pad(gamepadId: number): Gamepad - returns the gamepad status object

mix(obj1: object, obj2: object, ...): object - shallow copy all properties of passed objects into a new object

augment(target: object, obj1: object, ...) - augment object by deeply copying properties from all supplied objects

supplement(target: object, obj1: object, ...) - supplement object by deeply copying missing properties from all supplied object

before(obj: object, functionName: string, patchFunction: function) - chain the supplied **patchFunction** before **obj.functionName**

after(obj: object, functionName: string, patchFunction: function) - chain the supplied **patchFunction** after **obj.functionName**

chain(fn1: function, fn2: function): function - returns a chain container function that applies both functions

isFun(target: any): boolen - determines if **target** is a function

isObj(target: any): boolen - determines if **target** is an object

isString(target: any): boolen - determines if **target** is a String

isNumber(target: any): boolen - determines if **target** is a Number

isFrame(target: any): boolen - determines if **target** is a Frame

isArray(target: any): boolen - determines if **target** is an Array

isEmpty(target: any | object | array): boolean - checks if the **target** object/array is empty

abs(n: number): number - returns the absolute value of **n**

pow(n: number, p: number): number - returns **n** raised to the power of **p**

sqrt(n: number): number - returns the positive square root of **n**

min(n1: number, n2: number, ...: number): number - returns the smallest of numbers

max(n1: number, n2: number, ...: number): number - returns the largest of numbers

ceil(n: number): number - returns the smallest integer greater than or equal to **n**

floor(n: number): number - returns the largest integer less than or equal to **n**

round(n: number): number - returns **n** rounded to the nearest integer

sin(a: number/radians): number - returns the sine of the angle **a**

cos(a: number/radians): number - returns the cosine of the angle **a**

tan(a: number/radians): number - returns the tangent of the angle **a**

acos(n: number): number - returns the arccosine of the number **n**

asin(n: number): number - returns the arcsine of the number **n**

atan(n: number): number/radians - returns the arctangent of the number **n**

atan2(y: number, x: number): number/radians - returns an angle whose tangent is **y/x**

rnd(n1: number, n2: number): number - returns a pseudo-random number between [0..1] or between provided values

RND(i1: number/integer, i2: number/integer): number - returns a pseudo-random integer between provided values

limit(v: number, n1: number, n2: number): number - returns the **v** value limited by values of **n1** and **n2**

within(v: number, n1: number, n2: number): boolean - returns **true** if the value **v** is within the specified range.

warp(v: number, n1: number, n2: number): number - returns a value warped within the provided range

lerp(start: number, stop: number, val: number, limitRange: boolean): number - returns a value between 0..1 extrapolated to the range between **start**..**stop**

vmap()

len(x: number, y: number): number - returns the length of the vector **[x, y]**

dist(x1: number, y1: number, x2: number, y2: number): number - returns the distance between two points

angleTo(x: number, y: number): number/radians - returns the angle in radians between the vector **[x, y]** and OX

bearing(x1: number, y1: number, x2: number, y2: number) - returns the angle of direction vector from **[x1, y1]** to **[x2, y2]** in relation to OX axis

$$()

kill(node: Node) - kill a node

sfx(sound: AudioClip, volume: number[0..1], panorama: number[-1..1]) - play a sound effect

sleep(t: number): Promise - sleep for **t** seconds

print(line: string) - output a line on the text console

input(message: string): Promise - string - input a value form the text console

ask(question: string): Promise - string - ask a value in the most convenient way for the current mode

say(message: string) - show a message in the most convenient way for the current mode

cls() - clear the text console

rx(n: number): number - relative x coordinate, where **n** sets horizontal screenposition between [0..1]

ry(n: number): number - relative y coordinate, where **n** sets vertical screenposition between [0..1]

save() - save current drawing context state

restore() - restore previously saved drawing context state

scale(horizontal: number, vertical: number) - scale the drawing context by **horizontal** and **vertical** factors

rotate(a: number) - rotate the drawing context on the angle **a** (in radians)

translate(x: number, y: number) - translate drawing context to provided coordinates **x** and **y**

clip(x: number, y: number, w: number, h: number) - clip the drawing context to provided rectangular area

smooth() - draw images with anti-aliasing

blocky() - draw images without anti-aliasing (pixel-art mode!)

alpha(alpha: number) - set global alpha for drawing operations

stroke(v: string/hex | number/integer | number/float , w: number/integer | number/float , u: number/integer | number/float , a: number/integer | number/float) - set stroke mode and asign a line color

lineWidth(width: number) - set line width

fill(fillColor - set fill mode and asign a fill color

background(color - fills background with provided color

line(x1: number, y1: number, y2: number, y2: number) - draw a line between 2 provided points

plot(x: number, y: number) - draw a point with lineWidth as it's canvas size

triangle(x1: number, y1: number, x2: number, y2: number, x3: number, y3: number) - draw a triangle

quad(x1: number, y1: number, x2: number, y2: number, x3: number, y3: number, x4: number, y4: number) - draw a quad

rect(x: number, y: number, w: number, h: number) - draw a rectangle

circle(x: number, y: number, r: number) - draw a cirlce

ellipse(x: number, y: number, verticalRadius: number, horizontalRadius: number, angle: number) - draw an ellipse

arc(x: number, y: number, radius: number, startAngle: number, endAngle: number) - draw an arc centered at **x**/**y** with given **radius** and angles

earc(x: number, y: number, radiusX: number, radiusY: number, rotation: number, startAngle: number, endAngle: number) - draw an elliptical arc centered at **x**/**y**

polygon(x1 - draw a polygon from array of points

moveTo(x: number, y: number) - move to the next point of the shape

lineTo(x: number, y: number) - adds a line to current path

arcTo(x1: number, y1: number, x2: number, y2: number, radius: number) - adds an arc to

the current path using 2 provided control points and radius

quadraticTo(controlPointX: number, controlPointY: number, x: number, y: number) - adds a quadratic Bezier curve to current path

bezierTo(controlPointX1: number, controlPointY1: number, controlPointX2: number, controlPointY2: number, x: number, y: number) - adds a cubic Bezier curve to current path

closePath() - close current path

shape() - stroke or fill current shape according to assigned mode

font(font: string/font | number/fontSize) - set current font

alignLeft() - align text to the left

alignCenter() - align text to the center

alignRight() - align text to the right

baseTop() - base text on top

baseMiddle() - base text in the middle

baseBottom() - base text at the bottom

text(text: string, x: number, y: number) - draw text

textWidth(text: string): number - determine the width of provided text for current font settings

textHeight(): number - determine the height of text for current font settings

image(image: Image, x: number, y: number, w: number, h: number, dx: number, dy: number, dw: number, dh: number) - draw an image

rgb(red: number[0..1], green: number[0..1], blue: number[0..1]) - get the hex string color representation of provided rgb floats

rgba(red: number[0..1], green: number[0..1], blue: number[0..1], alpha: number[0..1]) - get the hex string color representation of provided rgba floats

RGB(red: number[0..255], green: number[0..255], blue: number[0..255]) - get the hex string color representation of provided RGB integers

RGBA(red: number[0..255], green: number[0..255], blue: number[0..255], alpha: number[0..255]) - get the hex string color representation of provided RGBA integers

hsl(hue: number[0..1], saturation: number[0..1], lightness: number[0..1]) - get the hex string color representation of provided hsl floats

hsla(hue: number[0..1], saturation: number[0..1], lightness: number[0..1], alpha: number[0..1]) - get the hex string color representation of provided hsla floats