



# Blink Code Complexity Survey

*The good, the bad and the ugly of our code and architecture*

jbroman@chromium.org  
BlinkOn 9 – Sunnyvale

blink-dev >

## 😊/😞 Blink Code Complexity Survey (complain about what makes working in Blink hard)

1 post by 1 author  



Jeremy Roman

Mar 20



**Other recipients:** platform-arc...@chromium.org

Hey there, Blink-folk!

The results of an internal survey suggest that many of us feel hindered by technical debt and overly complicated code.

**We want to know what slows you down while developing web platform code in Chromium.**

Befuddled by garbage collection? Tell us!

Sick and tired of writing yet another wrapper class? Let us know!

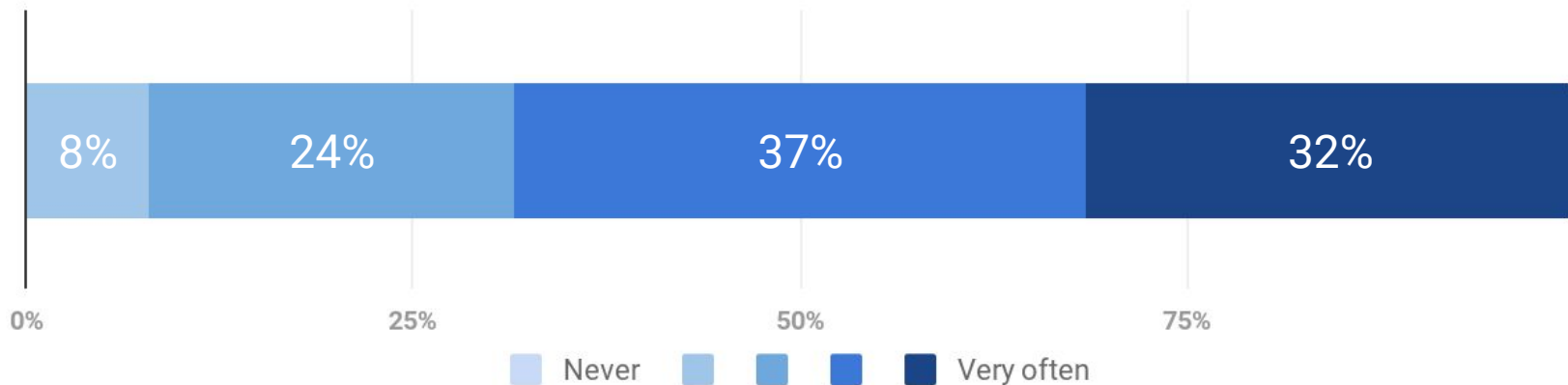
Please fill out this survey:

<https://goo.gl/forms/jRJLSzBcVcPTIIUz2>

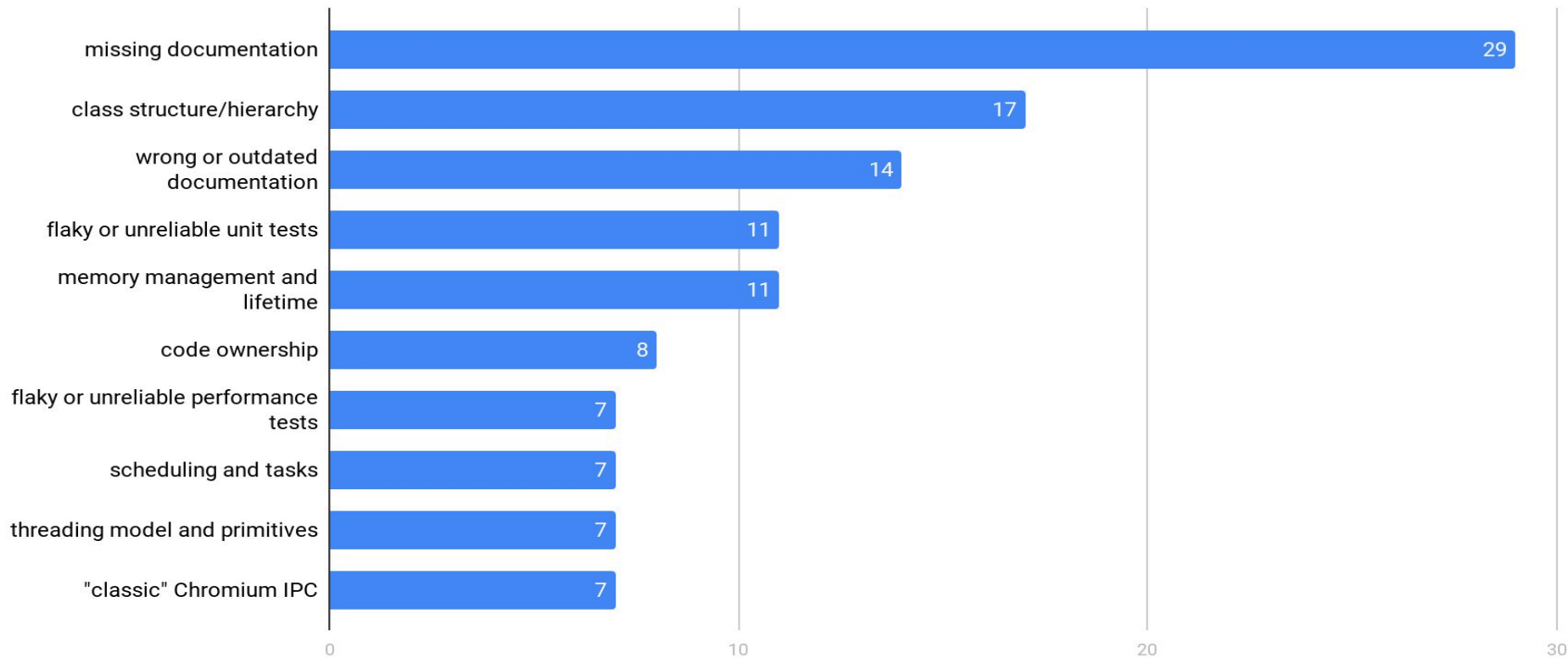
We'll share the results after the survey closes.

Click here to [Reply](#).

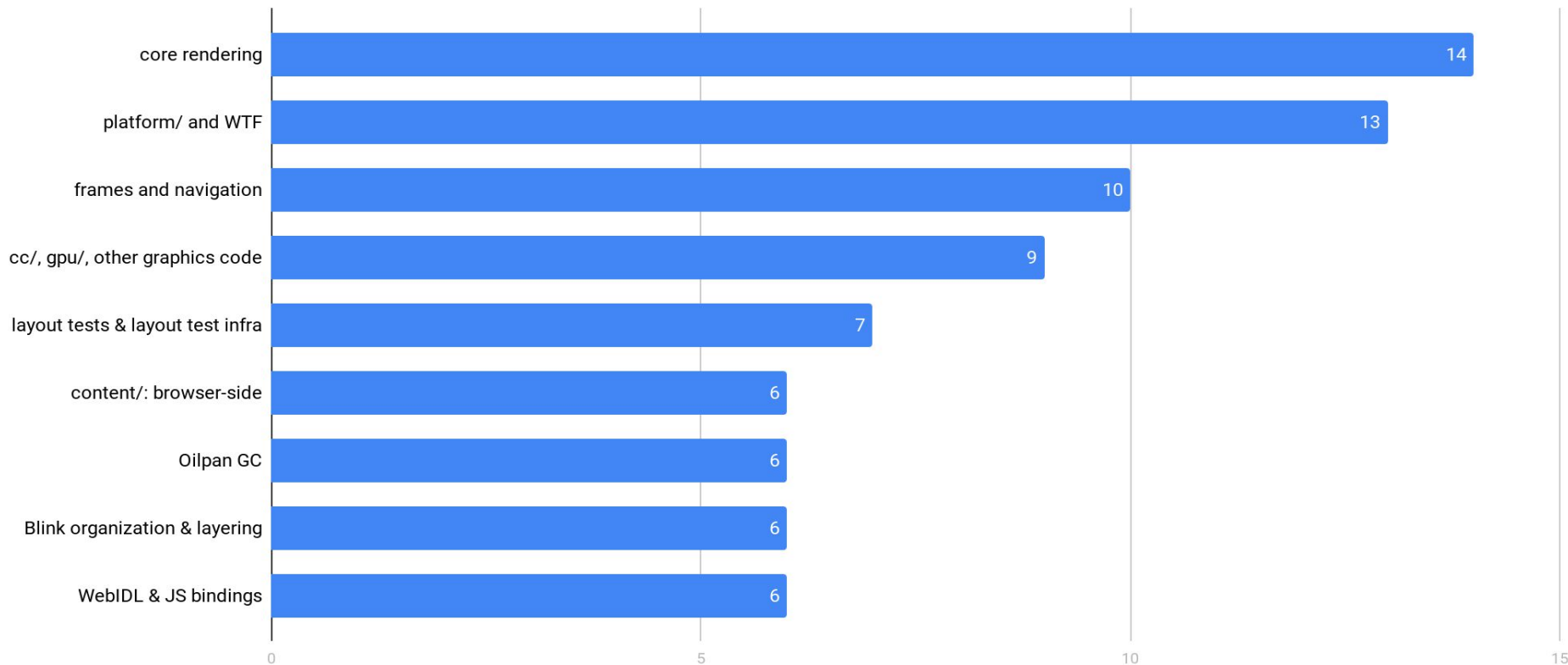
# “How often are you hindered by **technical debt** or **overly complicated code**?”



# “What **types** of technical debt or complexity issues frequently hinder your work in or around Blink?”



# “Which parts of Chromium have debt or complexity that slow you down?”



---

# Documentation




A lot of our systems, like layout and style are **incredibly complicated with very little documentation.**


...it's a nightmare. Compile errors, unhelpful test error messages, etc - I'm often left guessing, trial-and-erroring and copying existing code to figure out how to do things. I **try to find documentation but still often have to ask others for help.**



...prevalence in Blink of conditional statements that check for something, like some property on an object, and then do something different for that object, with **no comment to say why things should be that way.**




I still find many header files that lack a basic **"this does what" comment**...WebIDL and bindings are unevenly documented. I find myself either **blindly copying existing code** or reading the actual generated code in code search to work out what I need to do.



Incorrect assumptions and breaking invariants in other parts of the code. **Undocumented invariants and unit tests for undocumented classes/methods** with bad names where what the unit tests test is incomprehensible.

What does slow me down in existing code often is **lack of documentation**...we need a documentation sheriff whose job is to find and document answers.







I feel like there's a kind of **machismo in Blink underlying the lack of comments**, the undisciplined use of inheritance, the NIH code style, and the thread-hostile data structures. The overall message seems to be "if you can't deal with it, you're not good enough to code in Blink". I think the first step to changing this culture is to acknowledge it.

Upto date documentation and more specific examples of how to use "threading, task scheduling or IPC" will help



Comprehensive documentation that included **more context and links to other helpful resources** for those new to blink/chromium/c++(!) to actually understand things would be amazing.

---

# Code Structure




Going through the **Blink/content barrier** is always painful because of the use of different types on each side and the strange "no man's land" in the middle where neither set of types is appropriate.


Currently code structure designs seems to create strong black boxes that prohibits or scares people away from code designs that take advantage of existing code...Call it misc, util, helpers, don't care what, but right now **people are forced into copy/paste programming.**




Adequately invest into keeping architecture and layering sane, **focus on big picture** at least as much as on renaming classes and formatting.




Interfacing between Blink and `//chrome` or `//components` is a **real battle with abstraction layers**.



Logical entities are hard to locate due to complex code organization. Much of this complexity is added to support unit testing.



There's **too much inheritance of implementation and multiple inheritance**. It can be tricky to figure out which implementation of a method actually matters. Multiple inheritance interacts badly with Oilpan.




Unclear and undocumented contracts around many classes. Habit of just calling from core/ to core/ when you need it **makes reasoning about things very hard**.

---


# Development Process




Working sometimes **take ages** to get people to review contributions.



Have more **experienced engineers** that understand layering involved. Platform architecture team is doing great job, but having more eyes there would help. Having greater exposure to L6-L8 ICs/TLs could have helped preventing the web/ vs core/ vs core/exported mege mistakes.



Refactoring of the mojo/service layer should include a design document and/or public discussion. Commits related to the refactoring should link a bug report with the document/discussion.



**Code ownership** in Blink itself is not usually a problem, but the rules in //chrome are convoluted and automated tools frequently give the wrong answers.

---

# Testing



One of the biggest slowdowns is figuring out how to write test code that depends on the communication and scheduling of the **many threads involved and data transformations between them.**

Top issue: there is not a good way to unit test much of our codebase. For example, there's **no unit test framework** that lets you test the cc / blink boundary.



Perhaps a version of RenderingTest can be created which **sets up more of the plumbing** and access to various components.



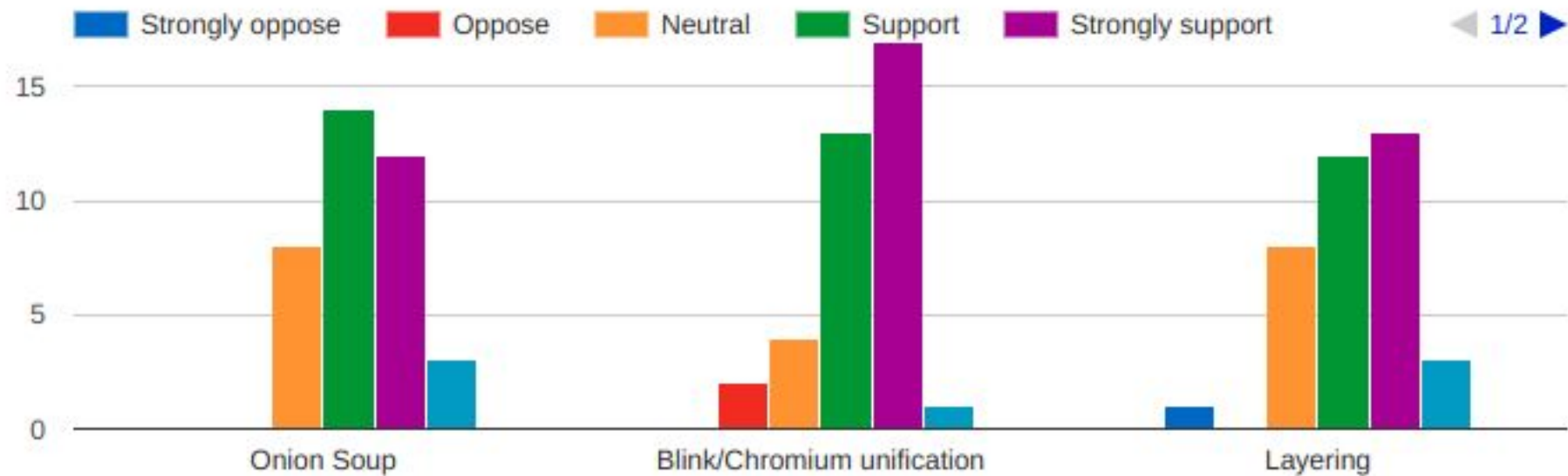
Unit tests are very time consuming to add where a **feature lacks existing tests.**





---



# Architecture Team Priorities





Finally get rid of **third\_party/WebKit**.

If the code is restructured or moved, make sure to **shorten the path** from /third\_party/WebKit/Source/ to something short like /engine or /blink



Kudos to the platform architecture team for their **open process** in approaching the Blink refactoring efforts. Teams refactoring Chromium subsystems should take your example and follow a similarly open approach.

I feel like platform architecture team **did not have a strong direction** supported by senior engineers which have understanding of existing picture and history, and was just pursuing random goals for a while...churn without clear benefits, bypassing owners of the changing code

