

Lab 0



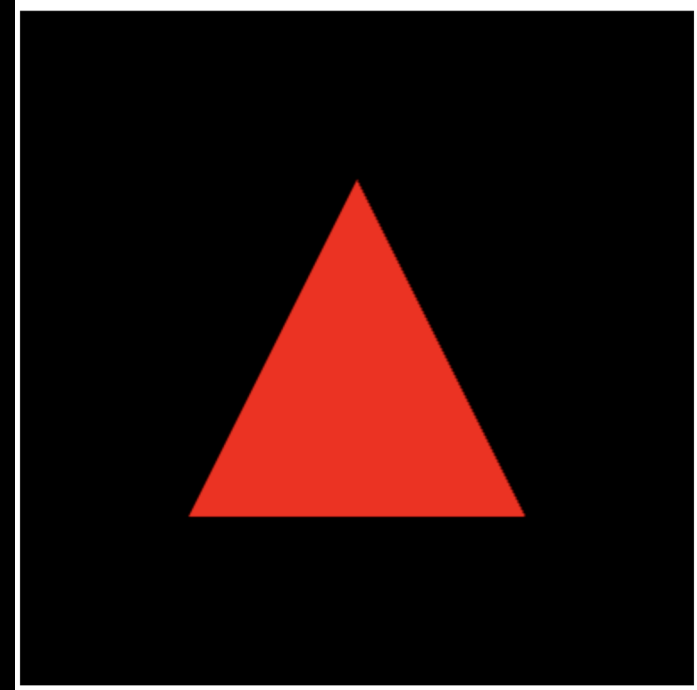
What You will Learn

- Familiar with a simple WebGL program
 - Run a WebGL program
 - Know the basic WebGL programming environment
 - Modify the code by instructions
 - Submit the code

What You Should Do Step by Step

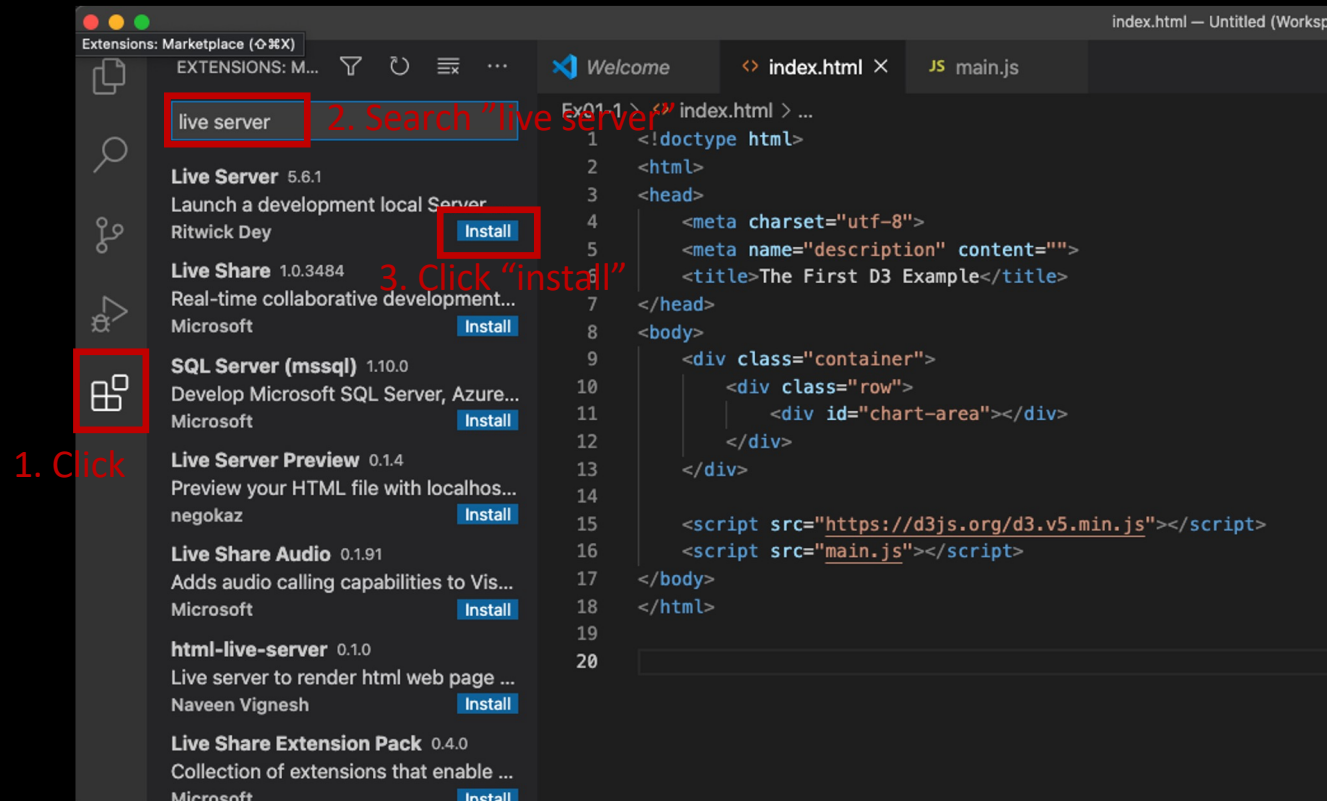
1. Download and Run

- Download the zip file which contains “index.html” and “WebGL.js” from Moodle (unzip the zip file)



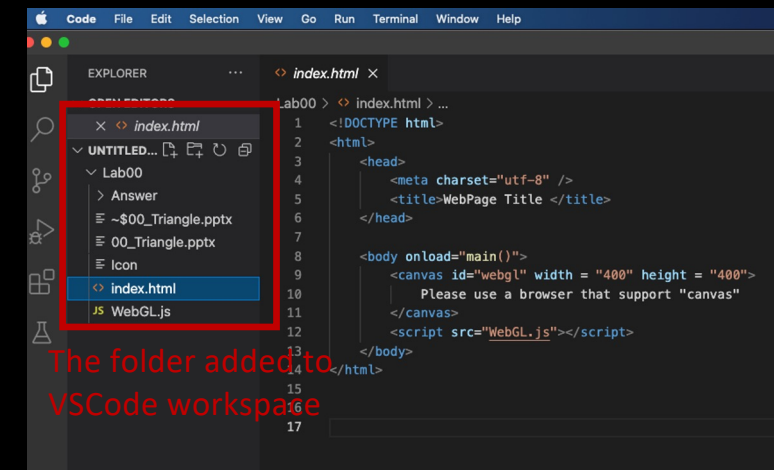
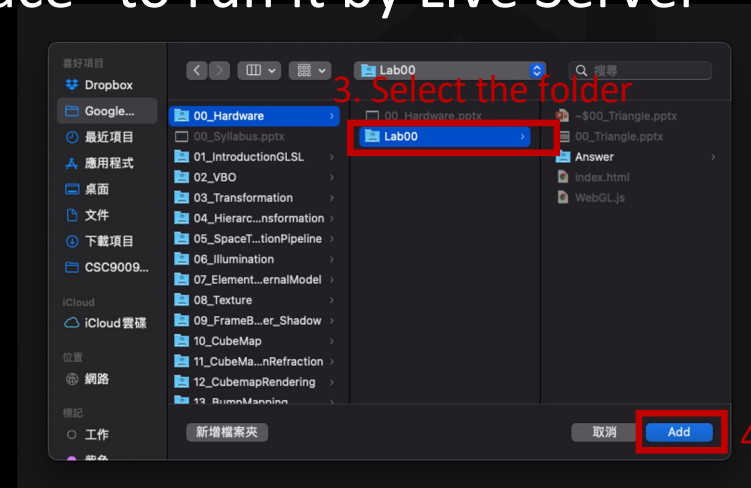
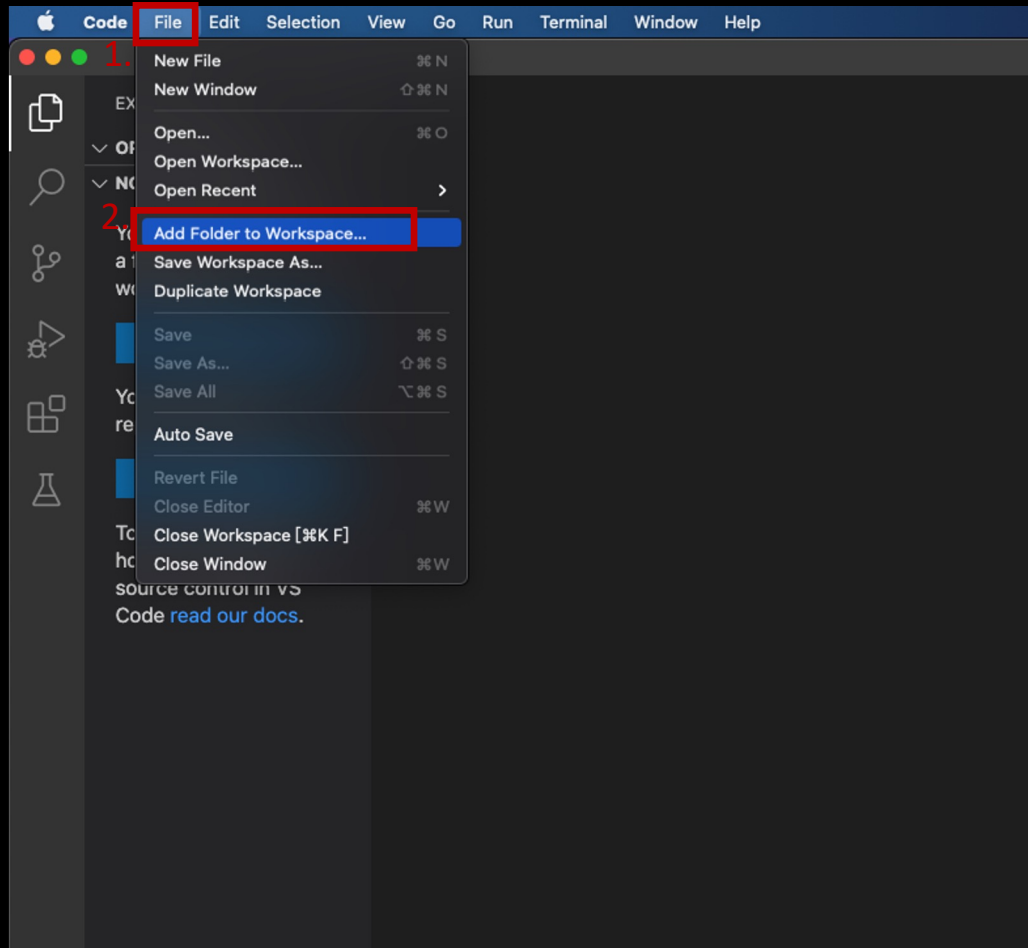
How to Run it?

- Open “VSCode”
- Install “Live Server” extension



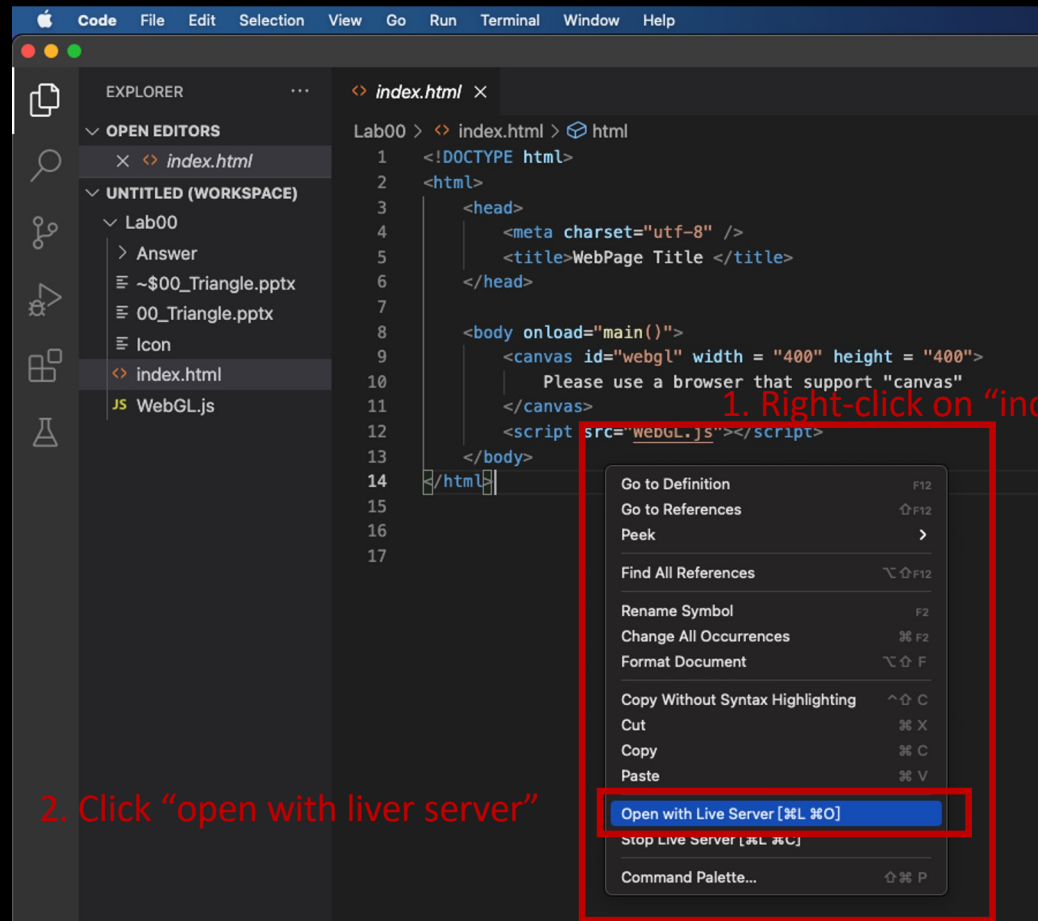
How to Run it?

- You have to add the folder into “workspace” to run it by Live Server

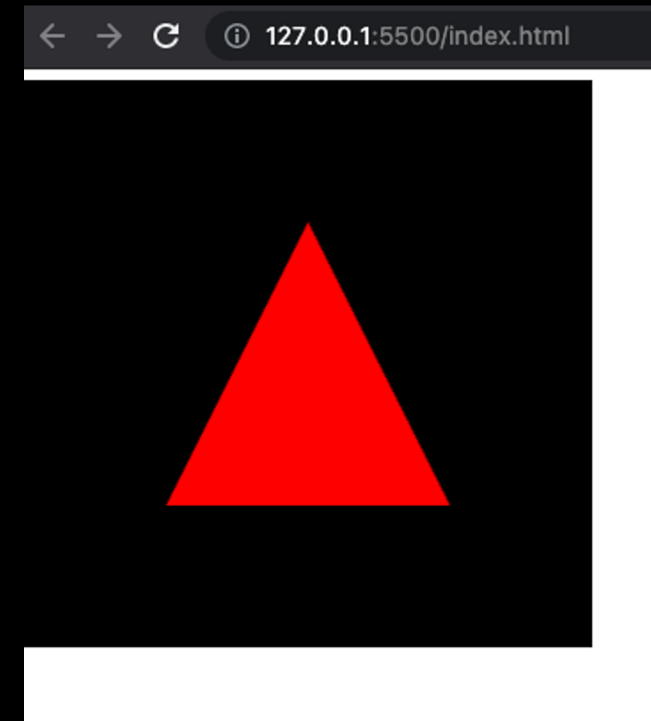


How to Run it?

- Run it by right-clicking on “index.html”



3. “live server” can create a http server and run “index.html” on your browser



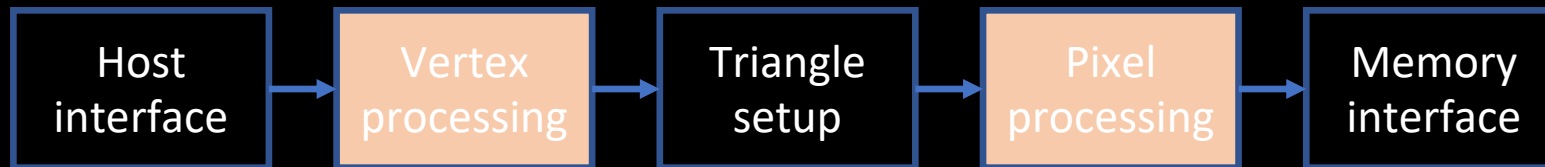
2. Index.html

- In “index.html”, we have a canvas (for WebGL to draw and set its size (Line 9))

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>WebPage Title </title>
6      </head>
7
8      <body onload="main()">
9          <canvas id="webgl" width = "400" height = "400">
10             Please use a browser that support "canvas"
11          </canvas>
12          <script src="WebGL.js"></script>
13      </body>
14  </html>
```

3. WebGL.js (Shaders)

- Open the files (index.html and WebGL.js) to edit
 - You should choose an editor you prefer (e.g. Visual Studio Code, Subline Text...)
- “Read and understand” the code, and map this code to the rendering pipeline we just learned
 - Line1 ~ Line7 is the Vertex processing in the pipeline (**Vertex shader**)
 - Line9 ~ Line14 is the pixel processing in the pipeline (**Fragment shader**)
 - **They are so-called GLSL (C-like language)**
 - **The main code segment to render images (run in graphics card to speed up the rendering)**



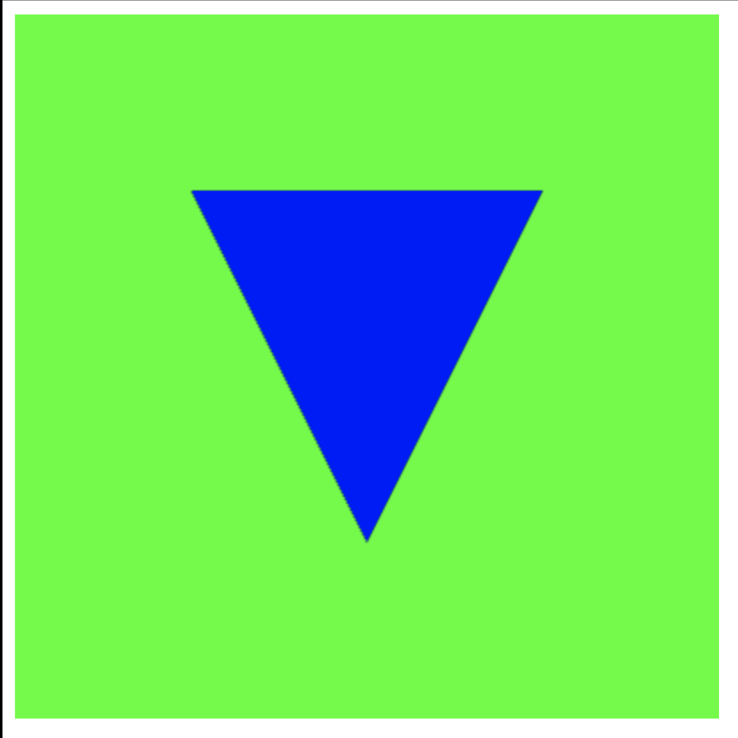
```
1  var VSHADER_SOURCE = `
2      attribute vec4 a_Position;
3      void main(){
4          //gl_Position is key variable in GLSL (pass vertex location to fragment shader)
5          gl_Position = a_Position;
6      }
7  `;
8
9  var FSHADER_SOURCE = `
10     void main(){
11         //gl_FragColor is key variable in GLSL (assign color of a pixel)
12         gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
13     }
14 `;
15
```

4. WebGL.js

- Trace the main function in WebGL.js and read the comments in code
 - Step 1 and 2: get the canvas for WebGL to draw later
 - Step 3: “Compile” the shader programs (vertex and fragment shaders)
 - You do not have to understand the details of the two functions, “createProgram” and “compileShader”, so far.
 - Step 4: tell WebGL which shader program you want to use (you may have multiple shader programs later)
 - Step 5: array of triangle vertices and number of vertices
 - Line93~Line107: pass the triangle vertices to shader programs
 - You do not need to know the details so far
 - Step 6: assign the background color and clear screen by the color
 - Step7: “draw” the triangle

What You Should Do for “Submission”

- Modify the code to draw
 - an “up-side down” “blue” triangle
 - on a “green” background



Where to modify

- WebGL.js

```
// var n = initVertexBuffers(gl, renderProgram);
///// 5. prepare the vertices for draw (we just draw 2D object here)
///// These are vertices of a triangle in 2D
var vertices = new Float32Array(
  [0.0, 0.5, -0.5, -0.5, 0.5, -0.5]
);

var n = 3; /// number of vertices

var vertexBuffer = gl.createBuffer(); ///// create a vertex buffer to store the triangle vertices
if(!vertexBuffer) {
  console.log('Failed to create the buffer object');
}

///// bind buffer and pass the vertices data
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

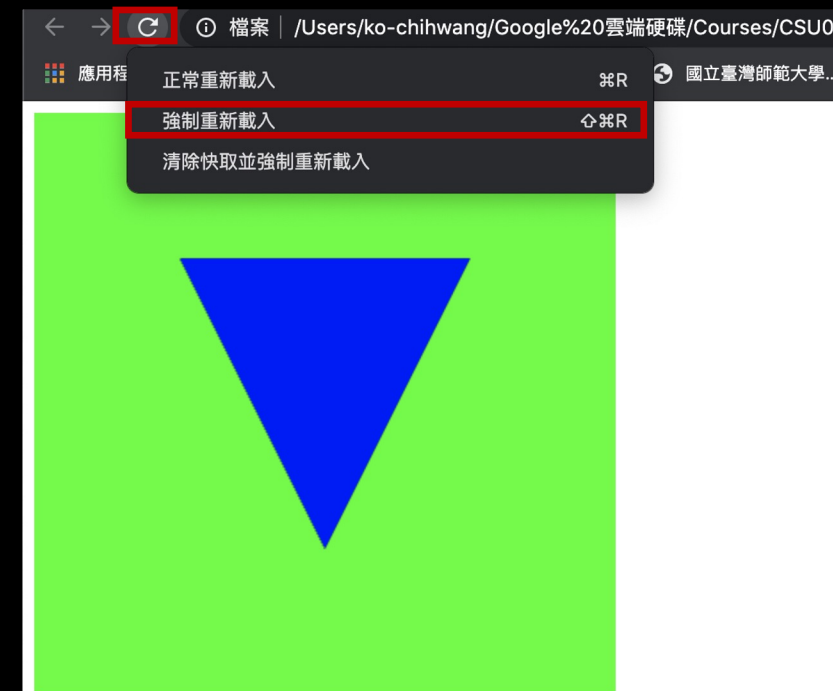
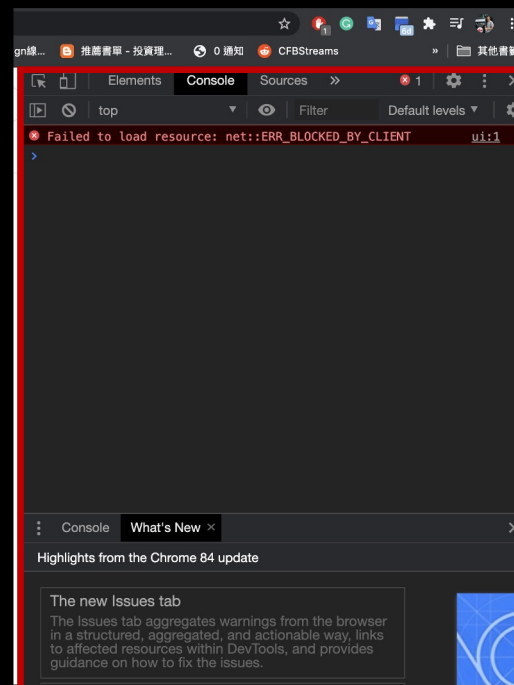
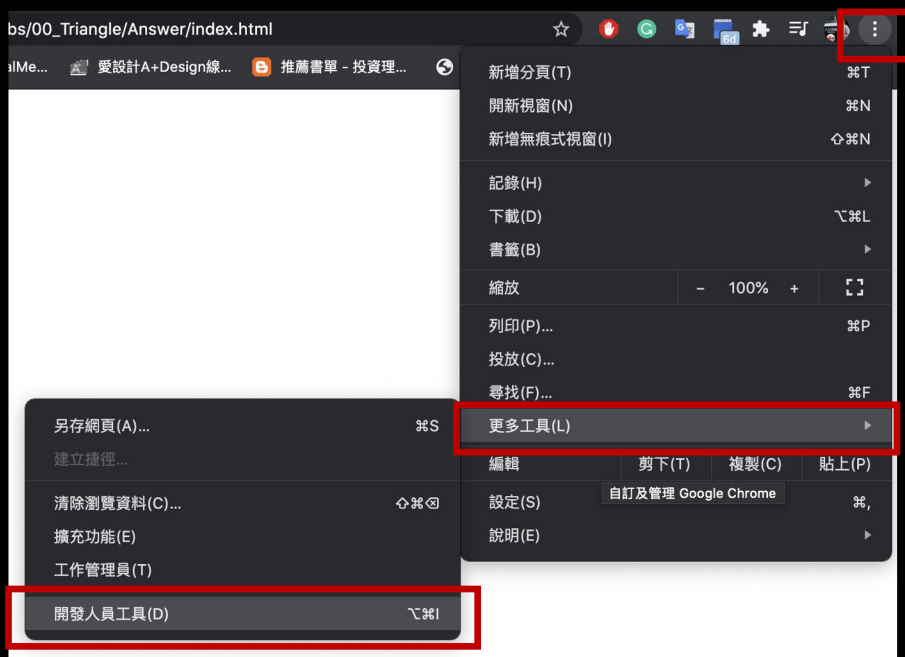
///// get the reference of the variable in the shader program
renderProgram.a_Position = gl.getAttribLocation(renderProgram, 'a_Position');
if( renderProgram.a_Position < 0 ) console.log("renderProgram.a_Position < 0"); //check you get the correct value

gl.vertexAttribPointer(renderProgram.a_Position, 2, gl.FLOAT, false, 0, 0); //setting of the vertex attribute
gl.enableVertexAttribArray(renderProgram.a_Position); //enable the vertex buffer

///// 6. clear the screen by designated background color
gl.clearColor(0.0, 0.0, 0.0, 1.0); //background color
gl.clear(gl.COLOR_BUFFER_BIT); // clear

///// 7. draw the shape
gl.drawArrays(gl.TRIANGLES, 0, n);
```

- What if the browser does not load your new code
 - You can force the browser to refresh its cache
 - Here is an example for Chrome
 - Open the developer tool
 - Long press on the refresh button, then select “hard reload”



Submission Instruction

- Create a folder
 - Put the html and js files in the folder
 - Zip the folder
 - Rename the zip file to your student ID
 - For example, if your student ID is “40312345s”, rename the zip file to “40312345s.zip”
 - Submit the renamed zip file to Moodle
- Make sure
 - you put all files in the folder to zip
 - You submit the zip file with correct name
- You won't get any point if
 - the submitted file does not follow the naming rule,
 - TA cannot run your code,
 - or cannot unzip your zip file.