

# Chromium Project Ideas

## Google Summer of Code 2022



Google  
Summer of Code

### Best Practices

*Thank you for your interest in Chromium Google Summer of Code!*

Please reach out to [chromium-gsoc@chromium.org](mailto:chromium-gsoc@chromium.org) or in our [discord server](#) with questions about project ideas and your proposal. Chromium Admins and engineering mentors will provide feedback.

For more information on getting started with chromium, please read [Chromium GSoC FAQs](#). Example [template](#) for contributor project proposal.

Since Chromium is a huge project and Summer of Code is in just a few months, you will be a lot more successful tackling these projects if you contributed patches to Chromium before. To support your application, we recommend that you get started by fixing the starter bugs and submitting patches (CLs) to add features or resolve issues.

If you would like to get early feedback on your proposal, send the proposal to mentors and cc [chromium-gsoc-admins@chromium.org](mailto:chromium-gsoc-admins@chromium.org), we will get back to you in coming weeks.

We are looking forward to working with you for the summer!

## 1. Fuzzing Mojo Interfaces

### Project Description:


Chromium security rests in part on the strength of its sandbox, which creates a security boundary between users' private data and untrusted code from the Internet. This project invites contributors to help us protect web users by strengthening the sandbox through fuzzing Chromium inter-process communication channels, known within Chromium as Mojo interfaces. Fortunately a framework called MojoLPM already exists to create Mojo fuzzers. We need your help to bring MojoLPM to interfaces that don't yet have fuzzer coverage.

### Stretch goal:

Propose and implement new ways to improve Mojo IPC fuzzing.

**Project Size:** Medium

**Mentors:** [bookholt@chromium.org](mailto:bookholt@chromium.org), [ajgo@chromium.org](mailto:ajgo@chromium.org)

**Project doc:**  Chromium Project Proposal Doc: Fuzzing Mojo IPC

## 2. Automate Drag-and-Drop tests

### Project Description:

Blink has multiple drag-and-drop API tests that rely upon a Chromium-specific automation library, and the web-platform-tests (WPT) project has lots of drag-and-drop tests without automation at all. Within the last few years, work has been done to add a cross-browser automation API to the web-platform-tests project and it is now possible to automate a subset of the drag-and-drop tests. We'd like to convert as many of the Blink-specific tests as possible into web-platform-tests and also de-dupe/expand coverage by automating the existing web-platform-tests.

**Project Size:** Either Medium or Large

**Mentors:** [awillia@chromium.org](mailto:awillia@chromium.org), [miketaylr@chromium.org](mailto:miketaylr@chromium.org)

**Project doc:**  Drag and Drop WPT Automation GSoC Proposal

## 3. Align Performance APIs with Web Platform Tests

### Project Description:


The web performance specifications define how browsers are supposed to be behaving when it comes to the various web performance APIs, and Web Platform Tests ensure that this behavior is actually implemented. At the same time, Chromium is not always fully spec compliant. [Resource Timing](#), [Navigation Timing](#), [Paint Timing](#), [Reporting](#) and [Preload](#) all have tests which Chromium is failing.

The project would be composed of analyzing the various test failures, understanding the reasons for them and the complexity of solving them and then starting to tackle those reasons one by one.

Due to the distributed nature of the issues and the disparate causes, this project seems highly scalable, in the sense that we can easily divide labor if more folks come in.

**Project Size:** Large

**Mentors:** [yoavweiss@google.com](mailto:yoavweiss@google.com), [lclelland@google.com](mailto:lclelland@google.com)

**Project doc:**  Chromium Project Proposal Doc - Performance APIs spec compliance

## 4. Expose Render Blocking Status in Resource Timing

### Project Description:

The status of resources with regards to render-blocking is currently exposed in traces (and e.g. in WebPageTest), but not in Resource Timing. Given the enthusiastic reception of that status in WebPageTest, it'd be neat if we'd expose the same information in Resource Timing, and allow RUM providers to highlight render-blocking resources.

The project would require the student to take the feature through:

- Specification (maybe with some help, as this would require integration with HTML and Fetch)
- Implementation
- Tests
- Blink's shipping process

**Project Size:** Large

**Mentors:** [yoavweiss@google.com](mailto:yoavweiss@google.com), [haoliuk@google.com](mailto:haoliuk@google.com)

**Project doc:**  [Chromium Project Proposal Doc - Resource Timing render blocking status](#)

## 5. Audio Worklet for Emscripten

### Project Description:

The AudioWorklet is a fast and secure replacement of ScriptProcessor. However, the usage of ScriptProcessor is still significantly higher (0.24%) than AudioWorklet (0.043%) as of Mar 2022.

Due to its design flaw, the ScriptProcessor is considered harmful to excellent user experience (e.g. P0 security exploitation and glitching audio) and it makes developers question Chrome as a viable application platform for audio. It is also officially deprecated from the Web Audio API specification. So lowering its usage is a clear priority for the future of the web platform.

Fortunately, there is a clear uptick in AudioWorklet, and it can be accelerated with a strategic effort. Many developers are using Emscripten to port the existing C++/C audio code to Web Assembly, but Emscripten is still using ScriptProcessorNode for audio functionality. By updating Emscripten to use AudioWorklet instead, the developer experience can be tremendously improved.

**Project Size:** Medium

**Mentors:** [hongchan@chromium.org](mailto:hongchan@chromium.org), [mjwilson@chromium.org](mailto:mjwilson@chromium.org)

**Project Doc:**  [Chromium Project Proposal Doc: Audio Worklet for Emscripten](#)

## 6. Add Screenshots to Support Tool

### Project Description:

Support Tool is a tool in Chrome browser that is aimed to collect and export debug data for solving user issues. First version of the Support Tool is planned to be ready for testing in the first half of 2022 ([go/support-tool-v1-design](https://go/support-tool-v1-design)). Support Tool V1 will have only basic functionalities. We have planned additional features for the Support Tool to implement during second half of 2022 (please see design doc's Features For Future Versions) like screenshot/screen-recording and Google drive integration. The intern project will be about adding Screenshots to the data packets Support Tool collects.

Screenshot is a very helpful feature we will have in the Support Tool. Screenshots are a great help to Support Engineers to understand the ongoing issue and having them in support packets is requested by them. This feature will enable users to take screenshots when reproducing the issue. The Support Tool gives users the option to mask the PII (Personally Identifiable Information) in the collected data. After taking the screenshots, users will be able to blur the areas they want in the screenshot they took.

**Project Size:** Either Large or Medium

**Mentors:** [iremuguz@google.com](mailto:iremuguz@google.com), [pmarko@chromium.org](mailto:pmarko@chromium.org)

**Project Doc:** (public link) [Chromium Project Proposal Doc - Screenshots For Support Tool](#) (restricted view link, please use the public one above) [Chromium Project Proposal Doc](#)

## 7. VirtIO Balloon Performance Improvements

### Project Description:


ChromeOS heavily utilizes virtualization to enable a wide range of features. The most prominent is running android apps on ChromeOS using a platform called ARCVM. On ChromeOS devices with small (~4GB) system RAM, care must be taken to efficiently share memory between ChromeOS and Android. ChromeOS uses a mechanism called the VirtIO Balloon to facilitate this memory sharing. The balloon itself is a relatively simple mechanism that allocates memory in the guest kernel (Android) and then reports those pages to the host (ChromeOS) as available to use - thereby granting the host more memory.

The goal of this project is to explore performance improvements to the VirtIO Balloon driver in the linux kernel and its communication with crosvm (the virtual machine monitor used by ChromeOS). The current implementation of the balloon driver can be CPU intensive when inflating/deflating the balloon, and we'd like to minimize the impact the balloon has on both the host and the guest.

The VirtIO balloon is a core component of ChromeOS and this project has the potential for big impact on performance of virtualization on ChromeOS. Additionally, this project is a rare opportunity to dive deep into OS internals, work directly with the linux kernel, and get some great experience with low level system performance.

**Project Size:** Either Large or Medium

**Mentors:** [kalutes@chromium.org](mailto:kalutes@chromium.org)

**Project Doc:**  GSoC22 - VirtIO Balloon Performance Improvements

## 8. Make ChromeStatus.com frontend into SPA

### Project Description ([Github](#)):

Refactor the chromestatus.com frontend to change it from HTML page templates that use web components into a single-page-app (SPA). The student would need to learn about web components and SPAs, convert several HTML templates into web components, and implement a top-level web component for the whole app and build URI routing logic. Parts of the app that currently use server-side HTML generation would need to be reimplemented as JSON APIs.

**Project Size:** Medium

**Mentors:** [jrobbins@google.com](mailto:jrobbins@google.com), [kyleju@google.com](mailto:kyleju@google.com)

**Project doc:** [WebStatus Frontend Re-Architecture](#)

## 9. Aggregation Jobs Control Plane


### Project Description:

As part of phasing out support for [third-party cookies](#), more privacy-protecting approaches to ads measurement are being explored. Among them is a multi-party-computation (MPC) based [aggregation service](#) integrated with the [Chrome Attribution Reporting API](#). The MPC protocol is based on cryptographics and relies on splitting measurements between two parties where only aggregation results are visible to the requestor. This new approach offers many opportunities to contribute and improve the usability and observability of the system. As part of this project you'll help design a control plane to monitor and control the status of aggregation jobs.

(Some experience in web application design and development may be useful)

**Project Size:** Large

**Mentors:** [taoliao@google.com](mailto:taoliao@google.com), [rkubis@google.com](mailto:rkubis@google.com)

**Project Doc:**  2022 Chromium Project Proposal Doc - Aggregation Jobs Control Plane

**Internal Project Doc (use doc above):**

 2022 Chromium Project Proposal Doc - Privacy Preserving Computation Technology

## 10. Perfetto: Visualize Dropped Frame Warnings

### Project Description:

Perfetto is a tracing tool, and can be used to record performance traces of websites.

There is already plenty of support for visualizing timing of Events, Animations, and Frames, as well as support for annotating key events, such as "Dropped Frame" warnings during smoothness "janks".


These Dropped Frame warnings are very useful, but the correctness is a bit lacking. We have been developing new Smoothness metrics and it would be excellent to update the perfetto tracing tool to visualize these "dropped frame" warnings in a consistent manner.

We have existing expertise in the Smoothness metric, tracing, and perfetto-- but there is plenty of room for excellent quality of life improvements for both chromium and web developers.

(Some SQL experience may be useful.)

**Project Size:** Large

**Mentors:** [mmocny@chromium.org](mailto:mmocny@chromium.org), [lanwei@chromium.org](mailto:lanwei@chromium.org)

**Project Doc:**  [Chromium GSoC 2022 Project Proposal: Animation Smoothness Alerts in ...](#)

**Internal Project Doc (see doc above):** [Chromium GSoC 2022 Project Proposal: Animation Smoothness Alerts in Perfetto UI](#)

## 11. Reducing Chrome's Presentation Latency on Windows

### Project Description:

Chrome relies on DXGI swapchain flips to provide back pressure using the default number of pending frames (3). This is suboptimal from a latency perspective since a small hiccup can increase latency and not recover until Chrome stops rendering (e.g. until current scroll or animation completes). Using frame latency waitable events to drive Chrome's rendering should improve latency and also allow recovering from hiccups by dropping frames rather than increasing latency

(<https://docs.microsoft.com/en-us/windows/uwp/gaming/reduce-latency-with-dxgi-1-3-swap-chains>). In a similar vein, we would like to improve Chrome's rendering on variable refresh rate displays by configuring the swap chain correctly i.e. with "allow tearing" flag.

**Project Size:** Medium

**Mentors:** [sunnyps@google.com](mailto:sunnyps@google.com), [zmo@google.com](mailto:zmo@google.com)

## 12. Happy Eyeball Version 2 (RFC 8305)

### Project Description:

Implement Happy Eyeballs Version 2 ([RFC 8305](#)) for Chrome to improve network connectivity using concurrency. You will touch Chrome network stack code (//net) to implement the algorithm described in the RFC 8305 in C++.

Here is an excerpt from the RFC.

**Abstract:**

Many communication protocols operating over the modern Internet use hostnames. These often resolve to multiple IP addresses, each of which may have different performance and connectivity characteristics. Since specific addresses or address families (IPv4 or IPv6) may be blocked, broken, or sub-optimal on a network, clients that attempt multiple connections in parallel have a chance of establishing a connection more quickly.

**Project Size:** Medium

**Mentors:** [toyoshim@google.com](mailto:toyoshim@google.com), [ricea@google.com](mailto:ricea@google.com), [bashi@google.com](mailto:bashi@google.com)


## 13. WebGPU Backend of OpenCV's Deep Neural Network

**Project Description:**

Complete the WebGPU backend of OpenCV's Deep Neural Network module, begun under an earlier GSOC project in <https://github.com/opencv/opencv/pull/18066>. Revise the code to work with the current version of the WebGPU and WSGL specifications; implement more operators to pass more tests; and get the WebGPU backend landed as optional.

**Project Size:** Medium

**Mentors:** [kbr@google.com](mailto:kbr@google.com), [vrabaud@google.com](mailto:vrabaud@google.com)

**Project doc:**  GSOC2022 - WebGPU

## 14. Support new IR in V8 Turbolizer

**Project Description:**

V8 is adding a new intermediate representation (IR) to the TurboFan optimizing JavaScript compiler. For the old IR, there is Turbolizer, a web-based [tool](#) to inspect the IR, which is important for debugging the optimizing compiler.

This project is about extending Turbolizer to also support the new IR. Turbolizer is written in TypeScript, so this project mostly involves TypeScript coding, but adapting V8's C++ code outputting the IR data as JSON might also be necessary.

**Project Size:** Medium

**Mentors:** [tebbi@google.com](mailto:tebbi@google.com), [nicohartmann@google.com](mailto:nicohartmann@google.com)

**Project doc:**  GSoC 2022 - V8 Turbolizer

## 15. lld improvements

**Project Description:**

Chrome uses LLVM's lld linker on all platforms. This project is about improving several features in lld, to make Chrome developers more productive, and to improve the quality of the generated binary.

Lld consists of three fairly independent linkers, one each for COFF (Windows), Mach-O (macOS/iOS), and ELF (rest).

Possible improvements include:

- Researching and implementing a better arm branch island insertion algorithm (ELF, Mach-O)
- Improving the Mach-O port's diagnostics by implementing techniques already used in the COFF and ELF ports:
  - Dedup "undefined symbol" errors by the undefined symbol: Emit "undef symbol s, referenced by foo, bar, baz" instead of printing "undef symbol s, referenced by foo", "undef symbol s, referenced by bar", "undef symbol s, referenced by baz"
  - Look at debug information to print "undef symbol s, referenced by bar.cc:488, see bar.o" instead of just "undef symbol s, referenced by bar.o"
- Increasing parallelism in the Mach-O so that it loads input files in parallel, using techniques used in the COFF port
- Making the COFF port's /winsysroot: flag (<https://reviews.llvm.org/D118070>) work without an explicit /machine: flag (<https://crbug.com/1300005>)
- The ELF port does typo correction for undefined symbols. It'd be nice to port this to the Mach-O port, and possibly to the COFF port (need to check if it makes sense with the different mangling).

Any subset of these would be valuable.

**Project Size:** Variable

**Mentors:** [thakis@chromium.org](mailto:thakis@chromium.org), [hans@chromium.org](mailto:hans@chromium.org)