WebGL Cheat Sheet v0.2

http:

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

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

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

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

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

Framebuffers

Object **createFramebuffer(** *voic*
Create a framebuffer obj

void **deleteFramebuffer(** *Obj*
Delete a framebuffer obj

void **bindFramebuffer(** *ulong*
Bind a framebuffer, targe

ulong **checkFramebufferStatu**
Return the framebuffer c
framebuffer object. Retur
FRAMEBUFFER_COMPLE
FRAMEBUFFER_INCOMP
FRAMEBUFFER_INCOMP
FRAMEBUFFER_INCOMP
FRAMEBUFFER_UNSUPP

ulong **framebufferRenderbuffe**
ulong **att**, *ulong* **rbta**
Attach a renderbuffer obj
Accepted values for attac
DEPTH_ATTACHMENT
STENCIL_ATTACHMENT

any **getFramebufferAttachn**
ulong **target**, *ulong* a
Return attachment param
object. Accepted values f
FRAMEBUFFER_ATTACH
FRAMEBUFFER_ATTACH
FRAMEBUFFER_ATTACH
FRAMEBUFFER_ATTACH
CUBE_MAP_FACE

ulong **framebufferTexture2D(**
ulong **textarget**, *Obj*
Attach a texture image to
Accepted values for texta
TEXTURE_2D
TEXTURE_CUBE_MAP_P
TEXTURE_CUBE_MAP_N
TEXTURE_CUBE_MAP_P
TEXTURE_CUBE_MAP_N
TEXTURE_CUBE_MAP_P
TEXTURE_CUBE_MAP_N

void **pixelStorei(** *ulong* **pname**
Set pixel storage modes.
PACK_ALIGNMENT

Array **readPixels(** *long* **x**, *long*
ulong **height**, *ulong*
Read a block of pixels fro
format values are:
ALPHA   RGB
Accepted type values are:
UNSIGNED_BYTE
UNSIGNED_SHORT_4_4
UNSIGNED_SHORT_5_5
UNSIGNED_SHORT_5_6

bool **isFramebuffer(** *Object* **b**
Determine if an object is

any **getParameter(** *ulong* **pna**
Relevant parameters:
RED_BITS
BLUE_BITS
FRAMEBUFFER_BINDING

Textures

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

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

void **bindTexture(** *ulong* **targ**
Bind a texture to a textur
for target are:
TEXTURE_2D

void **activeTexture(** *ulong* **tex**
Select active texture unit

any **getTexParameter(** *ulong*
Return parameter, pname
TEXTURE_WRAP_S
TEXTURE_WRAP_T

void **texParameterf(** *ulong* **ta**
void **texParameteri(** *ulong* **ta**
Set texture parameters.

void **texImage2D(** *ulong* **targe**
ulong **intformat**, *ulo*
border, *ulong* **forma**
Specify a two-dimensiona
WebGLArray of pixel data
type values. Accepted val
are:
ALPHA   RGB
LUMINANCE   LUMINA

void **texImage2D(** *ulong* **targe**
[*bool* **flipY**], [*bool* **as**
Specify a two-dimensiona
an ImageData object or a
HTMLCanvasElement or H

void **texSubImage2D(** *ulong* t
long **xoffset**, *long* **yo**
height, *ulong* **forma**
Specify a two-dimensiona
WebGLArray of pixel data

void **texSubImage2D(** *ulong* t
long **xoffset**, *long* **yo**
flipY], [*bool* **asPreMu**
Specify a two-dimensiona
either an ImageData obje
HTMLCanvasElement or a

void **copyTexImage2D(** *ulong*
ulong **intformat**, *lon*
ulong **height**, *long* **b**
Copy pixels into a 2D tex
framebufferTexture2D fo

void **copyTexSubImage2D(** *ul*
ulong **intformat**, *lon*
x, *long* **y**, *ulong* **widt**
Copy a two-dimensional t

void **generateMipmap(** *ulong*
Generate a complete set

bool **isTexture(** *Object* **buffer**
Determine if an object is

any **getParameter(** *ulong* **pna**
Relevant parameters:
TEXTURE_BINDING_2D
TEXTURE_BINDING_CUE
MAX_TEXTURE_SIZE
MAX_CUBE_MAP_TEXTU
ACTIVE_TEXTURE
MAX_TEXTURE_IMAGE_
MAX_VERTEX_TEXTURE_
MAX_COMBINED_TEXTU

Stencil buffer

void **enable|disable(** STENCIL
Enable/disable stencil tes

void **stencilFunc(** *ulong* **func**,
Set front and back functio
stencil testing. Paramete
NEVER   LESS
GREATER   NOTEQUAL

void **stencilFuncSeparate(** *ul*
long **ref**, *ulong* **mask**
Set front and/or back fun
stencil testing. Accepted
FRONT   BACK

void **stencilMask(** *ulong* **mask**
Control the front and bac
the stencil planes.

void **stencilMaskSeparate(** *ul*
Control the front and/or
in the stencil planes.

void **stencilOp(** *ulong* **sfail**, *ul*
Set front and back stencil
for sfail, dpfail and dppas
KEEP   ZERO
REPLACE   INVERT

void **stencilOpSeparate(** *ulon*
ulong **dpfail**, *ulong* d
Set front and/or back ste

void **clearStencil(** *long* **s)**
Specify the clear value fo

any **getParameter(** *ulong* **pna**
Relevant parameters:
STENCIL_TEST
STENCIL_FUNC
STENCIL_REF
STENCIL_WRITEMASK
STENCIL_BACK_FAIL
STENCIL_BITS
STENCIL_BACK_VALUE_
STENCIL_BACK_PASS_D
STENCIL_BACK_PASS_D
STENCIL_PASS_DEPTH_
STENCIL_PASS_DEPTH_

Array data

Object **createFloatArray(** *Array*
Object **createByteArray(** *Array*
Object **createUnsignedByteArr**
Object **createShortArray(** *Array*
Object **createUnsignedShortAr**
Object **createIntArray(** *Array* va
Object **createUnsignedIntArray**
Create WebGL array obje

void **drawArrays(** *ulong* **mode**,
Render primitives from ar
values are:
POINTS   LINES
LINE_STRIP   TRIANG
TRIANGLE_FAN

void **drawElements(** *ulong* **mo**
ulong **type**, *ulong* **of**
Render primitives from ar
values are:
UNSIGNED_BYTE

Multisampling

void **enable|disable(** SAMPLE
If enabled, the fragment's
temporary coverage value

void **enable|disable(** SAMPLE
If enabled, use the alpha
sample location to detern

void **sampleCoverage(** *float* v

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

Specify multisample cover

any **getParameter(** *ulong* **pna**
Relevant parameters:
SAMPLE_COVERAGE_VA
SAMPLE_COVERAGE_INV
SAMPLE_BUFFERS
SAMPLES

Misc.

void **viewport(** *long* **x**, *long* **y**,
Set the viewport.

void **lineWidth(** *float* **width)**
Specify the width of raste

void **flush(** *void* **)**
Force execution of GL cor

void **finish(** *void* **)**
Block until all GL executio

void **clear(** *ulong* **mask)**
Clear buffers to preset va
of one or more of
COLOR_BUFFER_BIT
STENCIL_BUFFER_BIT

void **enable|disable(** DITHER
Enable/disable dithering

void **colorMask(** *bool* **red**, *boo*
bool **blue**, *bool* **alpha**
Enable and disable writing
components.

void **clearColor(** *float* **red**, *flo*
float **blue**, *float* **alph**
Specify clear values for th

void **scissor(** *long* **x**, *long* **y**, *ul*
Define the scissor box.

ulong **getError(** *void* **)**
Return error information.
OUT_OF_MEMORY
INVALID_VALUE
INVALID_FRAMEBUFFER
NO_ERROR

any **getParameter(** *ulong* **pna**
Parameters values:
VIEWPORT
MAX_VIEWPORT_DIMS
COLOR_CLEAR_VALUE
SCISSOR_BOX
LINE_WIDTH
ALIASED_POINT_SIZE_R
ALIASED_LINE_WIDTH_
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 (2010-0
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-
http://trac.webkit.org/browser/trunk/WebCore/html/canvas/WebGLRenderingContext.cpp (2010-0