **Strongtalk**
- *Isolates design inspired by* **Erlang**
- *Syntax and constructs inspired by* **JavaScript** & **C#**

Designed for a VM

- *Straight forward semantics*
- *Simple object model*
- *No class initialization*
- *Applications are declared*

Dart Sample

```
class Point {
  Point(this.x, this.y);
  var x, y;
  operator +(other) => new Point(x + other.x, y + other.y);
  scale(factor) => new Point(x * factor, y * factor);
  toString() => "($a,$y)";
}

main() {
  var a = new Point(10, 10);
  var b = new Point(2, 3).scale(10);
  print("result = ${a+b}");
}
```

Dart Sample With Types

```
class Point {
  Point(this.x, this.y);
  num x, y;
  Point operator +(Point other)
     => new Point(x + other.x, y + other.y);
  Point scale(num factor) => new Point(x * factor, y * factor);
  String toString() => "($a,$y)";
}

main() {
  Point a = new Point(10, 10);
  Point b = new Point(2, 3).scale(10);
  print("result = ${a+b}");
}
```

Covariant Generic Types

```
main() {
  List<Apple> apples = tree.pickApples();
  printFruits(apples);
}

void printFruits(List<Fruit> fruits) {
  for (Fruit each in fruits) each.print();
}
```

"List<Apple> is a subtype of List<Fruit>"

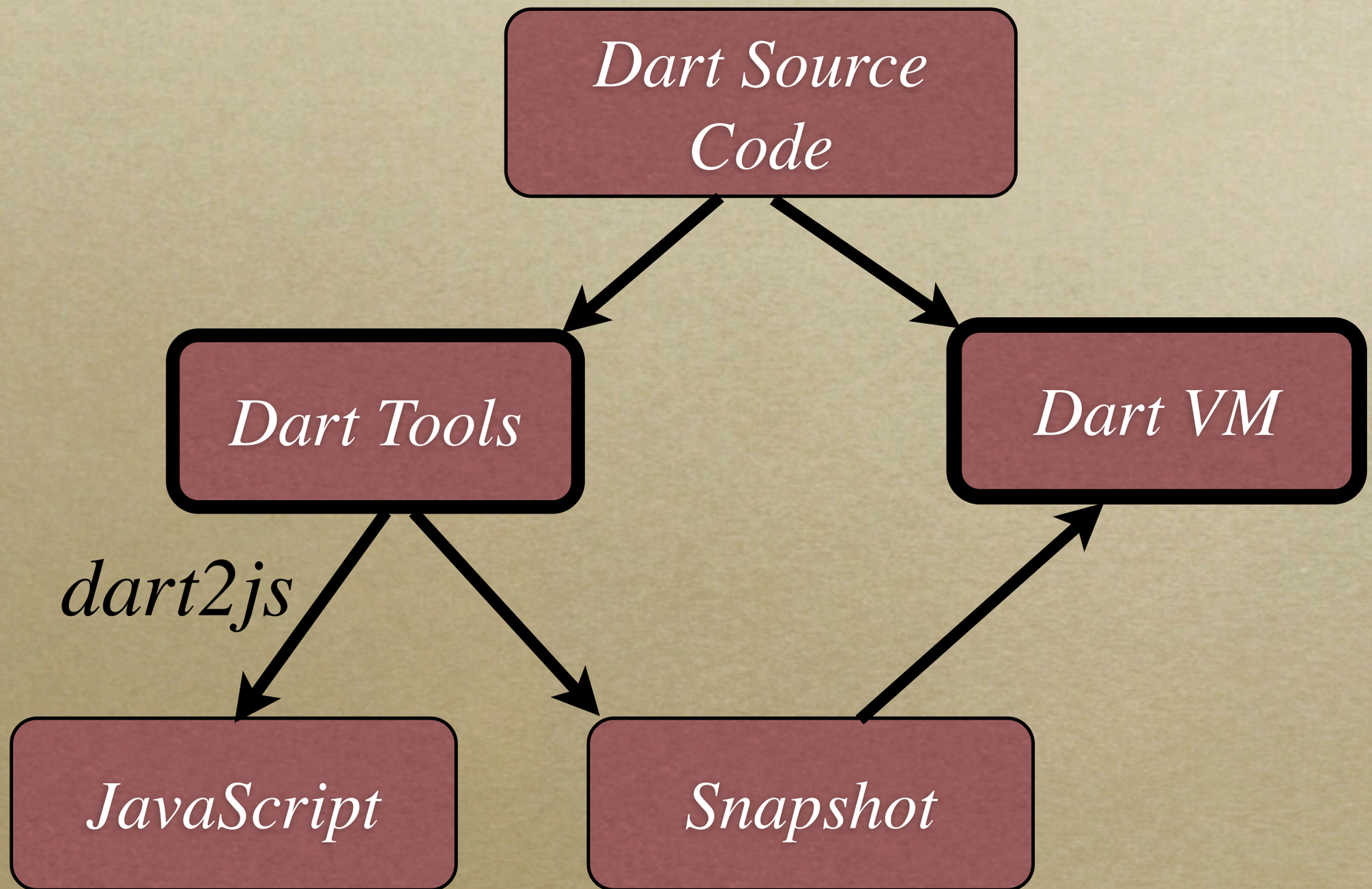
Dart Optional Type System

- *Not your traditional static type system*
  - *Used for specifying programmer intent*
  - *Types can be introduced gradually*
  - *Adding types to fields, variables, and method signatures does not change behavior*

Dart Optional Type System

- *Allows implicit down casting*
  - `T x = o;` → `assert(o == null || o is T);`
  - *Verified only in checked mode*
  - *Type checker only issues warnings*

Platform Independence



Current Dart Performance

Goal chart	Scores			Relative to v8	
Benchmark	v8	dart	dart2js	dart	dart2js
DeltaBlue	272.36	308.69	185.91	113.34%	68.26%
Richards	396.80	445.05	247.75	112.16%	62.44%
NBody	15931.00	17099.00	10724.50	107.33%	67.32%
BinaryTrees	9.06	8.36	8.28	92.24%	91.44%
Mandelbrot	169.00	166.75	132.80	98.67%	78.58%
Fannkuch	3500.50	4192.50	2277.00	119.77%	65.05%
Meteor	6.69	5.44	2.20	81.29%	32.86%
BubbleSort	23327.48	24045.50	15662.50	103.08%	67.14%
Fibonacci	9193.50	13509.00	9403.50	146.94%	102.28%
Loop	34386.48	28574.50	35699.50	83.10%	103.82%
Permute	11081.00	16202.00	7516.50	146.21%	67.83%
Queens	118511.50	177172.49	94533.50	149.50%	79.77%
QuickSort	17325.50	14916.00	9594.50	86.09%	55.38%
Recurse	14005.50	19867.00	14453.00	141.85%	103.20%
Sieve	101199.21	103316.00	86780.50	102.09%	85.75%
Sum	74434.11	59871.50	75389.00	80.44%	101.28%
Tak	3062.00	4861.50	2489.00	158.77%	81.29%
Takl	8925.00	15516.00	8939.50	173.85%	100.16%
Towers	4965.50	5541.50	3016.00	111.60%	60.74%
TreeSort	7016.00	7786.00	5646.00	110.97%	80.47%
Geo. mean	3988.95	4505.16	3000.25	112.94%	75.21%

Dart SDK

- *Dart language specification*
- *Dart libraries*
- *Dart virtual machine*
- *Translator to JavaScript*
- *Dart IDE*
- *Integration with Chromium*

Things I Have Learned

- *Always start small with a small team*
- *Focus on solving the hardest problem first*
- *Competitive situation fuels motivation*
- *Only hire people that are smarter...*

Things I Have Learned...

- *Open source projects are great*
- *Help industry*
- *Keep your work honest*

Things I Have Learned...

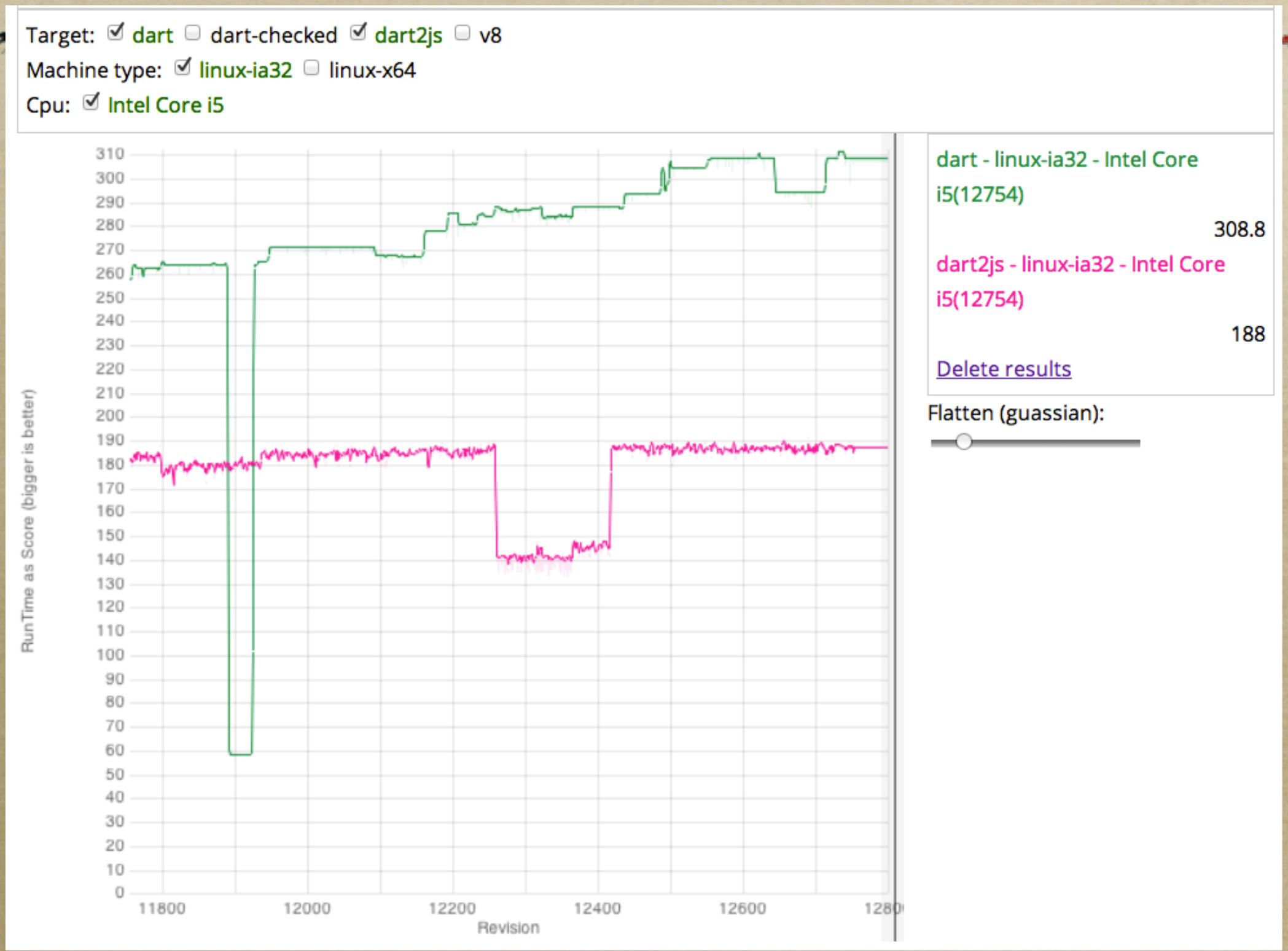
- *Track performance from day one*
  - *Run all benchmarks ...*
    - *on all revisions ...*
    - *on all platforms automatically*

Things I Have Learned...

- *Track performance from day one*
  - *Run all benchmarks ...*
    - *on all revisions ...*
    - *on all platforms automatically*

... if not, performance deteriorates

Dart Buildbot



Things I Have Learned...

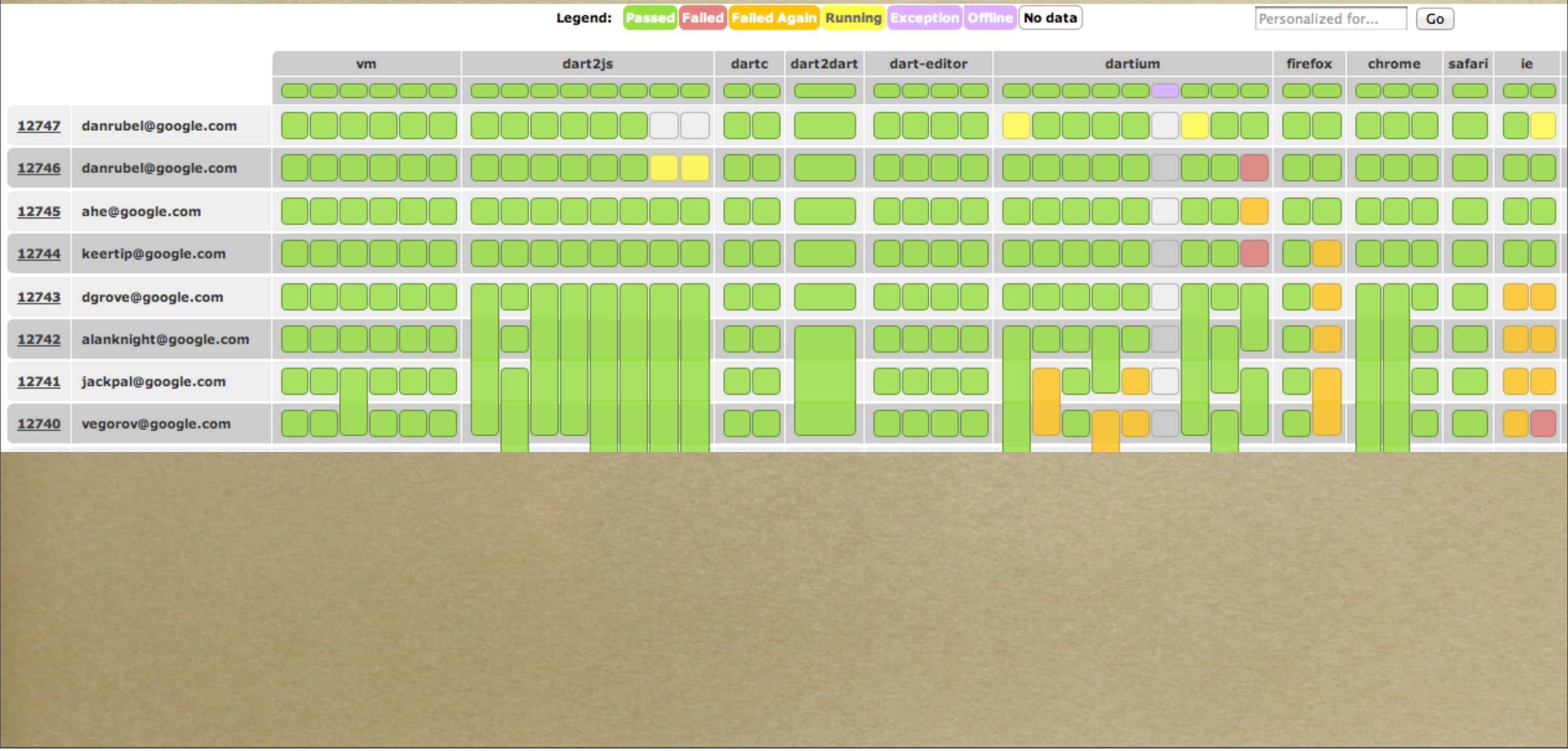
- *Testing must run from day one*
  - *Execute all tests ...*
  - *for all revisions ...*
  - *on all platforms automatically*

Things I Have Learned...

- *Testing must run from day one*
  - *Execute all tests ...*
  - *for all revisions ...*
  - *on all platforms automatically*

... if not, quality deteriorates

Dart Buildbot



Summary

- *Speed will continue to drive innovation for web applications*
- *Programmer productivity will be key as web applications get larger*

http://code.google.com/p/v8

http://dartlang.org

VM Summary

	Precise GC	*Adaptive optimizations*	*Incremental execution*	*Multi-threaded*	*Bytecodes*	*in C++*	*Open source*
Beta	●			|	|		
Self	●	●	●	|	●	●	●
Strongtalk	●	●	●	|	●	●	●
Hotspot	●	●	|	●	●	●	|
Monty	●	●		|	●	●	|
OOVM	●		●	|	●	●	
V8	●	●	●			●	●
Dart	●	●	|	/		●	●