API Tutorial

Introducing the new Daily React Hooks library

Speed up development time, simplify code, and optimize performance in React-based real-time audio and video apps



Jess Mitchell, Francine Boulanger

Published: 7 Mar 2022 (Updated: 25 Oct 2022) • 9 min read



UPDATE: Daily now supports 25 active participants with 15,000 viewers and 1,000 person large calls, all with cams and mics on.

> *If you're new to Daily and want to learn how build a custom video app, check out our tutorial on building a video app with Daily and React.*

Building real-time video and audio applications encompasses a lot of complex state management and event handling: Calls can host between

two and 15,000 participants, each with a different set of parameters that need to be tracked and managed. Participants will come and go, toggle their microphone on and off, and switch devices. This means a lot of information to process and handle, which can lead to complex code. As well as leading to longer development cycles, this complexity can make optimizing application performance more difficult.

That's why we're excited to announce Daily React Hooks, our latest React library. Daily React Hooks can help to optimize performance and simplify many of the complexities that arise from managing state and handling events in React real-time video and audio apps. By building with Daily React Hooks, developers can focus more on customizing and developing their app, reducing time spent on client optimization, state synchronization, and event handling.

This library leverages React's own internal state management tools and connects to Daily's JavaScript API, simplifying code so that it's easier to debug and maintain. For this reason, development teams already using Daily with React or React-compatible development frameworks like Next.js can benefit by converting their code. You can do this all at once, or over time—it's really up to you. There's no obligation if you're already happy with your React codebase, or if you don't want to create another JIRA ticket for your team, but we'll walk through the process below so you can see how easy it is.

Whether you're just starting out with Daily or already use our APIs in production, we recommend that all React developers building with Daily consider using Daily React Hooks for the best possible performance. We've tested these hooks extensively and even use them internally for Daily Prebuilt, our plug-and-play video interface. They've not only simplified our own code internally, but also fixed bugs and made our code much easier to read. 💪

We'd love to hear your experience using Daily React Hooks: if you have any feedback for us, let us know at help@daily.co.

Ready to convert your code? Read on!

What exactly are Daily React Hooks?

Daily React Hooks are a series of custom React Hooks that connect to Daily's JavaScript API. They offer significant advantages to React developers who want to integrate real-time video and audio into their app, or who are already using Daily's JavaScript API. Daily React Hooks:

- Simplify code, making it faster to integrate and easier to maintain
- Reduce component re-renders, significantly improving overall app performance
- Abstract away some of the complexity of error handling and state management, so that developers can focus on building engaging apps

---

Integrating Daily React Hooks into your React app

To use Daily Hooks, you'll first need to install the `@daily-co/daily-react-hooks` library in your React project, as well as `@daily-co/daily-js` and `recoil` (two dependencies):

```
npm install @daily-co/daily-react-hooks @daily-co/daily-js recoi

# or with yarn

yarn add @daily-co/daily-react-hooks @daily-co/daily-js recoil
```

Installation instructions for Daily React Hooks

Once it has been installed, you can import the different custom Daily hooks into your app components like so:

```
import { useDailyEvent } from '@daily-co/daily-react-hooks';
```

Wrapping your app components in the `DailyProvider` component

To get started, let's look at what is typically the first step in starting a custom Daily call: creating our call object instance. This is done via the Daily `createCallObject` factory method, and often called right before a call is joined by the local user.

The call object needs to be made available to `daily-react-hooks` so that it can be used to interact with Daily's JavaScript library, `daily-js`. Once the call object is created, we can pass the call object instance to Daily React Hooks provider component, `DailyProvider`.

Note: There is a second option for the `DailyProvider` component — instead of passing the call object itself, you can pass call object props to `DailyProvider`, which `DailyProvider` will then use to create the call object for you. We'll focus on the first option here, though.

```
import DailyIframe from '@daily-co/daily-js';
import { DailyProvider } from '@daily-co/daily-react-hooks';
import React, { useState, useEffect } from 'react';

const DAILY_URL = 'https://DOMAIN.daily.co/ROOM-NAME';

export default function App() {
  const [callObject, setCallObject] = useState(null);


  useEffect(() => {
    if (!DailyIframe) return;

    const newCallObject = DailyIframe.createCallObject();
    setCallObject(newCallObject);
    newCallObject.join({ url: DAILY_URL });
  }, []);


  // Pass the call object as a prop to the DailyProvider
  // and render the app UI that needs access to
  // Daily React Hooks as child components
  return (
     <DailyProvider callObject={callObject}>
       {children}
```

```
      </DailyProvider>
  )
}
```

Example of using DailyProvider component

Once the call object is linked to the `DailyProvider` component, the call object can be accessed by any child components in your app.

Leveraging the `useDaily` hook

Next, let's take a look at how to actually access the call object in a child component.

Accessing your call object instance via Daily React Hooks is now as simple as importing the `useDaily` hook into your component. For example, let's look at this abbreviated `VideoControls` component example:

```
import { useDaily } from '@daily-co/daily-react-hooks';
import React, { useState, useCallback } from 'react';

export const VideoControls = () => {
  const callObject = useDaily();
  const [muted, setMuted] = useState(false);

  const toggleCamera = useCallback(() => {
    callObject.setLocalVideo(muted);
    setMuted(!muted);
  }, [ callObject, muted ]);
 ...
```

useDaily hook example

Here, we import the `useDaily` hook from `daily-react-hooks` and assign it to the variable `callObject` in our `VideoControls` component.

Next, in our `toggleCamera` function – which would be called when you press a button to toggle the local camera's state — we can access the Daily API via the `setLocalVideo` method attached to any call object

instance. When this is called, `daily-js` will toggle the state of the local camera (either on or off).
The main benefit here is not having to worry about passing the call object around as a prop or creating your own React Context. We can also avoid re-renders that might happen if we had our own context managing our call object. `useDaily` handles all of that for you so you can simply import the call object where it's needed.

There is one issue in this code example, though. We're assuming the local participant's camera is unmuted to start when we initialize the `muted` state as `false`. In reality, it could be on or off to start – we can't assume, but we can find out from the local participant object available on the call object.

Additional Daily React Hooks: `useLocalParticipant` example

In addition to storing your call object instance, Daily React Hooks also simplify your app-level logic with a variety of additional hooks related to Daily calls. For example, we can access information related to your participant list in any given call.

Looking at the same example as above, let's update it to use the Daily `useLocalParticipant` hook.

```
import {
  useDaily,
  useLocalParticipant,
} from '@daily-co/daily-react-hooks';
import React, { useMemo, useCallback } from 'react';

export const VideoControls = () => {
  const callObject = useDaily();
  const localParticipant = useLocalParticipant();

  /* This variable could be used to determine
   which camera icon to show (muted or unmuted)
   in your "toggle camera" button */
  const camOn = useMemo(() => {
    return localParticipant?.video
  }, [localParticipant?.video])
```

```
  const toggleCamera = useCallback(() => {
    callObject.setLocalVideo(!camOn);
  }, [callObject, camOn]);

  ...
```

useDaily and useLocalParticipant example

In this block of code, we are retrieving the local participant by importing the `useLocalParticipant` hook and assigning it to the `localParticipant` variable in our component. We can then check the `localParticipant`'s `video` key (a boolean value) to see if the camera is on.

Previously in code not using Daily React Hooks, we would have had to access the call object and call Daily's `participants()` instance method to then retrieve the local participant. Now, we can instead access `localParticipant` directly without having to search for it.

This is not only simpler, it can also help avoid re-renders by having fewer hooks dependencies – especially ones like the call object that change often.

Hooking into Daily events

Another major benefit of using Daily React Hooks is being able to listen for Daily events. Let's use the example of the local participant joining a call, also known as the `'joining-meeting'` event.

If you want to show the local participant a loading animation in between requesting to join the call and actually joining the call, you can listen for the `'joining-meeting'` event.

Then, to know when the local participant has officially joined, you can listen for the `'joined-meeting'` event.

Below, we use the `useDailyEvent` hook to listen for both and update our local state based on which event has been triggered most recently.

```
import { useDailyEvent } from '@daily-co/daily-react-hooks';
import React, { useState, useCallback } from 'react';

export const InCall = () => {
  const [showLoader, setShowLoader] = useState(false);

  ...
  useDailyEvent(
    'joining-meeting,
    useCallback(
      () => setShowLoader(true),
      []
    )
  );

  useDailyEvent(
    'joined-meeting,
    useCallback(
      () => {
        setShowLoader(false)
        // Continue by updating your app UI to render the call U
      },
      []
    )
  );

  ...
  return (
    <>
      /* Assume there's some call-specific UI rendered
      along with the conditional loader */
      ...
      {showLoader && <Loader />}
      ...
    </>
  )
}
```

Example of useDailyEvent hook in action

Let's go through this example a bit more to show exactly how the `useDailyEvent` hook works.

First, we import the Daily `useDailyEvent` hook, which can listen for any Daily event. It accepts two parameters:

1. The event name it should listen for
2. The callback function that should be used when the event triggers

We also import React's `useCallback` and `useState` hooks, which are used as well.

```
import { useDailyEvent } from '@daily-co/daily-react-hooks';
import React, { useCallback, useState } from 'react';
```

In our component, we can then initialize our loading state with React's built-in `useState` hook.

```
const [showLoader, setShowLoader] = useState(false);
```

With that set up, we can now get to handling our Daily events. For the `'joining-meeting'` event, we use `useDailyEvent` and pass the event name and our callback as parameters.

In the callback function, we update our local loading state to be `true` so we know when to render our `Loader` component.

```
useDailyEvent(
  'joining-meeting,
  useCallback(
    () => setShowLoader(true),
    []
  )
);
```

Note: The callback function passed to `useDailyEvent` has to be a stable reference – meaning it will not change – which can be achieved by wrapping the function with React's `useCallback` hook.

We then use `useDailyEvent` a second time for the `'joined-meeting'` event. When this is called, we know we can remove the loader and show our regular call UI (whatever that looks like in your

app). Therefore, when `'joined-meeting'` is triggered, we can reset our loading state in the callback to prevent the `Loader` component from being rendered.

```
useDailyEvent(
  'joined-meeting,
  useCallback(
    () => {
      setShowLoader(false)
      // Continue by updating your app UI to render the call UI
    },
   []
  )
);
```

One of the benefits here is not needing to worry about "cleaning up" any event listeners. Typically in React, you would write your Daily event listener using the Daily `on` method, and then have to use the Daily `off` method when you are no longer listening for that event (e.g. when the call ends or the component unmounts).

Daily's `useDailyEvent` hook handles all of that for you and helps avoid memory leaks. All you need to do is provide the callback function to describe what needs to happen when a specific event is triggered.

Which aspects of `daily-js` are covered by Daily React Hooks?

Anything developers can do with `daily-js` directly can be done with Daily React Hooks.

Available custom Daily Hooks include:

- `useDaily` : our core Daily Hook that returns the current call object in custom Daily apps
- `useActiveParticipant` : returns the most recent/current active speaker
- `useAppMessage` : provides access to sending and receiving app messages often used for features like text chat

- `useDailyEvent` : accepts a callback for listening to specific Daily events
- `useDevices` : returns all available information needed to list or set media devices, as well as information about their state and errors
- `useLiveStreaming` : returns information about current live streams happening in the call
- `useLocalParticipant` : returns the local participant in an active call
- `useMediaTrack` : returns a specific participant's track and state
- `useNetwork` : returns state and information about the local user's network quality
- `useParticipant` : returns a specific participant based on the `session_id` provided
- `useParticipantsIds` : returns a list of participant IDs that can be filtered and sorted on request
- `useReceiveSettings` : returns information on the current receive settings
- `useRecording` : returns state and information on a recording in progress
- `useRoom` : a convenience hook for accessing the call's room configuration, similar to the `room()` instance method
- `useScreenShare` : returns a list of current screens being shared in the call
- `useThrottledDailyEvents` : similar to `useDailyEvent` , this accepts a callback for a Daily event, which can be throttled with a specified timeout
- `useWaitingParticipants` : returns a list of participants waiting to join a private call that requires participants to "knock" before joining

These hooks simplify many API features, but if you have to, you can always grab the call object with the `useDaily` hook and build as you would with `daily-js` directly.

Wrapping up

To learn more about Daily React Hooks and how to use them, check out our Daily React Hooks docs. We've added everything you'll need to know for switching your React code over.

If you're looking for more examples of how to use Daily React Hooks, be sure to keep an eye on our Twitter to see upcoming React demos putting them into action. In the meantime, feel free to reach out to our support team for additional information. We're always happy to help!

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 →



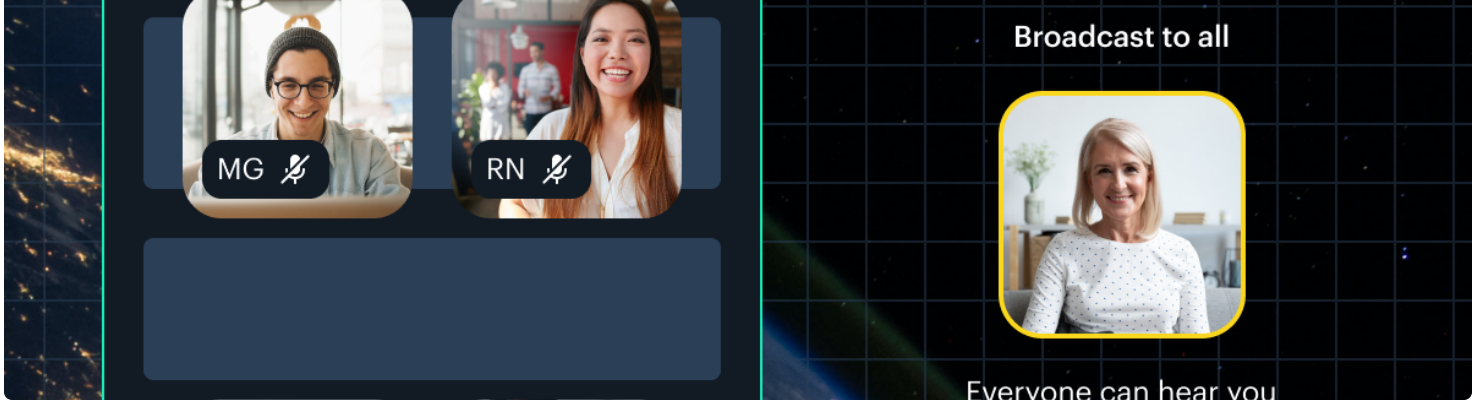
API

Daily use cases: Building a real-time video fitness class app with Daily

Introducing Daily's latest fitness class demo app features



Jess Mitchell
10 Mar 2022 – 4 min read



Spatialization

Creating interactive focus zones in Daily's spatialization demo (Part 5)

Broadcasting and gathering in small groups

Liza Shulyayeva
3 Mar 2022 – 13 min read