Collab Chrome Extension

Tutorial: Build a custom Daily video call widget and embed it into any Notion page

How we added video and audio-only calls to the Daily Collab Chrome extension demo using Daily's custom APIs



Jess Mitchell
Published: 26 Aug 2021 (Updated: 14 Jan 2022) • 11 min read



> *[14/01/22] This demo app uses manifest v2. Manifest v2 is being deprecated by Chrome in 2022 and will no longer be accepted into the Chrome store. The Daily code below is still valid, however.*

This tutorial is part of a series on how we built Daily Collab, our latest Chrome extension demo.

Daily built the Daily Collab Chrome extension demo to showcase a new example of how quickly you can jump into a video call from any webpage. With Daily Collab, you can start an audio or video call from any Notion page and transcribe the audio from the call directly into that page.

In this tutorial, we'll cover how to embed a Daily call directly into a webpage via a Chrome extension. We won't go into the details of building the React video app itself since there are already several tutorials for Daily's customizable call object mode and Daily Prebuilt. Instead, we'll focus on getting a Daily call to display and update in any Notion page.

Note: We're using Notion as an example but you can embed your Daily video in any webpage with just a couple tweaks to this code.

Getting started

To build Daily Collab, we used React boilerplate specifically set up for a Chrome extension. You can absolutely build this without React and just use plain JavaScript or another framework. In fact, we started with plain JavaScript but switched to React after seeing our designer's plans for this app. (Just kidding, Steve. 💕)

What is true, though, is that there is quite a bit of state management to be aware of related to authorization, which Daily calls are live on which pages, and keeping track of call participants. We decided on React to help simplify handling these state-based UI updates.

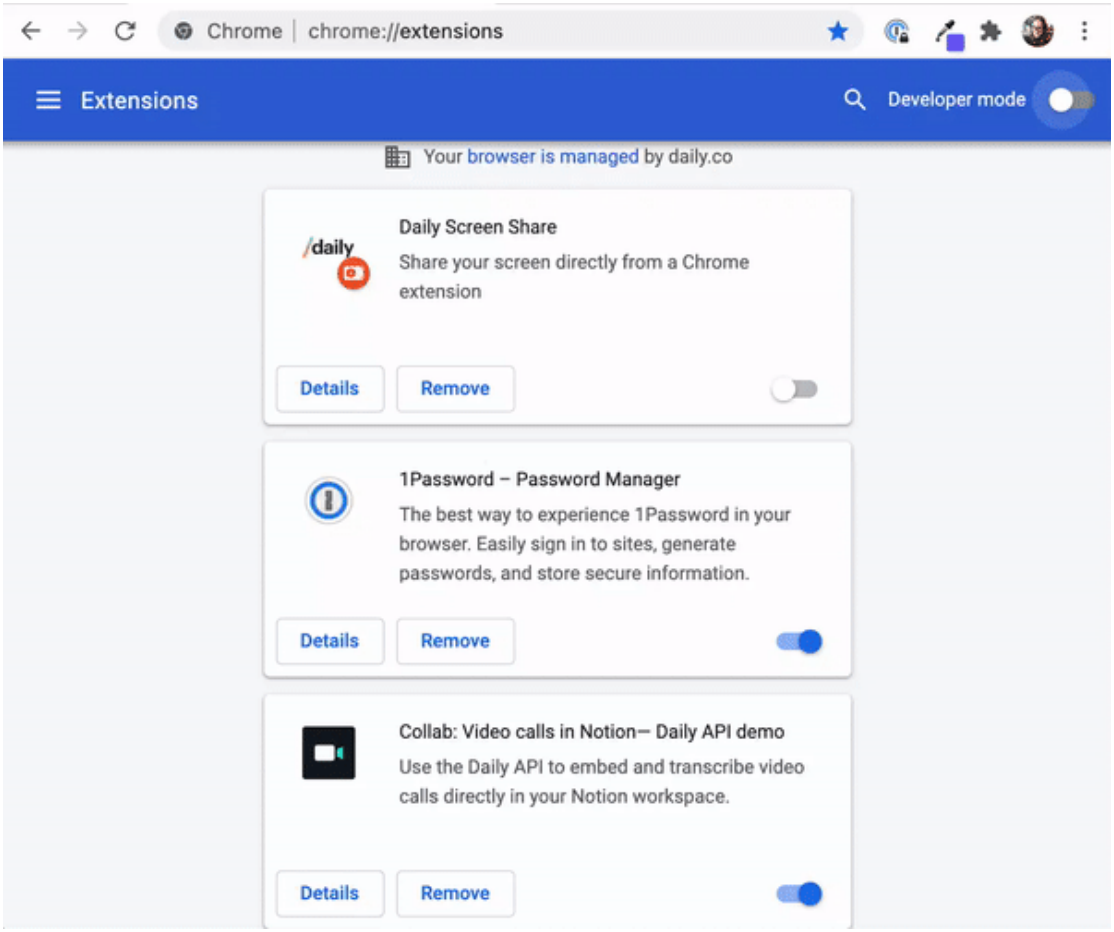
In the next section, we'll step through the Daily Collab code base to explain which parts are relevant to the embedded Daily call.

If you'd like to get a local version of Daily Collab up and running, clone the repo and run the following commands in your terminal:

```
npm install
npm start
```

Then install the Chrome extension by:

1. Visiting `chrome://extensions/`
2. Check `Developer mode`
3. Click on `Load unpacked extension`
4. Select the `build` folder from your local copy of Daily Collab



Add the Daily Collab app locally to your Chrome extensions

If there were no errors, Daily Collab will be available and visible in any Notion page you visit.
Tip: Refresh any Notion pages that were already open prior to installing the Chrome extension to have Daily Collab show on the page.

Injecting a React app into the DOM of another web app

There are lots of details related to specifically getting a Daily call injected into the DOM of your current webpage, but let's start with the basics: adding a new element to the DOM via a Chrome extension. Thankfully, it's pretty straightforward!

With Daily Collab, it's as simple as prepending the content script's HTML to the DOM's `body` element. In the app's root, a `div` element is created to wrap the entire React app included in the content script's code. That `div` is then prepended to the `document.body` on load.

```
class App extends React.Component {
 render() {
   return (
     // Pretend these components are just <div>s for now
     <CallProvider>
       <Main />
     </CallProvider>
   );
 }
}

const app = document.createElement('div');
app.id = 'daily-collab-root';

// Prepend the new <div> to current page's <body> element
document.body.prepend(app);
ReactDOM.render(<App />, app);
```

Content/index.js

Once it's added to the DOM, you can work with the React app it renders like you would any other React app.

Understanding specifically how a Daily video call is embedded in the DOM

If you've looked through the source code for Daily Collab, you may have noticed there's a lot going on in there.

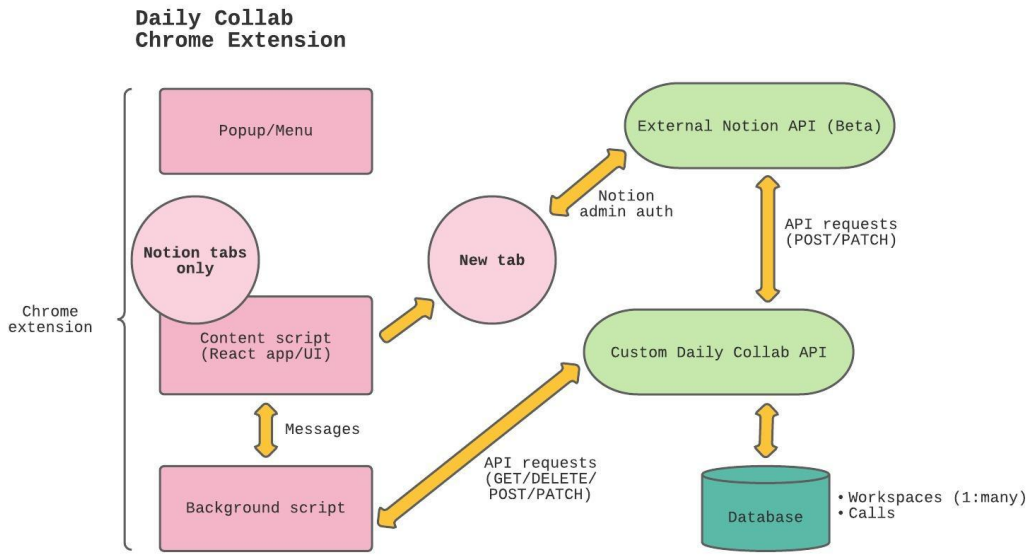
To get a better understanding of how the Daily call gets embedded, let's look at how the pieces all fit together.

Daily Collab overview

Before getting into the nitty gritty, let's cover some basic details about how Daily Collab will work:

- Daily Collab is a Chrome extension that embeds an entire React app via a content script in the DOM of any Notion page. (To review what content scripts and other Chrome extension concepts are in more detail, review our previous post on the anatomy of a Chrome extension.)
- The content script only runs on URLs permitted in the `manifest.json` file. In Daily Collab's case, it's anything that matches `https://www.notion.so/*`, with a couple exclusions.
- The content script communicates with the background script to make external API calls and take other "behind the scenes" actions.
- The background script runs independently from specific browser tabs and is like the brain of the extension. It is active regardless of the content script's URL restrictions.
- To send a message to a specific tab (i.e. content script) we have to keep track of the tab we're trying to communicate with.
- The background script and content script have to send messages to each other to communicate any information.

The anatomy of Daily Collab

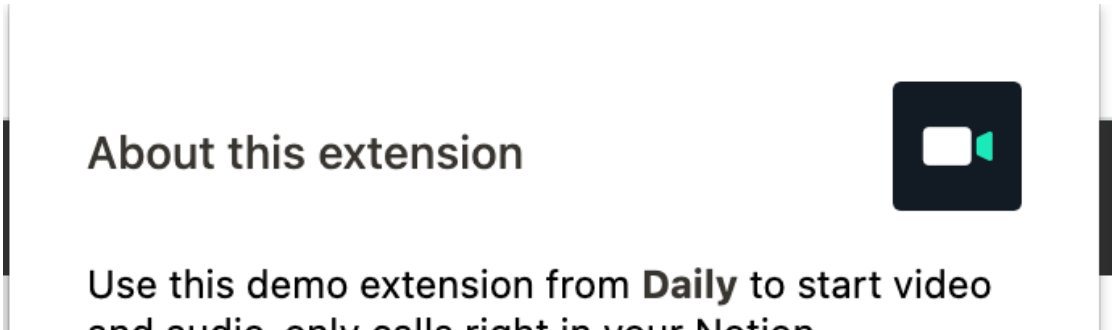


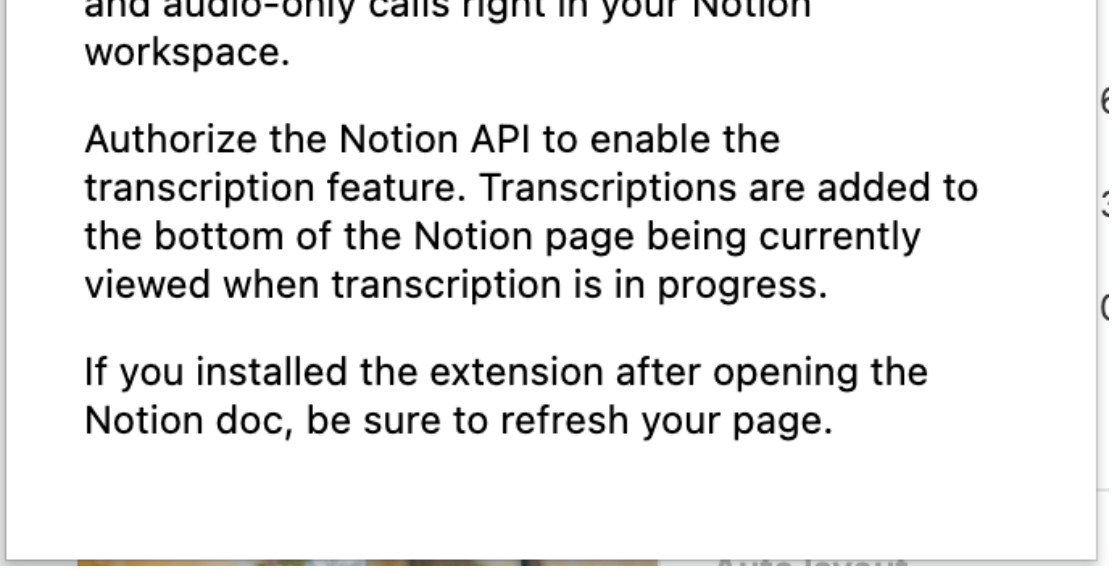
Daily Collab Chrome extension structure

Daily Collab uses a background script*, a content script, two APIs, a custom database, and a Chrome extension popup menu.

**Note: Daily Collab currently uses Manifest V2. In Manifest V3, background scripts are replaced with background service workers. To learn more about why, check out our previous post on Chrome extensions.*

The popup menu provides additional extension information for the user but no additional functionality. That means it doesn't need to communicate with the other areas of the extension.





Daily Collab popup menu

The content script and background script are where Daily Collab gets its functionality. On page load, the content script—which can access the DOM—will load if the page is permitted by the `manifest.json`'s permissions. (See under `content_scripts`.)

In this case, it only loads on Notion URLs.

```
{
 "manifest_version": 2,
 "version": "0.0.0.1",
 "name": "Collab: Video calls in Notion- Daily API demo",
 "description": "Use the Daily API to embed and transcribe video
 "background": { "service_worker": "background.bundle.js" },
 "icons": {
   "128": "icon-128.png"
 },
 "browser_action": {
   "default_popup": "popup.html",
   "default_icon": "icon-34.png"
 },
 "permissions": ["https://daily-notion-api.vercel.app/*", "tabs"
 "content_scripts": [

   {
     "matches": ["https://www.notion.so/*"],
     "exclude_matches": [
       "https://www.notion.so/onboarding",
       "https://www.notion.so/install-integration/*",
       "https://www.notion.so/my-integrations/*"
     ],
     "js": ["contentScript.bundle.js"]
```

```
  }
],
"web_accessible_resources": ["icon-128.png", "icon-34.png"],
"content_security_policy": "script-src 'self' 'unsafe-eval'; ob
}
```

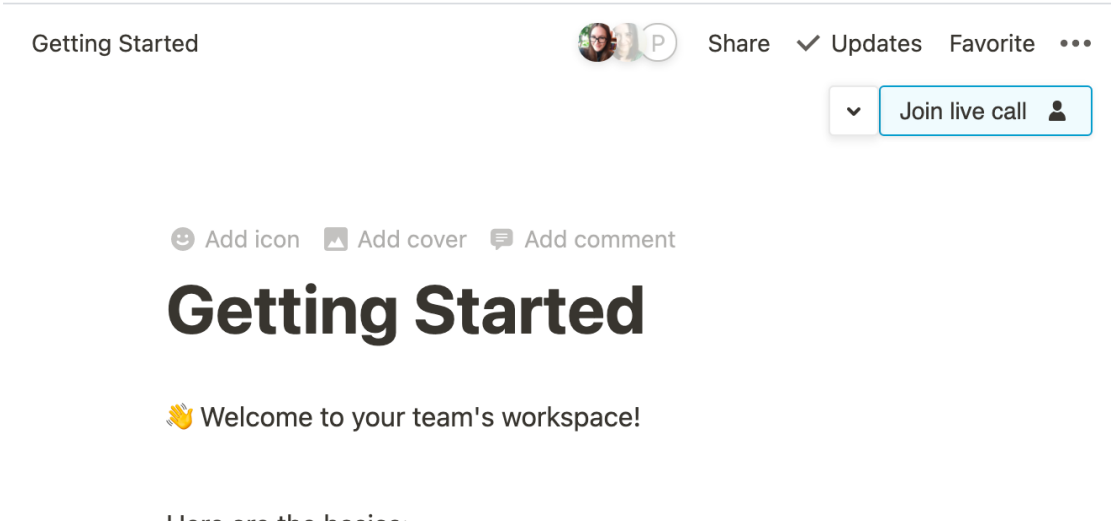
manifest.json

Note: If you were trying to embed the video widget in a different webpage, you could do that by updating the `content_scripts` ***'s*** `matches` ***value.***

The background script also loads, although it's not specifically concerned with Notion; it loads regardless of which page you're viewing.

Once loaded, the background script is available to receive messages from any open tab's content script. Basically, anything in the React app's UI related to a live call will first require the content script to message the background script to get or give a call status update.

This is also how multiple people on different computers can visit the same Notion page and join the same call.



The Join button shows when someone else has already started a call for that page

Hold up: How does the extension know which Daily call to join?

We are so glad you asked! 😉

Daily calls are associated with Daily rooms. (The room is ***where*** you have the call.)

So, if different people visiting the same Notion page need to be added to the same Daily room, how does the background script know which Daily room to tell the content script to use? That's where the custom Daily Collab API comes in!

Daily Collab API: Associating Notion docs with Daily rooms

One of the more complicated aspects of Daily Collab is that it doesn't need to just work for the local user; the state for the local user needs to stay in sync for anyone else viewing the same Notion page.

If someone starts a call on a specific Notion page, anyone else viewing that page with Daily Collab installed needs to know there's a live call and have their UI updated to join that specific Daily room.

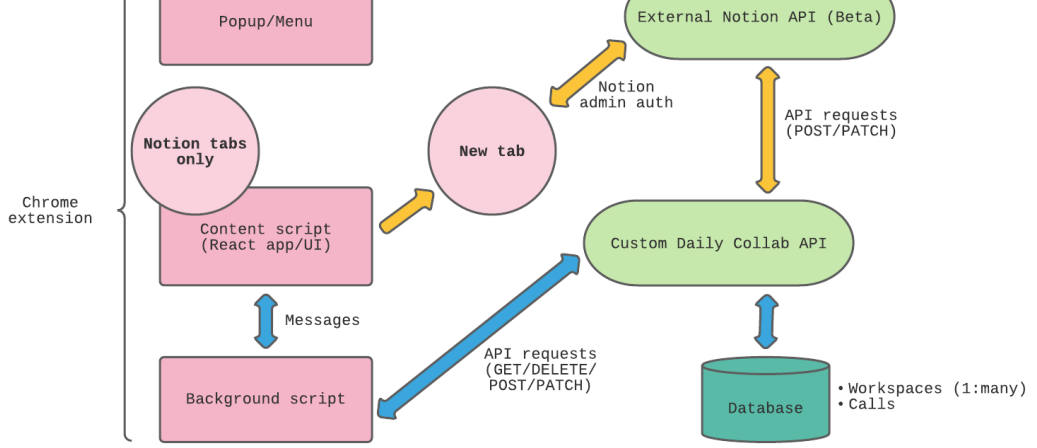
To handle this, we created a custom API that would keep Notion pages associated with the Daily room they were using. The API has a designated database that adds a row to it each time a call is started on a

Notion doc. When the last person leaves the call, the room is deleted from Daily and the item is removed from the database. This is because we only need to know if there is a call currently happening for each Notion page.

Note: The Daily Collab API source code is currently private because it uses an upcoming Daily feature: transcription. We promise we'll share it with you as soon as we can. ❤️

Getting the right Daily call with the Daily Collab API





The route for the React app to know which Notion pages have live Daily calls, information which is stored in the database

Now that we know we have a custom solution for associating Notion pages with Daily calls, let's look at this diagram again to see how these pieces work together:

- The React app (content script) polls every second for information on the current Notion page
- The background script makes a GET request to our custom API.
- The endpoint then checks the Daily Collab database to see if there's a row for that Notion page.
- If there is, the GET request returns a status `200` to the background script with the item in the response body.
- If there isn't, the GET request returns a `404` (not found) to the background script.

Here's an example of the response body returned when a call is live on the current Notion page:

```
{
   audioOnly: false
   createdDate: "2021-07-06T16:40:07.817Z",
   dailyId: "fd5165ac-9565-4e77-aed9-ffbb3bfd0294",
   // this is the Daily room
   dailyUrl: "https://example.daily.co/5L6lJkXl1fK4gP2nqwert",
   docUrl: "https://www.notion.so/dailyco/Getting-Started-8e4907
   id: 178,
   isTranscribing: false,
   notionId: "Getting-Started-8e4907f0171248f1b137f5598bdqwert",
   useTranscription: false,
```

```
  workspaceId: 7,
}
```

Notion page item from Daily Collab database

This response lets us know if the call is audio-only or video, the Notion page's exact URL, if the call currently has transcription available and in use, and, most importantly: the Daily room URL.

From there, the background script can send a message to the content script with all of that information so the UI can update as needed.

```
chrome.tabs.sendMessage(
  tab.id,
  {
    dailyUrl,
    audioOnly: room.audioOnly,

    isTranscribing: room.isTranscribing,
    useTranscription: room.useTranscription,
    docUrl: room.docUrl,
  },
  () => {}
);
```

Background/index.js

Since this check happens on a one second interval, the UI is able to always stay up-to-date.

Receiving the message from the content script

For the message from the background script to have any impact on the Chrome extension's UI, it needs to be explicitly received by the content script.

Since the content script is using a React app, Daily Collab can use the React context API to manage the app's state. The received message is handled by the `CallProvider` context, which contains most of the app's state.

A `useEffect` hook is included in `CallProvider.jsx` that adds a listener for any messages received:

```
useEffect(() => {
  if (backgroundListenerAdded) return;
  function receiveDailyUrlInfo(request) {
    // ... handle message
    setBackgroundListenerAdded(true);
  }

  chrome.runtime.onMessage.addListener(receiveDailyUrlInfo);
}, [backgroundListenerAdded, messageReceived, callUrl]);
```

Content/contexts/CallProvider.jsx

Once that listener has been added, it gets triggered each time the background script sends a message. From there, there's a callback function to read the message and determine how to update the app's state based on what the background script has sent.

For example, if a `dailyUrl` `key` is included in the message, we know it's a change related to the Daily room. That means we need to update the app's state to show that a room is available to join, or reset the UI if the previous room is no longer available to join.

```
// Set Daily room info provided by Daily Collab API via the back
// If there's no Daily room info, assume a call is not live and
if (keys.includes('dailyUrl') && request?.dailyUrl) {
  setDailyUrl(request.dailyUrl);
  setCallType(request.audioOnly ? 'audio' : 'video');
  setIsTranscribing(request.isTranscribing);
  setUseTranscription(request.useTranscription);
  setCallUrl(request.docUrl);
} else if (keys.includes('dailyUrl') && !request?.dailyUrl) {
  resetCallState();
}
```

Content/contents/CallProvider.jsx

In the `Main` component, we can then render a "Join" or "Start live call" button based on whether a Daily room— `dailyUrl` — is set in the state.

```
return (
  <div className="authorized-container">
    <Authorization />
    {dailyUrl ? (
      <JoinCallButton buttonLoading={buttonLoading} />
    ) : (
      <StartCallButton />
    )}
    <style jsx>{`
      .authorized-container {
        display: flex;
      }
    `}</style>
  </div>
);
```

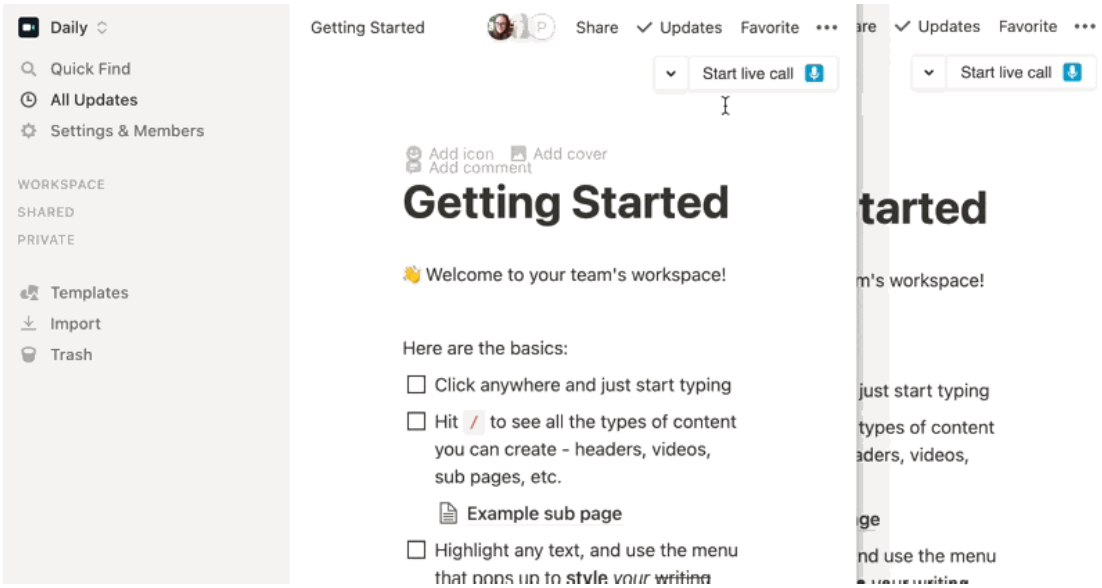
Content/components/Main.jsx

Understanding how UI updates work from the user's perspective

Now that we've gone through the code that updates the React app's UI, let's put this all together with a quick example. If two people are looking at the same Notion page and want to join a call together, the following steps will happen:

- `Person A` will press the "Start live call" button, which will trigger a message from the content script to the background script that a call was started. This message will trigger the background script to send a POST request to the Daily Collab API to create a new call item in the database for that specific Notion page.
- `Person B`, who is viewing that same Notion page, wants to join the same call as `Person A`. Since there is one second polling set up for the React app to know if a live call is available, `Person B`'s content script will send a message to the background script every second to check the database for a status update.
- Now, since `Person A`'s request to start a call created an item in the database, when `Person B` polls for an update, the background script will get that information from the database as soon as it's available. The background script then sends a message back to `Person B`'s content script with call details.

- Once that message is received by the message listener in the React context provider, `Person B` 's UI will update because the React app state now knows a call is available to join. The "Start live call" button will become a "Join live call" button.



Multiple tabs of the same Notion page staying in sync for any live calls with a Chrome extension alarm

And tada! 🎉 Both people can join and have their UI stay in sync. It's a journey, but it works!

Wrapping up

Building a Chrome extension that needs to embed a video call and stay in sync with other people certainly requires a lot of moving parts, and this was just one of them. To learn more about the Daily Collab Chrome extension, stay tuned for our upcoming tutorial on Daily Collab's transcription feature using the Notion API.

transcription feature using the Notion API.

In the meantime, check out our previous (and less involved) Chrome extension demos, including:

- How to update a Daily call's UI
- How to share your screen instantly with Daily, which included using the Chrome extension's pop up menu

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 Collab Chrome Extension

Build your own "prejoin" call UI in a custom Daily video chat app

9 Sep 2021 - 9 min read

Tutorial: Append Daily call transcription text with the Notion REST API

2 Sep 2021 - 8 min read

Authorizing Daily's Chrome extension transcription feature with Notion's API

19 Aug 2021 - 8 min read

See all 5 posts →



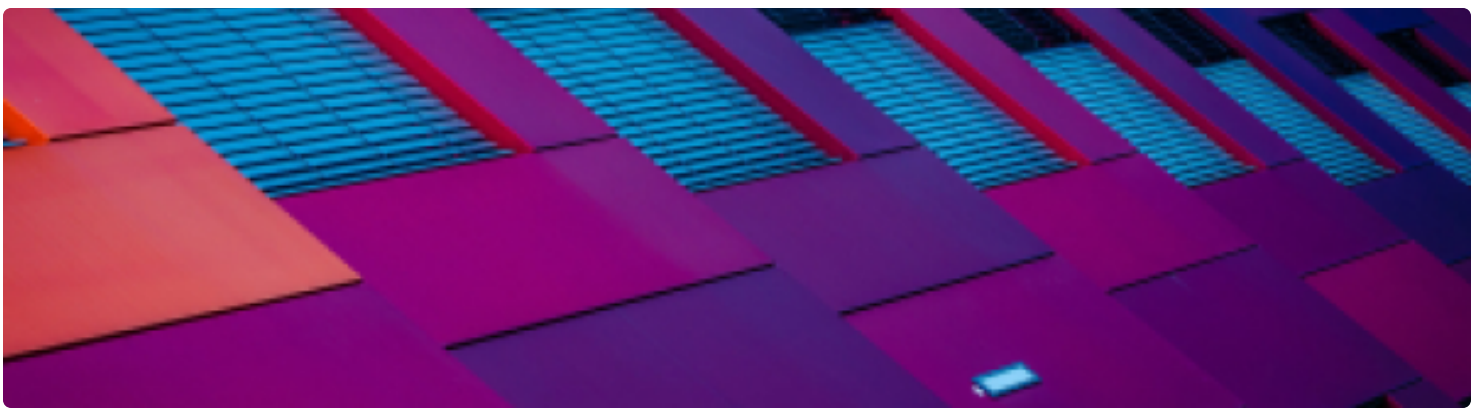
Code & Tutorials

Build a video chat app with Vue and Daily Prebuilt

Embed Daily Prebuilt and add custom video call controls to any Vue app



Jess Mitchell
30 Aug 2021 – 12 min read



Code & Tutorials

Using Docker and userspace networking to simulate real-world networks

Introducing Synthetic Network, Daily's open source tool for testing applications with configurable network conditions



Vanessa Pyne, Max Rottenkolber
23 Aug 2021 – 4 min read

Daily API: Developer Tips to Build Video and Audio Chat © 2022