WebGL Cheat Sheet v0.2

http://blog.nihilogic.dk/

Note: It is implied that all functions and symbolic names are methods and properties on a WebGL context object

Buffers

Object **createBuffer(** *void* **)**
Create a WebGLBuffer buffer object

void **deleteBuffer(** *Object* **buffer)**
Delete a WebGLBuffer buffer object

void **bindBuffer(** *ulong* **target**, *Object* **buffer)**
Bind a buffer object. Accepted values for target are:
ARRAY_BUFFER   ELEMENT_ARRAY_BUFFER

void **bufferData(** *ulong* **target**, *Object* **dta**, *ulong* **usage)**
Create and initialize a buffer object's data store.
Accepted values for usage are:
STREAM_DRAW   STATIC_DRAW
DYNAMIC_DRAW

void **bufferData(** *ulong* **target**, *long* **size**, *ulong* **usage)**
Set the size of a buffer object's data store.

void **bufferSubData(** *ulong* **target**, *ulong* **offset**, *Object* **data)**
Update a subset of a buffer object's data store.

any **getBufferParameter(** *ulong* **target**, *ulong* **value)**
Return parameter, pname, of a buffer object:
BUFFER_SIZE   BUFFER_USAGE

bool **isBuffer(** *Object* **buffer)**
Determine if an object is a buffer object.

any **getParameter(** *ulong* **pname)**
Relevant parameters:
ARRAY_BUFFER_BINDING
ELEMENT_ARRAY_BUFFER_BINDING

Renderbuffers

Object **createRenderbuffer(** *void* **)**
Create a renderbuffer object

void **deleteRenderbuffer(** *Object* **buffer)**
Delete a renderbuffer object.

void **bindRenderbuffer(** *ulong* **target**, *Object* **buffer)**
Bind a renderbuffer, target must be RENDERBUFFER.

any **getRenderbufferParameter(** *ulong* **target**, *ulong* **pname)**
Return parameter, pname, of a renderbuffer object:
RENDERBUFFER_WIDTH
RENDERBUFFER_HEIGHT
RENDERBUFFER_INTERNAL_FORMAT
RENDERBUFFER_RED_SIZE
RENDERBUFFER_GREEN_SIZE
RENDERBUFFER_BLUE_SIZE
RENDERBUFFER_ALPHA_SIZE
RENDERBUFFER_DEPTH_SIZE
RENDERBUFFER_STENCIL_SIZE

void **renderbufferStorage(** *ulong* **target**, *ulong* **format**, *ulong* **width**, *ulong* **height)**
Create and initialize a renderbuffer object's data store.
Accepted values for format are:
RGBA4   RGB565
RGB5_A1   DEPTH_COMPONENT16
STENCIL_INDEX8

bool **isRenderbuffer(** *Object* **buffer)**
Determine if an object is a renderbuffer object.

any **getParameter(** *ulong* **pname)**
Relevant parameters:
RENDERBUFFER_BINDING
MAX_RENDERBUFFER_SIZE

Framebuffers

Object **createFramebuffer(** *void* **)**
Create a framebuffer object

void **deleteFramebuffer(** *Object* **buffer)**
Delete a framebuffer object.

void **bindFramebuffer(** *ulong* **target**, *Object* **buffer)**
Bind a framebuffer, target must be FRAMEBUFFER.

ulong **checkFramebufferStatus(** *ulong* **target)**
Return the framebuffer completeness status of a framebuffer object. Return values are:
FRAMEBUFFER_COMPLETE
FRAMEBUFFER_INCOMPLETE_ATTACHMENT
FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT
FRAMEBUFFER_INCOMPLETE_DIMENSIONS
FRAMEBUFFER_UNSUPPORTED

ulong **framebufferRenderbuffer(** *ulong* **target**, *ulong* **att**, *ulong* **rbtarget**, *Object* **rbuffer)**
Attach a renderbuffer object to a framebuffer object.
Accepted values for attachment are:
DEPTH_ATTACHMENT   COLOR_ATTACHMENT0
STENCIL_ATTACHMENT

any **getFramebufferAttachmentParameter(** *ulong* **target**, *ulong* **attachment**, *ulong* **pname)**
Return attachment parameters of a framebuffer object.
Accepted values for attachment are:
FRAMEBUFFER_ATTACHMENT_OBJECT_TYPE
FRAMEBUFFER_ATTACHMENT_OBJECT_NAME
FRAMEBUFFER_ATTACHMENT_TEXTURE_LEVEL
FRAMEBUFFER_ATTACHMENT_TEXTURE_
CUBE_MAP_FACE

ulong **framebufferTexture2D(** *ulong* **target**, *ulong* **att**, *ulong* **textarget**, *Object* **tex**, *ulong* **level)**
Attach a texture image to a framebuffer object.
Accepted values for textarget are:
TEXTURE_2D
TEXTURE_CUBE_MAP_POSITIVE_X
TEXTURE_CUBE_MAP_NEGATIVE_X
TEXTURE_CUBE_MAP_POSITIVE_Y
TEXTURE_CUBE_MAP_NEGATIVE_Y
TEXTURE_CUBE_MAP_POSITIVE_Z
TEXTURE_CUBE_MAP_NEGATIVE_Z

void **pixelStorei(** *ulong* **pname**, *long* **param)**
Set pixel storage modes. Accepted pname values are:
PACK_ALIGNMENT   UNPACK_ALIGNMENT

Array **readPixels(** *long* **x**, *long* **y**, *ulong* **width**, *ulong* **height**, *ulong* **format**, *ulong* **type)**
Read a block of pixels from the frame buffer. Accepted format values are:
ALPHA   RGB   RGBA
Accepted type values are:
UNSIGNED_BYTE
UNSIGNED_SHORT_4_4_4_4
UNSIGNED_SHORT_5_5_5_1
UNSIGNED_SHORT_5_6_5

bool **isFramebuffer(** *Object* **buffer)**
Determine if an object is a framebuffer object.

any **getParameter(** *ulong* **pname)**
Relevant parameters:
RED_BITS   GREEN_BITS
BLUE_BITS   ALPHA_BITS
FRAMEBUFFER_BINDING

Program objects

Object **createProgram(** *void* **)**
Create a program object

void **validateProgram(** *Object* **program)**
Validate a program object

void **linkProgram(** *Object* **program)**
Link a program object

void **useProgram(** *ulong* **program)**
Install a program as part of current rendering state

void **deleteProgram(** *Object* **program)**
Delete a program object

any **getProgramParameter(** *Object* **pgm**, *ulong* **pname)**
Return parameter, pname, from a program object:

LINK_STATUS	INFO_LOG_LENGTH
DELETE_STATUS	VALIDATE_STATUS
ATTACHED_SHADERS	ACTIVE_UNIFORMS
ACTIVE_ATTRIBUTES	
ACTIVE_ATTRIBUTE_MAX_LENGTH	
ACTIVE_UNIFORM_MAX_LENGTH	

string **getProgramInfoLog(** *Object* **program)**
Return the information log for a program object

bool **isProgram(** *Object* **program)**
Determine if an object is a program object.

any **getParameter(** *ulong* **pname)**
Relevant parameters: CURRENT_PROGRAM

Shaders

Object **createShader(** *ulong* **shaderType)**
Create a shader object. Parameter shaderType must be VERTEX_SHADER or FRAGMENT_SHADER.

void **compileShader(** *Object* **shader)**
Compile a shader object

void **attachShader(** *Object* **program**, *Object* **shader)**
void **detachShader(** *Object* **program**, *Object* **shader)**
Attach/detach a shader object.

void **deleteShader(** *Object* **shader)**
Delete a shader object

any **getShaderParameter(***Object* **shader**, *ulong* **pname)**
Return parameter, pname, from a shader object:

SHADER_TYPE	DELETE_STATUS
COMPILE_STATUS	INFO_LOG_LENGTH
SHADER_SOURCE_LENGTH	

string **getShaderInfoLog(** *Object* **shader)**
Return the information log for a shader object

string **getShaderSource(** *Object* **shader)**
void **shaderSource(** *Object* **shader**, *string* **source)**
Get/set the source code in a shader object

Array **getAttachedShaders**[1] **(** *Object* **program)**
Return the shader objects attached to a program.

bool **isShader(** *Object* **shader)**
Determine if an object is a shader object.

any **getParameter(** *ulong* **pname)**
Relevant parameters:

SHADER_COMPILER	MAX_VARYING_VECTORS

Culling

void **enable|disable(** CULL_FACE **)**
void **cullFace(** *ulong* **mode)**
Specify facet culling mode, accepted values are:

FRONT	BACK	FRONT_AND_BACK

void **frontFace(** *ulong* **mode)**
Define front/back-facing mode: CW or CCW

any **getParameter(** *ulong* **pname)**
Parameters: CULL_FACE_MODE or FRONT_FACE

Textures

Object **createTexture(** *void* **)**
Create a texture

void **deleteTexture(** *Object* **texture)**
Delete a texture.

void **bindTexture(** *ulong* **target**, *Object* **texture)**
Bind a texture to a texturing target. Accepted values for target are:

TEXTURE_2D	TEXTURE_CUBE_MAP

void **activeTexture(** *ulong* **texture)**
Select active texture unit.

any **getTexParameter(** *ulong* **target**, *ulong* **pname)**
Return parameter, pname, of a texture:

TEXTURE_WRAP_S	TEXTURE_MAG_FILTER
TEXTURE_WRAP_T	TEXTURE_MIN_FILTER

void **texParameterf(** *ulong* **target**, *ulong* **pname**, *float* **v)**
void **texParameteri(** *ulong* **target**, *ulong* **pname**, *long* **v)**
Set texture parameters.

void **texImage2D(** *ulong* **target**, *long* **level**, *ulong* **intformat**, *ulong* **width**, *ulong* **height**, *long* **border**, *ulong* **format**, *ulong* **type**, *Object* **data)**
Specify a two-dimensional texture image from a WebGLArray of pixel data. See readPixels for accepted type values. Accepted values for intformat and format are:

ALPHA	RGB	RGBA
LUMINANCE	LUMINANCE_ALPHA	

void **texImage2D(** *ulong* **target**, *long* **level**, *Object* **data**, [*bool* **flipY**], [*bool* **asPreMultipliedAlpha**] **)**
Specify a two-dimensional texture image from either an ImageData object or a HTMLImageElement, HTMLCanvasElement or HTMLVideoElement.

void **texSubImage2D(** *ulong* **target**, *long* **level**, *long* **xoffset**, *long* **yoffset**, *ulong* **width**, *ulong* **height**, *ulong* **format**, *ulong* **type**, *Object* **data)**
Specify a two-dimensional texture subimage from a WebGLArray of pixel data.

void **texSubImage2D(** *ulong* **target**, *long* **level**, *long* **xoffset**, *long* **yoffset**, *Object* **data**, [*bool* **flipY**], [*bool* **asPreMultipliedAlpha**] **)**
Specify a two-dimensional texture subimage from either an ImageData object or a HTMLImageElement, HTMLCanvasElement or a HTMLVideoElement.

void **copyTexImage2D(** *ulong* **target**, *long* **level**, *ulong* **intformat**, *long* **x**, *long* **y**, *ulong* **width**, *ulong* **height**, *long* **border)**
Copy pixels into a 2D texture image. See framebufferTexture2D for accepted target values.

void **copyTexSubImage2D(** *ulong* **target**, *long* **level**, *ulong* **intformat**, *long* **xoffset**, *long* **yoffset**, *long* **x**, *long* **y**, *ulong* **width**, *ulong* **height)**
Copy a two-dimensional texture subimage.

void **generateMipmap(** *ulong* **target)**
Generate a complete set of mipmaps for a texture.

bool **isTexture(** *Object* **buffer)**
Determine if an object is a texture.

any **getParameter(** *ulong* **pname)**
Relevant parameters:

TEXTURE_BINDING_2D
TEXTURE_BINDING_CUBE_MAP
MAX_TEXTURE_SIZE
MAX_CUBE_MAP_TEXTURE_SIZE
ACTIVE_TEXTURE
MAX_TEXTURE_IMAGE_UNITS
MAX_VERTEX_TEXTURE_IMAGE_UNITS
MAX_COMBINED_TEXTURE_IMAGE_UNITS

Blending

void **enable|disable(** BLEND **)**
Enable/disable blending

void **blendFunc(** *ulong* **sfactor**, *ulong* **dfactor)**
Specify pixel arithmetic. Accepted values for sfactor and dfactor are:

ZERO	ONE
SRC_COLOR	DST_COLOR
SRC_ALPHA	DST_ALPHA
CONSTANT_COLOR	CONSTANT_ALPHA
ONE_MINUS_SRC_ALPHA	ONE_MINUS_DST_ALPHA
ONE_MINUS_SRC_COLOR	ONE_MINUS_DST_COLOR
ONE_MINUS_CONSTANT_COLOR	
ONE_MINUS_CONSTANT_ALPHA	

In addition, sfactor can also be SRC_ALPHA_SATURATE

void **blendFuncSeparate(** *ulong* **srcRGB**, *ulong* **dstRGB**, *ulong* **srcAlpha**, *ulong* **dstAlpha)**
Specify pixel arithmetic for RGB and alpha components separately.

void **blendEquation(** *ulong* **mode)**
Specify the equation used for both the RGB blend equation and the Alpha blend equation. Accepted values for mode are:

FUNC_ADD	FUNC_SUBTRACT
FUNC_REVERSE_SUBTRACT	

void **blendEquationSeparate(** *ulong* **modeRGB**, *ulong* **modeAlpha)**
Set the RGB blend equation and the alpha blend equation separately.

void **blendColor(** *float* **red**, *float* **green**, *float* **blue**, *float* **alpha)**
Set the blend color

any **getParameter(** *ulong* **pname)**
Relevant parameters:

BLEND	BLEND_COLOR
BLEND_DST_RGB	BLEND_SRC_RGB
BLEND_DST_ALPHA	BLEND_SRC_ALPHA
BLEND_EQUATION_RGB	BLEND_EQUATION_ALPHA

Depth buffer

void **enable|disable(** DEPTH_TEST **)**
Enable/disable depth testing.

void **depthFunc(** *ulong* **func)**
Specify the value used for depth buffer comparisons. Parameter func is one of:

NEVER	LESS	EQUAL	LEQUAL
GREATER	NOTEQUAL	GEQUAL	ALWAYS

void **depthMask(** *bool* **flag)**
Enable or disable writing into the depth buffer.

void **depthRange(** *float* **nearVal**, *float* **farVal)**
Specify mapping of depth values from normalized device coordinates to window coordinates.

void **clearDepth(** *float* **depth)**
Specify the clear value for the depth buffer

void **enable|disable(** POLYGON_OFFSET_FILL **)**
Enable/disable polygon offset.

void **polygonOffset(** *float* **factor**, *float* **units)**
Set the scale and units used to calculate depth values.

any **getParameter(** *ulong* **pname)**
Relevant parameters:

DEPTH_TEST	DEPTH_RANGE
DEPTH_WRITEMASK	DEPTH_CLEAR_VALUE
DEPTH_FUNC	DEPTH_BITS
POLYGON_OFFSET_UNITS	POLYGON_OFFSET_FACTOR

Stencil buffer

void **enable|disable(** STENCIL_TEST **)**
Enable/disable stencil testing.

void **stencilFunc(** *ulong* **func**, *long* **ref**, *ulong* **mask)**
Set front and back function and reference value for stencil testing. Parameter func is one of:

NEVER	LESS	EQUAL	LEQUAL
GREATER	NOTEQUAL	GEQUAL	ALWAYS

void **stencilFuncSeparate(** *ulong* **face**, *ulong* **func**, *long* **ref**, *ulong* **mask)**
Set front and/or back function and reference value for stencil testing. Accepted values for face are:

FRONT	BACK	FRONT_AND_BACK

void **stencilMask(** *ulong* **mask)**
Control the front and back writing of individual bits in the stencil planes.

void **stencilMaskSeparate(** *ulong* **face**, *ulong* **mask)**
Control the front and/or back writing of individual bits in the stencil planes.

void **stencilOp(** *ulong* **sfail**, *ulong* **dpfail**, *ulong* **dppass)**
Set front and back stencil test actions. Accepted values for sfail, dpfail and dppass are:

KEEP	ZERO	INCR	INCR_WRAP
REPLACE	INVERT	DECR	DECR_WRAP

void **stencilOpSeparate(** *ulong* **face**, *ulong* **sfail**, *ulong* **dpfail**, *ulong* **dppass)**
Set front and/or back stencil test actions.

void **clearStencil(** *long* **s)**
Specify the clear value for the stencil buffer.

any **getParameter(** *ulong* **pname)**
Relevant parameters:

STENCIL_TEST	STENCIL_CLEAR_VALUE
STENCIL_FUNC	STENCIL_FAIL
STENCIL_REF	STENCIL_VALUE_MASK
STENCIL_WRITEMASK	STENCIL_BACK_FUNC
STENCIL_BACK_FAIL	STENCIL_BACK_REF
STENCIL_BITS	STENCIL_BACK_WRITEMASK
STENCIL_BACK_VALUE_MASK	
STENCIL_BACK_PASS_DEPTH_FAIL	
STENCIL_BACK_PASS_DEPTH_PASS	
STENCIL_PASS_DEPTH_FAIL	
STENCIL_PASS_DEPTH_PASS	

Array data

Object **createFloatArray(** *Array* **values)**
Object **createByteArray(** *Array* **values)**
Object **createUnsignedByteArray(** *Array* **values)**
Object **createShortArray(** *Array* **values)**
Object **createUnsignedShortArray(** *Array* **values)**
Object **createIntArray(** *Array* **values)**
Object **createUnsignedIntArray(** *Array* **values)**
Create WebGL array objects from JS arrays.

void **drawArrays(** *ulong* **mode**, *long* **first**, *ulong* **count)**
Render primitives from array data. Accepted mode values are:

POINTS	LINES	LINE_LOOP
LINE_STRIP	TRIANGLES	TRIANGLE_STRIP
TRIANGLE_FAN		

void **drawElements(** *ulong* **mode**, *ulong* **count**, *ulong* **type**, *ulong* **offset)**
Render primitives from array data. Accepted type values are:

UNSIGNED_BYTE	UNSIGNED_SHORT

Uniform variables

ulong **getUniformLocation(***Object* **program**, *string* **name)**
Return the location of a uniform variable.

Object **getActiveUniform(** *Object* **program**, *ulong* **idx)**
Return information about an active uniform variable.
Returns an object: { size: ..., type: ..., name: ... }.

any **getUniform(** *Object* **program**, *ulong* **location)**
Return the value of a uniform variable

void **uniform[1234][if](** *ulong* **location**, **...)**
Specify 1-4 float or int values of a uniform variable.

void **uniform[1234][if]v(** *ulong* **location**, *Array* **v)**
Specify the value of a uniform variable as an array of 1-4 float or int values.

void **uniformMatrix[234]fv(** *ulong* **location**, *bool* **transpose**, *Object* **value)**
Specify the value of a matrix uniform variable using arrays of float values.

any **getParameter(** *ulong* **pname)**
Relevant parameters:
- MAX_VERTEX_UNIFORM_VECTORS
- MAX_FRAGMENT_UNIFORM_VECTORS

Attribute variables

ulong **getAttribLocation(** *Object* **program**, *string* **name)**
Return the location of an attribute variable.

Object **getActiveAttrib(** *Object* **program**, *ulong* **idx)**
Return information about an active attribute variable.
Returns an object: { size: ..., type: ..., name: ... }.

any **getVertexAttrib(** *Object* **idx**, *ulong* **pname)**
Return a generic vertex attribute parameter. Accepted pname values are:
- VERTEX_ATTRIB_ARRAY_ENABLED
- VERTEX_ATTRIB_ARRAY_SIZE
- VERTEX_ATTRIB_ARRAY_STRIDE
- VERTEX_ATTRIB_ARRAY_TYPE
- VERTEX_ATTRIB_ARRAY_NORMALIZED
- VERTEX_ATTRIB_ARRAY_BUFFER_BINDING
- CURRENT_VERTEX_ATTRIB

void **vertexAttribPointer(** *ulong* **idx**, *long* **size**, *ulong* **type**, *bool* **norm**, *long* **stride**, *ulong* **offset)**
Define an array of generic vertex attribute data.
Accepted type values are:

FIXED	BYTE	UNSIGNED_BYTE
FLOAT	SHORT	UNSIGNED_SHORT

void **vertexAttrib[1234]f(** *ulong* **idx**, **...)**
Specify 1-4 float values of a generic vertex attribute.

void **vertexAttrib[1234]fv(** *ulong* **idx**, *Array* **v)**
Specify the value of a generic vertex attribute as an array of 1-4 float values.

void **bindAttribLocation(** *Object* **program**, *ulong* **idx**, *string* **name)**
Associate a generic vertex attribute index with a named attribute variable.

void **enableVertexAttribArray(** *ulong* **idx)**
void **disableVertexAttribArray(** *ulong* **idx)**
Enable or disable a generic vertex attribute array

any **getParameter(** *ulong* **pname)**
Relevant parameters:
- MAX_VERTEX_ATTRIBS

Multisampling

void **enable|disable(** SAMPLE_COVERAGE **)**
If enabled, the fragment's coverage is ANDed with the temporary coverage value.

void **enable|disable(** SAMPLE_ALPHA_TO_COVERAGE **)**
If enabled, use the alpha value at the corresponding sample location to determine each bit.

void **sampleCoverage(** *float* **value**, *bool* **invert)**
Specify multisample coverage parameters.

any **getParameter(** *ulong* **pname)**
Relevant parameters:
- SAMPLE_COVERAGE_VALUE
- SAMPLE_COVERAGE_INVERT
- SAMPLE_BUFFERS
- SAMPLES

Misc.

void **viewport(** *long* **x**, *long* **y**, *ulong* **w**, *ulong* **h)**
Set the viewport.

void **lineWidth(** *float* **width)**
Specify the width of rasterized lines.

void **flush(** *void* **)**
Force execution of GL commands in finite time.

void **finish(** *void* **)**
Block until all GL execution is complete.

void **clear(** *ulong* **mask)**
Clear buffers to preset values, mask is the bitwise OR of one or more of

COLOR_BUFFER_BIT	DEPTH_BUFFER_BIT
STENCIL_BUFFER_BIT	

void **enable|disable(** DITHER **)**
Enable/disable dithering of color comps or indices.

void **colorMask(** *bool* **red**, *bool* **green**, *bool* **blue**, *bool* **alpha)**
Enable and disable writing of frame buffer color components.

void **clearColor(** *float* **red**, *float* **green**, *float* **blue**, *float* **alpha)**
Specify clear values for the color buffers.

void **scissor(** *long* **x**, *long* **y**, *ulong* **width**, *ulong* **height)**
Define the scissor box.

ulong **getError(** *void* **)**
Return error information. Error values are:

OUT_OF_MEMORY	INVALID_ENUM
INVALID_VALUE	INVALID_OPERATION
INVALID_FRAMEBUFFER_OPERATION	
NO_ERROR	

any **getParameter(** *ulong* **pname)**
Parameters values:
- VIEWPORT
- MAX_VIEWPORT_DIMS
- COLOR_CLEAR_VALUE
- SCISSOR_BOX
- LINE_WIDTH
- ALIASED_POINT_SIZE_RANGE
- ALIASED_LINE_WIDTH_RANGE
- COLOR_WRITEMASK
- SUBPIXEL_BITS

Notes: [1] Not implemented in one or more browsers.
Sources: https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html (2010-02-16)
http://www.khronos.org/opengles/sdk/docs/man/ (2009-10-23)
http://mxr.mozilla.org/mozilla-central/source/content/canvas/src/WebGLContextGL.cpp (2010-02-16)
http://trac.webkit.org/browser/trunk/WebCore/html/canvas/WebGLRenderingContext.cpp (2010-02-16)