

> sketch.js

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   background(255, 0, 0);  
7 }
```

宽 高  
决定颜色  
红 绿 蓝

①function 函数: blocks of code 代码块  
②setup 设置: a series of things that we want to do when we first load the code. 第一次加载代码时想做的一系列事情。(只想做一遍)

③draw 画: 相当于一个循环动作, 可以接下来继续画. pool 循环  
④RGB 颜色: 红. 绿. 蓝组成颜色. 在RGB颜色中, 通常有3个值, 它们代表红. 绿. 蓝, 从0-255 (总共有256个值, 因为是从0开始的, 所以0也包含在内). 可以添加第四个值来代表透明度(同样是0-225).  
也可以写单数字, 单数字相当于从黑(0)到纯白(255).

```
4  
5 function draw() {  
6   background(220);  
7 }  
8 rect(50, 100, 40, 60);  
9 }
```

⑤轴 轴 宽 高

⑥形状: rectangle 长方形的缩写.  
⑦从左上角开始的, 首先是形状在画布上的位置(xy轴), 其次是形状的宽和高.

⑦完成一项要加分号.

Preview



得到一个描黑边的白色长方形

Play Sketch

sketch.js

Preview

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   background(220);  
7 }  
8 stroke(255, 0, 0);  
9 fill(255, 255, 0, 255);  
10 rect(50, 100, 40, 60);  
11  
12 }
```

描边(RGB颜色)  
填充(RGB颜色)

1 function setup() {  
2 createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6 background(220);  
7 }  
8 stroke(255, 0, 0);  
9 fill(255, 255, 0);  
10 rect(50, 100, 40, 60);  
11  
12 fill(0, 0, 255);  
13 rect(150, 100, 40, 60);  
14 }



另外一个形状就另起一行重复以上操作.

# key coding: variable 变量

```
> sketch.js•  
1 let x = 0; ①  
2  
3 function setup() {  
4   createCanvas(400, 400);  
5 }  
6  
7 function draw() {  
8   background(220);  
9  
10  fill(255, 255, 0);  
11  rect(x, 100, 40, 60);  
12  
13  x = x + 3; ②  
14  
15  console.log(x);  
16 } 按控制台日志  
③ Console 519  
522 ↓ 出现矩形的移动  
525 日志  
528
```

① 让  $x=0$ , 其中,  $x$  就是 variable 的名字. 当我们将其其中一个值用  $x$  代表, 那么, 那个值就是 0.  
(这里呈现的图像是长方形在画布的最左边, 因为  $x$  轴的值被 variable 替换了).

②  $x=x+3$  是下一个循环, 再下一个循环中,  $x$  将变成 3. 然后又变成  $x=x+3$  ( $x=3+3, x=6+3, x=9+3, x=12+3 \dots \dots$ ) 这里呈现的图像是矩形一直向右移动, 并再也不回来.

③ 得到每次经过时  $x$  的值 (默认帧率是 30 or 60).

## system variable (不需要额外顶格写)

```
7 function draw() {  
8   background(220);  
9  
10  fill(255, 255, 0);  
11  rect(mouseX, mouseY, 40, 60);  
12  ④ ————— T
```

④ 这就是 system variable 系统变量.  
 $mouseX$  是图形跟着鼠标横向移动.  
 $mouseY$  就是纵向移动.

## key coding 2: conditionals (检查某事是否正确, 然后根据答案做些事情)

```
13  x = x + 3;  
14  
15  ⑤ if(x > width) {  
16    x = 0;  
17  } 这里不是 mouseX  
18  
// x = x + 3;  
// if(x > width) {  
//   x = 0;  
// }
```

⑤ 如果 ( $x >$  宽/数字) { 大括号并另起一行  
就让  $x=0$ ;  
} 这段代码可以让图形重新回起点  
选中这段代码然后按 (ctrl) 可以让它  
们暂时不运行.

```

1 let x = 0;
2
3 function setup() {
4   createCanvas(400, 400);
5 }
6
7 function draw() {
8   background(220);
9
10  if (mouseX > width / 2) { ①
11    fill(255, 0, 0);
12  } else { ②
13    fill(255, 255, 0);
14  }
15
16  rect(50, 100, 40, 60);
17

```

图像呈现的是当鼠标挪过宽的一半，就……  
长方形变成黄色。

①如果鼠标在 x 由 > 200  
写作宽 / 2 (width / 2) 也可以。  
就……  
else 如果不。

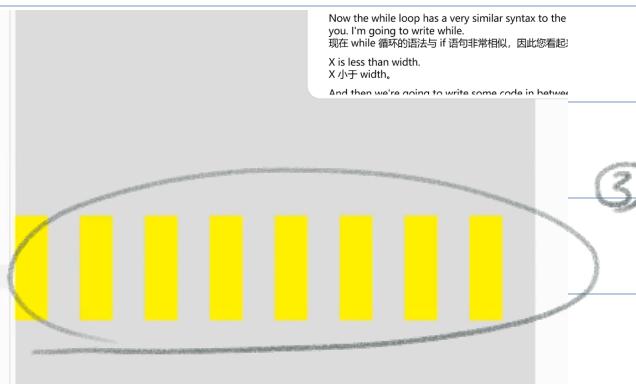
## workshop 1 :

```

7
8 function draw() {
9
10  fill(255, 255, 0);
11  noStroke();
12
13  while (x < width){ ①
14    rect(x, 200, 25, 80); ②
15    x += 50; ③
16
17  // x = x + 50;
18  // rect (x, 200, 25, 80); ④
19  // x = x + 50;
20  // rect (x, 200, 25, 80); ⑤
21

```

复制大小一致的图形可以使用 while 循环



①while语法和if语法极其相似，  
以图片为例，while和if的区别在于：while语法会在某个条件成立的情况下继续这么做（只要x留有宽度，就会发生一些事情，这里要发生的事情是我们将绘制一个矩形）。

②让函数执行  $x = x + 50$ ，等同于每隔 50 像素生成一个一模一样的矩形。 $x += 50$  是  $x = x + 50$  的速记方法。

③在函数生成后，在画布上每隔 50 像素生成一个矩形，直到 x 到达画布的边界。

④在大括号内缩进的文本，就是对 x variable 的操作。

# 如何制作你上桌便自动画布边框就自行返回

```
let x = 0;  
let move = 3;①  
6 }  
7  
8 function draw() {  
9   background(220);  
10  
11  if (x > width){  
12    move = -3; ②  
13  }  
14  
15  if (x < 0){  
16    move = 3; ③  
17  }  
18  
19  rect(x, 200, 100, 50);  
20  
21  x = x + move; ④  
22 }
```

①首先需要添加另一个变量(变量名称自拟)

变量名称能够让你知道该变量的用途，因此最好赋予有意义的名字。  
②第一个条件代表了x大于画布宽的情况，我们想让它返回，因此让move=-3。  
③第二个条件中的x小于0的话，代表图像会离开左侧画布，因此让move=3。

④这一段函数便是让图像动起来的conditional.

```
for (x = 0; x < width; x += 50){  
  rect(x, 200, 25, 80); ⑤  
}
```

for loop和while loop的用法相

似，但是for loop可以拥有更简洁的代码。

⑤首先x=0(既画布最左边)，退出条件是x小于宽度，然后就是让x=x+50(既每隔50像素生成一个图形)

```
1 function setup() {  
2   createCanvas(550, 450);  
3   background(220);  
4   for (x = 0; x < 30; x++) {①  
5     lollipop(random(0, width), random(0, height), random(0, 200),  
6     color(255, 0, 0));  X I Y ③  
7   } 颜色
```

①for loop的用法，其中x++的意思是x=x+1。

②lollipop是自创function，其中括号中的random是可以随机生成的数字。random右边的括号中分别

对应的还是最小值和最大值，因此random的数字不会超过这个范围。

③这个参数是介于0-200的随机数(向函数中的size)。

```
0 function draw() {  
1 }  
2  
3 function lollipop(x, y, size, flavour) {  
4     fill(255);  
5     rect(x, y - size, 10, size);  
6     fill(flavour);  
7     ellipseMode(CORNER);  
8     circle(x - 10, y - size - 30, 30);  
9 }  
0
```

flavour为其着色

① 归到函数本身。其中x, y, size, flavour都是自己的名字。我们用x, y, size画一个矩形，用flavour为其着色

## workshop 2.

```
JS sketch.js > makeHouse  
1 function setup() {  
2     createCanvas(400, 400);  
3     background(220);  
4  
5     setTimeout(makeHouse, 2000, 50, 100, 50, color(255, 0, 0));  
6 }  
7  
8 function makeHouse(xPos, yPos, stories, colour) {  
9     fill(colour);  
10    noStroke();  
11    rect(xPos, yPos, 50, stories);  
12    triangle(xPos, yPos, xPos + 25, yPos - 20, xPos + 50, yPos);  
13 }  
14
```

① setTimeout: 是JavaScript原生函数，将makeHouse子程序启动的②毫秒后执行。(这里的图像是2秒后出现)

② 2000在此是时间，单位是毫秒。2000毫秒就是2秒。

```
6 function mousePressed(){ ③  
7     setTimeout(makeHouse, 2000, random(0, width), random(0, height))  
8 }  
9
```

③ mousePressed是P5JS的内置函数，作用是当鼠标按下时，会执行后续代码。

```
function makeHouse(xPos, yPos, stories, colour) {  
    fill(colour);  
    noStroke();  
    rect(xPos, yPos, 50, stories);  
    triangle(xPos, yPos, xPos + 25, yPos - 20, xPos + 50, yPos);  
    setTimeout(makeHouse, 2000, random(0, width), random(0, height))  
}  
④
```

④ 也可以将setTimeout放进自拟函数中，这样每当我们使用这个自拟函数时都将触发setTimeout。

```
1 function setup() {  
2     createCanvas(400, 400);  
3     background(220);  
4     setInterval(makeRedSquare, 1000); ⑤  
5 }
```

```
js sketch.js > ⌂ mousePressed  
1 let counter = 0;  
2  
3 let countInterval = setInterval(makeRedSquare, 1000);  
4  
5 function setup() {  
6     createCanvas(400, 400);  
7     background(220);  
8 }  
9  
10 function makeRedSquare(){  
11     noStroke();  
12     fill(255, 0, 0);  
13     rect(random(0,300), random(0,300), 60, 60);  
14     counter++;  
15     if (counter > 5){  
16         clearInterval(countInterval); ⑥  
17     }  
18 }  
19  
20 function mousePressed() {  
21     setInterval(makeRedSquare, 1000); ⑦  
22 }
```

⑤ setInterval和setTimeout用法相同。

⑥ clearInterval：停止这个区间的操作（在此是生成五个图形后停止生成）

⑦ 在停止生成图形后再次点击鼠标便会重复上述步骤（既生成五个图形后停止）。