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

<table>
<thead>
<tr>
<th>Language</th>
<th>Code Lines</th>
<th>Comment Lines</th>
<th>Comment Ratio</th>
<th>Blank Lines</th>
<th>Total Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>5,819,797</td>
<td>1,196,317</td>
<td>17.1%</td>
<td>1,092,288</td>
<td>8,108,402</td>
</tr>
<tr>
<td>JavaScript</td>
<td>3,884,509</td>
<td>1,202,625</td>
<td>24.9%</td>
<td>870,402</td>
<td>6,057,536</td>
</tr>
<tr>
<td>HTML</td>
<td>2,513,253</td>
<td>124,985</td>
<td>4.7%</td>
<td>309,884</td>
<td>2,948,122</td>
</tr>
<tr>
<td>C</td>
<td>2,397,973</td>
<td>639,339</td>
<td>21.1%</td>
<td>400,372</td>
<td>3,437,704</td>
</tr>
<tr>
<td>Rust</td>
<td>801,636</td>
<td>154,096</td>
<td>16.1%</td>
<td>94,367</td>
<td>1,050,099</td>
</tr>
<tr>
<td>XML</td>
<td>720,847</td>
<td>16,434</td>
<td>2.2%</td>
<td>41,484</td>
<td>778,835</td>
</tr>
<tr>
<td>Python</td>
<td>602,906</td>
<td>174,837</td>
<td>22.5%</td>
<td>157,613</td>
<td>935,356</td>
</tr>
<tr>
<td>Java</td>
<td>327,613</td>
<td>122,062</td>
<td>27.1%</td>
<td>67,216</td>
<td>516,891</td>
</tr>
<tr>
<td>Assembly</td>
<td>226,062</td>
<td>24,811</td>
<td>9.9%</td>
<td>30,734</td>
<td>281,607</td>
</tr>
<tr>
<td>CSS</td>
<td>225,507</td>
<td>14,186</td>
<td>5.9%</td>
<td>32,804</td>
<td>272,497</td>
</tr>
<tr>
<td>Autoconf</td>
<td>104,687</td>
<td>1,843</td>
<td>1.7%</td>
<td>14,112</td>
<td>120,642</td>
</tr>
<tr>
<td>Shell script</td>
<td>87,890</td>
<td>16,985</td>
<td>16.2%</td>
<td>13,209</td>
<td>118,089</td>
</tr>
<tr>
<td>Objective-C</td>
<td>57,098</td>
<td>8,665</td>
<td>14.2%</td>
<td>11,818</td>
<td>77,581</td>
</tr>
<tr>
<td>Make</td>
<td>49,540</td>
<td>14,709</td>
<td>22.8%</td>
<td>13,116</td>
<td>77,665</td>
</tr>
<tr>
<td>OpenGL-SHADING</td>
<td>32,369</td>
<td>34,703</td>
<td>51.7%</td>
<td>10,956</td>
<td>77,428</td>
</tr>
<tr>
<td>Perl</td>
<td>17,019</td>
<td>3,358</td>
<td>16.5%</td>
<td>3,804</td>
<td>24,181</td>
</tr>
<tr>
<td>NSIS</td>
<td>10,449</td>
<td>2,959</td>
<td>22.1%</td>
<td>2,155</td>
<td>15,603</td>
</tr>
<tr>
<td>CMmake</td>
<td>7,301</td>
<td>2,005</td>
<td>21.5%</td>
<td>1,550</td>
<td>10,856</td>
</tr>
<tr>
<td>TeX/LaTeX</td>
<td>6,007</td>
<td>3,220</td>
<td>34.6%</td>
<td>752</td>
<td>10,079</td>
</tr>
<tr>
<td>DOS batch script</td>
<td>3,294</td>
<td>163</td>
<td>4.7%</td>
<td>524</td>
<td>3,961</td>
</tr>
<tr>
<td>Automake</td>
<td>3,212</td>
<td>222</td>
<td>6.5%</td>
<td>298</td>
<td>3,732</td>
</tr>
</tbody>
</table>
About: Firefox: Code

Patches landed per nightly cycle
We run a few tests... with a few different platforms and options.

1506 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?
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

Mozilla
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

Main repositories

mozilla-central

mozilla-beta

mozilla-release

BETA
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 behavior between browsers leading to rendering issues or JS errors
Code quality?

Mozilla
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:

```c
/* !!! 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

➔ 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
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%)
What does it mean when a file has 0 coverage?
A bug, for sure!

- Dead code
- No test

61 removed files
13272 removed lines.
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
**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

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
CI

➔ Launched (almost) on every commit

➔ Can be used by individual developers
   ◆ Platforms
   ◆ With testsuite selection
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
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)
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