Camera Effects Coordination

Breakout Session - TPAC 2024

Mark A. Foltz (mfoltz@google.com)

September 25, 2024

Ground Rules

This meeting operates under

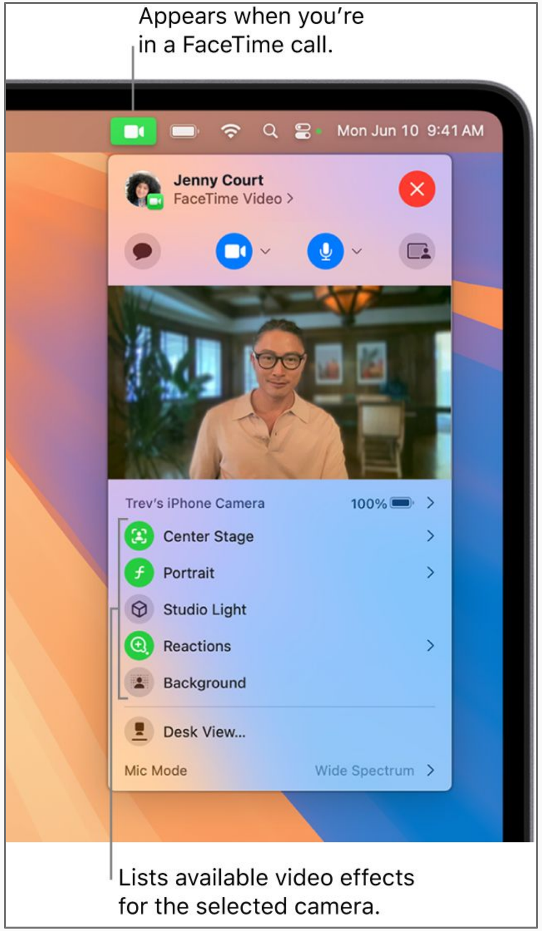
- The W3C Code of Ethics and Professional Conduct
- The W3C Antitrust and Competition Guidance
- For in-person attendees, this year's TPAC health policies

Agenda

1. Context & problem statement
2. Effects Coordination strategies
3. Proposed solution
4. Comparison with constraint based approach
5. Next steps and discussion

Context & Problem Statement

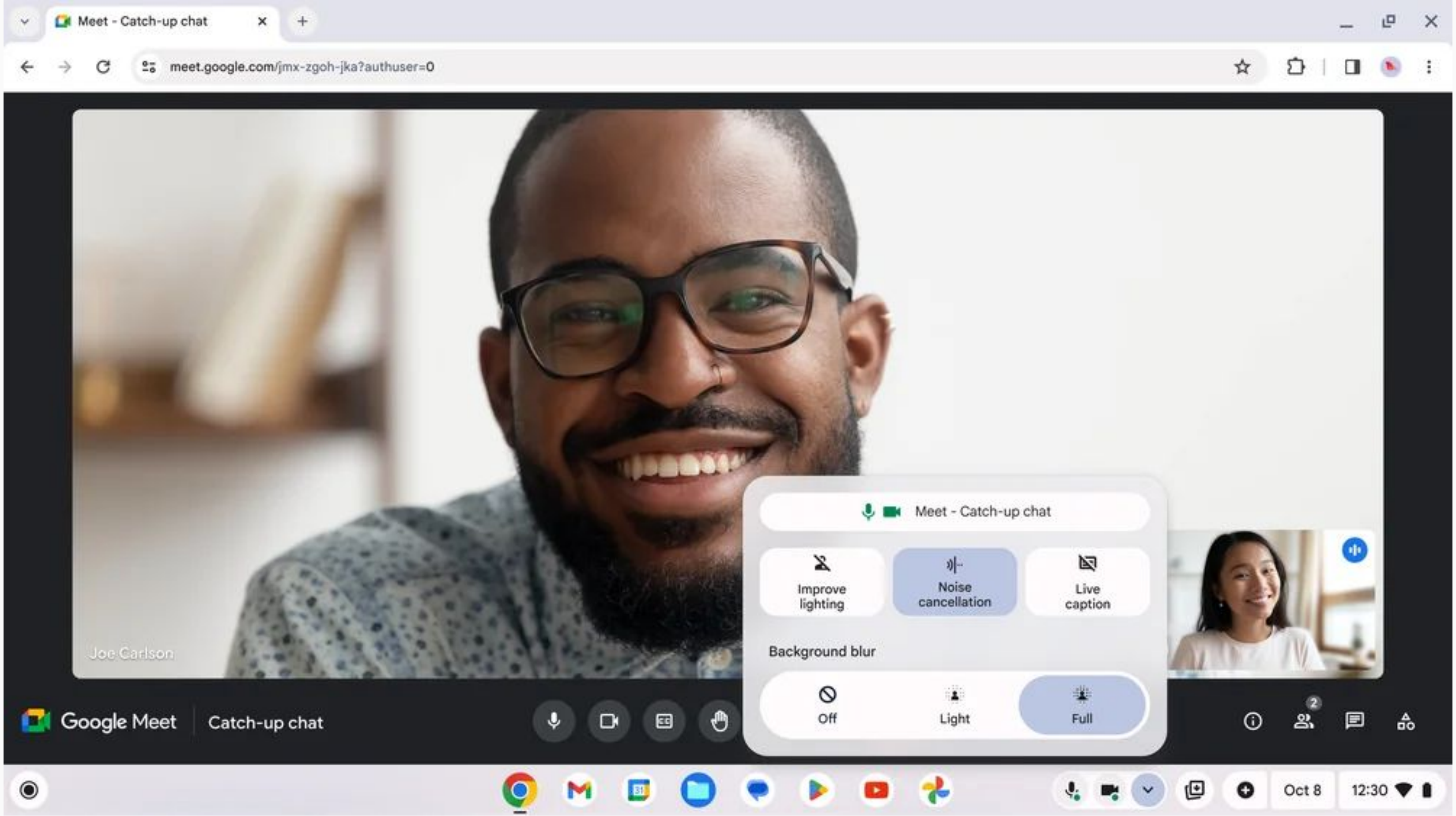
[Windows Studio Effects]



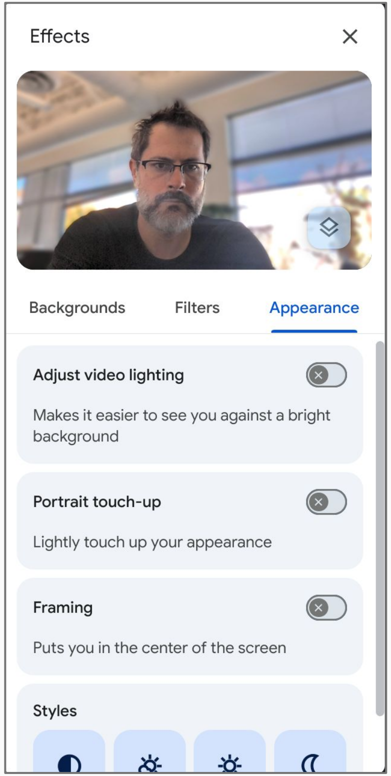
Appears when you're in a FaceTime call.

Lists available video effects for the selected camera.

[MacOS Video Effects]



Chromebook Plus



Effects

Backgrounds | Filters | Appearance

Adjust video lighting

Makes it easier to see you against a bright background

Portrait touch-up

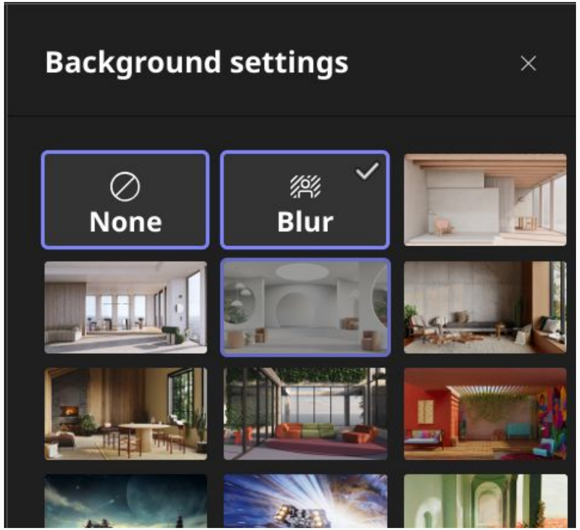
Lightly touch up your appearance

Framing

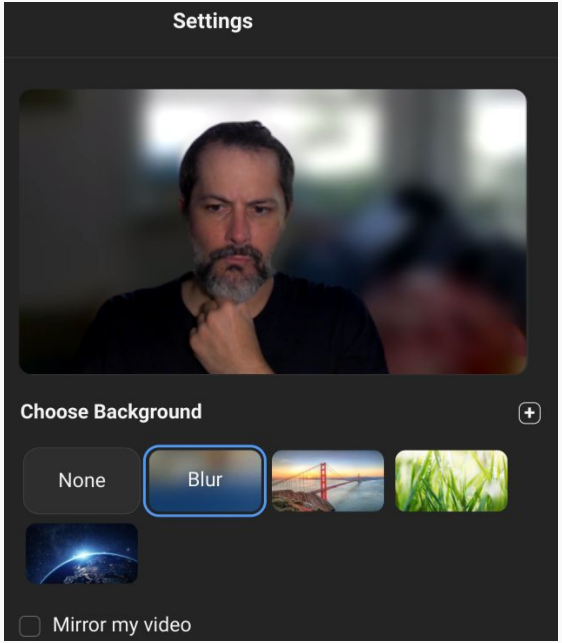
Puts you in the center of the screen

Styles

meet.google.com



teams.microsoft.com

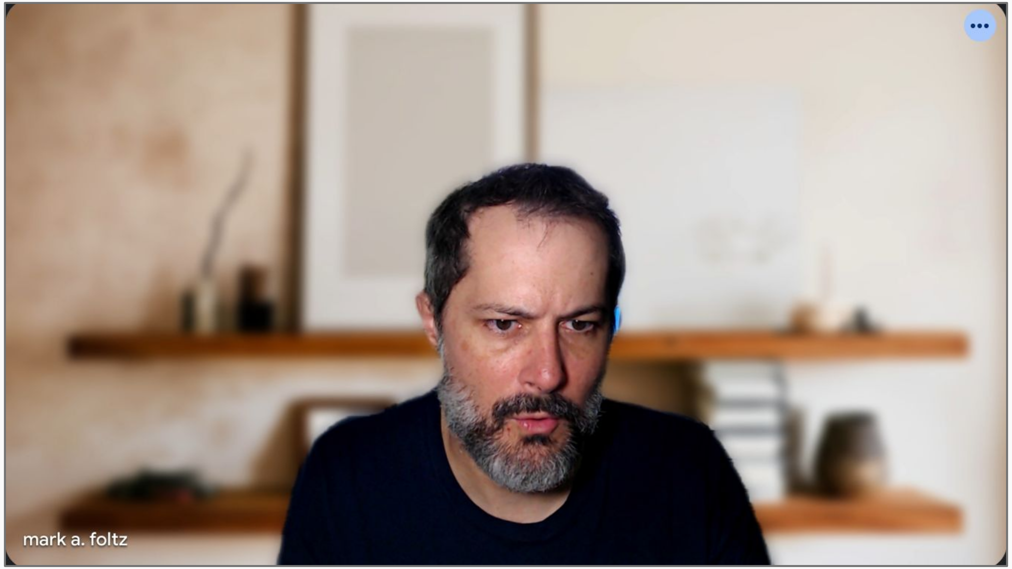


□ Mirror my video

app.zoom.us

Application Provided Effects

Double effects, double controls



ChromeOS background + Meet blur

Coordination Strategies

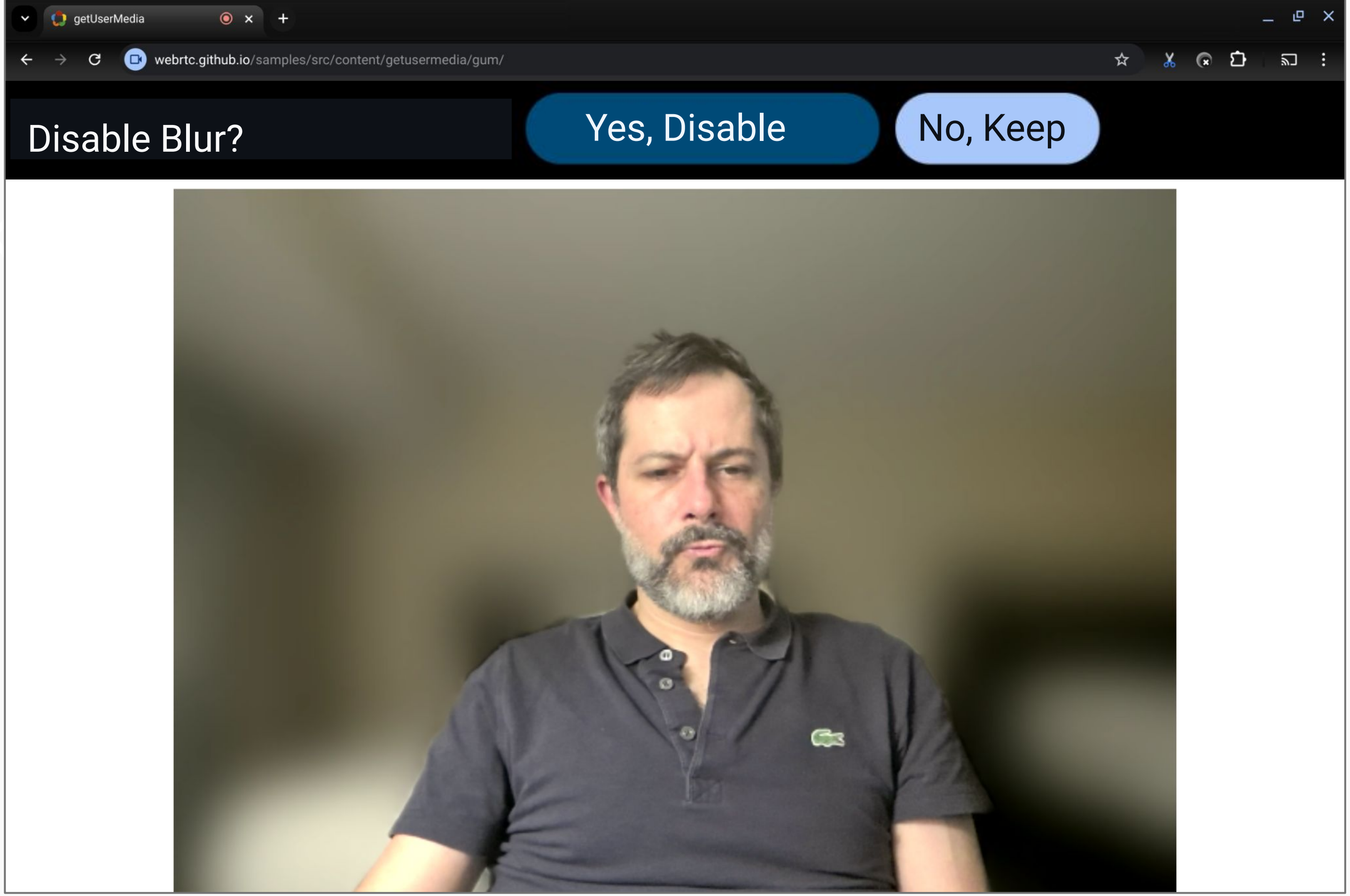
Feature Detection & Observing State

- Allow the application to know if blur is supported on the track
- Allow the application to know blur status
  - MediaStreamTrack property
  - Event “soon” after a state change takes place (disabled => enabled, enabled => disabled)
  - Frame by frame state
- Can notify the user, disable redundant effects, etc.

Capabilities

Allow the application to know how much control they have over effects.

- Can they enable the effect if it’s disabled?
- Can they disable the effect if it’s enabled?


getUserMedia
webrtc.github.io/samples/src/content/getusermedia/gum/
Disable Blur?
Yes, Disable
No, Keep

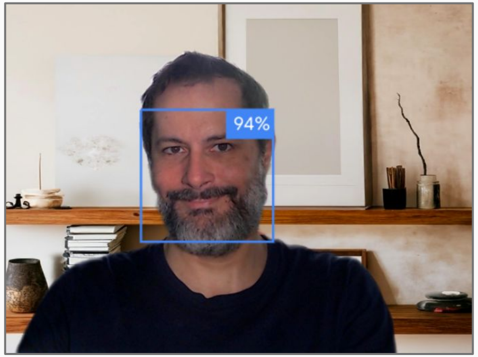
Ask the user

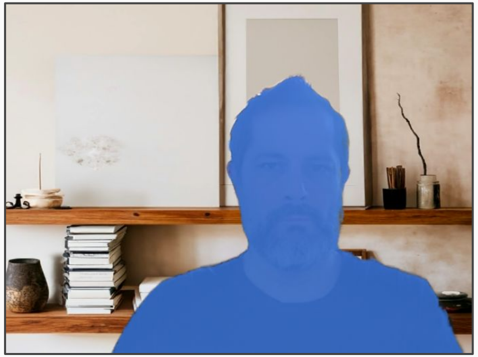
Direct Control

- Direct control of effect state (enable/disable)
- Problems
    - Effects today are per-camera or per-browser
    - Changes affect all sites simultaneously accessing the camera
    - Changes affect the next site that uses the camera
    - Sites could “fight” over the setting

Effect Intermediate Data

- Access to per-frame intermediate data used in effects processing
- Separate from whether pixels are actually modified
- Add background segmentation mask mediacapture-extensions#142





Proposed Solution

All code snippets from:
https://markafoltz.github.io/camera-effects/

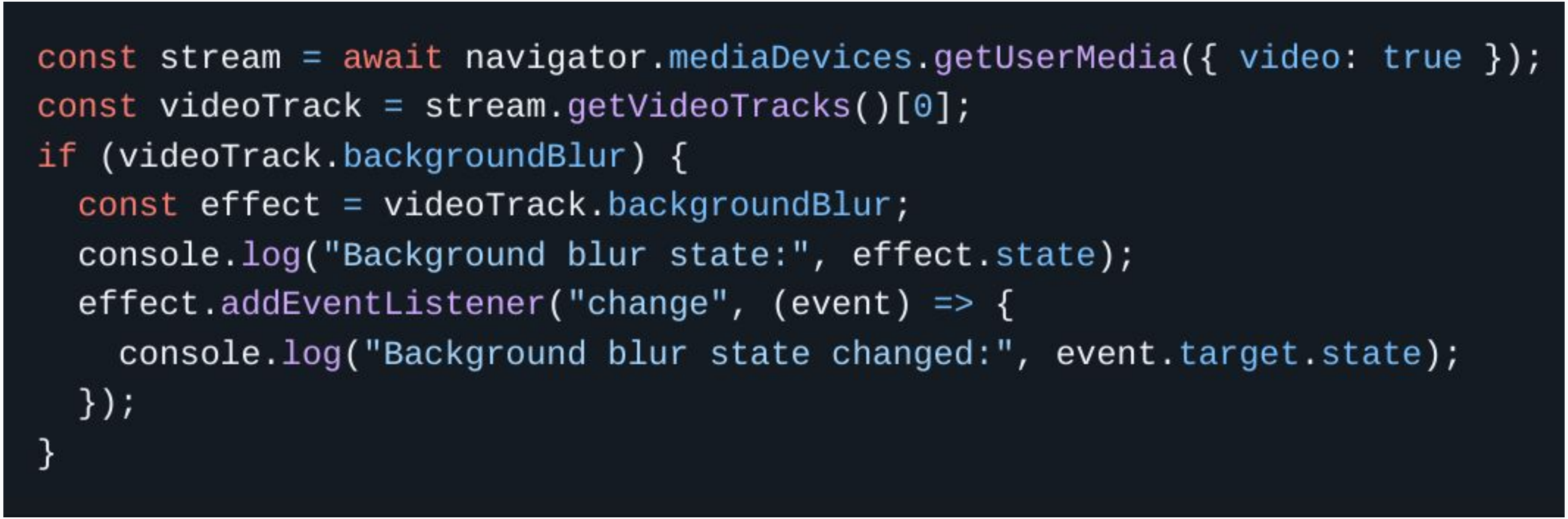
Goals

- Allow Web developers to easily access and monitor changes in platform blur.
- Enable Web developers to build new features that respond to changes in background blur.
- Provide a consistent and easy-to-use API for accessing platform effect state.

Non-goals

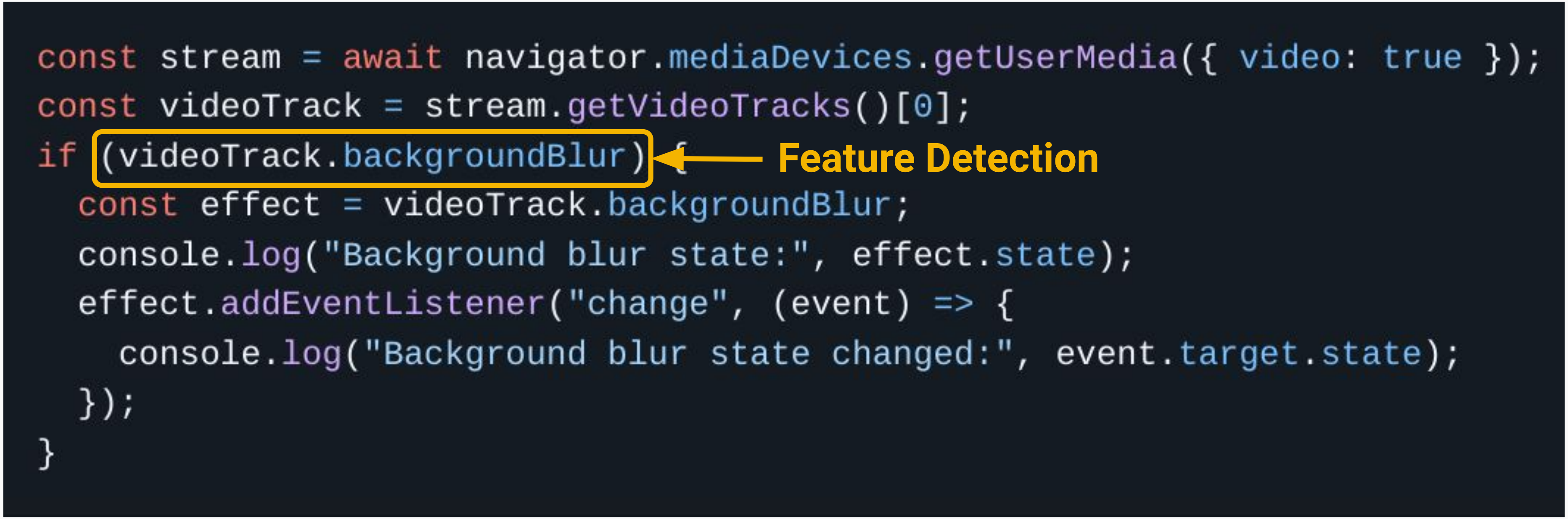
- This API does not provide a way to control platform effects. That functionality may be exposed in a future API.
- This API does not attempt to polyfill effects in platforms/browsers that do not support them.
- This API doesn’t include all possible platform effects. More effects may be exposed as future extensions of the API.

Observing State (via track)



```javascript
const stream = await navigator.mediaDevices.getUserMedia({ video: true });
const videoTrack = stream.getVideoTracks()[0];
if (videoTrack.backgroundBlur) {
  const effect = videoTrack.backgroundBlur;
  console.log("Background blur state:", effect.state);
  effect.addEventListener("change", (event) => {
    console.log("Background blur state changed:", event.target.state);
  });
}
```

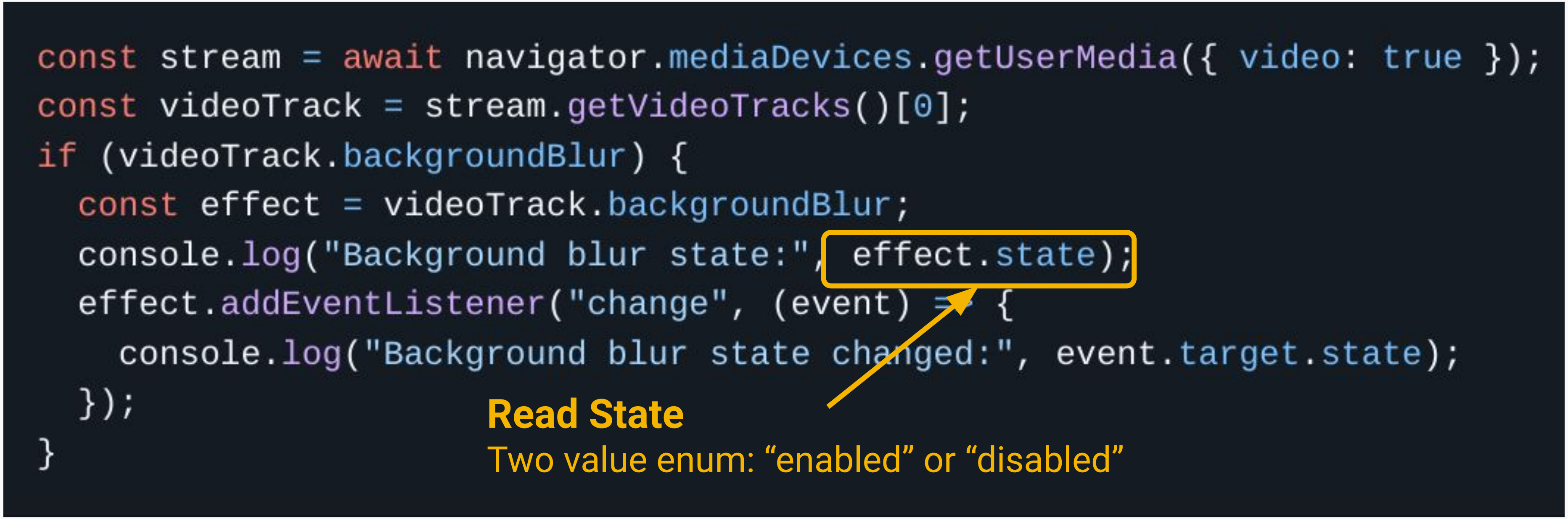
Feature Detection



```
const stream = await navigator.mediaDevices.getUserMedia({ video: true });
const videoTrack = stream.getVideoTracks()[0];
if (videoTrack.backgroundBlur) {
  const effect = videoTrack.backgroundBlur;
  console.log("Background blur state:", effect.state);
  effect.addEventListener("change", (event) => {
    console.log("Background blur state changed:", event.target.state);
  });
}
```

Feature Detection

Observing State (using the track)

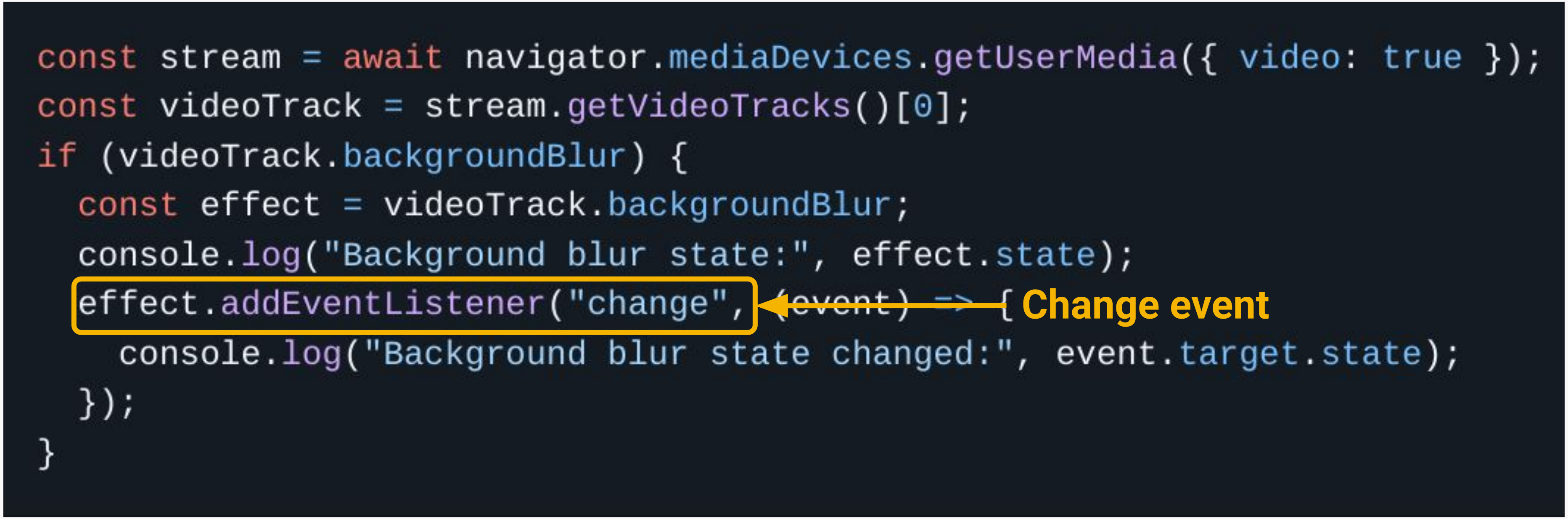


```
const stream = await navigator.mediaDevices.getUserMedia({ video: true });
const videoTrack = stream.getVideoTracks()[0];
if (videoTrack.backgroundBlur) {
  const effect = videoTrack.backgroundBlur;
  console.log("Background blur state:", effect.state);
  effect.addEventListener("change", (event) => {
    console.log("Background blur state changed:", event.target.state);
  });
}
```

Read State

Two value enum: “enabled” or “disabled”

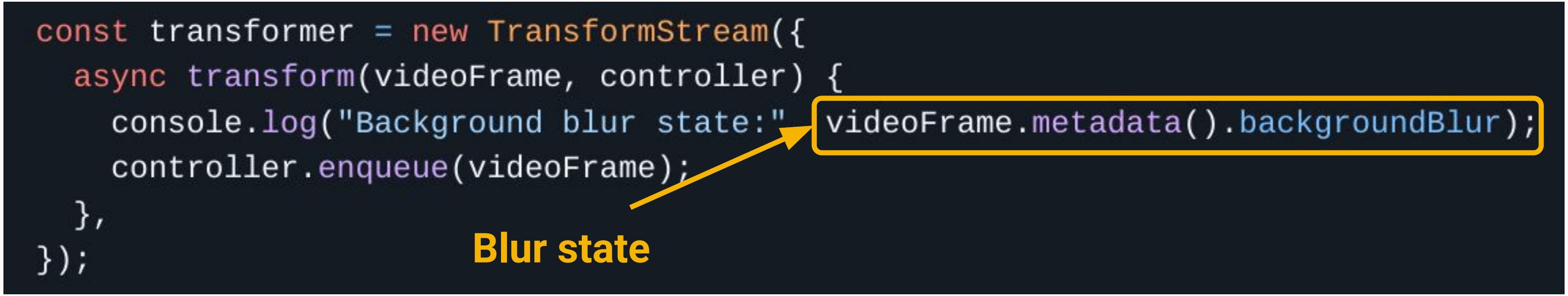
Observing State Changes



```
const stream = await navigator.mediaDevices.getUserMedia({ video: true });
const videoTrack = stream.getVideoTracks()[0];
if (videoTrack.backgroundBlur) {
  const effect = videoTrack.backgroundBlur;
  console.log("Background blur state:", effect.state);
  effect.addEventListener("change", (event) => {
    console.log("Background blur state changed:", event.target.state);
  });
}
```

Change event

Observing State (on each frame)



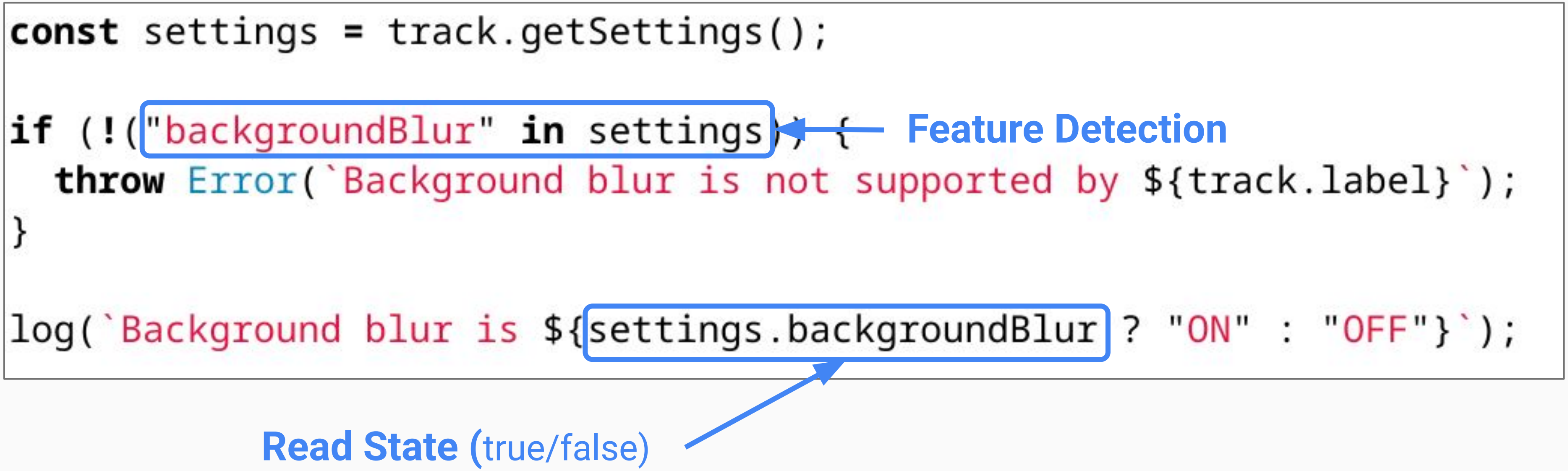
```
const transformer = new TransformStream({
  async transform(videoFrame, controller) {
    console.log("Background blur state:"  videoFrame.metadata().backgroundBlur);
    controller.enqueue(videoFrame);
  },
});
```

Blur state

Constraint based approach

Code snippets from:
https://googlechrome.github.io/samples/image-capture/background-blur.html

Feature Detection & Observing State

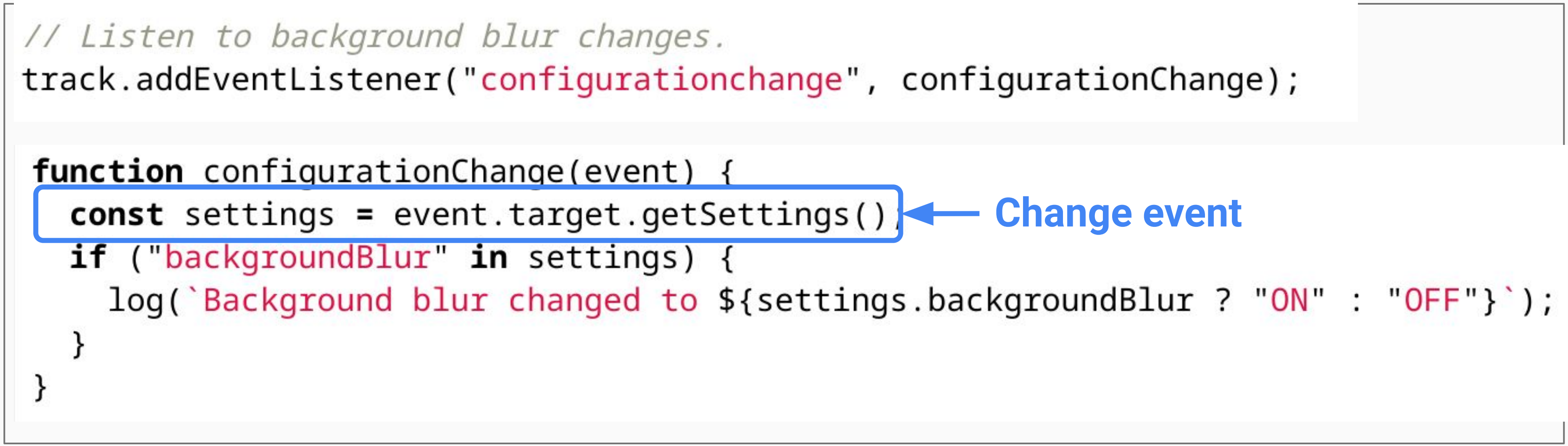


```
const settings = track.getSettings();

if (!("backgroundBlur" in settings)) {
  throw Error(`Background blur is not supported by ${track.label}`);
}

log(`Background blur is ${settings.backgroundBlur ? "ON" : "OFF"}`);
```

Feature Detection

Read State (true/false)

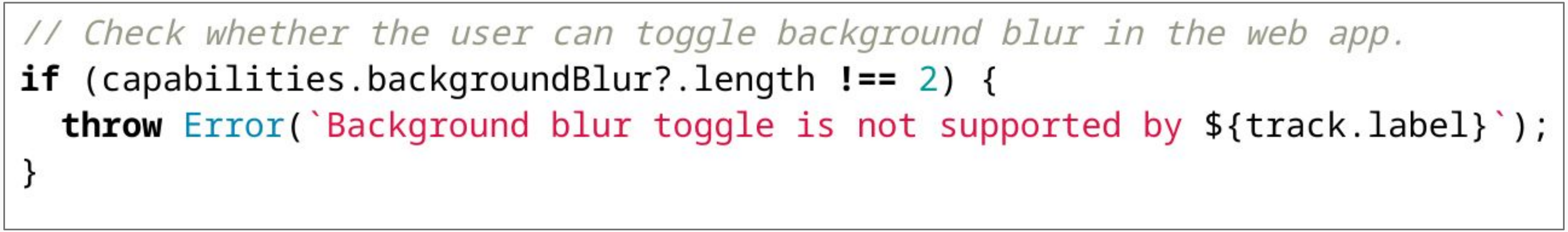
Observing State Changes



```
// Listen to background blur changes.
track.addEventListener("configurationchange", configurationChange);
```

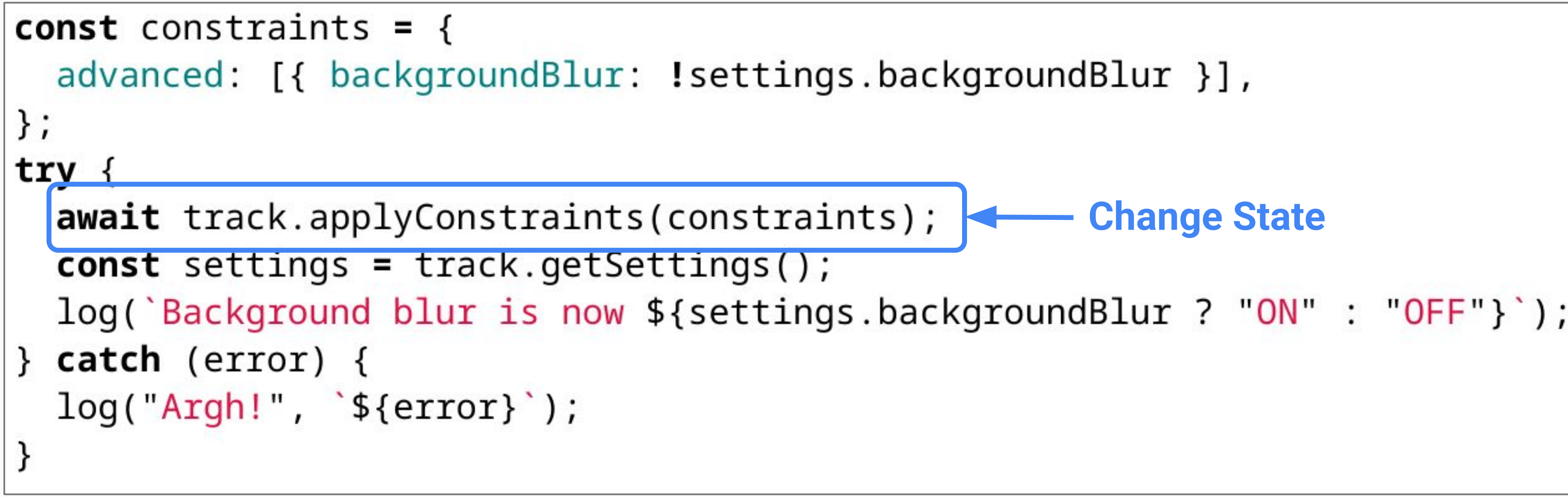
```
function configurationChange(event) {
  const settings = event.target.getSettings();
  if ("backgroundBlur" in settings) {
    log(`Background blur changed to ${settings.backgroundBlur ? "ON" : "OFF"}`);
  }
}
```

Detecting Capabilities



```
// Check whether the user can toggle background blur in the web app.
if (capabilities.backgroundBlur?.length !== 2) {
  throw Error(`Background blur toggle is not supported by ${track.label}`);
}
```

Direct Control



```
const constraints = {
  advanced: [{ backgroundBlur: !settings.backgroundBlur }],
};
try {
  await track.applyConstraints(constraints);
  const settings = track.getSettings();
  log(`Background blur is now ${settings.backgroundBlur ? "ON" : "OFF"}`);
} catch (error) {
  log("Argh!", `${error}`);
}
```

Change State

Comparison of approaches

	Property	Event	Per-frame status	Capabilities	Ask the User	Direct Control
Constraints	✅	✅		✅	X	✅
Proposal	✅	✅	✅			

X - Not compatible

✅ - Supported

Empty - Possible extension

Next Steps & Discussion

Can we combine the best of both?

- “Enabled”, “Disabled” enums instead of booleans?
- Improve eventing when stream settings change?
- Add a Promise-based API to allow “ask the user” scenarios
- Add effect status to VideoFrameMetadata
- Continue to pursue exposing intermediate data (face landmarks, segmentation)

Open & Future Questions

- Per-stream effects support throughout the stack (browser and OS)
- Access to effects-free streams
-