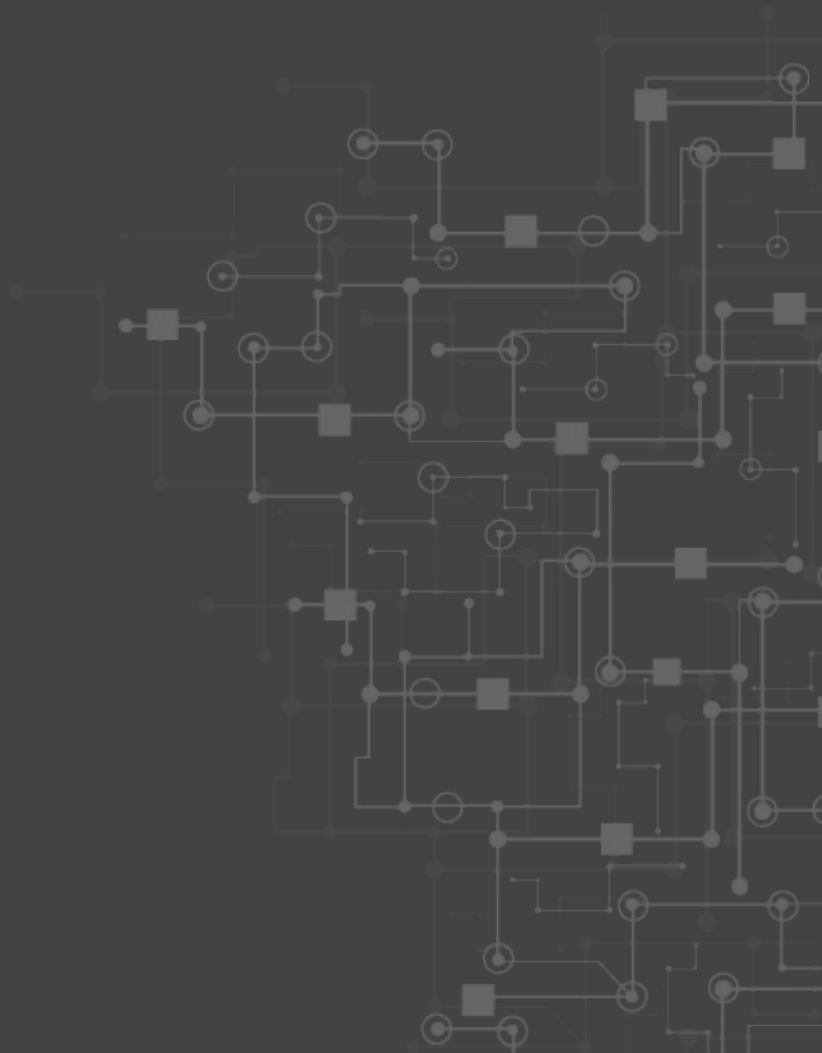
mozilla

Firefox quality

Mozilla Paris | FOSDEM | Feb 3rd 2018



Bonjour !

Je suis Sylvestre Ledru

Je parle de Firefox Quality

Twitter @SylvestreLedru

Bonjour !

Bonjour !

Bonjour !

The Firefox scale

mozilla

About:Firefox

- We release every 6 to 8 weeks

- 7 major releases published (one ESR) in 2017
  - Including Firefox Quantum 57
  - + 26 minor releases

About:Firefox:code

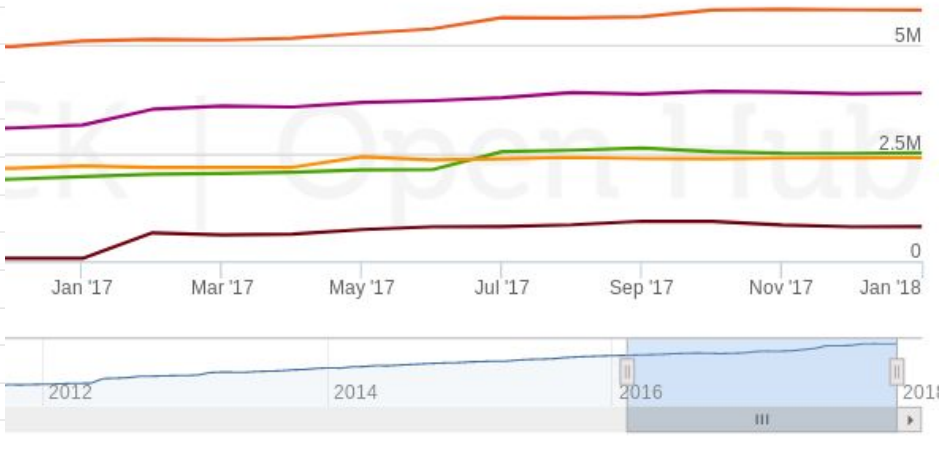
- ➔ One of the biggest and complex software
  A bit of legacy & technological debt
  (Netscape was opensourced 20 years ago)

- ➔ … has had 399 221 commits made by 5 356 contributors
  representing 17 920 130 lines of code

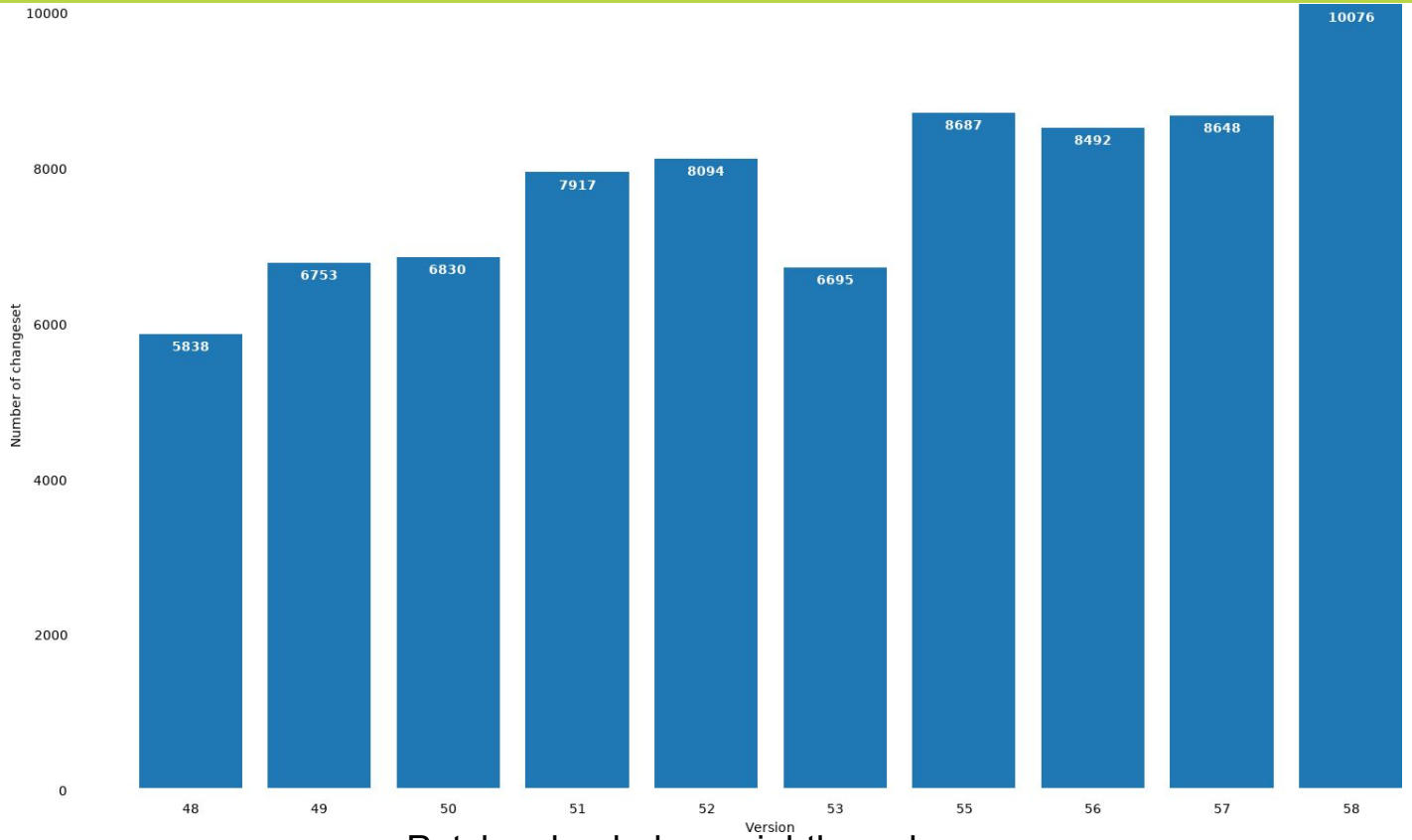
- ➔ 60 104 commits last year
- ➔ 1267 different contributors over last year

About:Firefox:code:languages

Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines
C++	5,819,797	1,196,317	17.1%	1,092,288	8,108,402
JavaScript	3,894,509	1,292,625	24.9%	870,402	6,057,536
HTML	2,513,253	124,985	4.7%	309,884	2,948,122
C	2,397,973	639,359	21.1%	400,372	3,437,704
Rust	801,636	154,096	16.1%	94,367	1,050,099
XML	720,947	16,434	2.2%	41,484	778,865
Python	602,906	174,837	22.5%	157,613	935,356
Java	327,613	122,062	27.1%	67,216	516,891
Assembly	226,062	24,811	9.9%	30,734	281,607
CSS	225,507	14,186	5.9%	32,804	272,497
Autoconf	104,687	1,843	1.7%	14,112	120,642
shell script	87,896	16,985	16.2%	13,209	118,090
Objective-C	57,098	8,665	13.2%	11,818	77,581
Make	49,840	14,709	22.8%	13,116	77,665
OpenGL Shading	32,369	34,703	51.7%	10,356	77,428
Perl	17,019	3,358	16.5%	3,804	24,181
NSIS	10,449	2,959	22.1%	2,195	15,603
CMake	7,301	2,005	21.5%	1,550	10,856
TeX/LaTeX	6,097	3,230	34.6%	752	10,079
DOS batch script	3,294	163	4.7%	524	3,981
Automake	3,212	222	6.5%	298	3,732



About:Firefox:Code



Patches landed per nightly cycle

About:Firefox:Continuous Integration

- We run a few tests... with a few different platforms and options
- 1 506 hours for the average full CI run
- Numbers from November 2017:
  - 8 319 189 tasks
  - 299.8 machine years
  - 927 333 unique machines

How to ship quality?

mozilla

Quality?

- Three types of QA:
  - Catch issues during development phase
  - Automated tests & testsuites when the code land
  - Pre release channel (nightly, beta, etc)

Pre release testing

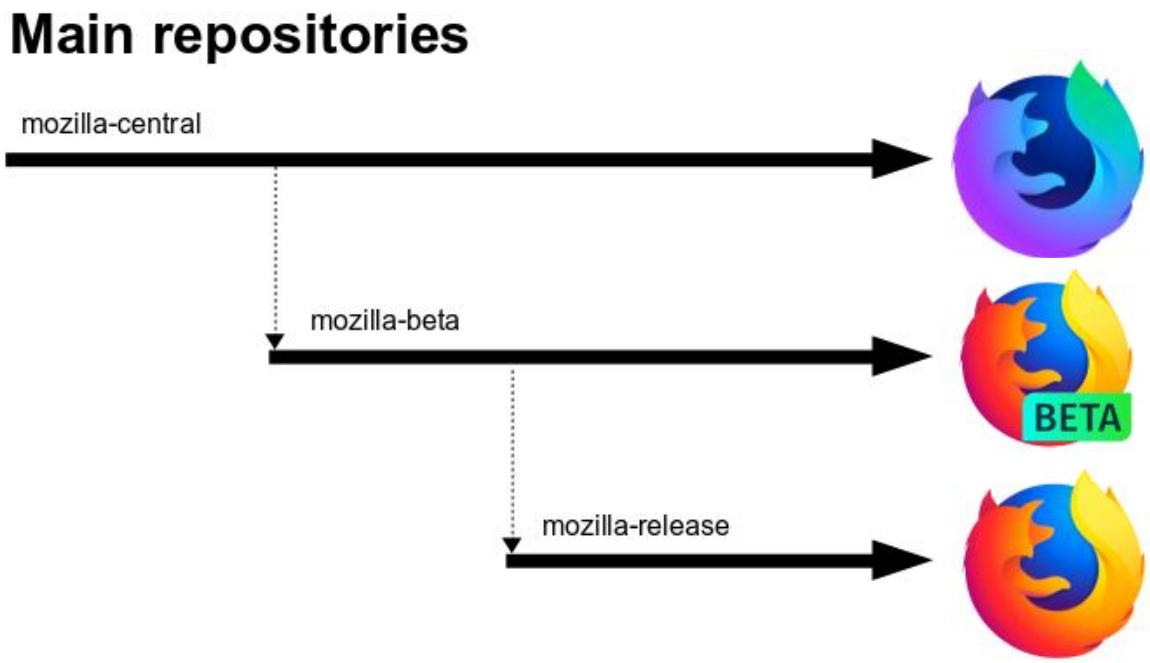
moz://a

Pre release testing

- The Web is a crazy platform
- All possible combinations of
  - HTML
  - CSS
  - Javascript (+ asm.js & WebAssembly)
  - Media format (Images, Audio, Video, etc)
  - Network
  - OS
  - ...

Pre release testing

➔ Release management - train model



Pre release testing

- We rely a lot on users on prerelease channel
  - Experiments (A/B testing) on pre-release channels
- Nightly - two nightlies per day
  - Hundred thousand of users
- Beta - 2 per week Desktop – 1 for Mobile
  - Millions of users

Manual testing

- Teams which test manually the new features

- Three colors
  - Green - Let’s ship it
  - Orange - We have to fix a few bugs
  - Red - Won’t be able to ship in this cycle

About:Firefox:Nightly

- Presented last year by Pascal Chevrel on this stage

- Reboot of the nightly community
  - Paved the way for 57
  - Doubled the nightly population
  - 1184 bugs reported by the nightly community
  - @FirefoxNightly jumped from 9500 to 16600 followers

About:Firefox:Sumo

- ➔ Gather feedback from users
- ➔ Identify some hard issues like “my Firefox only shows blank pages”
- ➔ Share it to release management and other teams

Pre release testing - web compat

- ➔ Platform to report Web compatibility issues
- ➔ Different of behavior between browsers leading to rendering issues or JS errors

julienw commented on 19 Jun 2017

URL: https://mobile.twitter.com/
Browser / Version: Firefox 56.0
Operating System: Android
Problem type: Something else - I'll add details below

Steps to Reproduce

1. Navigate to https://mobile.twitter.com/ with a valid account
2. Click the "retweet" icon on any tweet, and choose "Quote Tweet"
3. Start typing something

Expected Behavior:
What you type is what gets written.

Actual Behavior:
The first letter is *also* inserted at the end, you have to manually delete it.

Note: this doesn't happen on Mobile Twitter on Desktop. So could be something about how the virtual keyboard works.

From webcompat.com with ❤

webcompat-bot added **browser-firefox** **status-needstriage** labels on 19 Jun 2017

Code quality?

moz://a

Static analysis / linting

- C & C++ are hard languages like really really hard!
- How to detect programming mistakes
  - Related to the language designs
  - Usage of our APIs
  - Limit the code legacy

- Example:

```
/* !!! Should move this into its own .c and un-static it. */
static char *errStrings[] = {
    "Operation completed successfully.\n",
    "ERROR: NSS_Initialize() failed.\n"
    "ERROR: Unable to set initial password on the database.\n"
};
```

Static analysis / linting

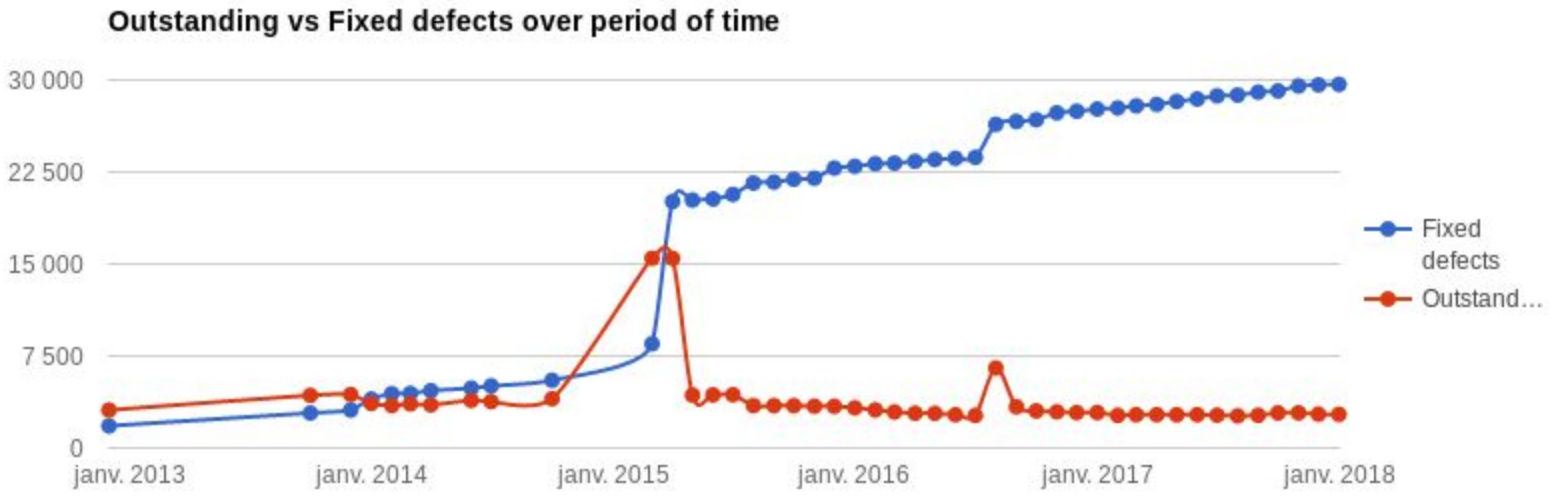
- Clang analyzer: 23 checkers
  - Dead code, insecure functions, etc

- Mozilla's: 26 checkers
  - Security issues, bad usages of API, best practices

- clang-tidy : 28 checkers
  - Best practices, coding style, performances, C++ 11, 14 or 17 upgrade

Static analysis / linting

➔ Once the code land, Coverity can catch others



SA tools that we use

- We use other tools for other languages
  - Javascript - Eslint
  - Python - flake8
  - Java (android) - findbug
  - Bash - shellcheck
  - Typos - codespell

toolkit/modules/NewTabUtils.jsm (Diff revision 3)

```
+20                                  + 36 lines
                                     37 XPCOMUtils.defineLazyModuleGetter(this, "pktApi",
+20                                  + 2113 lines
```

The issue has been resolved. Show all issues

```
Error: Please use ChromeUtils.defineModuleGetter instead of XPCOMUtils.defineLazyModuleGetter [eslint:
mozilla/use-chromeutils-import]
```

- For every commit – average of 12 minutes analysis
- We contribute upstream and sponsor some projects (ex: LLVM)...

Crash analysis

- When a crash occurs
  - Handled by breakpad
  - Sent to https://crash-stats.mozilla.com/
  - Doing some voodoo magic on them

Operating System

Operating System	Count	Percentage
Windows 7	247	98.0%
Windows 10	3	1.2%
Windows 8.1	2	0.8%

Product *

Product	Version	Count	Percentage	Installations
Firefox	58.0.1	252	27.7%	232
Firefox	57.0.4	203	22.3%	133

Crash analysis

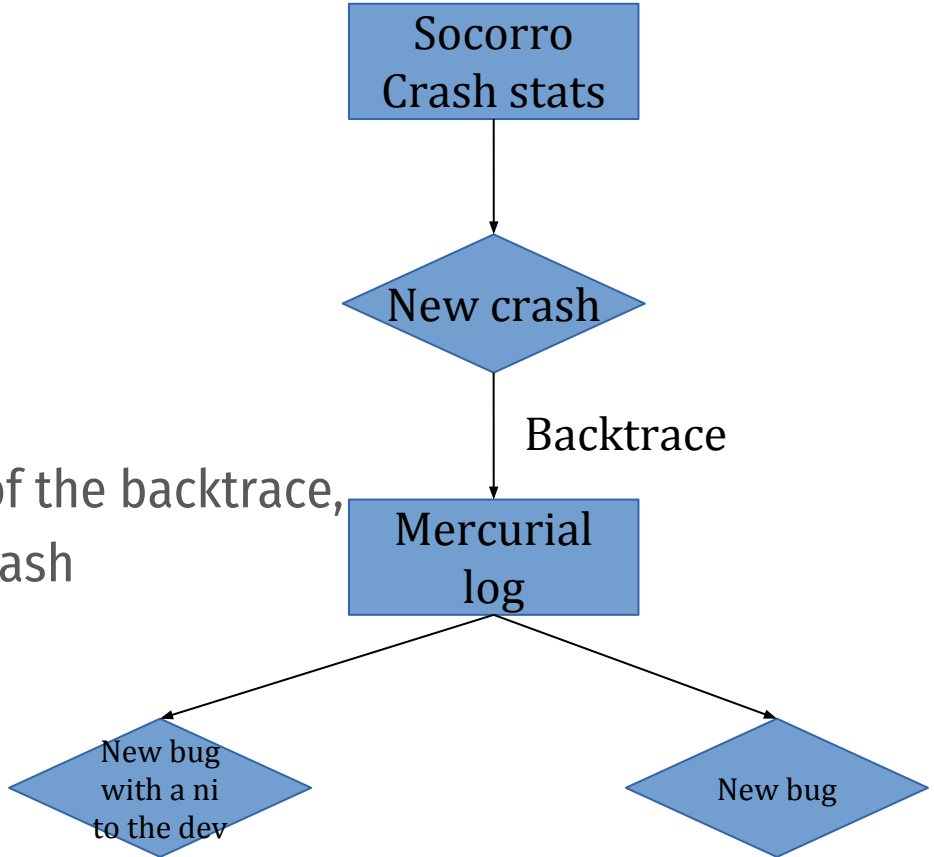
➔ Data mining on the results

Correlations for Firefox Release

```
(100.0% in signature vs 05.85% overall) reason = EXCEPTION_ACCESS_VIOLATION_WRITE
(95.03% in signature vs 01.32% overall) GFX_ERROR "[D3D11] failed to get compositor device." = true
(95.03% in signature vs 01.40% overall) GFX_ERROR "[D3D11] Failed to init compositor with reason: " = true
(72.79% in signature vs 00.21% overall) address = 0x198
(100.0% in signature vs 31.34% overall) Module "winsta.dll" = true
```

Crash analysis - clouseau

- Look at new crash signatures
- Extract the backtrace
- Look at the recent VCS history
- If a change touched one level of the backtrace, it might be the source of the crash
- 212 bugs reported



Code coverage

- Understanding of the quality of the testsuites

- (afaik) First time done on this scale
- We had to:
  - Add JS code coverage support in the Firefox JS engine
  - Add code coverage support in the Rust compiler
  - Patch: gcc, llvm, clang & compiler-rt
  - Develop an alternative to lcov called grcov dropping the processing time from more than 24 hours to less than 5 minutes

Code coverage - the results

- Current code coverage results (Windows & Linux)
  - C++
    - 2 913 824 lines
    - 1 620 227 covered lines (55,6%)
  - JS
    - 586 383 lines
    - 426 906 covered lines (72,8%)

```
216  1  handleEvent: function(event) {
217  1    switch (event.type) {
218         case "DOMTitleChanged":
219  1        this._titleChangedHandler(event);
220           break;
221         case "DOMLinkAdded":
222           this._linkAddedHandler(event);
223           break;
224         case "MozScrolledAreaChanged":
225  1        this._mozScrollAreaChanged(event);
226           break;
227         case "MozDOMFullscreen:Request":
228           this._mozRequestedDOMFullscreen(event);
229           break;
230         case "MozDOMFullscreen:NewOrigin":
231           this._mozFullscreenOriginChange(event);
```

Code coverage - Side effect

- What does it mean when a file has 0 coverage?

  A bug, for sure!
  - Dead code
  - No test

- 61 removed files

  13272 removed lines.

☑ Show third-party files ☐ Show headers ☑ C/C++ ☑ JavaScript

8721 files.

gfx - 1449 files and 44162 functions
media - 1371 files and 25100 functions
dom - 1113 files and 16703 functions
devtools - 633 files and 11872 functions
toolkit - 483 files and 9503 functions
modules - 477 files and 7381 functions
js - 409 files and 16460 functions
security - 394 files and 5119 functions
intl - 365 files and 13533 functions
layout - 301 files and 4362 functions
netwerk - 243 files and 5327 functions
browser - 228 files and 4228 functions
third_party - 188 files and 4701 functions

Fuzzing

- Send invalid, unexpected, or random data as inputs
- We are testing:
  - JavaScript features, DOM, Layout, CSS, Stylo,
  - Media file formats (images, audio, video)
- Last 2 y, over 600 security bugs

```
RegExp.prototype[Symbol.split].call({})


Backtrace:

Program received signal SIGSEGV, Segmentation fault.
0x0000000000a531a0 in JSVAL_TO_STRING_IMPL (l=<error reading variable: Canno
/dist/include/js/Value.h:778
#0  0x0000000000a531a0 in JSVAL_TO_STRING_IMPL (l=<error reading variable: C
/dist/include/js/Value.h:778
#1  toString (this=0x8) at js/src/opt64/dist/include/js/Value.h:1272
#2  getSource (this=0x7ffff7e6e140) at js/src/vm/RegExpObject.h:451
#3  js::regexp_construct_no_sticky (cx=0x7ffff6907800, argc=<optimized out>,
#4  0x0000000000879fd1 in CallJSNative (args=..., native=0xa53130 <js::regex
JS::Value*)>, cx=0x7ffff6907800) at js/src/jscntxtinlines.h:235
[...]
#28 main (argc=<optimized out>, argv=<optimized out>, envp=<optimized out>)
rax     0x8     8
rbx     0x7ffff6907800  140737330051072
rcx     0x0     0
```

Other best practices

- Once or twice a day, compiler Firefox trunk with -Werror on:
  - Build with gcc snapshot packages from Debian experimental (currently version 8)
  - Clang trunk (currently version 7)

- Find new issues in our code

- Find bugs in the compiler

Depends on: (new) class-memaccess, 1411037, 1411049, 1426997, ~~build-gcc-7~~, ~~1409284~~, ~~1409285~~, ~~1409326~~, ~~1409382~~, ~~1410379~~, ~~1411027~~, ~~1411034~~, ~~1411056~~, ~~1424866~~, ~~1424867~~, ~~1430729~~, ~~1431109~~

Dependency tree / graph

See Also: https://gcc.gnu.org/bugzilla/show_bug...
https://gcc.gnu.org/bugzilla/show_bug...
https://gcc.gnu.org/bugzilla/show_bug...
https://gcc.gnu.org/bugzilla/show_bug...
https://gcc.gnu.org/bugzilla/show_bug...
https://gcc.gnu.org/bugzilla/show_bug...

Automation

mozilla

Crazy CI

Linux opt	B Cpp GTest Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+28) R-e10s(+20) W-e10s(+19) X(+8) [tier 2](BR)
Linux pgo	B
Linux debug	B Cpp GTest Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+25) M-e10s(+40) R(+9) R-e10s(+22) SM(arm) W(+18) W-e10s(+19) X(+12) [tier 2](BR)
Linux Stylo Disabled opt	Cpp M(+4) M-e10s(+20) R-e10s(+9) SYsd-e10s(sy) W-e10s(wpt6* +17)
Linux Stylo Disabled debug	Cpp M(+4) M-e10s(+40) R-e10s(+9) W-e10s(+18)
Linux x64 opt	B Bb Cpp GTest Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn MnH S V Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+34) R-e10s(+28) SM(+6) SY-e10s(sy) T-e10s(+16) Tss-e10s(tp6) W-e10s(+19) X(+8) [tier 2](+3)
Linux x64 pgo	B Cpp GTest Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+28) R-e10s(+20) T-e10s(+16) Tss-e10s(tp6) W-e10s(wpt6* wpt12* +17) X(+8) [tier 2](TV)
Linux x64 asan	Bd Bo BoR Bof Cpp GTest Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(6* +41) R-e10s(+20) X(+8) [tier 2](TV)
Linux x64 debug	B Bb Bf Cpp GTest H Jit1 Jit2 Jit3 Jit4 Jit5 Jit6 Mn MnH S Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+64) R-e10s(+28) SM(+7) SM[tier 2](rust) W-e10s(+19) X(+10) [tier 2](TV* BR)
Linux x64 QuantumRender opt	M-e10s[tier 2](+5) M-e10s(+7) R-e10s(+13)
Linux x64 QuantumRender debug	M-e10s(+7) M-e10s[tier 2](+16) R-e10s(+13)
Linux x64 Stylo Disabled opt	Cpp M(+4) M-e10s(+28) R-e10s(+9) SYsd-e10s(sy) W-e10s(+18)
Linux x64 Stylo Disabled debug	Cpp M(+4) M-e10s(bc6* bc14* +46) R-e10s(+9) W-e10s(+18)
Linux x64 Stylo-Seq opt	SYss-e10s[tier 2](sy)
Linux x64 NoOpt debug	[tier 2](B)
OS X 10.10 opt	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+27) R-e10s(R* +3) SY-e10s(sy) T-e10s(h2 +14) W-e10s(+6) X(X) [tier 2](TV)
OS X 10.10 debug	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(bc2* +25) R-e10s(+5) W-e10s(+11) X(X) [tier 2](TV)
OS X Cross Compiled opt	B
OS X Cross Compiled debug	B
OS X Cross Compiled NoOpt debug	[tier 2](B)
Windows 7 opt	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+34) R-e10s(Rg21 +99) SY-e10s(sy) T-e10s(+14) Tss-e10s(tp6) W-e10s(+13) X(X) [tier 2](TV)
Windows 7 pgo	Cpp Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(bc2* +32) R-e10s(+99) SY-e10s(sy) T-e10s(g g2 +12) Tss-e10s(tp6) W-e10s(+13) X(X) [tier 2](TV)
Windows 7 debug	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(dt6* +32) R-e10s(+27) W-e10s(+13) X(X) [tier 2](TV)
Windows 10 x64 opt	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(+37 +4) R-e10s(+4) SY-e10s(sy) T-e10s(+13) Tss-e10s(tp6) W-e10s(+13) X(X) [tier 2](TV)
Windows 10 x64 pgo	Cpp Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(i* g gl1 +29) R-e10s(+4) SY-e10s(sy) T-e10s(d g2 +12) Tss-e10s(tp6) W-e10s(+13) X(X) [tier 2](TV)
Windows 10 x64 debug	Cpp GTest Jit Mn MnH Fxfn-l-e10s(en-US) Fxfn-r-e10s[tier 2](en-US) M(+4) M-e10s(h1* gl1 +35) R-e10s(+5) W-e10s(+13) X(X) [tier 2](TV)
Windows 10 x64 QuantumRender opt	M-e10s[tier 2](+2) R-e10s[tier 2](C C)
Windows 10 x64 QuantumRender debug	M-e10s[tier 2](+2) R-e10s[tier 2](C)
Windows 2012 opt	B Bs S SM(p) [tier 2](BR)
Windows 2012 pgo	B Bs
Windows 2012 debug	B Bs S SM(cgc p)
Windows 2012 NoOpt debug	[tier 2](B)
Windows 2012 x64 opt	B Bs S [tier 2](BR)
Windows 2012 x64 pgo	B Bs
Windows 2012 x64 debug	B Bs S
Windows 2012 x64 NoOpt debug	[tier 2](B)
Android 4.0 API16+ opt	B
Android 4.0 API16+ debug	B
Android 4.2 x86 opt	B gv M(+2) X(+6)
Android 4.2 API16+ opt	A[tier 2](t)
Android 4.3 API16+ opt	Cpp gv M(23 +35) R(+68) X(+8) [tier 2](TV)
Android 4.3 API16+ debug	Cpp gv M(17 +57) R(+158 +3) X(+8) [tier 2](+11)
Android 4.4 API16+ opt	A[tier 2](t)
Android 5.0 AArch64 opt	B
Android 6.0 API16+ opt	A[tier 2](+2)
Android API16+ Gradle opt	Bng M[tier 2](+35) R[tier 2](+68) X[tier 2](+8) [tier 2](+2)
Gecko Decision Task opt	D
Linting opt	Bugzilla Doc ES f8 py-compat spell
windows-mingw32-32 debug	[tier 2](B)

CI

- Launched (almost) on every commit

- Can be used by individual developers
  - Platforms
  - With testsuite selection

WPT - Web Platform Tests

- A W3C-coordinated attempt to build a cross-browser testsuite for the Web-platform stack

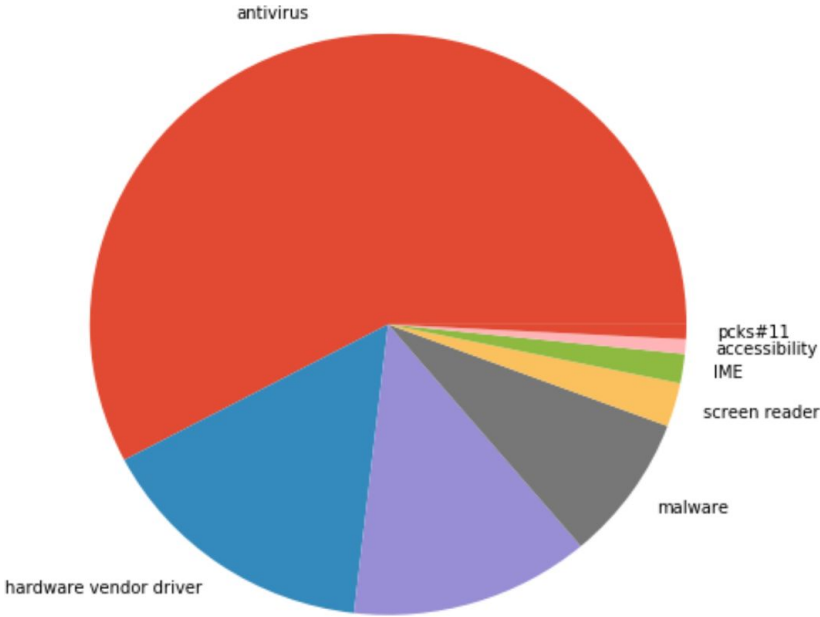
- Takes ~184 minutes on a Linux PGO build

Despite all that

mozilla

Predicting release quality is tough.....

- We still have issues after we published to users
- Huge trunk caused by Malware, Antivirus or security software
- Some web compat issues
  (example: Outlook web in 58)



mozilla

Thanks to all the persons who helped me gathering these stats (coop, marco, pascal, calixte, etc)
Shameless advertising: we are looking for interns to work on static analysis & code coverage