```
<html>

<!-- VERTEX SHADER SOURCE CODE: Written in GLSL -->
<script id="vertex-shader" type="x-shader/x-vertex">
#version 300 es

in vec4 aPosition;

void main()
{
  gl_Position = aPosition;
}
</script>

<!-- FRAGMENT SHADER SOURCE CODE: Written in GLSL -->
<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es

precision mediump float;

out vec4  fColor;

void main()
{
    fColor = vec4( 1.0, 0.0, 0.0, 1.0 );
}
</script>

<canvas id="gl-canvas" width="512" height="512"> </canvas>

<!-- Written in Javascript -->
<script>


// This compiles and links the shaders to create a GPU program
object
// The GLSL code above is parsed and provided as the source code
function initShaders( gl, vertexShaderId, fragmentShaderId )
{
    var vertShdr;
    var fragShdr;

    var vertElem = document.getElementById( vertexShaderId );
    if ( !vertElem ) {
        alert( "Unable to load vertex shader " + vertexShaderId );
        return -1;
    }
    else {
        vertShdr = gl.createShader( gl.VERTEX_SHADER );
        gl.shaderSource( vertShdr, vertElem.textContent.replace(/
^\s+|\s+$/g, '' ));
        gl.compileShader( vertShdr );
        if ( !gl.getShaderParameter(vertShdr, gl.COMPILE_STATUS) )
{
            var msg = "Vertex shader failed to compile.  The error
```

```
log is:"
 53          + "<pre>" + gl.getShaderInfoLog( vertShdr ) + "</pre>";
 54             alert( msg );
 55             return -1;
 56         }
 57     }
 58
 59     var fragElem = document.getElementById( fragmentShaderId );
 60     if ( !fragElem ) {
 61         alert( "Unable to load vertex shader " +
fragmentShaderId );
 62         return -1;
 63     }
 64     else {
 65         fragShdr = gl.createShader( gl.FRAGMENT_SHADER );
 66         gl.shaderSource( fragShdr, fragElem.textContent.replace(/
^\s+|\s+$/g, '' ) );
 67         gl.compileShader( fragShdr );
68         if ( !gl.getShaderParameter(fragShdr, gl.COMPILE_STATUS) ) {
 69             var msg = "Fragment shader failed to compile.  The
error log is:"
 70          + "<pre>" + gl.getShaderInfoLog( fragShdr ) + "</pre>";
 71             alert( msg );
 72             return -1;
 73         }
 74     }
 75
 76     var program = gl.createProgram();
 77     gl.attachShader( program, vertShdr );
 78     gl.attachShader( program, fragShdr );
 79     gl.linkProgram( program );
 80
 81     if ( !gl.getProgramParameter(program, gl.LINK_STATUS) ) {
 82         var msg = "Shader program failed to link.  The error log
is:"
83            + "<pre>" + gl.getProgramInfoLog( program ) + "</pre>";
 84         alert( msg );
 85         return -1;
 86     }
 87
 88     return program;
 89 }
 90
 91
 92 // EXECUTION: Code executes starting here when we launch this file
 93 window.onload = function init()
 94 {
 95     canvas = document.getElementById( "gl-canvas" );
 96
97     //
 98     //   Grab the section of the screen for drawing.
 99     //   All graphic output is within the canvas
100     //
101
102     gl = canvas.getContext('webgl2');
```

```
103     if (!gl) { alert( "WebGL 2.0 isn't available" ); }
104
105     //
106     //  Initialize our data for a single triangle
107     //
108
109
110     ////////////////////////////////////////////////////////////////
////////////////
111     //  Load shaders and initialize attribute buffers
112     //
113     program = initShaders( gl, "vertex-shader", "fragment-
shader" );
114     gl.useProgram( program );
115
116
117     ////////////////////////////////////////////////////////////////
////////////////
118     // Define shapes (initialize points defining a triangle)
119     // The GPU expects typed arrays and uses 4D points (x,y,z,w)
120
121
122      var points = new Float32Array([
123      //   X        Y      Z  and  W are allowed to default here!
124          0.0 ,    0.8,
125         -0.4,    -0.2,
126          0.4,    -0.2
127         ]);
128
129
130
131
132    ////////////////////////////////////////////////////////////////
///////////////
133    // Load the data into the GPU Buffers
134    //
135     var pointsBuffer = gl.createBuffer();
134     gl.bindBuffer( gl.ARRAY_BUFFER, pointsBuffer );
137     gl.bufferData( gl.ARRAY_BUFFER, points, gl.STATIC_DRAW );
138
139     // Associate out shader variables with our data buffer
140
141     var aPosition = gl.getAttribLocation( program, "aPosition" );
142     gl.vertexAttribPointer( aPosition, 2, gl.FLOAT, false, 0, 0);
143     gl.enableVertexAttribArray( aPosition );
144
145
146
147     ////////////////////////////////////////////////////////////////
////////////////
148     // Configure WebGL settings and draw
149     //
150       // Configure area of canvas to map framebuffer
151     gl.viewport( 0, 0, canvas.width, canvas.height ); //
152
```

```
153     // Configure color to use to clear all pixels
154     gl.clearColor( 1.0, 1.0, 1.0, 1.0 );    // Format: R, G, B, A
Normalized [0.0,1.0]
155
156     // Render
157     gl.clear( gl.COLOR_BUFFER_BIT );     // Clear all pixels in the
framebuffer
158     gl.drawArrays( gl.TRIANGLES, 0,  3);     // Draw
159
160 };
161
162 </script>
163 </html>
164
```