The Script Behind It All

JavaScript

Learning Outcomes

1. Describe the relationship between Behavior Script and JavaScript
2. Identify the steps to add a JavaScript file to your AR experience

JavaScript

In our journey from simple checkboxes to Global Behaviors, we've seen how powerful "pre-built" logic can be. But to truly push the boundaries of STEAM and create a professional-grade comic book experience, we eventually move into the world of Custom Scripting.

The next activity is an introduction to using JavaScript (specifically the code in powAnimationFade.js) to create a "Dynamic Comic Pop" effect that goes far beyond a simple scale change.

Transitioning to Custom Scripts
While the Behavior Script is like using a remote control with pre-set buttons, writing your own script is like building the remote itself.

JavaScript

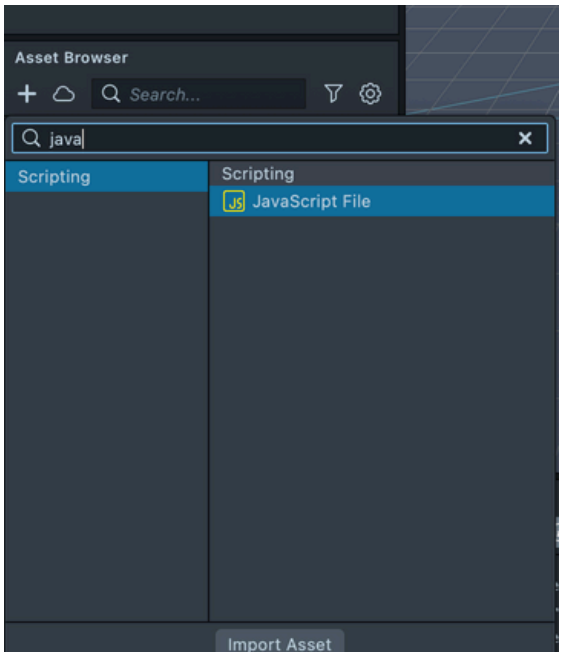
What is powAnimationFade.js doing?

This file is a specialized "recipe" that tells the AR engine exactly how to handle a comic book interaction. Instead of just getting bigger, it manages a Life Cycle for your art. Below is the JavaScript code we'll be using.

```
// @input SceneObject graphicObject
// @input float popSpeed = 0.05
// @input float maxScale = 30.0

var isPopped = false;
var currentScale = 0;

function onUpdate() {
   if (isPopped && currentScale < 1) {
      currentScale += script.popSpeed;

      // Multiply the progress (0 to 1) by our Max Scale
      var s = currentScale * script.maxScale;
      script.graphicObject.getTransform().setLocalScale(new vec3(s, s, s));
   }
}

function onTap() {
   isPopped = true;
   currentScale = 0;
}

script.createEvent("UpdateEvent").bind(onUpdate);
script.createEvent("TapEvent").bind(onTap);
```

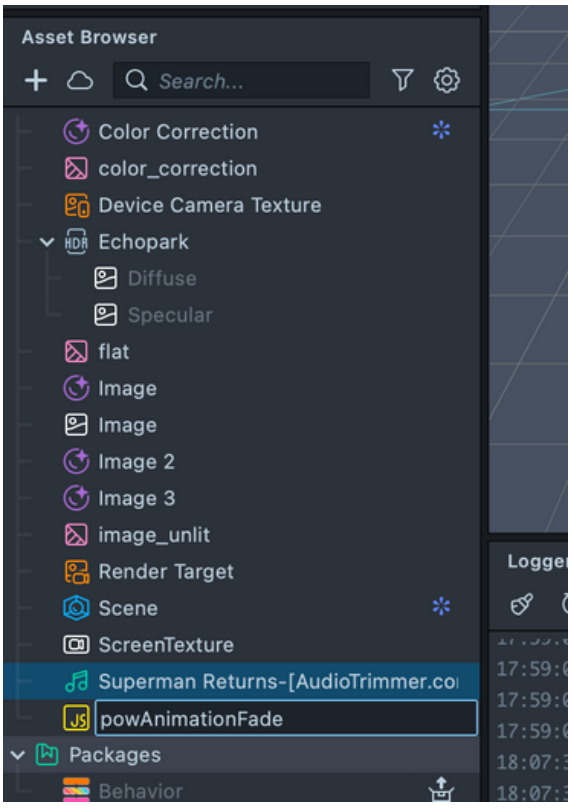
Adding a JavaScript Asset File

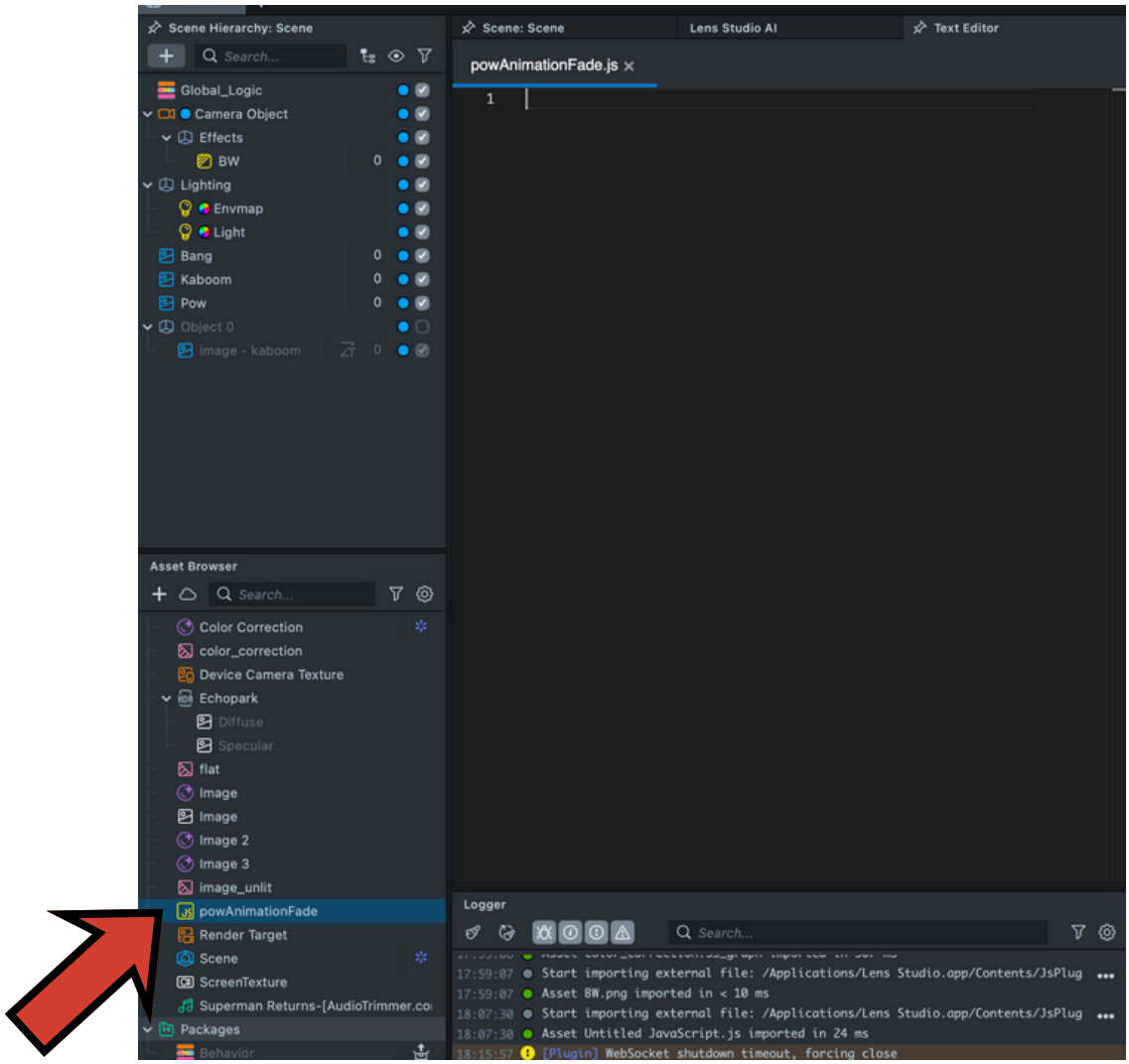
A JavaScript file is another type of asset. To add a blank script file, in the Asset Panel (lower right corner), do a search for “JavaScript” and click JavaScript File.



Rename the Asset

Rename the asset to “powAnimationFade”





Adding our JavaScript code

Lens Studio has a built in Text Editor which allows us to easily add our custom code to our AR lens.
To open the Text Editor, double click on the powAnimationFade asset you just created.
You can copy and paste the code provided in the file

The full code along with a short description (gray text) is provided on the next page. If you're up for a challenge - try typing it into the text editor instead of copying and pasting.
Save the file by going to File > Save or clicking CTRL + S.

```
// --- INPUTS: These create the adjustable boxes in the Inspector panel ---

// This tells the script which comic sticker we want to animate
// @input SceneObject graphicObject

// This controls how fast the "burst" happens (Science/Math link)
// @input float popSpeed = 0.05

// This sets the final size of the sticker (Engineering/Art link)
// @input float maxScale = 30.0

// --- VARIABLES: These store information that changes over time ---

var isPopped = false;  // A "True/False" switch to check if we have tapped yet
var currentScale = 0;  // Our starting point (0% grown)


// --- LOGIC: The "If/Then" instructions for the animation ---

function onUpdate() {
    // IF the user tapped AND the sticker isn't full size yet...
    if (isPopped && currentScale < 1) {

        // Add a small amount of growth on every "heartbeat" of the lens
        currentScale += script.popSpeed;

        // Multiply our progress (0 to 1) by our Max Scale to get the final 3D siz
        var s = currentScale * script.maxScale;

        // Apply that math to the sticker's X, Y, and Z dimensions
        script.graphicObject.getTransform().setLocalScale(new vec3(s, s, s));
    }
}

// --- EVENTS: These tell the lens to start "listening" for actions ---

// Listen for every frame update (the "Render Loop")
script.createEvent("UpdateEvent").bind(onUpdate);

// Listen for when the user touches the screen
script.createEvent("TapEvent").bind(onTap);
```

Knowledge Check

Bridging the Gap to Code

Instructions: Answer the following questions to demonstrate your understanding of how "no-code" tools and professional programming work together.

1. The Handshake: Behavior vs. JavaScript

In Lens Studio, what is the primary relationship between a Behavior Script and a custom JavaScript file?

A. They are unrelated; you can only use one or the other in a project.

B. Behavior Scripts are a "no-code" interface for common tasks, while JavaScript allows you to write custom, complex logic from scratch.

C. JavaScript is only for 3D models, while Behavior Scripts are only for 2D images.

D. Behavior Scripts are written in a different language like Python.

2. Creating a New Script

To add a new, blank JavaScript file to your project, which panel must you use?

A. The Objects Panel

B. The Logger Panel

C. The Asset Browser

D. The Preview Panel

3. The "Activation" Step

Once you have created a .js file in your Asset Browser, how do you make it "live" so that it actually runs in your AR experience?

A. You must drag the script file onto an object in the Objects Panel to attach it as a component.

B. It runs automatically as soon as you save the file.

C. You have to rename the file to "ActiveScript.js."

D. You must click the "Play" button in the Logger.

4. Short Answer: The Power of Custom Code

Imagine you want to create a game where a sticker moves faster every time the user scores a point. Why might you choose to write a JavaScript file for this instead of using a standard Behavior Script?

Answer:

_____

(Hint: Think about the difference between a simple "If-Then" rule and complex math that changes over time.)

JavaScript, Meet Object
Object, Meet JavaScript

Apply JavaScript to the Object

Learning Objectives

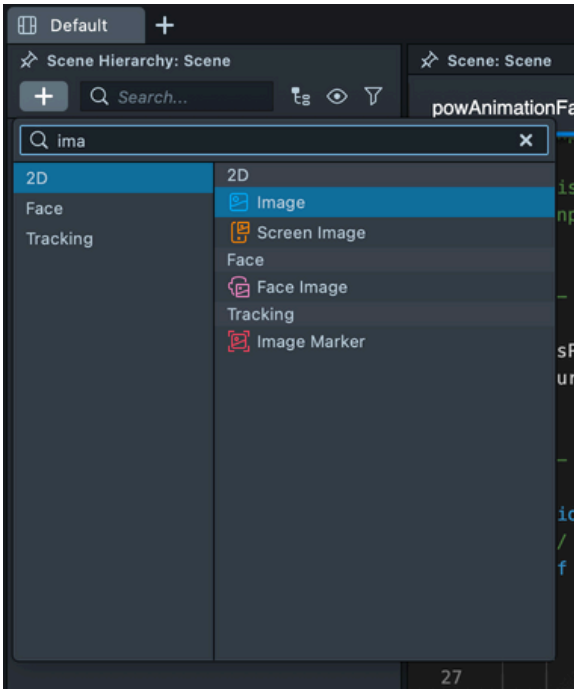
1. Describe how to add a custom JavaScript block to an object
2. Understand how to use the front and rear cameras to create a unique AR experience

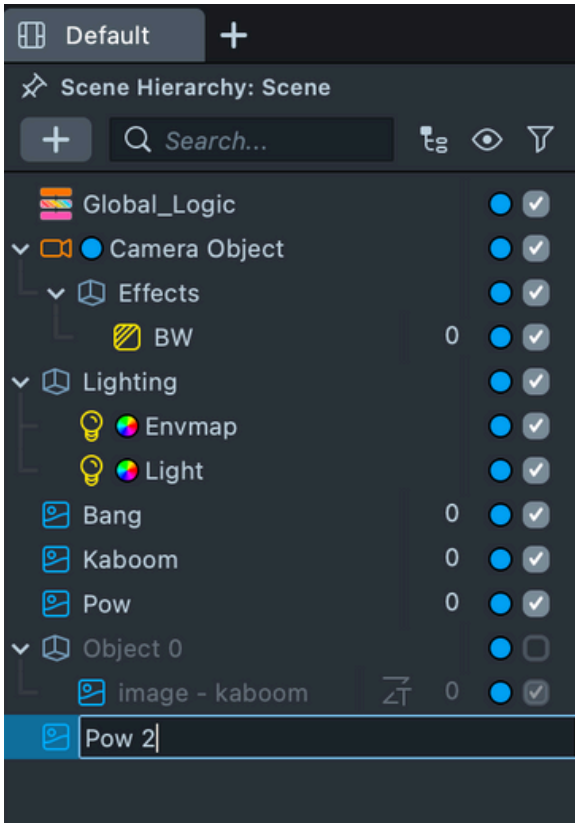
Applying JavaScript to an Object: Create the Object

With your script saved, you can now apply it to any object in your AR experience.

Add another object to your AR experience. In this demo, we are going to add another Pow image.

1. Add a new image object in the Scene Hierarchy (aka Object) panel.
2. Give it a unique name (e.g. “Pow 2”)

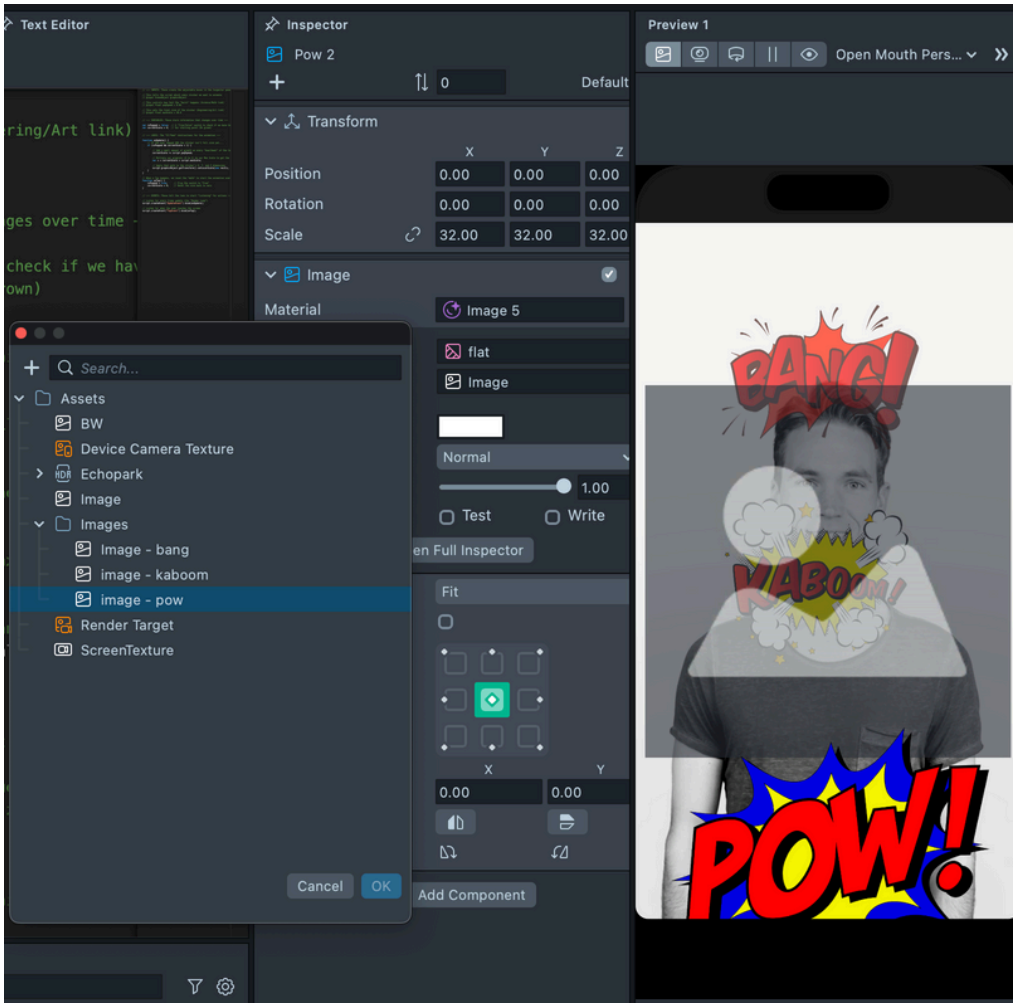




Applying JavaScript to an Object: Add the Asset

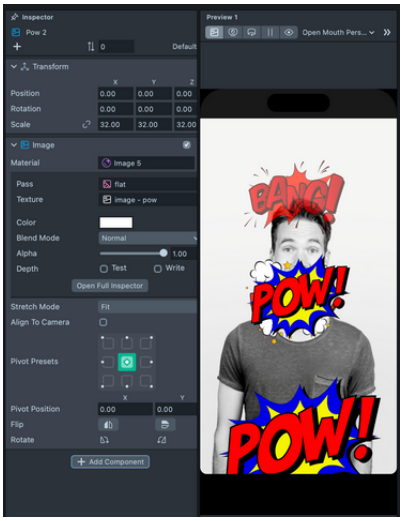
With your script saved, you can now apply it to any object in your AR experience.

In the Inspector Panel, we need to add an image to the object. In this demo, we're going to use the same Pow image asset as before. Assets can be easily reused in Lens Studio. It's the behaviors and other attributes that can make each asset unique in an AR experience.

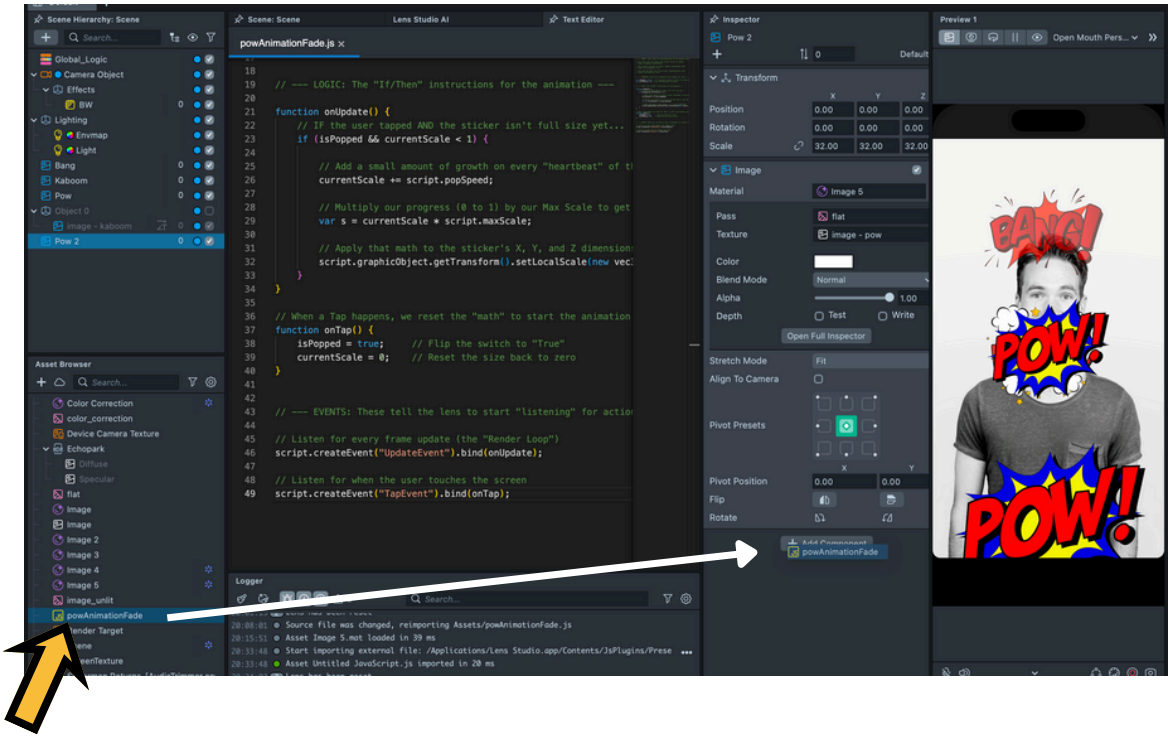


Applying JavaScript to an Object: Add the Script

We see the second “Pow” added to our preview. The image is added, now we need to add the JavaScript file of powAnimationFade.js

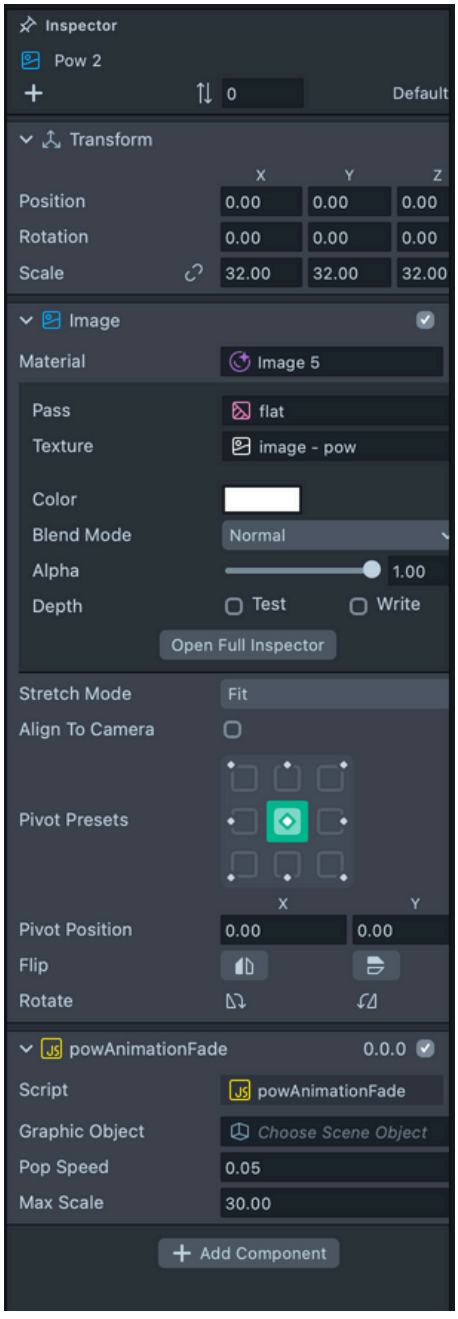


Select the powAnimationFade file from the Asset panel and drag it to the Inspector Panel

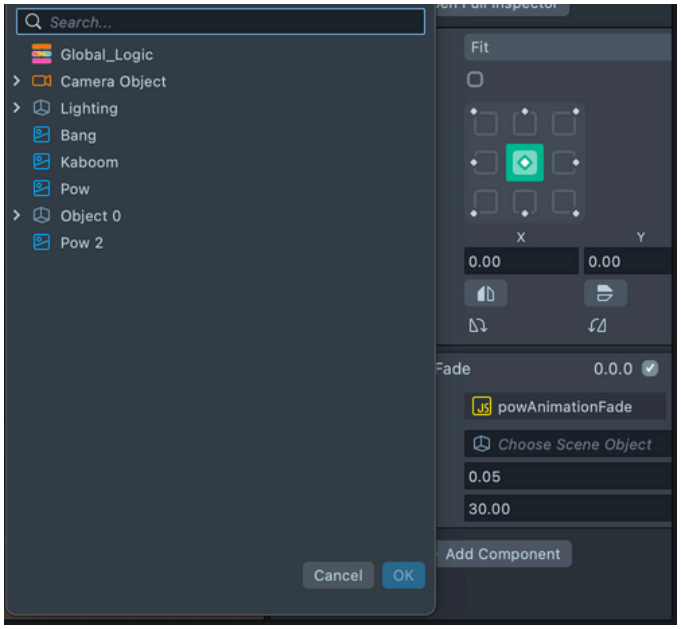


Applying JavaScript to an Object: Inspector Panel

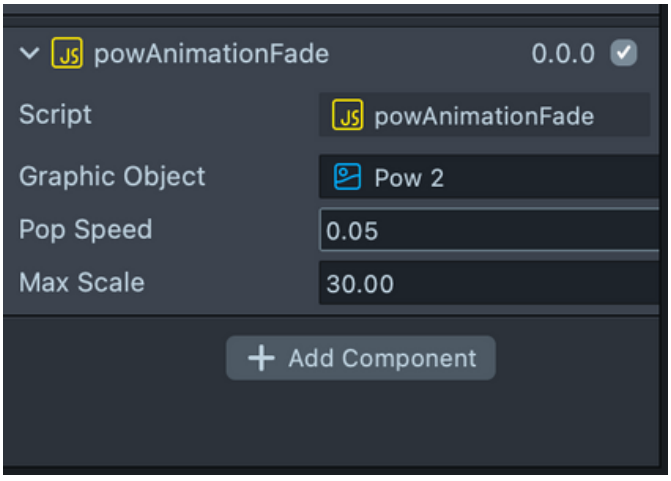
The powAnimationFade component has been added to the object itself but we still need to add the graphic object that this applies to.



This demo, we are selecting the Pow 2 object.



Adjust the sliders for the Pop Speed and Max Scale as desired.



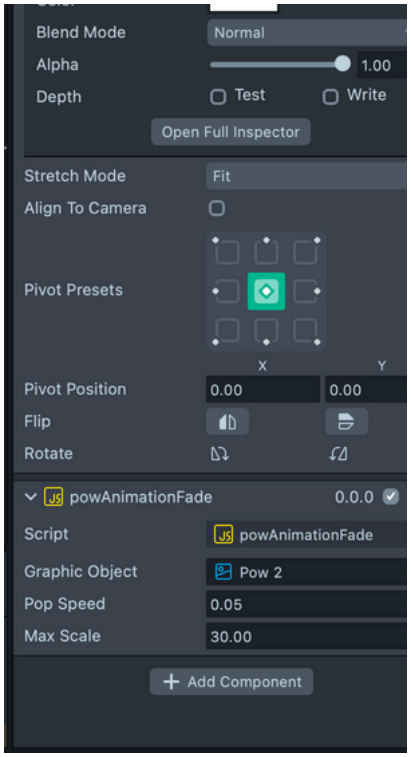
Applying JavaScript to an Object: Preview Panel

The powAnimationFade JavaScript file tells the Pow image to wait until you click on the Preview panel and the second Pow image will shrink down quickly to slowly grow again.

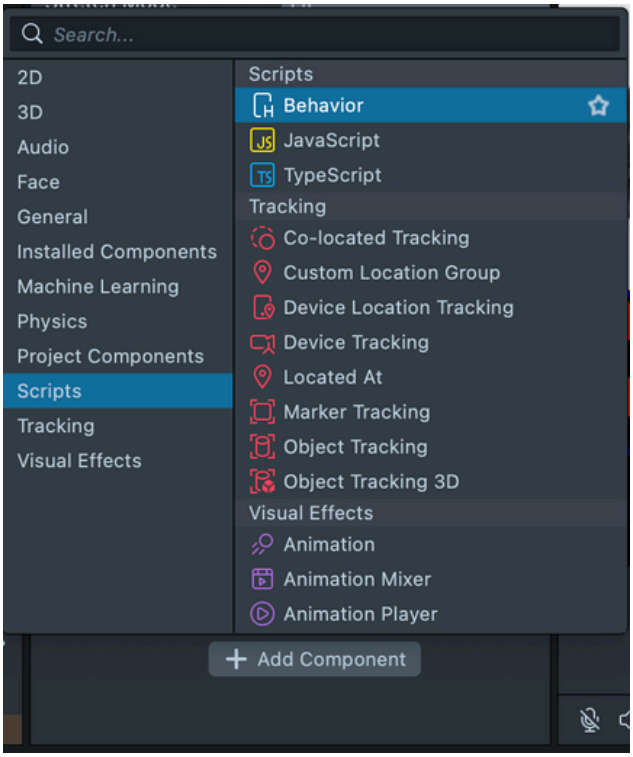
Change Your Mind?
Totally Fine

Having two POW images showing on the front camera is a little too much. We want to change so the Pow 2 image shows on the Rear Camera and not on the Front Camera.

Click on + Add Component



Go to Scripts > Behavior



The Behavior component has been added.