

### Buffer Objects

**Creates a new buffer object:** Object createBuffer() Note: Corresponding OpenGL ES function is GenBuffers

**Binds an empty array buffer to the object:** void bindBuffer(enum target, Object buffer) target: ARRAY\_BUFFER, ELEMENT\_ARRAY\_BUFFER

**Assigns the passed values to the target:** void bufferData(enum target, Object data, enum usage) target and usage: Same as for bufferData above

### Uniforms and Attributes

**Index of the attribute:** long getAttribLocation(Object program, string name)

**Index of the uniform variable:** int getUniformLocation(Object program, string name)

**Passes the information from buffer to attribute:** void vertexAttribPointer(uint index, int size, enum type, bool normalized, long stride, long offset) type: BYTE, SHORT, UNSIGNED\_{BYTE, SHORT}, FIXED, FLOAT index: [0, MAX\_VERTEX\_ATTRIBS - 1] stride: [0, 255] offset, stride: must be a multiple of the type size in WebGL

### Storage Qualifiers

**attribute** : Linkage between a vertex shader and OpenGL ES for per-vertex data

**uniform** : Value does not change across the primitive being processed, uniforms form the linkage between a shader, OpenGL ES, and the application

**varying** : Linkage between a vertex shader and fragment shader for interpolated data

### Programs and shaders

**Creates a shader object:** Object createShader(enum type) type: VERTEX\_SHADER, FRAGMENT\_SHADER

**Creates a program:** Object createProgram()

**Assigns a source program to the object:** void shaderSource(Object shader, string source)

**Attach the shader to the program:** void attachShader(Object program, Object shader)

**Compiles the shader:** void compileShader(Object shader)

**Links the attached shaders:** void linkProgram(Object program)

**Uses the combined shaders:** void useProgram(Object program)

### Writing to the draw buffer

**Draws the given array:** void drawArrays(enum mode, int first, long count) mode: POINTS, LINE\_STRIP, LINE\_LOOP, LINES, TRIANGLE\_STRIP, TRIANGLE\_FAN, TRIANGLES first: May not be a negative value.

**Draws the passed element:** void drawElements(enum mode, long count, enum type, long offset) mode: POINTS, LINE\_STRIP, LINE\_LOOP, LINES, TRIANGLE\_STRIP, TRIANGLE\_FAN, TRIANGLES type: UNSIGNED\_BYTE, UNSIGNED\_SHORT

### Precision

**highp** : Satisfies minimum requirements for the vertex language. Optional in the fragment language.

**mediump** : Satisfies minimum requirements for the fragment language. Its range and precision is between that provided by lowp and highp.

**lowp** : Range and precision can be less than mediump, but still represents all color values for any color channel.



By **luzvare**  
[cheatography.com/luzvare/](https://cheatography.com/luzvare/)

Not published yet.  
Last updated 17th April, 2020.  
Page 1 of 1.

Sponsored by **Readable.com**  
Measure your website readability!  
<https://readable.com>