Joint WebRTC WG/SCCG Meeting

September 14, 2023
17:00 - 18:30 Seville time
8:00 - 9:30 Pacific Time
15:00 - 16:30 UTC

W3C WG IPR Policy

- WebRTC WG abides by the W3C Patent Policy: https://www.w3.org/Consortium/Patent-Policy/
- Only people and companies listed at https://www.w3.org/groups/wg/webrtc/ipr/ are allowed to make substantive contributions to the WebRTC WG specs (first 2 technical items on the agenda)
- The Screen Capture CG works under the W3C Community Group CLA

W3C Code of Conduct

- This meeting operates under W3C Code of Ethics and Professional Conduct

- We're all passionate about improving WebRTC and the Web, but let's all keep the conversations cordial and professional

Safety Reminders

While attending TPAC, follow the health rules:

- Authorized masks are required indoors at all times. If you need to remove your mask during the meeting, please keep it short
- Daily test is expected

Please be aware of and respect the personal boundaries of your fellow participants

https://www.w3.org/2023/09/TPAC/health.html

Welcome!

- Welcome to the TPAC 2023 joint meeting of the WEBRTC WG and the SCCG.
- TPAC 2023 Schedule
- Link to slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- Will we be recording the session?
- Volunteers for note taking?

TPAC 2023 Meeting Schedule

- WebRTC WG: September 12, 2023 (concluded)
  - 11:30 - 16:30 Seville Time
  - Meeting info
  - Slides
- Joint WebRTC/SCCG: September 14, 2023
  - 17:00 - 18:30 Seville Time
  - Meeting Info
  - Slides
- Joint WebRTC/MEDIA September 15, 2023
  - 14:30 - 16:30 Seville Time
  - Meeting Info
  - Slides

Virtual Meeting Tips (Zoom)

- **Both local and remote participants need to be on irc.w3.org channel #webrtc.**
- **Use “Raise hand” to get into the speaker queue and “Lower hand” to get out of the speaker queue.**
- **To try out WebCodecs over RTCDatachannel (not RTP!) join using a Browser.**
- **Please use headphones when speaking to avoid echo.**
- **Please wait for microphone access to be granted before speaking.**
- **Please state your full name before speaking.**

Issues for Discussion Today

Agenda:

- 17:00 - 17:10: Introduction
- 17:10 - 17:40: Dynamic-switching between any display-surface types
- 17:40 - 18:00: getViewportMedia (capture-current-tab)
- 18:00 - 18:25: Status updates on work of mutual interest
- 18:25 - 18:30: Wrap-up and Next Steps (Chairs)

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

Dynamic-switching between any display-surface types

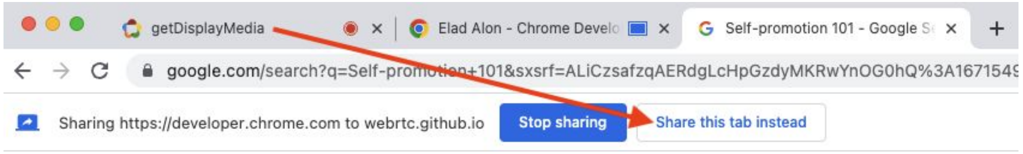
(under WebRTC WG Patent Policy)

Start Time: 17:10

End Time: 17:40

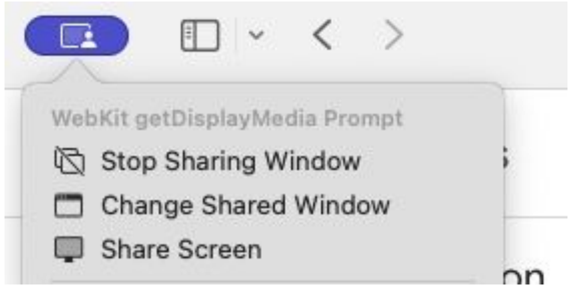
Dynamic-switching

- Several browsers allow users the ability to change what they're sharing "on the fly."
    - Chrome - change from one tab to another.
    - Safari - change between different windows, screens.
- Often easier for users than starting a new capture session.
- Managed by the surfaceSwitching option.
- Some limitations remain.



getDisplayMedia
Elad Alon - Chrome Develo
Self-promotion 101 - Google S
google.com/search?q=Self-promotion+101&sxsrf=ALiCzsafzqAERdgLcHpGzdyMKRwYnOG0hQ%3A167154
Sharing https://developer.chrome.com to webrtc.github.io
Stop sharing
Share this tab instead

Chrome (any platform)



Safari (macOS)

Challenge

Can we extend UA capabilities in this area without any spec changes? Should we just expect the UA to switch out the source?

If we did, there would be some challenges. In the next slides we will cover:

1. Audio
2. Methods
3. Auto-pause

Challenge #1: Audio

When a user chooses to dynamically change what they are sharing, they might also wish to either add or remove audio.

What if they wish to add audio? The current API shape does not support this.

The current API shape essentially says:

- “Audio from the start or never thereafter.”

That’s not a privacy-preserving API shape. It pushes users in a bad direction.

Side note (expecting certain objections)

Since Region Capture was introduced, MediaStreamTracks now exposes a method called cropTo(). It is only meaningful for tab-capture; otherwise it either:

- Throws an error
- The method is not exposed on the subtype of MST the UA instantiated.

In either case, we note that cropTo() belongs on the track, not on CaptureController, so that different clones of a track my be cropped to different targets. (Including having no target.)

Challenge #2: Methods

When changing captured surfaces, the set of methods which the application can validly invoke changes.

The aforementioned cropTo() is one example.

- Only valid when capturing a tab.
- Different capture-targets are valid depending on which tab.

This is a general problem.

- Web applications should not be required to check what they are capturing before every invocation.
- Many existing applications would be doing it only immediately after gDM resolves, then setting user-facing controls and forgetting about it.

Challenge #3: Auto-pause

As previously discussed in mediacapture-screen-share/issues/255:

- An application might set specific settings and user-facing controls based on what the user chooses to capture.
- When the user chooses to capture something else, the application needs to adapt, and might not wish for any new frames to be emitted until it has had time to adapt.
    - Events are not enough here, e.g. if the track is already plugged into a PeerConnection. By the time the app sees the events, some frames will have already been placed on the network.

Proposal outline

(Modulo known issue which is discussed later.)

Define a new event-listener. When one is registered, upon a dynamic-switch, concurrently:

1. Terminate the old tracks. (Behavior of onended events - TBD.)
2. Fire a new event with a new stream. (Potentially different number of tracks.)

That is:

```
const controller = new CaptureController();
controller.addEventListener('switch', (event) => {
  const videoElement = document.getElementById('myVideoElement');
  videoElement.srcObject = event.mediaStream;
});
…
navigator.mediaDevices.getDisplayMedia({ controller });
```

Proposal analysis

- Solves all three challenges.
- Up for discussions whether, when the event-listener is not registered, user agents:
  - MUST / MAY / SHOULD still offer the option to dynamic-switch.
  - MUST / MAY / SHOULD NOT offer the option to dynamic-switch.
  - [Unspecified]

P.S: Modulo

Invoking addEventListener should not have side effects.

If we decide to say user agents {MUST / MUST NOT / SHOULD / …} offer certain options depending on whether an event listener is registered, we might need to slightly change the API to fit this.

One option would be:

```
 CaptureController.enableDynamicSwitching(handler);
```

Discussion (End Time: 17:40)

-

getViewportMedia (capture-current-tab)

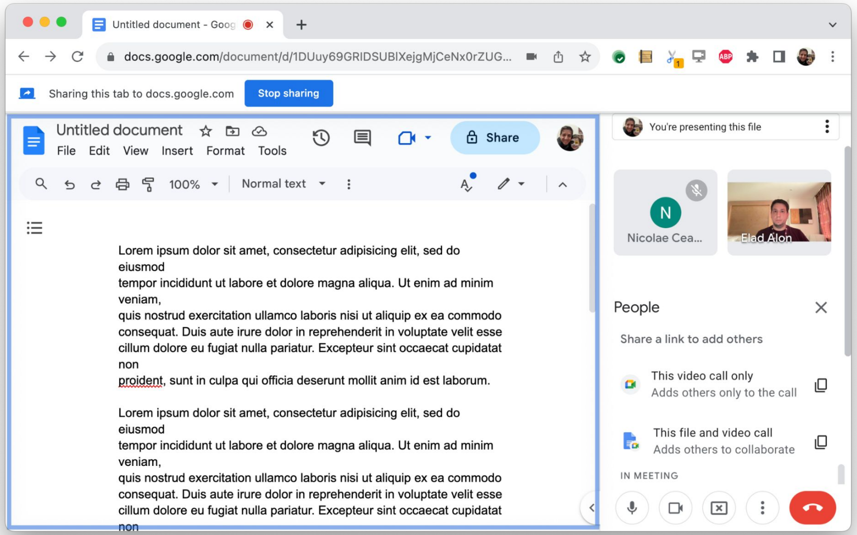
(under WebRTC WG Patent Policy)

Start Time: 17:40

End Time: 18:00

Capturing the current tab - use cases

- Embed a video-conferencing app into another app.
- Embed a third-party app into a video-conferencing app.



Untitled document - Goog
docs.google.com/document/d/1DUuy69GRIDSUBIXejgMjCeNx0rZUG...
Sharing this tab to docs.google.com
Stop sharing
Untitled document
File Edit View Insert Format Tools
Share
100%
Normal text
You're presenting this file
Nicolae Cea...
Elad Alon
People
Share a link to add others
This video call only
Adds others only to the call
This file and video call
Adds others to collaborate
IN MEETING

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non

Risks

When capturing one’s own tab, it is possible for the capturing application to embed arbitrary cross-origin content (that has neglected to opt-out) and read it.

Some users will approve self-capture without understanding the risk; sometimes they won’t even read the prompt at all.

Mitigations

Prior discussions result in consensus that the following set of mitigations is sufficient:

1. Cross-origin isolation
2. Opt-in by embedded documents

Challenges

1. Cross-origin isolation does not yet enjoy widespread adoption.
2. An opt-in mechanism for documents has not been agreed upon. (A candidate exists.)
3. The method of handling absent opt-in remains TBD.

Do we really need opt-in?

Given people’s position, it is not realistic to expect the opt-in requirement to be dropped today.

But if new information comes to light, or new arguments, we may re-examine the necessity of opt-in to begin with.

For now, let’s proceed under the assumption that we retain this requirement, and see how to shape it optimally.

Absent opt-in

Before we decide what mechanism should be used for opt-in, we should align on the desired properties.

Should absent opt-in block…

1. …the non-opt-in content from loading?
2. …the non-opt-in content from being captured? And if so…
   a. …stop capture of the entire tab?
   b. …stop capture of the non-opt-in content?

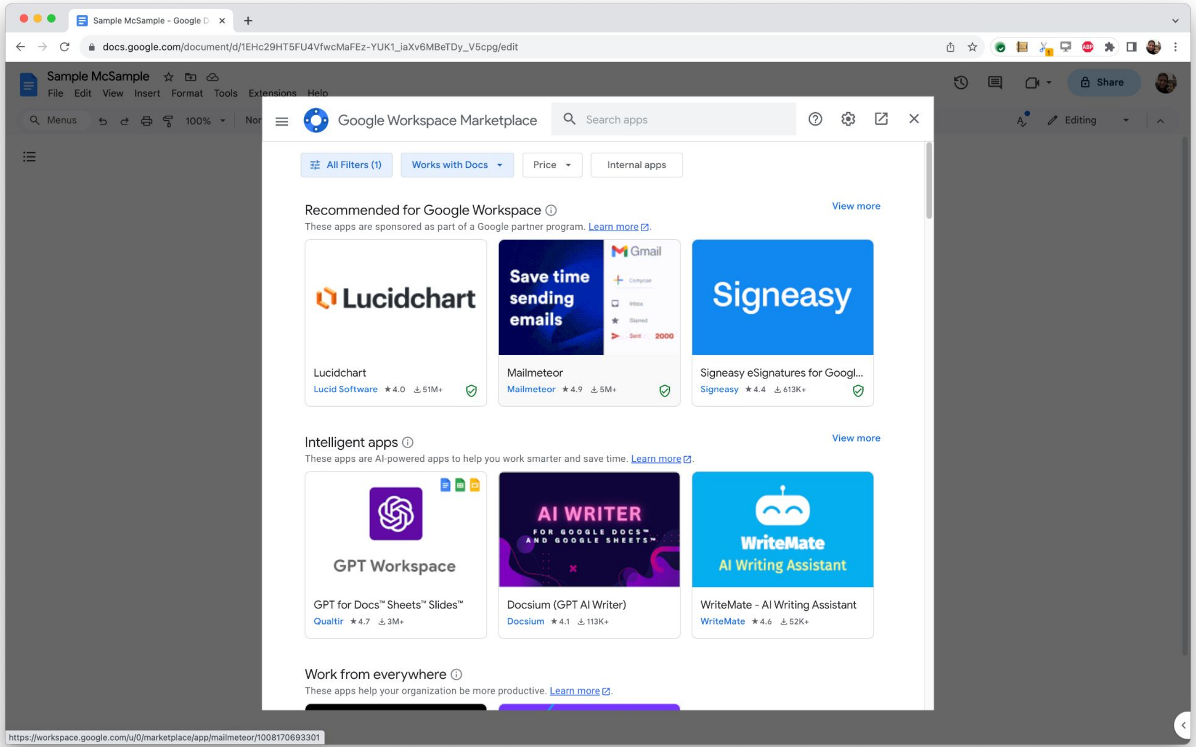
Preferred option (Elad)

I think we should seriously consider option 2a.

- gVM called with non-opt-in content already loaded?
  - Error.
- Non-opt-in content loaded while gVM capture active?
  - Don’t emit a frame until all such content is unloaded.

(Content loaded while dialog active is treated as content loaded after user approved the capture.)

Case study



Sample McSample - Google D

docs.google.com/document/d/1EHc29HT5FU4VfwcMaFEz-YUK1_iaXv6MBeTDy_V5cpg/edit

Sample McSample

File Edit View Insert Format Tools Extensions Help

Google Workspace Marketplace

Search apps

All Filters (1) | Works with Docs | Price | Internal apps

Recommended for Google Workspace

These apps are sponsored as part of a Google partner program. Learn more.

View more

Lucidchart — Lucid Software ★4.0 51M+

Mailmeteor — Mailmeteor ★4.9 5M+

Signeasy eSignatures for Googl... — Signeasy ★4.4 613K+

Intelligent apps

These apps are AI-powered apps to help you work smarter and save time. Learn more.

View more

GPT for Docs™ Sheets™ Slides™ — Qualtir ★4.7 3M+

Docsium (GPT AI Writer) — Docsium ★4.1 113K+

WriteMate - AI Writing Assistant — WriteMate ★4.6 52K+

Work from everywhere

These apps help your organization be more productive. Learn more.

https://workspace.google.com/u/0/marketplace/app/mailmeteor/1008170693301

Case study - analysis

Third-party content might be highly incentivized to comply with the imbeder’s requirements and keep being imbeded and loaded.

If absent opt-in blocks loading, embedders can end up unintentionally coercing third-party content to approve of being captured.

It’s **much more democratic** to allow third-party content to block capture and still be loaded.

Feasibility

Can implementations avoid race conditions and ensure non-opt-in content is never captured?

I believe so.

- During rendering, mark each surface with the set of origins permitted to capture it.
- Only emit the frame if the disjunction of all these sets includes the capturing origin.

Complicating factor - audio

Whether audio can be treated similarly should be discussed. (My intuition says yes, but it could be challenging - let’s loop in the experts.)

Error-handling by the capturer

To ensure graceful handling, we’ll want to add some mechanism to inform the capturer why the track was muted.

- Bespoke event?
- MuteReason?

Discussion (End Time: 18:00)

-

Status updates on work of mutual interest

Start Time: 18:00

End Time: 18:25

Screen Capture CG specs

The following specs, currently incubated in the SCCG, will be presented today:

1. Element Capture
2. Captured Surface Control
3. Captured Mouse Events

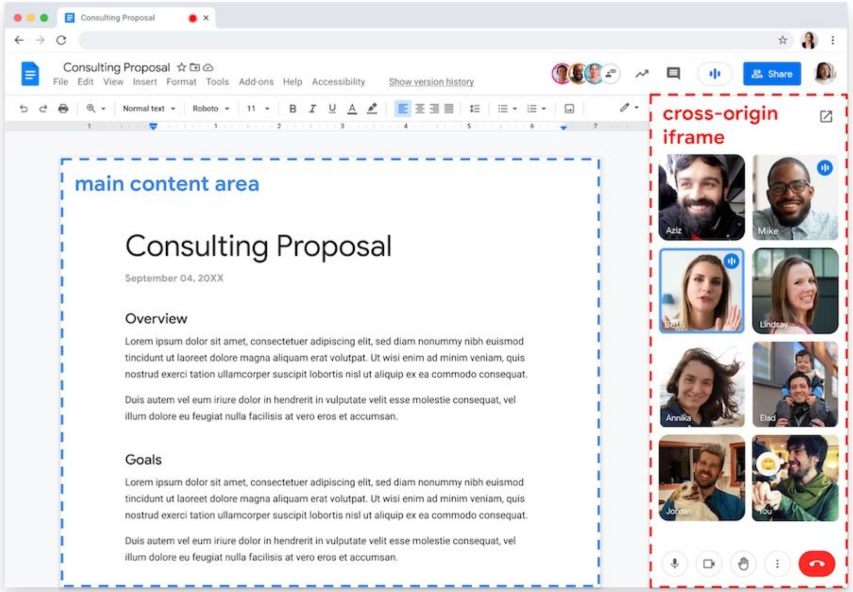
Element Capture

Recap

- Assume an **orthogonal, pre-existing** API such as getDisplayMedia or getViewportMedia was used to initiate tab-capture.
- **Region Capture** is an existing API that allows cropping video MediaStreamTracks.
  - Efficiently: Cropping done by the UA.
  - Robustly: Cropping tracks a target-element.
- **Element Capture** is essentially a variant of Region Capture that also removes occluding and occluded content.

Recap - Region Capture

1. Mint a CropTarget for a target-element.
2. Pass it, possibly cross-origin, to holder of MediaStreamTrack.
3. await mst.cropTo(cropTarget);
4. Frames cropped.
   a. Target moved? No problem.
   b. Target disappeared? No frames.



Consulting Proposal

September 04, 20XX

Overview

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

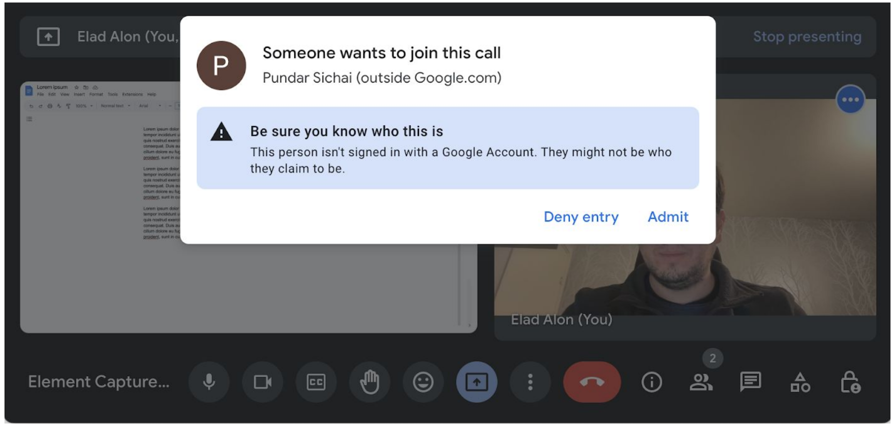
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Goals

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Recap - Region Capture Limitations



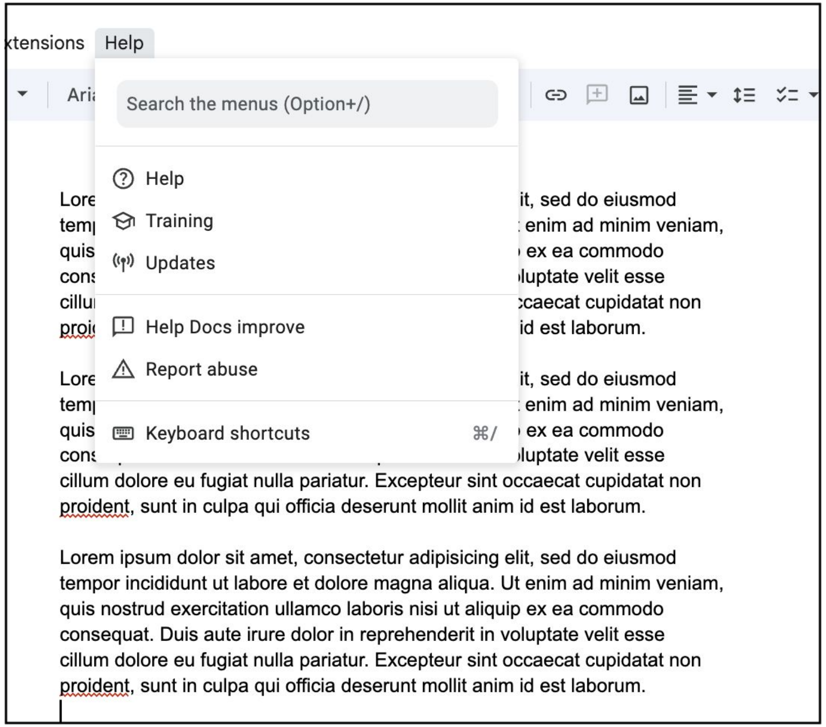
Someone wants to join this call

Pundar Sichai (outside Google.com)

Be sure you know who this is

This person isn't signed in with a Google Account. They might not be who they claim to be.

Deny entry Admit

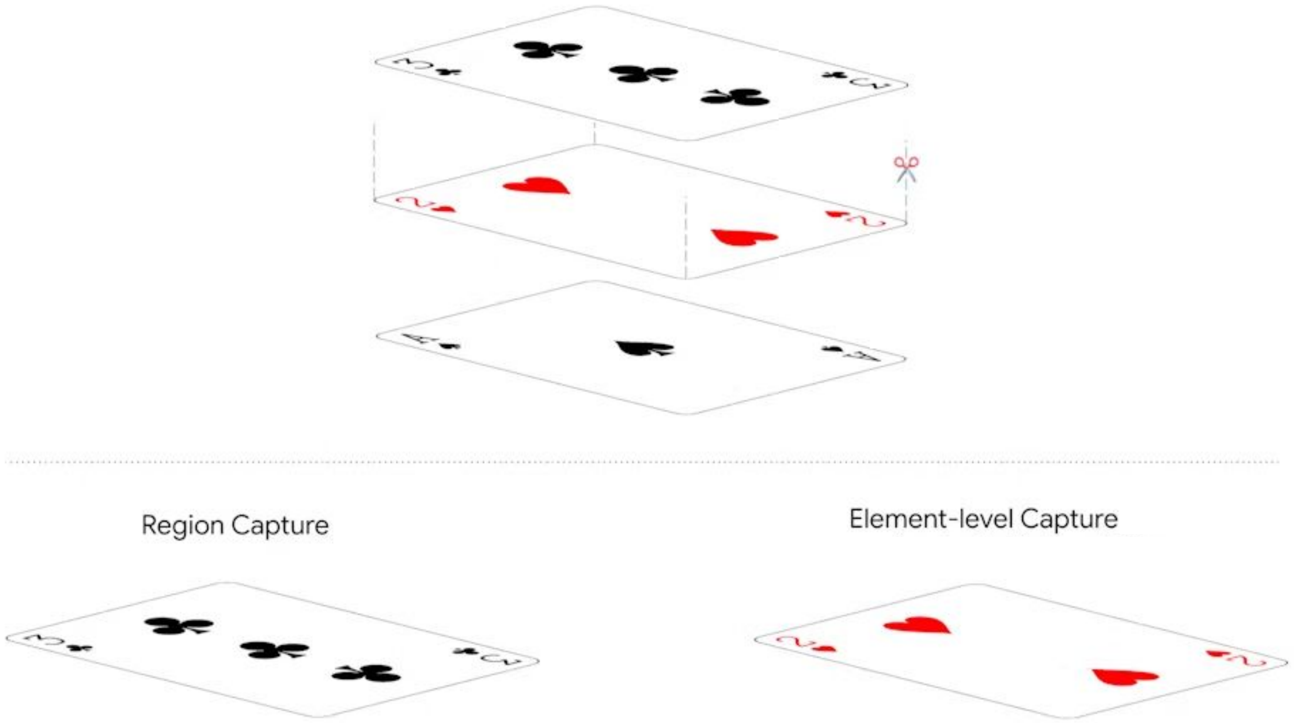


Lore it, sed do eiusmod temp enim ad minim veniam, quis ex ea commodo cons luptate velit esse cillu ccaecat cupidatat non proi id est laborum.

Lore it, sed do eiusmod temp enim ad minim veniam, quis ex ea commodo cons luptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Enter Element Capture



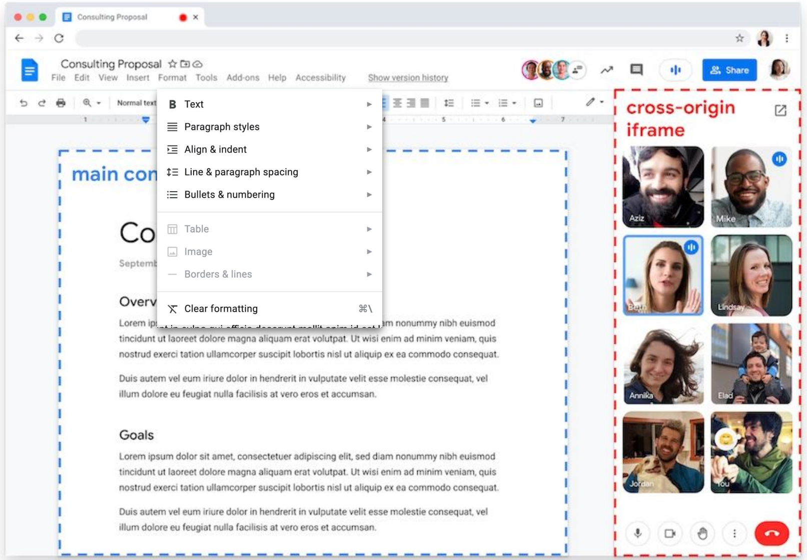
Element Capture - Delta from Region Capture

Differences in <u>results</u>:

- Element not descendant of the target element are not captured.
  - Neither occluded elements (behind),
  - nor occluding elements (in front).

Differences in <u>activation</u>:

- Use a RestrictionTarget instead of CropTarget.
- Use restrictTo() instead of cropTo().
- Some elements not valid targets.



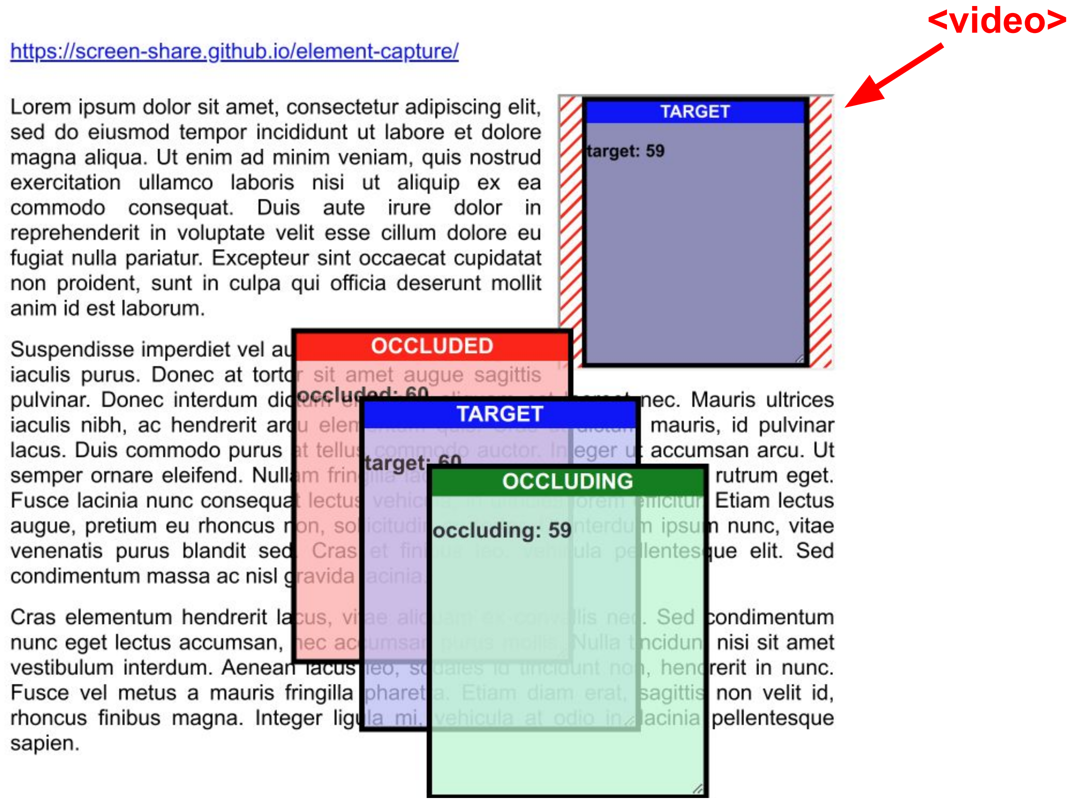
Lorem ip… m nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Element Capture - Demo



https://screen-share.github.io/element-capture/

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Suspendisse imperdiet vel au
iaculis purus. Donec at tortor sit amet augue sagittis
pulvinar. Donec interdum dic nec. Mauris ultrices
iaculis nibh, ac hendrerit arcu elem mauris, id pulvinar
lacus. Duis commodo purus at tellus commodo auctor. Integer ut accumsan arcu. Ut
semper ornare eleifend. Nullam fringilla rutrum eget.
Fusce lacinia nunc consequat lectus vehic efficitur Etiam lectus
augue, pretium eu rhoncus non, sollicitudi interdum ipsum nunc, vitae
venenatis purus blandit sed. Cras et finibus leo, vehicula pellentesque elit. Sed
condimentum massa ac nisl gravida lacinia.

Cras elementum hendrerit lacus, vitae aliquam ex convallis nec. Sed condimentum nunc eget lectus accumsan, nec accumsan purus mollis. Nulla tincidunt nisi sit amet vestibulum interdum. Aenean lacus leo, sodales id tincidunt non, hendrerit in nunc. Fusce vel metus a mauris fringilla pharetra. Etiam diam erat, sagittis non velit id, rhoncus finibus magna. Integer ligula mi, vehicula at odio in, lacinia pellentesque sapien.

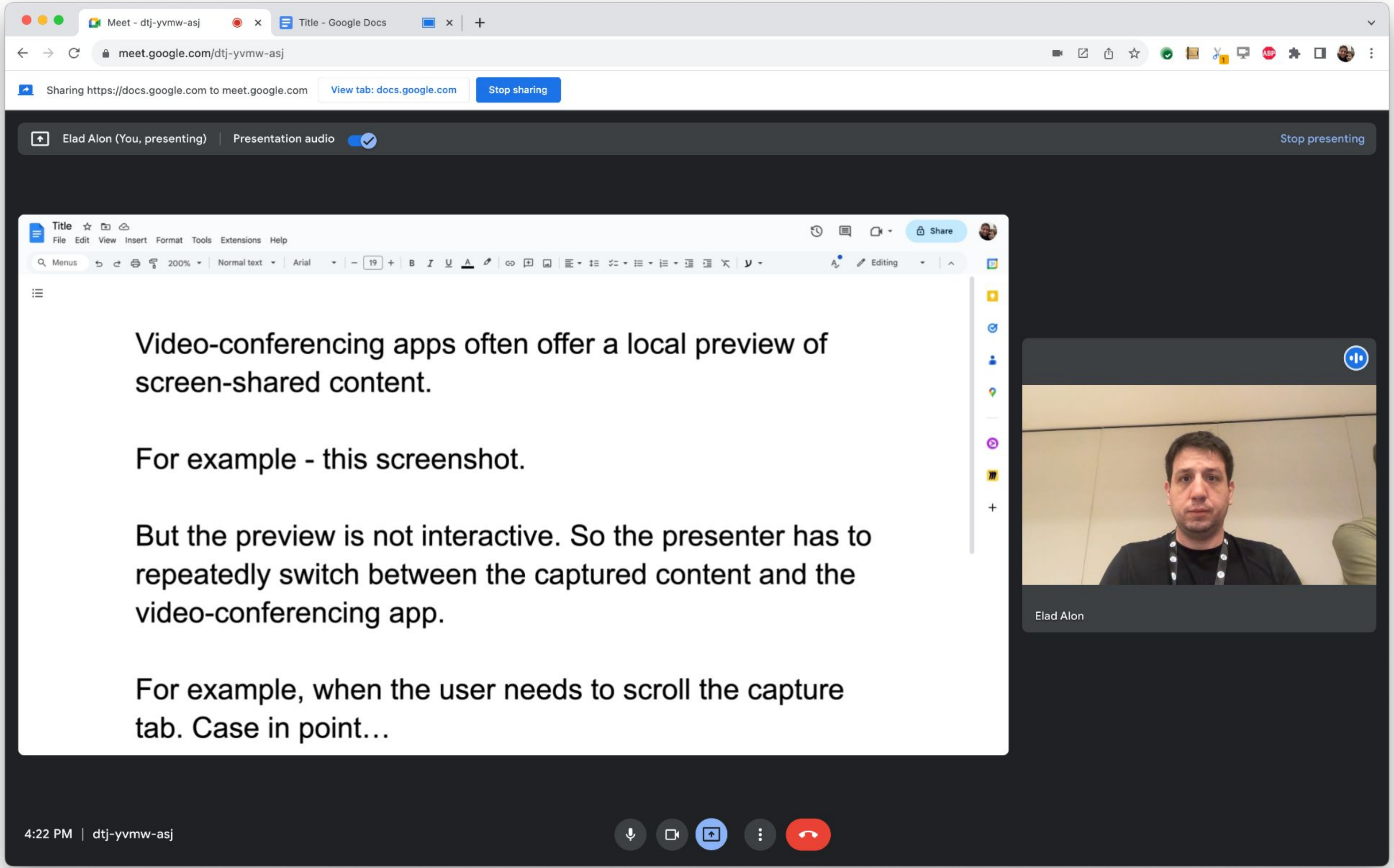
Risks

As always, the user’s privacy is a top priority. This API is safe:

- User grants permission to start capture using pre-existing means.
- User consistently warned that the app has permission to capture the entire tab.
- Even before this API, the contents of the captured tab could change at any time.
    - Navigation
    - Loading of new content
    - Layout changes
- Malicious applications can already subvert the users expectations and capture something without the user realizing it.
    - Content flashed briefly
    - Low opacity content
    - Piece by piece

Questions? (Element Capture)

Captured Surface Control


Meet - dtj-yvmw-asj
Title - Google Docs
meet.google.com/dtj-yvmw-asj
Sharing https://docs.google.com to meet.google.com
View tab: docs.google.com
Stop sharing
Elad Alon (You, presenting)
Presentation audio
Stop presenting
Title
File Edit View Insert Format Tools Extensions Help
Share
Editing
Elad Alon
4:22 PM | dtj-yvmw-asj

Video-conferencing apps often offer a local preview of screen-shared content.

For example - this screenshot.

But the preview is not interactive. So the presenter has to repeatedly switch between the captured content and the video-conferencing app.

For example, when the user needs to scroll the capture tab. Case in point…

Possible approaches, generally speaking

I see three categories of viable approaches:

1. UA offers users limited control of captured surface
2. UA offers users unlimited control of captured surface
3. UA offers apps limited control of captured surface

The first two options were previously discussed in the SCCG.

This presentation will be focused on the 3rd option.

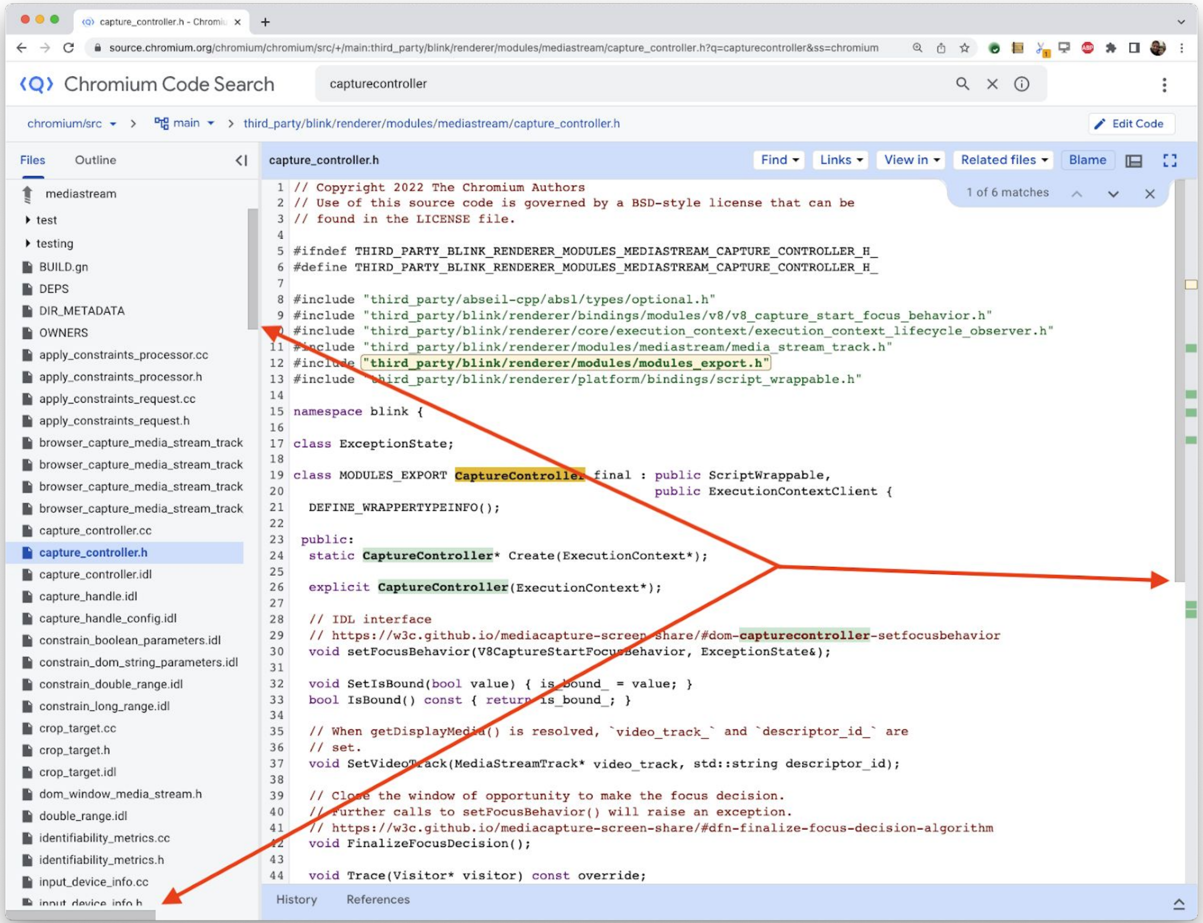
Limited app control of captured surface

After a <u>permissions prompt</u>, allow the capturing application <u>limited control</u> over the captured surface.

- Scrolling (over coordinates of choice)
- Zooming (of the captured tab)

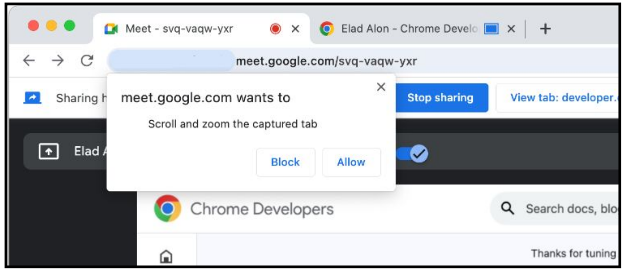
Benefits of this approach:

- Simple
- Relatively safe (scrolling rarely has side-effects - permission sufficient)
- Solves the challenge of “scroll what?” (Next slide)
- Allows delegation of (limited) control to remote participants


capture_controller.h - Chromiu
source.chromium.org/chromium/chromium/src/+/main:third_party/blink/renderer/modules/mediastream/capture_controller.h?q=capturecontroller&ss=chromium
Chromium Code Search
capturecontroller
chromium/src
main
third_party/blink/renderer/modules/mediastream/capture_controller.h
Edit Code
Files
Outline
mediastream
test
testing
BUILD.gn
DEPS
DIR_METADATA
OWNERS
apply_constraints_processor.cc
apply_constraints_processor.h
apply_constraints_request.cc
apply_constraints_request.h
browser_capture_media_stream_track
browser_capture_media_stream_track
browser_capture_media_stream_track
browser_capture_media_stream_track
capture_controller.cc
capture_controller.h
capture_controller.idl
capture_handle.idl
capture_handle_config.idl
constrain_boolean_parameters.idl
constrain_dom_string_parameters.idl
constrain_double_range.idl
constrain_long_range.idl
crop_target.cc
crop_target.h
crop_target.idl
dom_window_media_stream.h
double_range.idl
identifiability_metrics.cc
identifiability_metrics.h
input_device_info.cc
input_device_info.h
capture_controller.h
Find
Links
View in
Related files
Blame
1 of 6 matches
History
References

```
// Copyright 2022 The Chromium Authors
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef THIRD_PARTY_BLINK_RENDERER_MODULES_MEDIASTREAM_CAPTURE_CONTROLLER_H_
#define THIRD_PARTY_BLINK_RENDERER_MODULES_MEDIASTREAM_CAPTURE_CONTROLLER_H_

#include "third_party/abseil-cpp/absl/types/optional.h"
#include "third_party/blink/renderer/bindings/modules/v8/v8_capture_start_focus_behavior.h"
#include "third_party/blink/renderer/core/execution_context/execution_context_lifecycle_observer.h"
#include "third_party/blink/renderer/modules/mediastream/media_stream_track.h"
#include "third_party/blink/renderer/modules/modules_export.h"
#include "third_party/blink/renderer/platform/bindings/script_wrappable.h"

namespace blink {

class ExceptionState;

class MODULES_EXPORT CaptureController final : public ScriptWrappable,
                                              public ExecutionContextClient {
  DEFINE_WRAPPERTYPEINFO();

 public:
  static CaptureController* Create(ExecutionContext*);

  explicit CaptureController(ExecutionContext*);

  // IDL interface
  // https://w3c.github.io/mediacapture-screen-share/#dom-capturecontroller-setfocusbehavior
  void setFocusBehavior(V8CaptureStartFocusBehavior, ExceptionState&);

  void SetIsBound(bool value) { is_bound_ = value; }
  bool IsBound() const { return is_bound_; }

  // When getDisplayMedia() is resolved, `video_track_` and `descriptor_id_` are
  // set.
  void SetVideoTrack(MediaStreamTrack* video_track, std::string descriptor_id);

  // Close the window of opportunity to make the focus decision.
  // Further calls to setFocusBehavior() will raise an exception.
  // https://w3c.github.io/mediacapture-screen-share/#dfn-finalize-focus-decision-algorithm
  void FinalizeFocusDecision();

  void Trace(Visitor* visitor) const override;
```

Sample usage 1:
Initiate capture and obtain permission



meet.google.com wants to

Scroll and zoom the captured tab

Block Allow

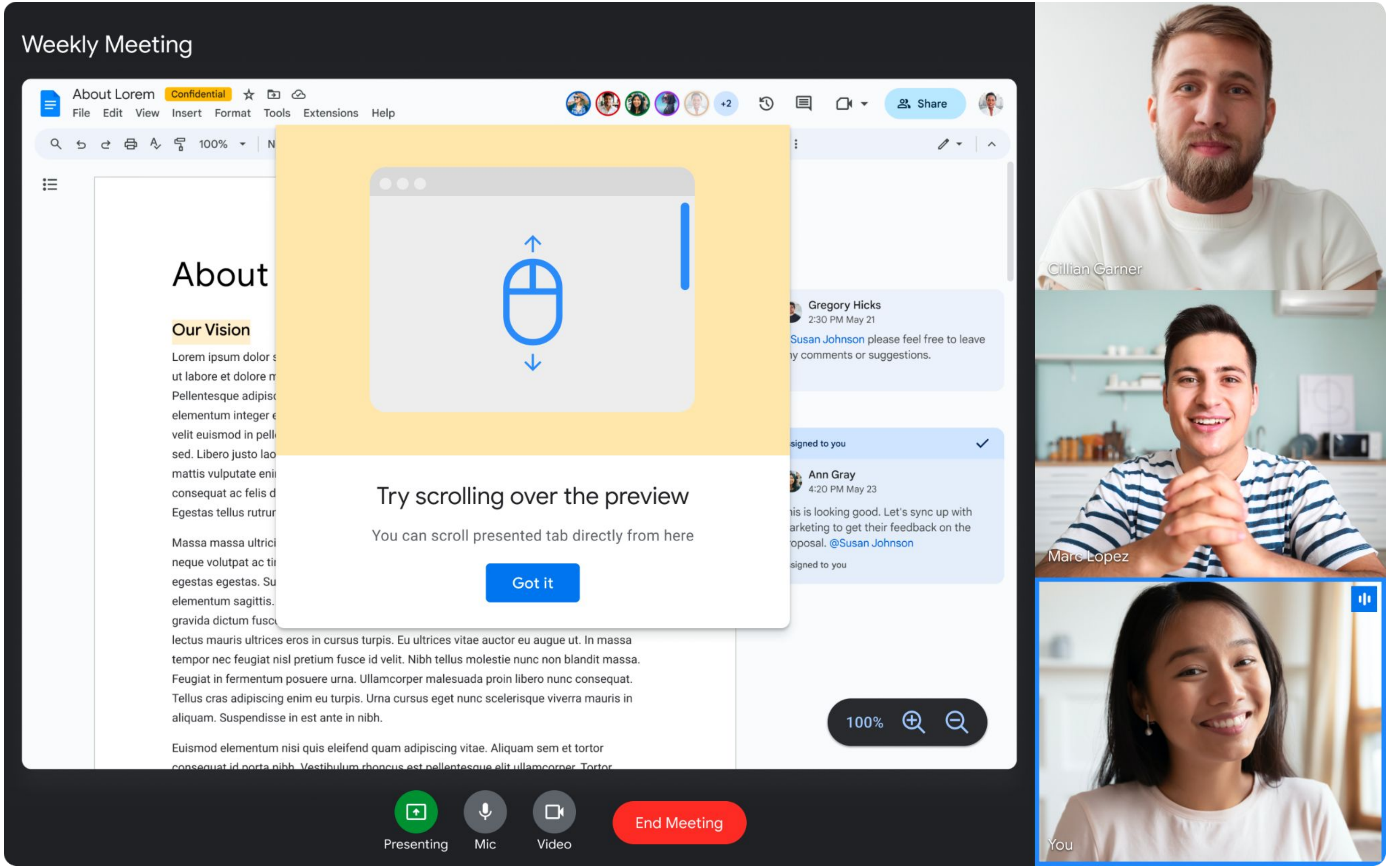
```
const controller = new CaptureController();
const stream =
    await navigator.mediaDevices.getDisplayMedia({ controller });
document.getElementById('video').srcObject = stream;

// Perform a null-action so as to prompt the user for permission.
try {
  await controller.sendWheel({});
} catch (e) {
  return;  // Permission denied. Bail.
}
```

Permission prompts are only displayed when attempting to use the API they gate access to. A null event is intentionally used to avoid performing an action before the user requests one.

Sample usage 2: Relay scroll events to captured surface

```
// Having obtained the user’s permission, we can now relay subsequent
// wheel events to the captured tab.
video.addEventListener("wheel", event => {
  const [x, y] = translateCoordinates(event.offsetX, event.offsetY);
  controller.sendWheel({
    x,
    y,
    wheelDeltaX: event.wheelDeltaX,
    wheelDeltaY: event.wheelDeltaY});
});
```

translateCoordinates() scales the coordinates in the video-element to those of the captured surface. Its implementation is left as an exercise for the reader.



About

Our Vision

Lorem ipsum dolor s
ut labore et dolore m
Pellentesque adipisc
elementum integer e
velit euismod in pell
sed. Libero justo lao
mattis vulputate eni
consequat ac felis d
Egestas tellus rutrur

Massa massa ultrici
neque volutpat ac ti
egestas egestas. Su
elementum sagittis.
gravida dictum fusc
lectus mauris ultrices eros in cursus turpis. Eu ultrices vitae auctor eu augue ut. In massa tempor nec feugiat nisl pretium fusce id velit. Nibh tellus molestie nunc non blandit massa. Feugiat in fermentum posuere urna. Ullamcorper malesuada proin libero nunc consequat. Tellus cras adipiscing enim eu turpis. Urna cursus eget nunc scelerisque viverra mauris in aliquam. Suspendisse in est ante in nibh.

Euismod elementum nisi quis eleifend quam adipiscing vitae. Aliquam sem et tortor

Try scrolling over the preview

You can scroll presented tab directly from here

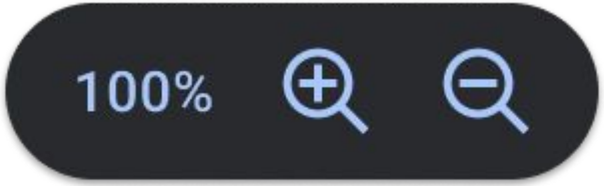
Gregory Hicks
2:30 PM May 21
Susan Johnson please feel free to leave
y comments or suggestions.

signed to you

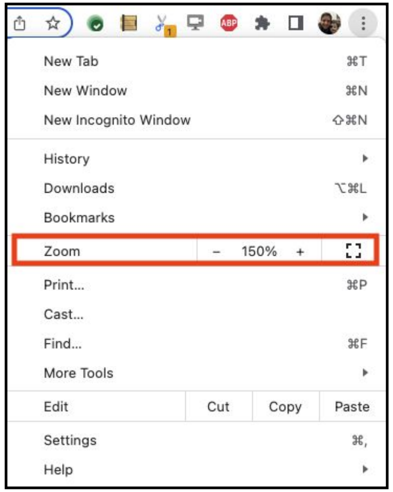
Ann Gray
4:20 PM May 23
is is looking good. Let's sync up with
arketing to get their feedback on the
oposal. @Susan Johnson
signed to you

Sample usage 3: Control zoom-level of captured tabs

```
const zoomInButton = document.getElementById('zoomInButton');
zoomInButton.addEventListener('click', async (event) => {
  const oldZoomLevel = await controller.getZoomLevel();
  const newZoomLevel =
      Math.min(oldZoomLevel + 10,
               controller.getMaxZoomLevel());
  controller.setZoomLevel(newZoomLevel);
});
```



Possible capturing-app UX



Effect in captured tab

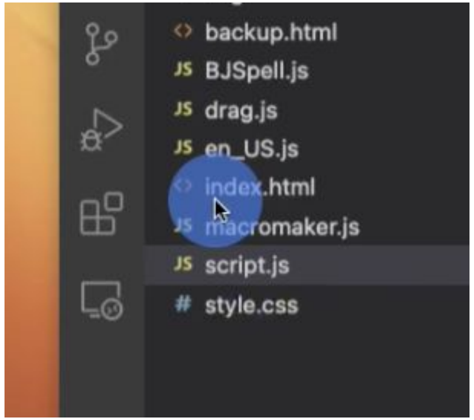
API shape

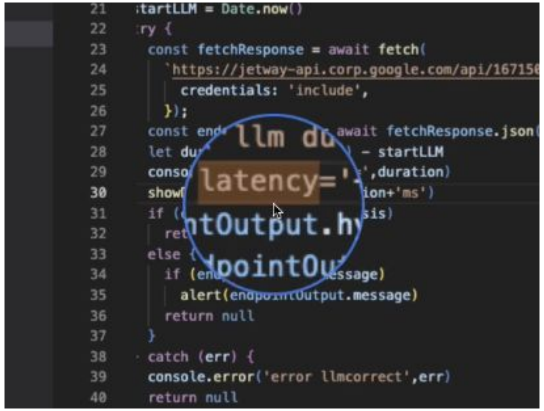
```
dictionary CapturedWheelAction {
  int x = 0;
  int y = 0;
  int wheelDeltaX = 0;
  int wheelDeltaY = 0;
};

partial interface CaptureController {
  // 0. Pre-existing and irrelevant methods omitted.
  ...

  // 1. Scrolling
  Promise<undefined> sendWheel(CapturedWheelAction action);

  // 2. Zoom-level
  int getMinZoomLevel();
  int getMaxZoomLevel();

  Promise<int> getZoomLevel();
  Promise<undefined> setZoomLevel(int zoomLevel);
};
```

Questions? (Captured Surface Control)

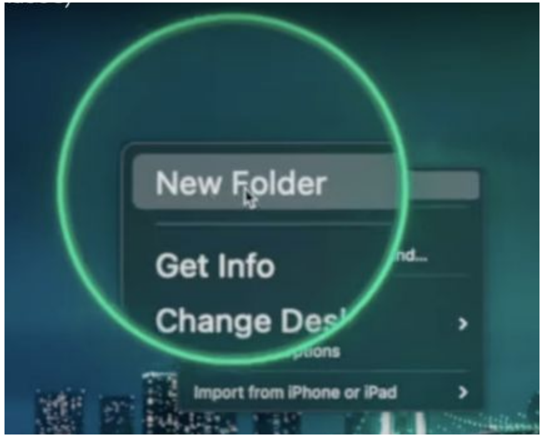
Captured Mouse Events

Cursor Enhancements (useful for mobile and accessibility)



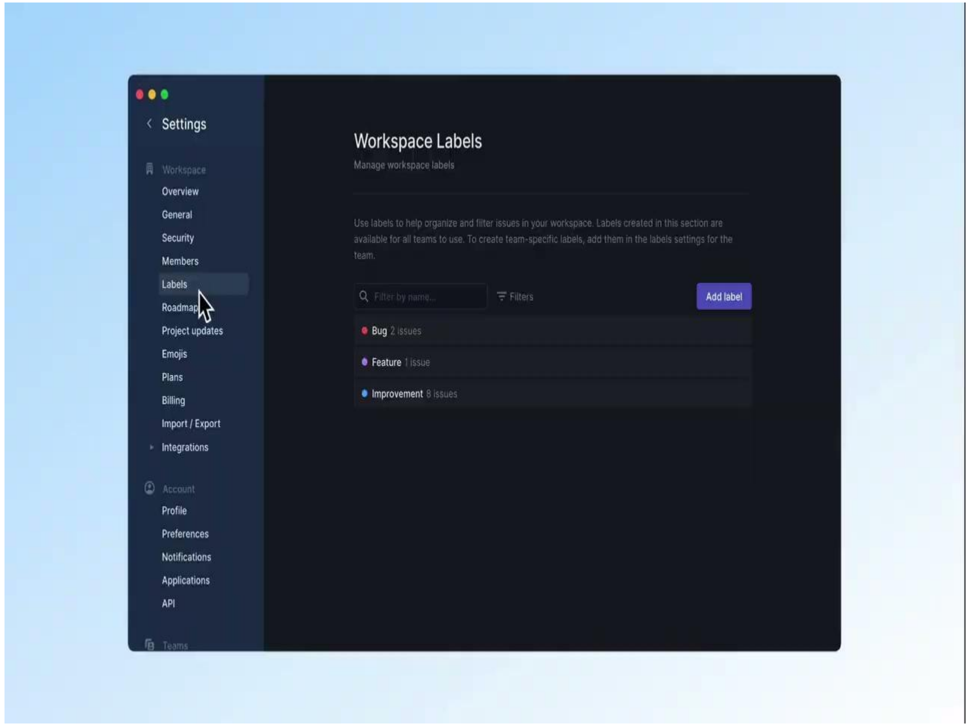


```
:tartLLM = Date.now()
:ry {
  const fetchResponse = await fetch(
    `https://jetway-api.corp.google.com/api/167150
      credentials: 'include',
    });
  const end    llm du   await fetchResponse.json(
  let du                 - startLLM
  conso  latency='      ',duration)
  show                  ion+'ms')
  if (  ntOutput.hy    sis)
    ret
  else {  dpointOu
    if (en             essage)
      alert(endpointOutput.message)
      return null
  }
  catch (err) {
  console.error('error llmcorrect',err)
  return null
```



https://www.ixeau.com/cursor-pro/
https://www.appblit.com/screegle

Post processing of recordings e.g. screen.studio



Workspace Labels

Manage workspace labels

Use labels to help organize and filter issues in your workspace. Labels created in this section are available for all teams to use. To create team-specific labels, add them in the labels settings for the team.

- Bug 2 issues
- Feature 1 issue
- Improvement 8 issues

https://www.screen.studio/

CPU, power and bandwidth savings

- In a perfect world, everything is free, and everything works.
- We don’t live in a perfect world.
  - Entire frames have to be re-scanned by encoders with every small change.
  - Delta frames are larger than an update of mouse coordinates.
  - Everything costs more CPU, power and bandwidth than the optimal.
- What if we…
  a. Stop capturing the cursor (encoder doesn’t re-scan input)
  b. Transmit mouse-coordinates separately from the frames
  c. Redraw the cursor on the receiver side
- Would that not lead to some savings?
  - Our initial measurements suggest that it would.

WebIDL

```
partial interface CaptureController {
  attribute EventHandler oncapturedmousechange;
};
```

WebIDL

```
[Exposed=Window]
interface CapturedMouseEvent : Event {
  constructor(DOMString type, optional CapturedMouseEventInit eventInitDict = {});
  readonly attribute long surfaceX;
  readonly attribute long surfaceY;
};
```

Extensions under discussion

Currently specified:

- Coordinates

Under discussion:

- Mouse-clicks (buttons)
- Modifier keys (most likely only in conjunction with mouse-clicks)

Questions? (Captured Mouse Events)

Discussion (End Time: 18:25)

-

Wrap-up and Next Steps (Chairs)

Start Time: 18:25

End Time: 18:30

Next Steps

- Content goes here

Thank you

<u>Special thanks to:</u>

WG and SG Participants, Editors & Chairs