P5.JS-CHEATSHEET

this is merely a guide for getting started.
for more detailed explainations, visit: www.p5js.org/reference
v03 – based on p5.js v0.8.0 // cc teddavis.org 2019 – fhnw hgk ivk

BASIC STRUCTURE *// very least we need to write to have a program!*

```
function setup(){
        // this code runs once
}

function draw(){
        // this code is looped
}
```

COMMENTS + DEBUG *// annotate and toggle code to find bugs*

```
// this is a single line comment
/*
   this is a multiline comment.
   nothing between here will be run or executed
*/
print(variableName);  // outputs the value to the console, used to learn value of variable
```

META *// attributes for sketch*

```
createCanvas(800, 500)  // sets canvas size to 800px * 500px
createCanvas(800, 500, WEBGL)  // uses 3D renderer, for 3D transofmations
createCanvas(windowWidth, windowHeight)  // full screen canvas
noLoop()  // stops draw() function from looping, use loop() to continue
```

PRIMITIVE SHAPES *// the core things we use to draw with*

```
point(x, y)  // draw single point at x and y on canvas
line(x1, y1, x2, y2)  // draws line from x1, y1 to x2, y2
rect(x, y, w, h)  // draws rectangle at given postition and size (width, height)
ellipse(x, y, w, h)  // draws ellipse at given postition and size
beginShape()  // starts complex form
    vertex(x, y)  // use as many vertex()'s as needed to draw shape
endShape()  // ends complex form. use endShape(CLOSE); to automatically close shape
```

STYLE ATTRIBUTES *// processed top to bottom = declare before drawing shape*

```
background(0)  // sets background to black (test having and not having in draw function )
noFill()  // turns off the fill of any object following this code
fill(255)  // turns fill on and sets color to white, short for fill(255, 255, 255);
fill(255, 145, 90, 150)  // same but with color (r, g, b) + optional alpha
color(255, 0, 0)  // create a color to use a variable and plug into stroke/fill
noStroke()  // turns off stroke
stroke(0)  // turns stroke back on and is black (use color as listed above)
strokeWeight(5)  // sets thickness of stroke (any value goes here)
noSmooth()  // turns off anti-aliasing for hard edge vectors, smooth() by default
rectMode(CENTER)  // sets x and y origin to CENTER or CORNER (see: ellipseMode, imageMode)
```

TYPOGRAPHY *// play with type + code*

```
textSize(12)  // set the fontsize in pts
textFont('Monaco')  // set the font based on built-in system fonts
textAlign(CENTER, CENTER)  // align type (HORIZONTALLY, VERTICALLY)
text('blah', x, y)  // draw type without bounding box
text('blah', x, y, w, h)  // draw type in bounding box
```

TRANSFORMATIONS *// manipulate shapes*

```
push()/pop()  // transform objects without effecting others, push() ... your code ... pop()
translate(x, y)  // move the 0, 0 coordinates, useful for rotating objects around their own center
rotate(radians(45))  // to rotate things using degrees, in 3D there's rotateY, rotateX, rotateZ
orbitControl(5)  // rotate a 3D environment with the mouse
```

RANDOM *// let the computer design for -er surprise you!*

```
random(100)  // generates a random float number from 0 » 100
random(75, 100)  // generates a random float number from 75 » 100
randomSeed(5)  // locks each request of random to that values 'gear' = consistant random
noise(foo)  // foo variable needs to grow with time, produces more organic walking range between 0 - 1
```

VARIABLE TYPES *// store a values to reference throughout code, useCamelCase*

```
let myNumber = 5  // integer or whole number
let myNumber = 3.14  // floating-point decimal number
let myText = "Hello World"  // string of text
let myChar = 'a'  // single character
let mySwitch = true  // boolean (true or false), used for if statements
let myArray = [12, 53, 23,...]  // array, store any variables as collection
```

BUILT-IN VARIABLES *// memorize these!*

```
width / height  // total canvas width / height, use 'width/2', 'height/2' for center!
mouseX / mouseY  // current X/Y mouse coordinate = goldmine for interaction!
pmouseX / pmouseY  // previous X / Y mouse coordinate, useful to know if mouse moved
frameCount  // starts at 0 and counts everytime the draw code loops, great for animation
```

MATH *// our favorite subject right?*

```
+ - * /  // add, subtract, multiply, divide = basic math operations
foo = foo + 5  // value = self + 5
foo += 5  // same as above, but less code!
foo++  // similar to above, however only adds 1 each time (also works with --)
round(foo)  // convert a float into an int, normal rounding rules apply
floor(foo)  // convert a float into an int, force rounding down
ceil(foo)  // convert a float into an int, force rounding up
map(inVal, inMin, inMax, outMin, outMax)  // scale values
sin(foo)  // produces value between -1 to 1, smaller changes = smaller periods between wave
abs(foo)  // absolute value, useful when comparing two numbers with subtraction
```

CONDITIONAL STATEMENT *// the world is composed of them! if this, do that!*

```
if(a == b){
        // if 'a' IS EQUAL to 'b' all code in between these { } will be executed
}else{
        // optional else, this code will run
}
```

RELATIONAL OPERATORS *// the foundation of if statements, used to filter instructions*

```
a == b  // a is EQUAL to b (note the use of two == signs)
a != b  // a is NOT EQUAL to b
a >  b  // a is GREATER than b
a <  b  // a is LESS than b
a >= b  // a is GREATER or EQUAL to b
a <= b  // a is LESS or EQUAL to b
```

LOGICAL OPERATORS *// you can require two or more things to be true*

```
if(a > 10 && a < 100){ }  // AND = both statements must be true
if(a < 10 || a > 100){ }  // OR = either statement must be true
```

MOUSE *// react to the mouse!*

```
if(mouseIsPressed){ }  // used in the draw{ } to know if mouse is constantly pressed
function mousePressed(){ }  // will only trigger once when mouse is pressed
function mouseReleased(){ }  // will only trigger once when mouse is released
```

KEYBOARD *// react to the keyboard keyboard!*

```
if(keyIsPressed){ }  // used in the draw{ } to know if any key constantly pressed
function keyPressed(){ }  // will only trigger once when key is pressed
function keyReleased(){ }  // will only trigger once when key is released
print(keyCode)  // use this to learn the keyCode for any special key on the keyboard
if(key == 'a'){ }  // is true if the letter a is pressed
if(keyCode == 32){ }  // alternative for key, useful for meta-keys
```

LOOPS *// let the computer do repetitive tasks! let the computer do repetitive tasks!*

```
// abstract: for(start; stop; count){... looped code ...}

// human: (variable i starts at 0; as long as i is less than 50, loop; add 1 to i on each loop){... loop this code ...}
for (let i = 0; i < 50; i++){  // 'i' is a unique on every loop, 1..2..3..4... use it!
        line(i*10, 0, width/2, height/2);
}
```