



# Custom JS Components

This [video](#) shows how to use a custom, [IFrame](#)-based [symbol](#) to implement your custom JavaScript:

To create this effect:

1. Create an HTML file that contains your JavaScript or a reference to your JavaScript.
2. Upload your HTML file, or HTML and JS files, to your DGLux5 project by dragging them into the [Project panel](#).
3. Upload `dgframe.js` to the same folder in the Project panel that contains your HTML and JS.
4. Create an IFrame component by dragging the HTML file from your Project panel to the [Document window](#).
5. Convert the IFrame to a [symbol](#):
  1. Right-click the IFrame component, and select  **Convert to Symbol**.
  2. Specify a name for the symbol.
6. Add custom parameters to the symbol:
  1. In the [Outline](#) or [Document window](#), right-click the symbol, and select **Edit Properties**.
  2. In the dialog that appears, drag the correct data types from the left panel into the right panel. For example, if you want one number and one string property, drag an instance of **number** and an instance of **string** from the left panel into the right panel.
  3. Double-click each property label in the right panel to name it. These names must match the names of your `dgParams` in step 9.
  4. Exit the symbol editor by double-clicking the Stage.
7. Update your HTML code:
  1. In the Project panel, click the HTML file.
  2. In the [Details panel](#), click  **Edit in Window** to open a pop-up for editing the HTML.
8. Add the following code to your HTML:

```
◦ <script type='text/javascript' src='dgframe.js'></script>
```

This ensures that `dgframe.js` is included, so that your HTML variables can correspond with your symbol properties.

9. Add `dgParams` to your HTML, as an object of functions. Refer to the example below. These dynamic variables must match the symbol property names in step 6.

The symbol properties now correspond with the dynamic properties in your script.

---

## Example HTML

```
<!DOCTYPE html>  
<!-- https://github.com/web-animations/web-animations-js -->
```

```
<html>
  <body>
    <div class="pulse" style="width:150px;">Hello world!</div>
    <script type='text/javascript' src='dgframe.js'></script>
    <script type='text/javascript'>
      //DEFINE VARIABLES
      var duration = 500;
      var header = "Hello world!";
      var element = document.querySelector(".pulse");
      var animate = {
        direction: "alternate",
        duration: 500,
        iterations: Infinity
      };
      //DEFINE DYNAMIC PROPERTIES
      var dgParams = {
        "duration": function(val) {
          if (typeof(val) == "number") {
            duration = val;
          }
        },
        "header": function(val) {
          if (typeof(val) == "string") {
            header = val;
          }
        }
      };

      //APPLY TEXT AND ANIMATION TO ELEMENT
      element.textContent = header;
      element.animate([
        {opacity: 0.5, transform: "scale(0.5)"},
        {opacity: 1.0, transform: "scale(1)"}
      ], animate);

      //DEFINE ON LOAD SCRIPT
      window.onload = function() {
        //DEFINE REQUEST ANIMATION FRAME FROM BROWSER
        window.requestAnimFrame = (function(){
          return window.requestAnimationFrame ||
            window.webkitRequestAnimationFrame ||
            window.mozRequestAnimationFrame ||
            function (callback) {
              window.setTimeout(callback, 1000 / 60);
            };
        })();

        //DEFINE WHAT HAPPENS ON LOOP
```

```
function update() {
  if (element.textContent != header) {
    element.textContent = header;
  }
  if (animate.duration != duration) {
    animate.duration = duration;
    element.animate([
      {opacity: 0.5, transform: "scale(0.5)"},
      {opacity: 1.0, transform: "scale(1)"}
    ], animate);
  }
}

//DEFINE THE LOOP ITSELF
function loop() {
  requestAnimationFrame(loop);
  update();
}

loop();
}
</script>
</body>
</html>
```

## dgframe.js

```
// test if a parameter value is dglux table
function dgIsParamTable(val){
  return (val != null && typeof(val)=='object' && val.hasOwnProperty('cols')
&& val.hasOwnProperty('rows'));
}

// interface to the dglux5 application
function onDGFrameMessage(e){
  var data = e.data;
  if (typeof(data)=='object'){
    if (data.hasOwnProperty('dgIframeInit')){
      dgIframeId = data['dgIframeInit'];
      if (window.parent != null){
        // the first post back shouldn't contain any data change
        window.parent.postMessage({'dgIframe':dgIframeId}, '*');
      }
    }
  }
}
```

```
} else if (data.hasOwnProperty('dgIframeUpdate')){
  var updates = data['updates'];
  if (typeof(updates)=='object'){
    if (typeof(dgParams) == 'object'){
      for (key in dgParams){
        if (updates.hasOwnProperty(key)){
          dgParams[key](updates[key]);
        }
      }
    }
  }
}

if (typeof(dgParamsUpdated) == 'function'){
  dgParamsUpdated();
}
}
}
}
}
window.addEventListener('message', onDGFrameMessage);

// push parameter changes back to dglux
function dgUpdateParams(changes) {
  if (dgIframeId != null) {
    window.parent.postMessage({'dgIframe':dgIframeId, changes:changes}, '*');
  } else {
    throw 'dgUpdateParams failed, handshake not finished';
  }
}
}
```

---

## Changing the Symbol Properties from JavaScript

Use the following syntax to change the symbol properties from JavaScript:

```
dgUpdateParams({'a':300});
```

- Replace a with the name of the property.
- Replace 300 with the new value of the property.

---

[Previous: Scripting and Syntax](#)

[Next: Using Custom div Elements](#)

From:  
<https://wiki.dglogik.com/> - **DGLogik**

Permanent link:  
[https://wiki.dglogik.com/dglux5\\_wiki:dgscript:custom\\_javascript](https://wiki.dglogik.com/dglux5_wiki:dgscript:custom_javascript)

Last update: **2021/09/20 14:43**

