API Tutorial

How to display device permission errors in a custom video or audio call app built with Daily

Adding a device permissions prompt using the Daily SvelteKit app as an example



Jess Mitchell
Published: 17 Mar 2022 (Updated: 17 Mar 2022) • 6 min read

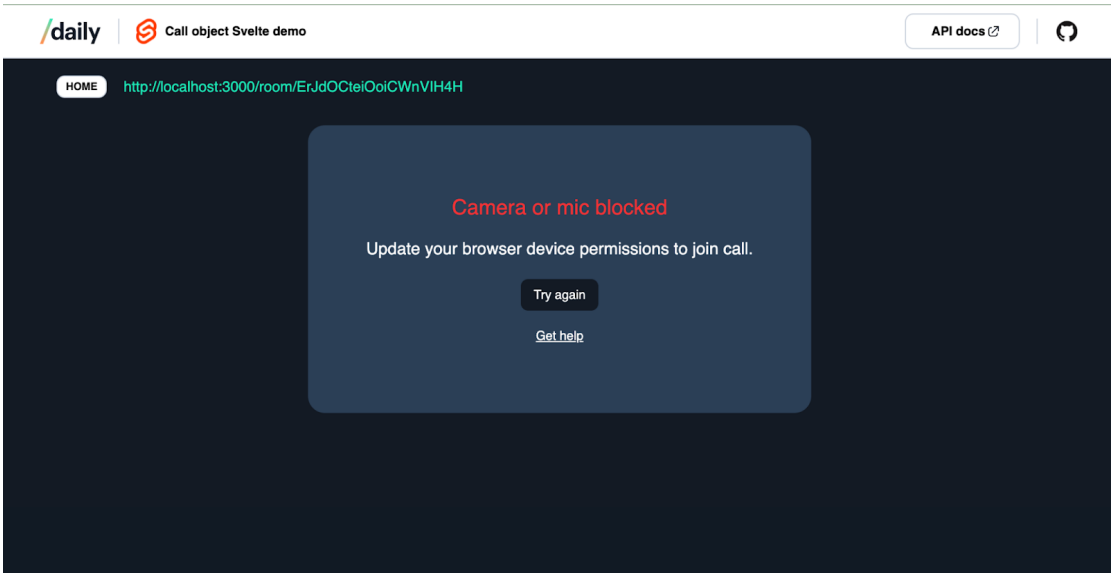


> *This is post four in a tutorial series on how to build a custom Daily app with SvelteKit.*

At Daily, one of the topics that comes up often while speaking to customers is the best way to handle device permissions in video and

audio-only calls. Daily has built-in tools to help you know when the browser is trying to access devices – like your camera or microphone – but it isn't always clear how to pass that message on to the user.

In today's tutorial, we'll look at how to know when there's a device error in a custom Daily app. We'll use the SvelteKit demo app we've been building as an example for how to listen for device errors. We'll then display an error message in the app for users to know how to proceed.



Demo app UI for device permission error

This tutorial references daily-js events that are compatible with any front-end framework, so read on even if you're not building with Svelte. If you are a Svelte developer, and you would like to catch up on how we built our app up until this point, you can check out our previous posts:

- Use SvelteKit's filesystem-based routing to create new Daily rooms on the fly
- Add video tile components to your SvelteKit app with Daily
- Build a custom text chat sidebar in a Svelte video app with Daily

This tutorial can also be read as a standalone post. Like any good buffet, take what you need! 🍕

Setting up our local environment

To run this project locally, first fork Daily's SvelteKit demo repo.

Next, you will need to rename `env.example` to `.env.local` and add your Daily API key and Daily domain to it. (Visit the project's README for more information on where to retrieve these values.)

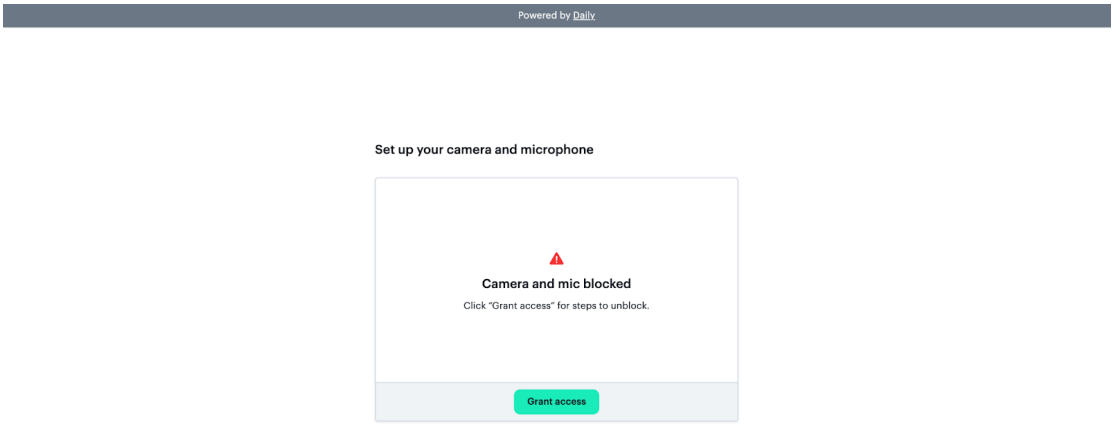
Lastly, from the project's root directory, run:

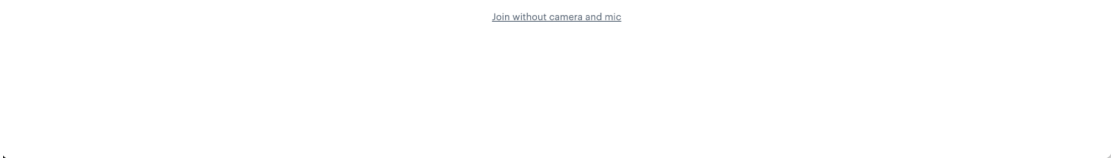
```
npm install
npm run dev
```

To view the app, open `http://localhost:3000` in the browser of your choice.

Listening for device errors with Daily

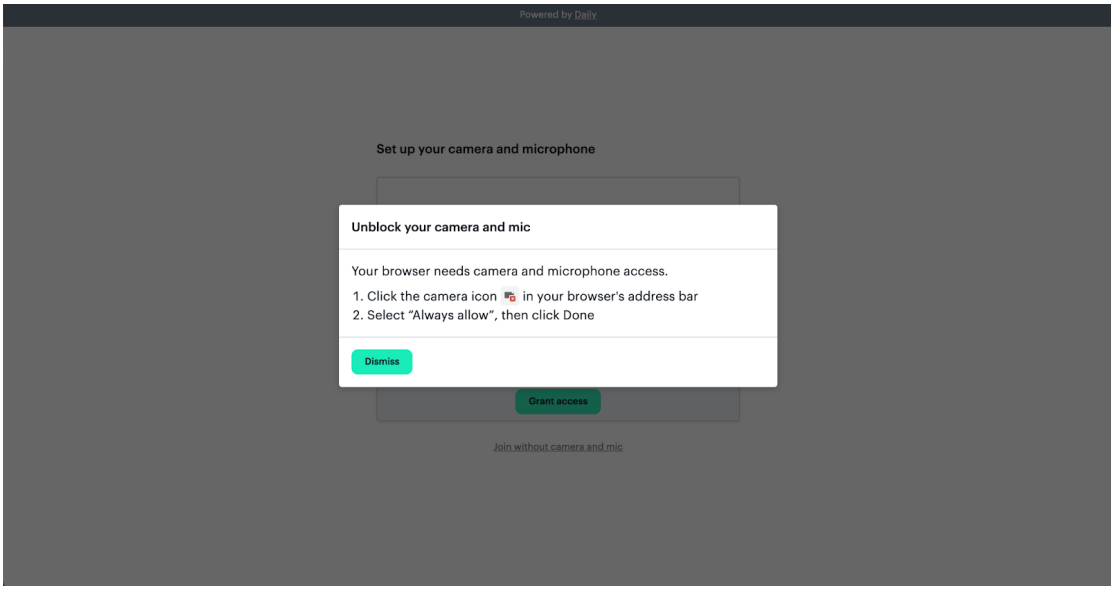
If you've ever joined a Daily Prebuilt call, you'll know we do what we can to help users join with any devices they want. Sometimes, however, device permissions are blocked in the browser or another app is already using that device. In cases like these, Daily isn't able to immediately let you turn your devices on for the call until you make updates on your end (in the browser or by closing other apps).





Daily Prebuilt device permission error in the pre-join UI

In the case of Daily Prebuilt, we've already built these device warnings in for you so your users will know how to join the call.



Daily Prebuilt additional information on unblocking device access

When it comes to a custom Daily app, we can add similar warnings and information but we need to build it ourselves.

To do this, we can rely on Daily's `'camera-error'` event. The `'camera-error'` event will be triggered if there are any issues accessing the cameras or microphones available. Admittedly, it's a bit of a misnomer in that sense, since it reports any ***device*** access errors.

These device errors include:

- The device is already in use by another app (like by one that rhymes with "Room")
- Permissions have been denied in the browser, which users can update to resolve themselves
- There are no devices found, which is one way to learn that the new webcam you just bought doesn't actually have a microphone built-in 😅

To learn more about these specific errors and device permissions in general, read our guide on the topic.

Using `'camera-error'` in our Svelte app

Now that we know what to look for, let's put it into action.

In our `[roomId].svelte` file, we instantiate our Daily call object to be used for building our in-call UI. We also add all our Daily event listeners, including one for `'camera-error'`.

```
<script>
…
const createAndJoinCall = async () => {
  …
  callObject = daily.createCallObject({
      url,
      userName: $username,

  });

  ...
  callObject
     .on('camera-error', handleDeviceError)
  …
}
…
</script>
```

[roomId].svelte

Here, we add our `'camera-error'` listener with the callback `handleDeviceError`.

```
<script>
…
let deviceError = false;
…

const handleDeviceError = () => {
   deviceError = true;
 };
```

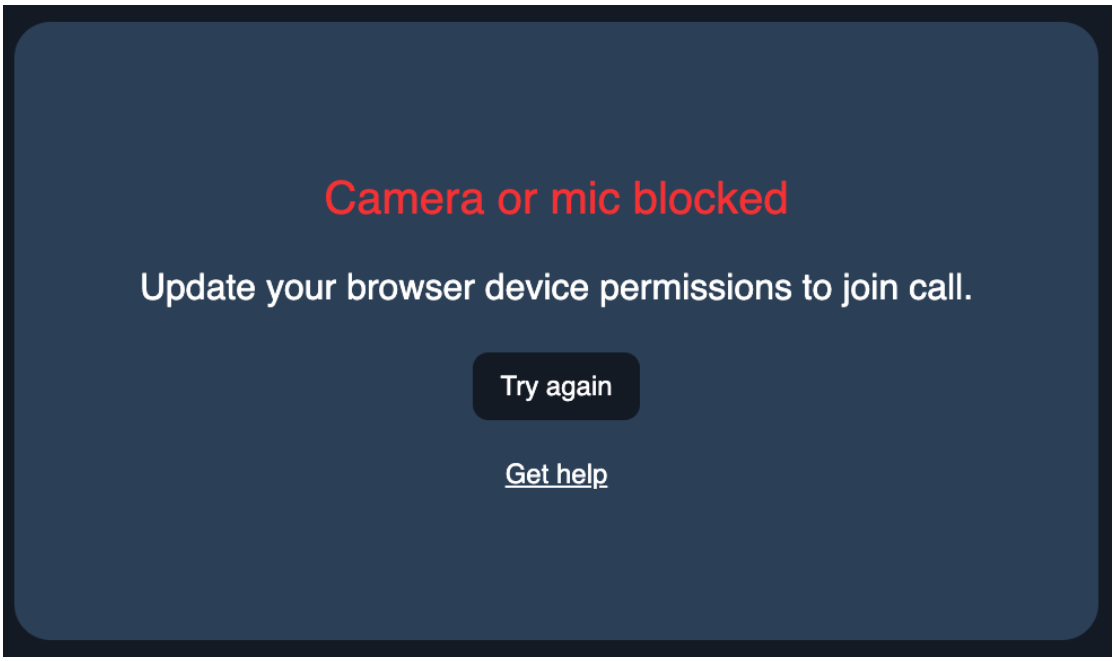
```
…
</script>
```

[roomId].svelte

`handleDeviceError` will get called whenever `daily-js` alerts us of an error related to the participant devices. Therefore, when it gets called, we can toggle our `deviceError` value from `false` to `true` because we know there is an error.

In this specific demo we don't go into further detail about the ***type*** of device error, but you definitely can. Daily Prebuilt, for example, handles a wide variety of device permission errors, including browser-specific ones, and it's also built with Daily's call object. How detailed you get is really up to you.

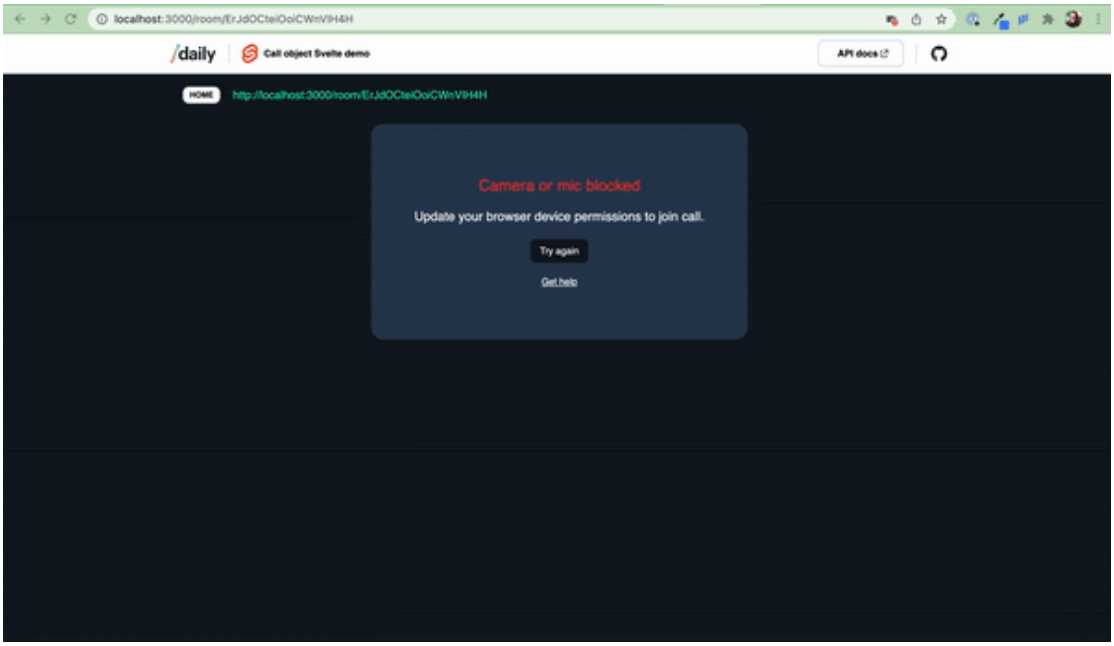
In this specific app, we show a card in the UI that lets users know there's a device error. We also link to a Daily help doc that steps through how to unblock your devices in the browser. This level of information resolves the majority of device permission errors, so it meets the scope for our demo's purposes.



Svelte demo app devices block UI

Once the call participant updates their browser permissions, they can click "Try again" which will clear the error and bring them back to the

home page to rejoin a call. The participant can also refresh the page, which some browsers will prompt them to do anyway after updating browser settings.



Unblocking devices in Chrome

How we show this card in the UI is fairly straightforward. Since we have the `deviceError` variable – a boolean value that's `true` when there's an error – we can conditionally render a Svelte component whenever there is an error. In this case, we want to conditionally render the `PermissionErrorMessage` component, which provides our user with additional information.

```
{#if loading}
 <div class="loading">
   <Loading />
 </div>
{:else if deviceError}
 <PermissionErrorMessage
   on:clear-device-error={clearDeviceError} />
```

```
{:else}
…
```

[roomId].svelte

In this code block, we show our loading animation (`<Loading />`) if our `loading` variable is `true` and we show our `PermissionsErrorMessage` component when `deviceError` is `true` . Otherwise, we just render our regular in-call UI, like our participant video tiles.

Note: Since these are abridged code snippets, check out the GitHub repo for the broader context of variables like `loading` .

The `PermissionErrorMessage` component also contains a custom Svelte event listener for the `clear-device-error` event.

```
<PermissionErrorMessage
    on:clear-device-error={clearDeviceError} />
```

When triggered, this event will call `clearDeviceError` :

```
const clearDeviceError = () => {
  deviceError = false;
  goHome();
};
```

[roomId].svelte

This function will clear the error state and send the user back home to rejoin the call. (Alternatively, they can just refresh the page, too. We try to avoid deadends in the UI, so we've provided a button to achieve the same effect as refreshing.)

Displaying device error UI

Let's quickly look at `PermissionErrorMessage` to see how this custom `clear-device-error` event is triggered.

```
<script>
 import { createEventDispatcher } from 'svelte';

 const dispatch = createEventDispatcher();

 const tryAgain = () => {
   dispatch('clear-device-error');
 };
</script>

<div class="wrapper">
 <div class="permissions-card">
   <h2>Camera or mic blocked</h2>
   <p>Update your browser device permissions to join call.</p>
   <button on:click={tryAgain}>Try again</button>
   <a
     href="https://help.daily.co/en/articles/2528184-unblock-cam
     rel="noreferrer noopener"
     target="_blank"
   >
     Get help
   </a>
 </div>
</div>
```

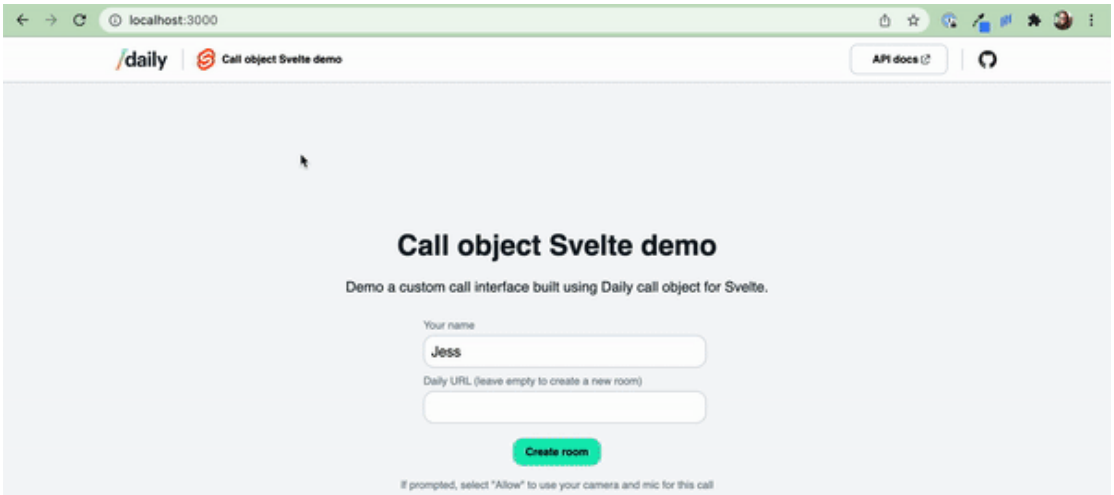
PermissionErrorMessage.svelte

Here, we have a card with some text and a link to the support doc mentioned before. We also have a “Try again” button that calls the `tryAgain` function on click.

`tryAgain` then takes advantage of Svelte’s ability to dispatch custom events – in this case `'clear-device-error'`.

Once it’s dispatched, our parent component `[roomId].svelte` will listen for that event, since we’ve already added the `clear-device-error` event listener on our instance of the `PermissionErrorMessage` component (described above).

And that’s really it!



Wrapping up

We hope this post helps you improve your own custom Daily app's user experience when handling device errors.

Displaying and clearing the device permissions error

For more information, be sure to read our guide on handling device errors as well as our help center article.

This post concludes our custom Daily SvelteKit demo series. Thanks for joining us for the ride and please let us know if there are any topics you would like us to cover. 🏄

Get updates from Daily

Don’t miss a thing! Subscribe to our monthly newsletter and we’ll keep you up-to-date with the latest product releases, tutorials, and more from Daily.

Email*

Country*

Join peerConnection - our video/WebRTC developer community.

By clicking submit below, you consent to allow Daily.co to store and process the personal information submitted above to provide you the content requested.

Submit

More in API Tutorial

Implementing cursor sharing with Daily's video call API

24 Oct 2022 - 11 min read

Add hand-raising to your custom video app with the Daily user data API

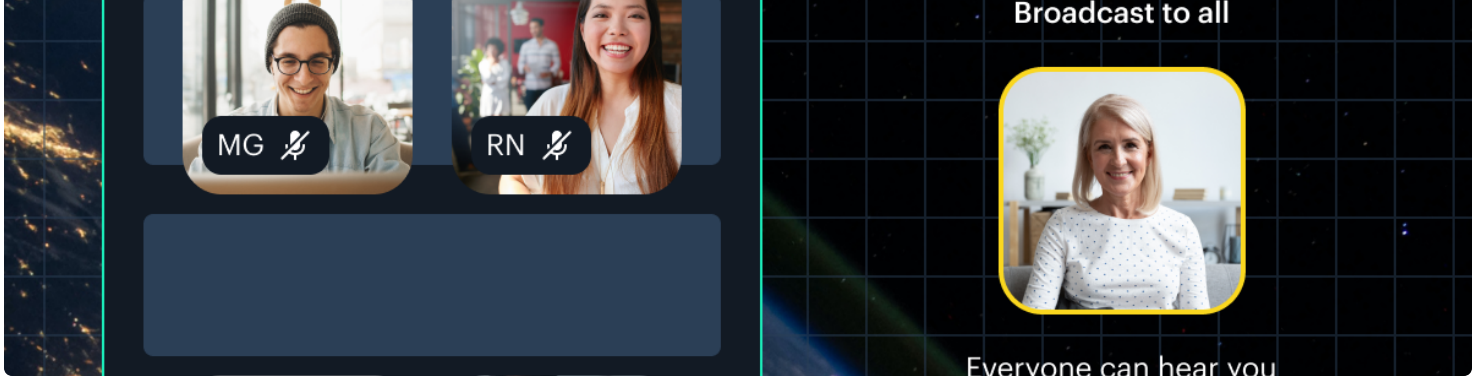
12 Sep 2022 - 12 min read

Stream with HLS while recording to a custom S3 bucket using Daily

30 Aug 2022 - 5 min read

See all 82 posts →





Spatialization

Adding screen sharing support to our spatialization demo (Part 6)

Tips and tricks for screen sharing with Daily



Liza Shulyayeva
21 Mar 2022 – 14 min read



API Tutorial

Build a custom video chat using Daily and web components

Add video calling to your app with only native HTML custom elements and plain JS



Ashley Blewer
14 Mar 2022 – 9 min read