Software Engineering and Open Source

(A few things I have learned doing Mozilla since 1998)

Brendan Eich <brendan@mozilla.org>
Happy to be here!
Agenda

• Problem statement

• Mozilla’s history and contributions

• The big picture

• Where we are all headed
Problem statement

- Software, especially “in the large”, remains quite challenging

- “What makes code strange” - Jeremy Ashkenas, CodeConf 2011

  - “I do not know of any other technology covering a ratio of 10^{10}.” - Dijkstra

  - “By evoking the need for deep conceptual hierarchies, the automatic computer confronts us with a radically new intellectual challenge that has no precedent in our history.” - Dijkstra again

  - “Science is knowledge which we understand so well that we can teach it to a computer; and if we don’t understand something, it is an art to deal with it.” - Knuth
The scale problem is worse over time

• Path dependence in any large codebase creates “lock-in” (see sunk cost fallacy)

• Rewriting is both a beginner’s error and something experienced programmers can defer for far too long...

• “Code by itself almost rots and it’s gotta be rewritten. Even when nothing has changed, for some reason it rots.” - Ken Thompson
Mozilla’s history: 1998

- An “escape pod” jettisoned by Netscape, the first big commercial open source conversion attempt - an act of desperation, a bit of executive legacy vanity

- 1998-2003 mozilla.org was a "virtual organization" with no legal independence

- staff@mozilla.org drawn from part- and full-time employees at several companies

- Mitchell Baker (MPL creator): governance and process

- Brendan Eich: technical architecture and oversight

- March 31, 1998: all contributors were @netscape.com
The beginner’s error: rewrite!

• October 1998: I declared it time to rebuild almost all of the code using:

• A new, cross-platform XML-based user-interface widget-set and front end, customizable similar to how web pages can be customized

• Gecko, a new, C++, standards-oriented HTML/CSS engine from Netscape, not nearly done but able to load pages and do basic layout

• A scriptable C++ component model, complete with IDL, IDL-derived type libraries, automatic JS <=> C++ type and method bridging

• Big gamble to bring in new developers attracted by green-field development and a new focus on correctly implemented Web standards
2000

- Netscape 6 - “Black and Blue”, circular buttons, slow, crashy, AOL IM built in...
- Based on “Mozilla 0.6” - our view was that the code was not nearly “1.0”
2001-2002

• After Netscape 6 flopped, Mitchell and I leveled the community playing field:

  • New Netscape hires had to prove merit before getting “commit access”

  • Mozilla binaries (available to testers who did not have a compiler or time to do their own builds) became competitive with Netscape releases

  • A roadmap plan of milestone releases leading up to “Mozilla 1.0”, running from 2001 into 2002, independent of any Netscape product release plans

• The community of volunteers and contributors employed by other companies than Netscape grew in strength and numbers from 2000 to 2002
2002-2003

• Mozilla 1.0 released in June 2002

• mozilla/browser, to be renamed Phoenix, then Firebird, was created

• Dave Hyatt and I wrote a new roadmap advocating unbundling the “Mozilla suite” into browser, mail, and other apps, each doing one thing well, with fewer features and an “add-ons” architecture so users could customize at will

• Mozilla downloads were at least as popular as Netscape product downloads

• July 2003: AOL laid off almost all Netscape employees and created the Mozilla Foundation, giving it initial funding, trademarks, and the project’s infrastructure

• Mozilla Foundation initial headcount: 10, with money for ~1.5 years
2004

• The new roadmap split the community, with a bare majority supporting separate browser and mail apps

• Firefox (renamed from Firebird) started rapid gains in market share and buzz

• Focus on end-user usability, Windows integration over Linux, unusual for open source software

• June 2004: Mozilla, Opera, and (not announced at first) Apple founded the WHATWG to create HTML5; I blogged about it and against the W3C

• The rest is perhaps better-known history...
<table>
<thead>
<tr>
<th>Period</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>Netscape</th>
<th>Mozilla</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010 Q4</td>
<td>58.27%</td>
<td>22.80%</td>
<td>9.23%</td>
<td>5.59%</td>
<td>2.24%</td>
<td>0.67%</td>
<td>0.03%</td>
</tr>
<tr>
<td>2010 Q3</td>
<td>60.29%</td>
<td>22.93%</td>
<td>7.53%</td>
<td>5.17%</td>
<td>2.40%</td>
<td>0.52%</td>
<td>0.02%</td>
</tr>
<tr>
<td>2010 Q2</td>
<td>60.01%</td>
<td>24.24%</td>
<td>7.00%</td>
<td>4.78%</td>
<td>2.33%</td>
<td>0.49%</td>
<td>0.12%</td>
</tr>
<tr>
<td>2010 Q1</td>
<td>61.43%</td>
<td>24.40%</td>
<td>5.67%</td>
<td>4.55%</td>
<td>2.37%</td>
<td>0.52%</td>
<td>0.15%</td>
</tr>
<tr>
<td>2009 Q4</td>
<td>63.67%</td>
<td>24.46%</td>
<td>4.04%</td>
<td>4.41%</td>
<td>2.29%</td>
<td>0.32%</td>
<td>0.11%</td>
</tr>
<tr>
<td>2009 Q3</td>
<td>66.80%</td>
<td>23.06%</td>
<td>2.86%</td>
<td>4.13%</td>
<td>2.07%</td>
<td>0.50%</td>
<td>0.09%</td>
</tr>
<tr>
<td>2009 Q2</td>
<td>68.06%</td>
<td>23.00%</td>
<td>2.13%</td>
<td>3.67%</td>
<td>2.05%</td>
<td>0.68%</td>
<td>0.06%</td>
</tr>
<tr>
<td>2009 Q1</td>
<td>69.13%</td>
<td>22.67%</td>
<td>1.56%</td>
<td>3.58%</td>
<td>2.18%</td>
<td>0.51%</td>
<td>0.06%</td>
</tr>
<tr>
<td>2008 Q4</td>
<td>71.99%</td>
<td>20.78%</td>
<td>1.16%</td>
<td>3.11%</td>
<td>2.15%</td>
<td>0.41%</td>
<td>0.07%</td>
</tr>
<tr>
<td>2008 Q3</td>
<td>74.93%</td>
<td>19.07%</td>
<td>0.37%</td>
<td>2.73%</td>
<td>2.06%</td>
<td>0.43%</td>
<td>0.07%</td>
</tr>
<tr>
<td>2008 Q2</td>
<td>76.24%</td>
<td>18.16%</td>
<td>2.73%</td>
<td>1.97%</td>
<td>0.48%</td>
<td>0.08%</td>
<td></td>
</tr>
<tr>
<td>2008 Q1</td>
<td>77.83%</td>
<td>16.86%</td>
<td>2.65%</td>
<td>1.84%</td>
<td>0.41%</td>
<td>0.13%</td>
<td></td>
</tr>
<tr>
<td>2007 Q4</td>
<td>79.16%</td>
<td>15.84%</td>
<td>2.39%</td>
<td>1.78%</td>
<td>0.50%</td>
<td>0.09%</td>
<td></td>
</tr>
<tr>
<td>2007 Q3</td>
<td>78.85%</td>
<td>14.69%</td>
<td>4.80%</td>
<td>0.51%</td>
<td>0.76%</td>
<td>0.11%</td>
<td></td>
</tr>
<tr>
<td>2007 Q2</td>
<td>78.76%</td>
<td>14.92%</td>
<td>4.66%</td>
<td>0.46%</td>
<td>0.83%</td>
<td>0.15%</td>
<td></td>
</tr>
<tr>
<td>2007 Q1</td>
<td>79.38%</td>
<td>14.35%</td>
<td>4.70%</td>
<td>0.50%</td>
<td>0.77%</td>
<td>0.19%</td>
<td></td>
</tr>
<tr>
<td>2006 Q4</td>
<td>80.69%</td>
<td>13.50%</td>
<td>4.06%</td>
<td>0.56%</td>
<td>0.86%</td>
<td>0.23%</td>
<td></td>
</tr>
<tr>
<td>2006 Q3</td>
<td>82.88%</td>
<td>11.89%</td>
<td>3.30%</td>
<td>0.62%</td>
<td>0.88%</td>
<td>0.26%</td>
<td></td>
</tr>
<tr>
<td>2006 Q2</td>
<td>84.03%</td>
<td>10.67%</td>
<td>3.25%</td>
<td>0.57%</td>
<td>0.98%</td>
<td>0.32%</td>
<td></td>
</tr>
<tr>
<td>2006 Q1</td>
<td>85.01%</td>
<td>9.77%</td>
<td>3.10%</td>
<td>0.53%</td>
<td>1.09%</td>
<td>0.36%</td>
<td></td>
</tr>
<tr>
<td>2005 Q4</td>
<td>85.88%</td>
<td>9.00%</td>
<td>2.80%</td>
<td>0.54%</td>
<td>1.25%</td>
<td>0.43%</td>
<td></td>
</tr>
<tr>
<td>2005 Q3</td>
<td>86.74%</td>
<td>7.97%</td>
<td>2.24%</td>
<td>0.55%</td>
<td>1.92%</td>
<td>0.48%</td>
<td></td>
</tr>
<tr>
<td>2005 Q2</td>
<td>87.24%</td>
<td>8.08%</td>
<td>1.89%</td>
<td>0.52%</td>
<td>1.62%</td>
<td>0.59%</td>
<td></td>
</tr>
<tr>
<td>2005 Q1</td>
<td>89.02%</td>
<td>6.17%</td>
<td>1.70%</td>
<td>0.49%</td>
<td>1.89%</td>
<td>0.67%</td>
<td></td>
</tr>
<tr>
<td>2004 Q4</td>
<td>91.35%</td>
<td>3.66%</td>
<td>1.50%</td>
<td>0.51%</td>
<td>2.09%</td>
<td>0.80%</td>
<td></td>
</tr>
</tbody>
</table>
Mozilla’s contributions: community

• In the modern Firefox era, 30-40% of patches come from volunteers

• ~700 volunteers contributing patches over a year (outsize impact)

• Firefox 4 available in 82 locales and 74 languages, the work of many volunteers

• ~50,000 nightly build testers

• 5M beta testers in the Firefox 4 release cycle

• 450M users
Mozilla embraced 20,391 worldwide patches from our Army of Awesome.

The average number of patches/year is 38 per contributor.
Mozilla embraced 6,660 worldwide patches from our Army of Awesome.

The average number of patches/year is 17 per contributor.
**Bugzilla**

**Bugzilla@Mozilla – Bug List: mybugs**

**Status:** UNCONFIRMED, NEW, ASSIGNED, REOPENED  
**Assignee:** brendan

Thu Apr 14 2011 22:13:22 PDT

29 bugs found.

<table>
<thead>
<tr>
<th>ID</th>
<th>Sev</th>
<th>Pri</th>
<th>Target</th>
<th>Whiteboard</th>
<th>OS</th>
<th>Assignee</th>
<th>Status</th>
<th>Resolution</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>335256</td>
<td>cri</td>
<td>--</td>
<td>---</td>
<td>[sg:dos]</td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>Denial of Service through the setTimeout() JavaScript function (recursive setTimeout/setInterval)</td>
</tr>
<tr>
<td>367425</td>
<td>nor</td>
<td>P3</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Javascript: Object construction time O(n^2) when getter closures used (Part II)</td>
</tr>
<tr>
<td>384115</td>
<td>nor</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>Mozilla 2 &quot;exploiting hardware parallelism&quot; meta- bug</td>
</tr>
<tr>
<td>445363</td>
<td>enh</td>
<td>P3</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Proper Tail Calls</td>
</tr>
<tr>
<td>476939</td>
<td>enh</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Implement Number.prototype.<em>iterator</em> by default</td>
</tr>
<tr>
<td>485791</td>
<td>enh</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>Self-host the E4X runtime in JS using proxies</td>
</tr>
<tr>
<td>486643</td>
<td>nor</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>deprecate sharp variables</td>
</tr>
<tr>
<td>493754</td>
<td>nor</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>Missed opportunity for flat closure, within funarg</td>
</tr>
<tr>
<td>508716</td>
<td>nor</td>
<td>--</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>NEW</td>
<td>---</td>
<td>Fluid dynamics simulator is slow</td>
</tr>
<tr>
<td>521670</td>
<td>nor</td>
<td>P3</td>
<td>mozilla2.0</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Make strict mode (new in ESS) disable undetected document.all emulation</td>
</tr>
<tr>
<td>545259</td>
<td>nor</td>
<td>P1</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>re-enable partial flat closure by reifying full scope chain on trace</td>
</tr>
<tr>
<td>565374</td>
<td>enh</td>
<td>P3</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Front end analysis for Proper Tail Calls</td>
</tr>
<tr>
<td>565534</td>
<td>enh</td>
<td>P3</td>
<td>---</td>
<td></td>
<td>All</td>
<td><a href="mailto:brendan@mozilla.org">brendan@mozilla.org</a></td>
<td>ASSI</td>
<td>---</td>
<td>Select JSOP_POP not JSOP_POPV for non-tail expression statements</td>
</tr>
</tbody>
</table>
## Tinderbox, now "tinderboxpushlog"

<table>
<thead>
<tr>
<th>Source</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:surkov.alexander@gmail.com">surkov.alexander@gmail.com</a></td>
<td>Thu Apr 14 19:12:21 2011 PDT</td>
<td>29ea31633ac6  Alexander Surkov  Bug 630486 - ASSERTION Want to</td>
</tr>
<tr>
<td><a href="mailto:cpierce@mozilla.com">cpierce@mozilla.com</a></td>
<td>Thu Apr 14 18:39:05 2011 PDT</td>
<td>24505fa65653  Chris Pearce  Bug 650157 - Push nsBuiltInDecoderRec</td>
</tr>
</tbody>
</table>

### Linux opt
- B  M(1 2 3 4 5 oth)  R(C J R Ripc)  X  T  T  T  T  T  T  T
- Linux debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
- Linux64 opt  B  M(1 2 3 4 5 oth)  R(C J R)  X  T  T  T  T  T  T  T  T
- Linux64 debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
- OS X opt  M(1 2 3 4 5 oth)  R(C J R)  X  T  T  T  T  T  T  T
- OS X debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
- OS X64 opt  B  M(1 2 3 4 5 oth)  R(C J R)  X  T  T  T  T  T  T  T  T
- OS X64 debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
- Win opt  B  M(1 2 3 4 5 oth)  R(C J R)  X  T  T  T  T  T  T  T  T
- Win debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
- WinXP opt  M(1 2 3 4 5 oth)  R(C J R)  X  T  T  T  T  T  T  T  T
- WinXP debug  B  M(1 2 3 4 5 oth)  R(C J R)  X
Code review and testing

• Code review before commit is required, in general

  • Large new modules may plead for post-commit review

  • No formal design review; important to catch design mistakes early

  • Code review of more, smaller patches landing early and often is best

• Tests are required as part of the patch

  • Previously we lacked this rule and suffered lack of test coverage and (prior to 2006) even test automation

• Valgrind [Nethercote, Seward] automated build test coverage
Fuzz-testing

- **jsfunfuzz**, Jesse Ruderman’s JavaScript travesty generator, has been enormously productive

  - More productive than static analysis frameworks (Coverity, e.g.)

  - Knows JS syntax and some semantics, bends and break rules, nests constructs in evil ways

  - Extremely effective when steered by an expert who tracks new code and language features, senses soft spots and attacks them with targeted fuzzer logic...

- Variations perform random DOM manipulation, markup generation, etc.
Automatic program analysis

• Collaboration with UCB project Oink, ultimately forked as Mozilla Pork, for:
  • experiments with automatic patch generation

• Second-generation analysis tools based on the GNU C Compiler (GCC):
  • dxr, a semantic source code cross-referencer
  • Dehydra, a scriptable static analysis tool (GCC plugin)
  • Treehydra, heavy-duty static analysis version of Dehydra (binds to JS reflection of GCC’s GIMPLE AST representation)
Brian Hackett’s **Sixgill**

Report: [Annotate] [Reanalyze] [Top] [Prev] [Next]
Goal: Assert [362] : this->mUnrootedGlobalCount
Dependent: TypeInvariant [XPCJSRuntime] :: this->mUnrootedGlobalCount
Steps: First Prev Next Last
Statement: annotation;

```
js/src/xpconnect/src/xpcjsruntime.cpp

346: void XPCJSRuntime::TraceXPConnectRoots(JSTracer *trc, JSBool rootGlobals)
347: {
348:     if(mUnrootedGlobalCount != 0)
349:         {
350:             JSContext *iter = nsnull, *acx;
351:             while((acx = JS_ContextIterator(GetJSRuntime(), &iter))
352:                 {
353:                 if(JS_HAS_OPTION(acx, JSOPTION_UNROOTED_GLOBAL))
354:                     {
355:                         NS_ASSERTION(nsXPConnect::GetXPConnect()->GetRequestDepth(acx)
356:                             == 0, "active cx must be always rooted");
357:                         NS_ASSERTION(acx->globalObject, "bad state");
358:                         JS_CALL_OBJECT_TRACER(trc, acx->globalObject,
359:                             "global object");
360:                         if(rootGlobals)
361:                             {
362:                             NS_ASSERTION(mUnrootedGlobalCount != 0, "bad state");
363:                             NS_ASSERTION(trc == acx->runtime->gcMarkingTracer,
364:                                 "bad tracer");
365:                             JS_ToggleOptions(acx, JSOPTION_UNROOTED_GLOBAL);
366:                             --mUnrootedGlobalCount;
367:                             }
368:             
```
Rust and Project Servo

• A new, safer systems programming language, Rust, instead of C++

  • Rust is a multi-paradigm, compiled language focused on safety, efficiency, programming-in-the-large, and a high degree of concurrency

• Research building parallel browser engine stages in Rust

• Experiment with Andreas Gal’s DOM implemented in JavaScript

• For a scalably-faster-on-manycore, much safer Browser from the Future

• Rust is good for Servers and other Software from the Future, too
// xfail-boot
// based on:

native "llvm" mod llvm {
    fn sqrt(float n) -> float = "sqrt.f64";
}

fn main() {

    let vec[int] inputs = vec(
        50000,
        500000
        //
        // Leave these commented out to
        // finish in a reasonable time
        // during 'make check' under valgrind
        // 5000000
        // 50000000
    );

    let vec[Body.props] bodies = NBodySystem.MakeNBodySystem();
Rust performance results (nbody)
// xfail-stage0
// Based on Isaac Gouy's fannkuchredux.csharp

use std;

import std._int;
import std._vec;

impure fn fannkuch(int n) -> int {

    fn permlimit(uint i) -> mutable int {
        ret i as int;
    }
    auto permlimit_ = permlimit; // Rustboot workaround

    auto perm = _vec.init_elt[mutable int](0, n as uint);
    auto perm1 = _vec.init_fn[mutable int](permlimit_, n as uint);
    auto count = _vec.init_elt[mutable int](0, n as uint);

    auto f = 0;
    auto i = 0;
    auto k = 0;
    auto r = 0;
Rust performance results (fannkuchredux)

- Within 2x (much more to do) of clang optimized performance, with safety
The big picture

• We at Mozilla like formal methods and invest in them, but *incrementally*.
  
  • Fielding a competitive browser still entails use of unsafe languages such as C and C++.
  
  • Being safe but so slow you lose users does not help.

• Manycore and security challenges motivate pragmatic and hybrid methods

• Art + Engineering require many hands at all layers of the system. Not all are ready for formal methods.

• Jeremy Ashkenas: “Ruby is not used for software engineering reasons”, rather for aesthetic and programmer productivity reasons
Where we are all headed

• When Mozilla began, **CVS** was the state of the art in open source version control

• Since then (see [http://revctrl.org/](http://revctrl.org/)), distributed peer-to-peer version control systems have emerged and gained widespread use

• 15 years ago, forking was a dire thing, to be avoided; “branches” were costly and unmaintainable in parallel for very long

• Now with DVCSes such as **Git** and **Mercurial**, forking is not always a bad thing, and often a good thing

• We are increasingly using [http://github.com/](http://github.com/), a Git-based “social hacking” site
697,802 people hosting over 1,968,966 git repositories

jQuery, reddit, Sparkle, curl, Ruby on Rails, node.js, ClickToFlash, Erlang/OTP, CakePHP, Redis, and many more

**git** /'git/

Git is an extremely fast, efficient, distributed version control system ideal for the collaborative development of software.

**git·hub** /'git.hab/

GitHub is the best way to collaborate with others. Fork, send pull requests and manage all your public and private git repositories.

---

Free public repositories, collaborator management, issue tracking, wikis, downloads, code review, graphs and much more...

---

**Team management**

30 seconds to give people access to code. No SSH key required. Activity feeds keep you updated on progress.

More about collaboration

**Code review**

Comment on changes, track issues, compare branches, send pull requests and merge forks.

More about code review

**Reliable code hosting**

We spend all day and night making sure your repositories are secure, backed up and always available.

More about code hosting

**Open source collaboration**

Participate in the most important open source community in the world today—online or at one of our meetups.

More about our community
Hi, BrendanEich

News Feed

- brson pushed to master at graydon/rust 15 minutes ago
  - 99a697b Add support for upper-case hex and binary output to #fmt.
  - 5c0f4c1 Add more commentary about ExtFmt
  - 4844e1c Add support for printing uints as lower-case hex to ExtFmt.
  - 5 more commits »

- brson pushed to master at graydon/rust about an hour ago
  - 330fdd2 Fix a typo in test expr-alt-generic-box1.rs

- graydon commented on pull request 328 on graydon/rust about 3 hours ago
  - Integrated

- graydon closed pull request 328 on graydon/rust about 3 hours ago
  - Tyepstate 2

- graydon pushed to master at graydon/rust about 3 hours ago
  - de50efe rm unused import
  - 7c6e6fc Make expr_while work in tyepstate_check
    View comparison for these 2 commits »

GitHub Jobs Update
GitHub's job board now has an iPhone app with push notifications, a new homepage, and more!

x hide this broadcast

View 30 new broadcasts

Your Repositories (4)

- graydon/rust
- zpao/v8monkey
- graydon/rust-mode
- BrendanEich/narcissus

Watched Repositories (2)
A work-in-progress programming language; not yet suitable for users

Make expr_while work in typestate_check

Also did some refactoring in typestate_check. All test cases in compile-fail that involve uninitialized vars now fail correctly! (All eight of them, that is.)

Tim Chevalier (author)
about 5 hours ago

graydon (committer)
about 4 hours ago
```rust
- print_expr(out, e);
+ print_expr(out, &e);
    log(s.get_str());
```
Blocks represent "impact". Impact is (lines added + lines deleted) for all non-merge commits during a week period.
Use a `\n` at a line-ending to suppress the newline. The slash is removed here once its job is done.

We treat all other single characters as a token. E.g.: `( )`, `. !
Multi-character operators are also literal tokens, so that Jison can assign the proper order of operations. There are some symbols that we tag specially here: `;` and newlines are both treated as a TERMINATOR, we distinguish parentheses that indicate a method call from regular parentheses, and so on.

```
suppressNewlines: ->
  @tokens.pop() if @value() is '\\'
  this

literalToken: ->
  if match = OPERATOR.exec @chunk
    [value] = match
    @tagParameters() if CODE.test value
  else
    value = @chunk.charAt 0
    tag = value
    prev = last @tokens
  if value is '=' and prev
    @assignmentError() if not prev[1].reserved and prev[1] in JS_FORBID
    if prev[1] in ['ll', '&&']
      prev[0] = 'COMPOUND_ASSIGN'
      prev[1] += '='
    return value.length
  if value is ';' then tag = 'TERMINATOR'
  else if value in MATH then tag = 'MATH'
  else if value in COMPAR then tag = 'COMPARE'
  else if value in COMPOUND_ASSIGN then tag = 'COMPOUND_ASSIGN'
  else if value in UNARY then tag = 'UNARY'
  else if value in SHIFT then tag = 'SHIFT'
  else if value in LOGIC or value is '?' and prev?.spaced then tag = 'LOGIC'
  else if prev and not prev.spaced
    if value is '(' and prev[0] in CALLABLE
      prev[0] = 'FUNC_EXIST' if prev[0] is '?'
      tag = 'CALL_START'
    else if value is '[' and prev[0] in INDEXABLE
      tag = 'INDEX_START'
      switch prev[0]
        when '?' then prev[0] = 'INDEX_SOCK'
        when ':' then prev[0] = 'INDEX_PROTO'
    @token tag, value
    value.length
```

**Token Manipulators**

Sanitize a heredoc or herecomment by erasing all external
A Daemon is the root object in a Pow process. It's responsible for starting and stopping an HttpServer and a DnsServer in tandem.

Create a new Daemon with the given Configuration instance. HttpServer and DnsServer instances are created accordingly.

The daemon stops in response to SIGINT, SIGTERM and SIGQUIT signals.

Start the daemon if it's stopped. The process goes like this:

- First, start the HTTP server. If the HTTP server can't boot, emit an error event and abort.
- Next, start the DNS server. If the DNS server can't boot, stop the HTTP server, emit an error event and abort.
- If both servers start up successfully, emit a start event and mark the daemon as started.

```javascript
{EventEmitter} = require "events"
HttpServer = require "../http_server"
DnsServer = require "../dns_server"

module.exports = class Daemon extends EventEmitter

  constructor: (@configuration) ->
    @httpServer = new HttpServer @configuration
    @dnsServer = new DnsServer @configuration

    process.on "SIGINT", @stop
    process.on "SIGTERM", @stop
    process.on "SIGQUIT", @stop

  start: ->
    return if @starting or @started
    @starting = true

    startServer = (server, port, callback) -> process.nextTick ->
      try
        server.listen port, ->
          callback null
          callback err

    pass = =>
      @starting = false
      @started = true
      @emit "start"

    flunk = (err) =>
      @starting = false
      try @httpServer.close()
      try @dnsServer.close()
      @emit "error", err

```
Conclusions

• Open source and distributed development obviously on the rise. Synergies between the two for social hacking, scaling up projects, recruiting new employees, and even partnering, all favor more open source and distributed development.

• Formal methods for verifying correctness, static and dynamic, are coming, but there are two large gaps to close:

  • Between the github.com world of JS, Ruby, etc., and Mozilla’s C++ and Rust systems programming “kernel languages”. Contracts, gradual typing?

  • Between kernel languages we use at large scale, and the sound but hard to scale verification systems used on, e.g., device drivers (SLAM)