Cheatsheets / **Learn p5.js**

codecademy

Animation

The `draw()` Function

The `draw()` function is automatically called after the `setup()` function, which runs once at the program's start. The `draw()` loop infinitely runs the code block inside the function from top to bottom.

```
function setup(){
  // Runs once at the start of the program
}
function draw(){
  // Loops infinitely after setup() is run
}
```

FPS

Frames Per Second (FPS) specifies the number of frames displayed every second. p5.js automatically runs the program at 60 frames per second. This means that the `draw()` function runs repeatedly 60 times per second.

The `frameCount` Variable

To keep track of the number of frames rendered, p5.js provides a built-in variable called `frameCount`. This variable stores the number of frames that have been displayed since the program started.
The value of the `frameCount` variable updates with every frame. The first time the `draw()` loop runs, the `frameCount` variable's value is one; the second time, the `frameCount` is two; and so forth.

```
function draw() {
    // Animate ellipse's x position to move
from left to right as the frameCount variable
increases
  ellipse(frameCount, height / 2, 100, 100);
}
```

The `frameRate()` function

p5.js will automatically run your code at 60 frames per second. However, you can manipulate the FPS value by using the `frameRate()` function, which will change the number of frames shown per second. The maximum number of frames that can be drawn is 60 FPS.

```
function setup() {
  createCanvas(400, 400);
  // Set frame rate to 24 FPS, which will
make the sketch run at a slower rate than the
default 60 FPS
  frameRate(24);
}
```

Incrementing Values

A value is incremented by writing the following expression:

```
x = x + 1
```

Above can be rewritten as:

```
x++;
```

or

```
x += 1;
```

In all three expressions, the `x` variable is taking its own value and adding 1 to it. The value 1 can be changed to any other number.

```
function draw(){
  ellipse(xPos, 100, 100, 100);
  // Increment x position by 5 every draw
loop
  xPos += 5;
}
```

Decrementing Values

A value is decremented by writing the following expression:

```
x = x - 1
```

Above can be rewritten as:

```
x--;
```

or

```
x -= 1;
```

In all three expressions, the `x` variable is taking its own value and subtracting 1 from it. The value 1 can be changed to any other number.

```
function draw(){
  ellipse(100, yPos, 100, 100);
  // Decrement y position by 5 every draw
loop
  yPos -= 5;
}
```

The `random()` Function

The `random()` function returns a random decimal value between 0 and 1.

When one numeric argument is given to the `random()` function, it returns a random decimal value between 0 and the value of the given argument.

When two numeric values are given as arguments, the function returns a random decimal value between the first argument and the second argument.

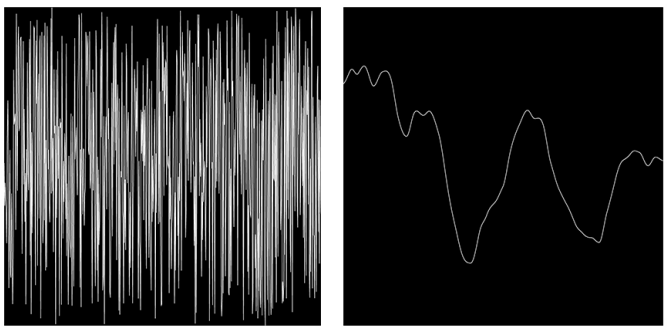
```
function draw(){
  // Generate a random value between 0 and
width
  let randomX = random(width);
  // Generate a random value between 15 and
100
  let randomSize = random(15, 100);

  circle(randomX, height / 2, randomSize);
}
```

The `noise()` Function

The `noise()` function returns a random decimal value between 0 and 1, based on Perlin noise.

Since the `noise()` function generates a naturally ordered sequence of random numbers, this function is useful for creating more natural random movements.



Random Noise

User-defined Functions

User-defined functions can be called inside the `draw()` and `setup()` function. If a user-defined function containing p5.js functions is called outside of the `draw()` or `setup()` function, they may not run as intended.

```
let xpos, ypos;

function setup(){
  // Call init() custom function inside
setup()
  init();
}

function draw() {
  // Call display() custom function inside
draw()
  display();
}

// Define custom functions outside of the
setup() and draw() functions
function init(){
  xpos = width / 2;
  ypos = height / 2;
}

function display() {
  fill(255);
  ellipse(x, y, 50, 50);
}
```

Function Parameters

Parameters are a way to introduce variations into your functions. Variables in the `draw()` function can be passed into the user-defined functions as arguments.

```
function draw() {
  // 0 and 5 are passed in as arguments
  // Circle is drawn at (0, 5) coordinate
  makeCircle(0, 5);

  // 10 and 15 are passed in as arguments
  // Circle is drawn at (10, 5) coordinate
  makeCircle(10, 15);
}

// Custom function with parameters for x and
y positions
function makeCircke(x, y) {
  circle(x, y, 20);
}
```