Inside Firefox

Robert O'Callahan
robert@ocallahan.org

Mozilla

An open Internet

An open Web

A level playing field

Our tool: Firefox

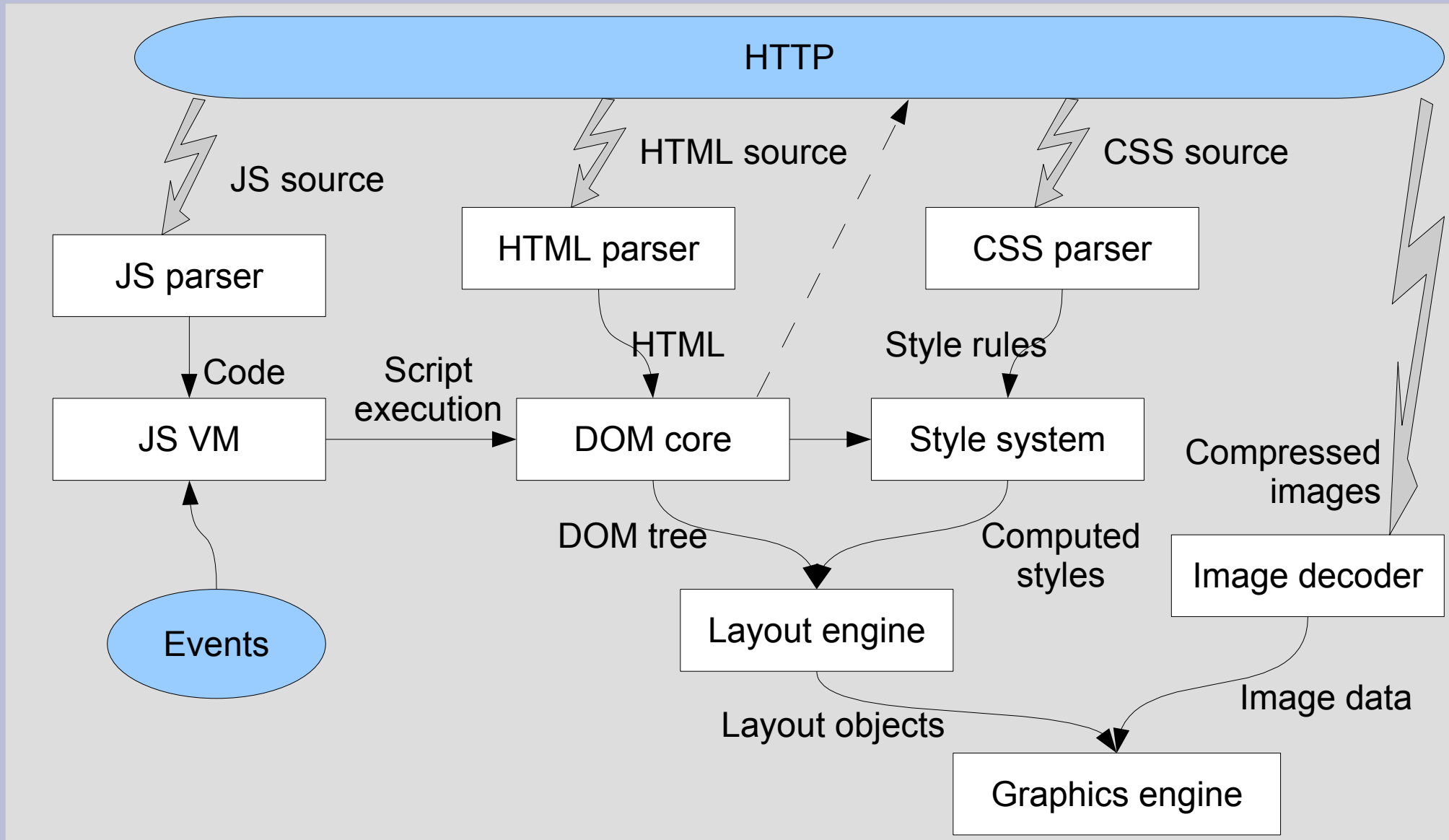
Overview

- Why Web browsers are hard
- Mozilla development processes and tools
- Running a successful open source project
- Standards and specifications in the real world
- The future of browsers and the Web

Circa 1995


THE COMPANY PAYS ME TEN DOLLARS FOR EVERY BUG I FIX IN MY CODE, RATBERT.
S. Adams E-mail: SCOTTADAMS@AOL.COM
I WANT YOU TO DO YOUR LITTLE RAT DANCE ON MY KEYBOARD SO I'LL HAVE LOTS OF BUGS TO FIX.
© 1995 United Feature Syndicate, Inc.(NYC)
HOW AM I DOING?
NOT SO GOOD. YOU JUST AUTHORED A WEB BROWSER.

Inside A Browser



Browser Requirements

- **Performance**
- **Compatibility**
- **Functionality**
- **Security**
- Usability
- Portability
- Internationalization
- ...

Performance

- #1 requirement
  - Key switching factor
- Both actual and perceived
  - Incremental results vs throughput
- Both time and space
- On a variety of devices (desktop to mobile)
- On a variety of workloads
  - Huge static documents to GMail

Compatibility

- #2 requirement
- Billions of Web pages, sloppy authors
- Dominant client “tolerates” errors
- Reverse engineering
  - Error recovery
  - Extensions to specs
  - Errors/omissions in specs
- Complex behaviours

Functionality

- The open Web platform must evolve
  - Or it will be replaced by proprietary platforms
- So we need to specify and implement
  - 2D and 3D graphics
  - Video and audio
  - Offline application execution
  - Client-side storage
  - Better programming models
  - GPU programming
  - ...

Security

- Evolving exploit technology
  - Double-free exploits
  - Method-call-on-freed-object exploits
  - Heap-buffer-overrun exploits
- **Almost any memory safety issue should be considered potentially exploitable**

More Bad News

- Evolving testing tools
  - Randomized “fuzz” testing very very effective!
  - Much more effective than human auditing and static analysis in terms of bugs/effort
  - Sophistication increasing quickly
  - Mozilla investing in fuzz testing and delta debugging (testcase minimization) tools
- Compounds with evolving exploits

Even More Bad News

- Protecting users from spoofing/phishing
- Slack domain registrars/certificate authorities
  - micros0ft.com vs microsoft.com; IDN worse
  - CA checking of name↔org binding is weak
  - “Evil domain” classification by Google, Microsoft
- Hardest problem: actually blocking users' risky behavior
  - Warnings are useless; users rationalize, dismiss, learn to ignore (Miller et al)
  - UI design issues

Mozilla Development

- 30-ish full time Gecko developers?
- 15-ish full time front-end developers?
- Plus volunteer community
- ~2.5M lines C++
- ~0.4M lines JS
- Plus bundled open-source libraries
  - cairo, SQLite, lcms, libbz2, zlib, libpng, libjpeg

Development Tools

- MSVC and gcc
- GNUmake build system
  - Unsuitable for IDEs; Eclipse CDT only hope!
- CVS (sucks), Bonsai
  - Main barrier to switching: CVS-integrated tools
  - Switching to *distributed source control* (hg)
- Tinderbox continuous build and test
- Bugzilla bug/issue tracking

QA Tools

- Automated regression tests
  - make check, reftests, mochitests
  - Relatively recent; expensive, but helpful
- Automated performance tests
- Custom fuzz testing, delta debugging
- Valgrind
- Breakpad
  - Captures stack traces from crashes in the field
- Manual testing, test minimization, and triage
  - An easy path for volunteers to contribute

Static Analysis

- Mozilla mixes C++/JS, dynamic architecture
- Complex heap invariants
- General-purpose static analysis doesn't tell us much interesting
  - Null-check bugs, some uninitialized vars
- Writing app-specific analyzers using *Dehydra*
- Writing automated refactoring tools
  - E.g. outparamdel, remove bogus OOM checks
- Very useful already, lots of scope for more

Performance Analysis

- Profiling tools suck
  - Quantify can't handle us
  - VTune, sysprof, oprofile, Shark
- Continuous performance tests
- Performance leakage
  - Performance tests are noisy
  - Observe performance degrading over time, tests too noisy to pin on any individual change
  - Comparing noisy profiles is hard
- Performance test suites are hard

Debugging

- gdb is rubbish
- Visual Studio: good implementation, bad concept
- Gather complete program execution traces for post-mortem analysis: *Chronicle...*

What We Don't Use, But Should

- Fault injection tools
  - Low memory, I/O, network
  - Lack of free tools here
- Coverage analysis

Bug Bottlenecks

- Finding bug: 5%
- Understanding bug: 40%
  - **Invest in debugging**
- Fixing bug: 10%
- Writing automated tests: 10%
- Dealing with regressions: 35%
  - Invest in ...?
  - “Code cleanup verifier”?

Open Source Processes

- Handling new contributors
  - Developers who ask “how can I help?” rarely contribute much of value
  - No follow up with questions, they just disappear
  - Strong contributors have usually made progress on a task before they make contact
- Contributors disappear at inconvenient times
  - Put them on the payroll
- Open vs in-house planning
- Strong leadership required

Open Source Issues

- Abusive bug reporters
- Counterproductive “contributors”
- Armchair architects

But...

- 30% of code non-MoCo
- Strong international flavour
- Facilitates remote work

Web Standards

- Browser incompatibilities → developer pain
- Single-vendor platforms are attractive
- Solution: standardize behaviour
  - *Essential for an open Web*
- Problems:
  - Existing specs vague
  - Dominant implementations buggy
  - Dominant implementations error-tolerant
  - → Non-compliant content and browsers

Error Tolerance

- Error tolerance is a competitive advantage
  - Tolerant clients favoured by authors and users
- Clients forced to reverse engineer dominant error recovery behaviour
- May as well **fully** specify error recovery
- XML bad, HTML good!

  Be liberal in what you accept
- Postel was right, but for the wrong reasons

Fixing Web Standards

- The Web as practiced is complex and ugly
- We can codify that practice and ignore the worst quirks
- Give authors and browser vendors something to aim at
- In way that is “compatible enough” with existing content and browsers

WHATWG and HTML5

- Fully open, “low budget” organization
- Created when the W3C wasn't interested
- Supported by Opera, Mozilla, Apple...
- Written specifications for de facto standards
  - e.g. innerHTML
  - 'window' object
  - HTML parsing
    - <b>**I *Like***</*b*> *Cheese*
  - And much much much more

The Spec Editor

- How to avoid design-by-committee *and* design-by-tyrant?
- HTML5 model
  - Single editor responsible for the spec
  - Must take all feedback into account via trackable process
  - Editor can be replaced by vote
- Very demanding, vastly underappreciated
  - Superstars needed

Specifying New Features

- Specifications need implementation experience
- **But,** shipping implementations constrain specifications
- Author-visible vendor prefixes
  - E.g. “-moz-grid”
- Publish experimental builds
- Pull features from release builds
- Communicate

Evolution Of The Web

- Eliminate “you can't do *that* on the Web”
  - 3D graphics (demo)
  - Heavy computation (JS JIT)
  - Local storage
  - Disconnected operation
  - ...

Offline Web Applications

- Run Web apps without a network connection
- Browser caches application resources
  - Manifest lists resources
  - No eviction
- HTML5 spec
- Most useful with client-side storage
  - HTML5 key-value storage
  - HTML5 SQL
- No change in security model!

Web Apps vs Desktop Apps

- Converging capability
- Fundamental distinction: *trust decision*
- Users should not be prompted to grant dangerous privileges
- Very few apps need broad privileges

New Challenges

- Multicore
  - vs Web's single-threaded programming model
- Overcoming mobile device limitations
- JS compilation
- Simplifying development
  - Tool support
- Evolving Web security model for cross-domain interaction

HTML Triumphant

- The universal, ubiquitous container
  - Yahoo Maps
- Where can we go from here?