API

WebinaRTC: Build a webinar app with React and Daily's prebuilt UI



Jess Mitchell
Published: 4 Mar 2021 (Updated: 25 Oct 2021) • 6 min read



> *This is the second post in our series on how to build your own webinar platform using Daily Prebuilt and a custom chat widget.*

As the need for online video calls grows, Daily's video APIs continue to evolve to help ensure our customers can support whichever use cases they encounter. Recently at Daily we've been discussing the webinar format more and how to help our customers (and our own team!) build a webinar app as efficiently as possible.

Knowing a proper webinar app could only be built with our call object mode, we decided to update our prebuilt UI with two new Daily properties and instance methods to facilitate an even faster developer experience.

Now that our prebuilt UI can accommodate the webinar use case, building a webinar app has never been faster. To show you how much we mean this, we'll be sharing an ongoing series on how to build your own webinar app with Daily's prebuilt UI and Create React App. Check out our introduction to our webinar series to see our full plan for what we'll be building.

What to expect when you're expecting a webinar app

Having recently built our own webinar app for internal use at Daily, we'll be including a series of features we found most helpful for the webinar experience. Today we'll start with the basics.

By the end of this first tutorial you'll have the foundation for a webinar app that you can continue to build along with us or customize yourself to meet your own needs.

In this tutorial, we'll be:

- Initializing a new create-react-app repository.
- Determining the structure of our React app and creating the files we'll need for upcoming posts.
- Adding the daily-js dependency to our React app.
- With `daily-js`, we'll embed a Daily call into a `WebinarCall` React component.

Bonus: we'll review creating a Daily room. The room settings will update as we add more features in this series to ensure the level of privacy we want.

By the end of this tutorial, you'll have a React app running locally that

has a Daily video call embedded in it using Daily's prebuilt UI.

Creating a React app

Our first step will be to initialize a React app using create-react-app.

```
npx create-react-app webinar-app
cd webinar-app
yarn install
yarn add styled-components
yarn start
```

(Note: If you've read other Daily tutorials, you might notice we usually use `npm`. We're using `yarn` here to stick with Create React App's defaults.)

These commands create the React app repository locally, then change directories to be in the React app. Once in the directory, it installs all the dependencies. It then adds styled-components, which we'll use instead of a CSS stylesheet. Lastly, it will start the front-end server. To confirm everything is working as expected you can visit `http://localhost:3000`.

Now that we've got our basic React app, we can embed our Daily call.

Adding the daily-js package

To embed Daily's prebuilt video call in your app, we'll need to install `daily-js`, Daily's front-end library. Install `daily-js` and restart your server by running the following in the command line:

```
yarn add @daily-co/daily-js
yarn start
```

Now we can use `daily-js` in our app!

Determining our app structure

For this initial implementation, our structure can stay pretty simple. We will have one new component: `WebinarCall.jsx`.

Under the `src` directory, add a directory called `views` and within `views` add a file called `WebinarCall.jsx`. (You can also delete `App.css` since we'll be using `styled-components` instead.)

Once you've done this, your project structure should look like this:

```
Project_directory
|_  public
|_  src
  |_  App.js
  |_  views
    |_  WebinarCall.jsx
  |_  index.js
|_ package.json
// plus some other files created by create-react-app
```

Since we'll be building on this app in future webinar posts, feel free to take a sneak peek at how this structure will grow to two views and several components as we add more features.

Building our call component

Let's start with the most basic version of a Daily call component and then review what's included below.

```
import React, { useRef, useEffect, useState } from "react";
import DailyIframe from "@daily-co/daily-js";
import styled from "styled-components";

const CALL_OPTIONS = {
 iframeStyle: {
   width: "100%",
   height: "100%",
   border: "1px solid #e6eaef",
   borderRadius: "6px 6px 0 0",
```

```
 },
 showLeaveButton: true,
 showFullscreenButton: true,
 //   showLocalVideo: false,
 //   showParticipantsBar: false,
};

const DEFAULT_HEIGHT = 400;

const WebinarCall = () => {
 const videoRef = useRef(null);
 const [height, setHeight] = useState(DEFAULT_HEIGHT);
 const [callframe, setCallframe] = useState(null);

 useEffect(() => {
   if (!videoRef || !videoRef?.current || callframe) return;
   CALL_OPTIONS.url = "https://[DOMAIN].daily.co/[ROOM-NAME]";

   const newCallframe = DailyIframe.createFrame(
     videoRef.current,
     CALL_OPTIONS
   );

   newCallframe.join().then(() => {
     setHeight((videoRef?.current?.clientWidth || 500) * 0.75);
     setCallframe(newCallframe);
   });
 }, [videoRef]);

 return (
   <div>
     <Header>Welcome to our webinar!</Header>
     <VideoContainer height={height}>
       <Callframe ref={videoRef} />
     </VideoContainer>
   </div>
 );
};

const Header = styled.div`
 font-size: 36px;
 text-align: center;
 padding: 24px;
`;
const VideoContainer = styled.div`
 margin: auto;
 max-width: 1000px;
 height: ${(props) => (props.hidden ? "100" : props.height)}px;
`;
const Callframe = styled.div`
```

```
  width: 100%;
  height: 100%;
`;

export default WebinarCall;
```

This code block can be broken into a few sections. It includes:

- The library imports (React, daily-js, styled-components).
- The callframe properties (`CALL_OPTIONS`) we'll pass the `daily-js` factory method.
- The component itself (`WebinarCall`).
- The styled components, which apply CSS styles to the element provided to it.

The `WebinarCall` component can be broken down into a few more sections:

- It initializes state values and a `ref` that will be passed to the `daily-js` factory method.
- It uses the `useEffect` hook to create and embed the Daily call.
- It returns the component elements for what will be rendered (a simple header and the video-related elements).

Let's take a closer look at the `useEffect` that initialized the call:

```
useEffect(() => {
   if (!videoRef || !videoRef?.current || callframe) return;
   CALL_OPTIONS.url = "https://[DOMAIN].daily.co/[ROOM-NAME]";

   const newCallframe = DailyIframe.createFrame(
     videoRef.current,
     CALL_OPTIONS
   );

   newCallframe.join().then(() => {
     setHeight((videoRef?.current?.clientWidth || 500) * 0.75);
     setCallframe(newCallframe);
   });
 }, [videoRef]);
```

Here we first want to check that the ref exists before trying to embed the call in it. We also want to make sure the call hasn't already been created, in which case we wouldn't need to create it again. Then we decide which Daily room we want to use. (We'll go into more detail about Daily room's below, but notice that the room URL in this code sample doesn't have the domain or room name set yet.)

Once it's ready to be created, we use the DailyIframe class imported at the top of the file and call the factory method createFrame, which accepts two parameters: the element that the iframe will be embedded in and the properties we defined already (`CALL_OPTIONS`).

Once the call has been joined we set the height of the iframe's container to be ¾ of the width and update the callframe state to prevent additional iframes from being added.

Adding our webinar call to our App component

Now that we have our call component, we just need to render it in our main App component.

```
import WebinarCall from "./views/WebinarCall";
import styled from "styled-components";

function App() {
 return (
   <AppContainer>
     <WebinarCall />
   </AppContainer>
 );
}

const AppContainer = styled.div`
 width: 100vw;
 height: 100vh;
 background-color: ghostwhite;
`;

export default App;
```

Once the call component is included, we have one final step to get this call working: deciding on which Daily room to use.

A room of one's own

To create a Daily room to use, you'll need to create a Daily account. Once you have an account, you can create a room through the dashboard. (Check out our room docs for more information on creating rooms.)

When creating a room through the dashboard, you can customize your settings however you'd like. For now you can create a basic `public` room to test that the app is working.

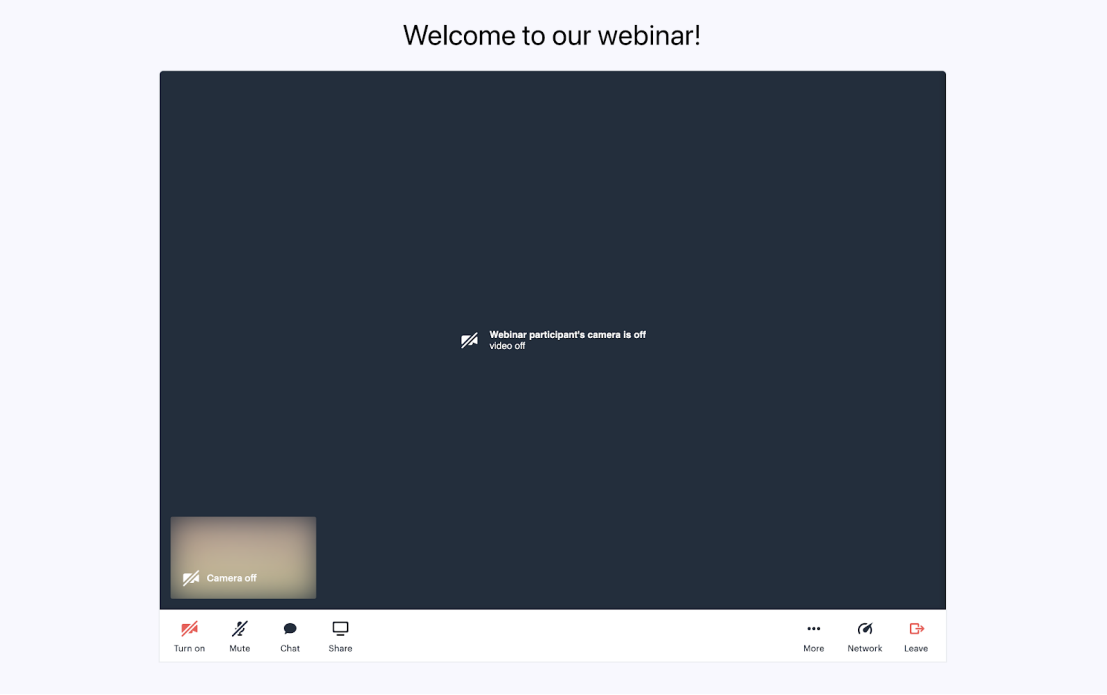
In a future tutorial we'll update the app to distinguish between hosts and participants with meeting tokens, so we'll need to change the room settings to be `owner_only_broadcast` . But, for now, let's keep things simple.

Once your room is created, you can copy your new room URL (`https://[DOMAIN].daily.co/[ROOM_NAME]`) and replace the placeholder URL in the code sample above.

Note: As is often the case with demo apps, this example isn't quite production ready, but it's a starting off point for our future features!

Testing our first call

Now that we have our React app and room URL configured, we can refresh our page and see our simplified webinar app.



You might be thinking, “Is this really a webinar app?” Well, not yet! So far we have our Daily call embedded in our app just waiting to be turned into a full webinar platform.

What’s next?

In upcoming blog posts we’ll build on this app further by adding features like admin authentication with Daily meeting tokens, building a chat for attendees to ask questions in, and adding a sign-in form for attendees.

If you’d like to get a head start, check out our webinar demo code. You can also learn more about meeting tokens in a previous blog post.

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

Working with video call participants' media tracks

10 Oct 2022 - 9 min read

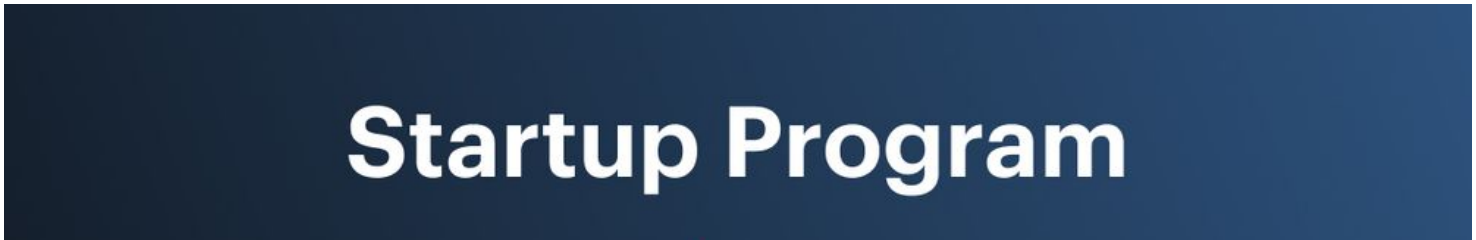
Obtaining, handling, and validating meeting tokens in your video application

26 Sep 2022 - 11 min read

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

30 Aug 2022 - 5 min read

See all 83 posts →



Join Daily's startup program

Up to $1,000 in call credits for qualifying startups, plus partnerships for accelerators and other programs.



Daily Team
9 Mar 2021 – 1 min read



API Tutorial

Introducing Daily's webinar series: Building an open source webinar app with Daily's prebuilt UI

This is the first post in our series on how to build your own webinar platform using Daily Prebuilt and a custom chat widget. As more services move online, Daily



Jess Mitchell
4 Mar 2021 – 3 min read