This guide explains how to create a custom Bubble plugin for your app. For demonstration purposes, we’ll create a simple camera recorder.

## **Features:**

1. Start the recorder
2. Display the recording video
3. Stop the recorder
4. Upload the video to the Bubble App and return an URL

# **Step 1: Create a new plugin**

Go to <https://bubble.io/my_plugins>. Click “New plugin”.

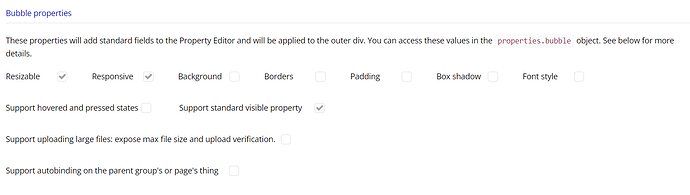
Name your plugin. Click “Create”.

# **Step 2: Add a new element**

A plugin element is a custom Bubble element that can be added to Bubble pages. We need an element to display the recording video.

Go to tab “Elements”, add a new element. Name your element.

The element should be resizable and responsive, so we select those properties. We also want to show the standard visible property.



[step11542×437 32.5 KB](https://us1.discourse-cdn.com/bubble/original/3X/6/a/6a044d06eed919fac5253e808fd0a1ae01508d81.png)

# **Step 3: Initialize functions**

Go to “Code initialize”, this JavaScript function is called when the element is visible on the page. It is a good practice to define functions here, and trigger these functions only when you need them.

We’ll create a function instance.data.start to access the camera, start recording, and display the recording video. We’ll also need a function instance.data.stop to stop the recording and save the video.

function(instance, context) {

let getCamera = async function () {

return await navigator.mediaDevices.getUserMedia({

audio: false,

video: true

});

}

let createRecorder = function(stream, mimeType) {

let recordedChunks = []; *// the stream data is stored in this array*

const mediaRecorder = new MediaRecorder(stream);

mediaRecorder.ondataavailable = function(e) {

if (e.data.size > 0) {

recordedChunks.push(e.data);

}

};

mediaRecorder.onstop = function() {

saveFile(recordedChunks);

recordedChunks = [];

};

mediaRecorder.start(200); *// For every 200ms the stream data will be stored in a separate chunk.*

return mediaRecorder;

}

let saveFile = function(recordedChunks) { *//This function creates a .webm file*

const blob = new Blob(recordedChunks, {

type: 'video/webm'

});

var reader = new FileReader();

reader.readAsDataURL(blob);

reader.onloadend = function() {

var filename = instance.data.file || 'video.webm';

var base64data = reader.result;

var uploadData = base64data.substr(base64data.indexOf(',') + 1);

context.uploadContent(filename, uploadData, function(err, url) { *//upload to the bubble database*

instance.publishState("url", url); *//return the url as an exposed state*

instance.triggerEvent("recorded"); *//trigger an event*

URL.revokeObjectURL(blob); *// clear from memory*

});

}

}

instance.data.start = async function(){

instance.data.streamWebcam = await getCamera();

let webcamRecorder = createRecorder(instance.data.streamWebcam, "video/webm");

*//display the video*

let video = document.createElement('video'); *//create the video element*

video.srcObject = instance.data.streamWebcam;

video.style.width = '100%';

video.style.height = '100%';

video.muted = true;

video.onloadedmetadata = function(e) {

video.play(); *//play the video on load*

};

instance.canvas.append(video); *//add element to canvas*

}

instance.data.stop = function() {

if (instance.data.streamWebcam) {

instance.data.streamWebcam.getTracks()

.forEach(track => track.stop()); *//stop all tracks*

}

}

}

# **Step 4: Create actions**

An action can be added to the Bubble workflows. We’ll create actions to trigger the functions we created in step 3.

### Action “Start”

Go to “Actions”, create a new action called “Start”.



[step21594×311 20.3 KB](https://us1.discourse-cdn.com/bubble/original/3X/1/1/1189e6341c8672d2b1def332951bc9b7782b66de.png)

Go to “Code actions”. Find function start. Add code to trigger the start function.

function(instance, properties, context) {

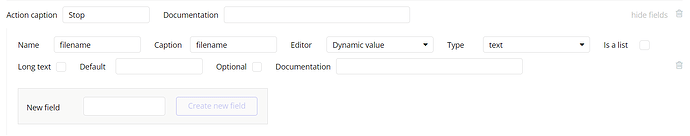
instance.data.start();

}

### Action “Stop”

Go to “Actions”, create a new action called “Stop”.

Add a new field filename. A field is a property you can fill in the Property Editor when you add the action to the workflow.



[step31587×313 18.8 KB](https://us1.discourse-cdn.com/bubble/original/3X/1/c/1cb42eb1009572db87f80ebbf5a85c2a81a41472.png)

* Name : The name of the field. We can get the value of this field by properties.filename
* Caption: This is the display name that appears in the editor.
* Editor: Select “Dynamic value”. This allows you to fill in a dynamic expression in the editor.
* Type: Type of the Dynamic value. Select “Text”. If you fill in an unexpected type of data in the editor, you’ll see an issue.

Go to “Code actions”. Find function start. Add code to trigger the start function.

function(instance, properties, context) {

instance.data.filename = properties.filename;

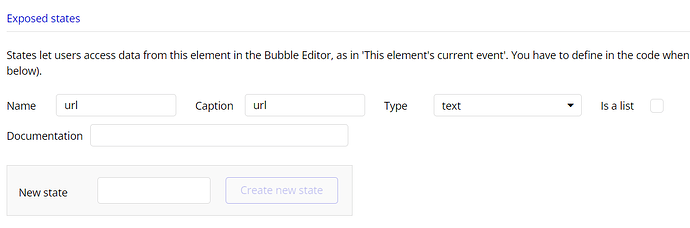
instance.data.stop();

}

# **Step 5: Add a state**

When a user stops the recording, the function saveFile() is triggered. This function creates a video file and uploads it to the Bubble database. An uploaded file has an URL. To get the URL, we have to create an exposed state. With an exposed state, you can expose the data from this custom plugin to other elements or actions in your app.

Go to “Exposed states”. Add a new state url. The type of data should be text.



[step41147×380 18.4 KB](https://us1.discourse-cdn.com/bubble/original/3X/e/9/e9d796c891a24d741222c3a6c6c80a0ea04c1b74.png)

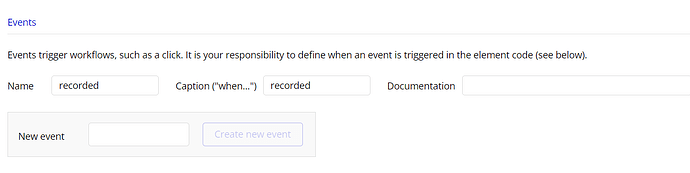
To expose the data, we have to add a line of code instance.publishState("<name>", <value>) to the saveFile() function, where <name> refers to the state’s name, and <value> refers to the value you want to expose. We have already done this in step 3. You should be able to see the code.

# **Step 6: Add an event**

An event is a Bubble workflow that triggers actions. You can create a new event and trigger this event in your code.

With the exposed state created in step5, we can get the URL of the video. However, the URL will be exposed only when the uploading process is completed. In another word, we cannot get the URL right after stopping the recording. We need an event to trigger the rest of the workflows after the unloading process is completed.

Go to “Events”. Create a new event “recorded”.



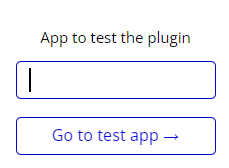
[step51287×323 17.1 KB](https://us1.discourse-cdn.com/bubble/original/3X/a/c/ac1042f92636c2877d969bf4a7501501f3dc91d7.png)

To trigger the event, we can add a line of code instance.triggerEvent("<name>"), where refers to the name of the event. We have already added the code in step 3.

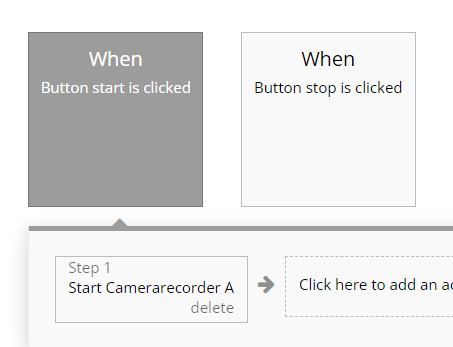
# **Step 7: Test your plugin**

You need a test app to test the plugin. Create an app if you don’t have one.

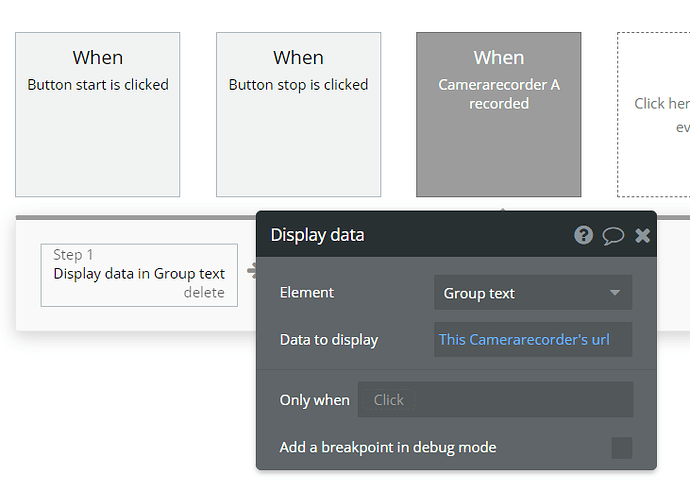
There is an input field under the plugin elements tree. Input the app name and click “Go to test app”



To test the function, put your plugin element on the page. Add a button to trigger the custom actions.



Add a custom event and display the exposed URL.



[step8731×521 26.5 KB](https://us1.discourse-cdn.com/bubble/original/3X/b/4/b4ed4247c5c874f3ee399a7cba166386954326d7.png)

# **Step 8: Publish your plugin**

Once your plugin is ready, you can publish it on an open-source license or a private license. Go to “Settings”. Under Publishing and license, select “Private” or “Open source (MIT)” as the Plugin distribution license. If you have selected “Private”, you need to authorize your apps to have access to this plugin.

Click Submit a new version to publish your plugin. Congratulations! You have created a custom camera recording plugin.