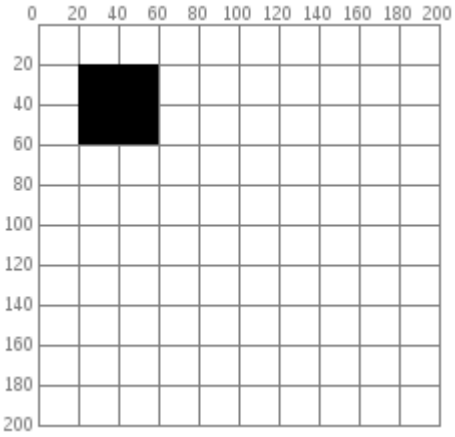
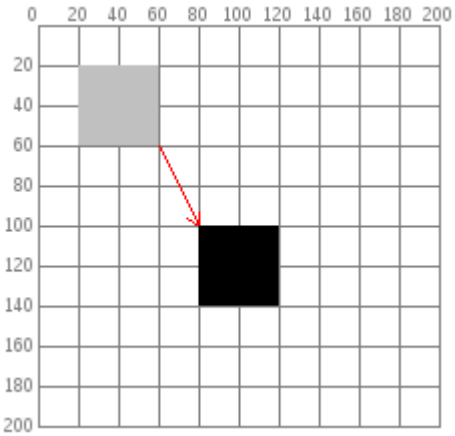
Translation

When you create a program here using ProcessingJS, the output is drawn to a canvas that acts like a piece of graph paper. To draw a shape, you specify its coordinates onto that graph.

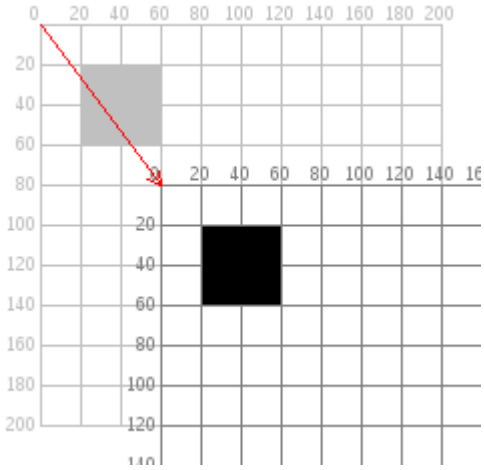
As an example, here is a simple rectangle drawn with the code `rect(20, 20, 40, 40)`. The coordinate system (a fancier word to describe that "graph paper") is shown in gray, and to keep our example images smaller, the coordinate system shown is 200 pixels by 200 pixels (instead of the default 400x400 size).



If you want to move the rectangle 60 units right and 80 units down, you can just change the coordinates by adding to the `x` and `y` starting point: `rect(20 + 60, 20 + 80, 40, 40)` and the rectangle will appear in a different place. (We put the arrow in there for dramatic effect.)



But there is a more interesting way to do it: **move the graph paper instead**. If you move the graph paper 60 units right and 80 units down, you will get exactly the same visual result. Moving the coordinate system is called ***translation***.



The important thing to notice in the preceding diagram is that, as far as the rectangle is concerned, it hasn't moved at all. Its upper left corner is still at (20,20). When you use transformations, the things you draw never change position; the coordinate system does.

Here is a program that draws the original rectangle, then draws it in red at the new location by changing its coordinates, then draws it in blue at the new location by moving the grid (`translate()`ing). The fill colors are translucent, so that you can see that the red and blue overlap to form a purple square that is at virtually the same place. Only the method used to move them has changed. Fiddle with the numbers below to see for yourself:

```
background(255, 255, 255);

noStroke();


// draw the original position in gray

fill(190, 190, 190);

rect(20, 20, 40, 40);


// draw a translucent red rectangle by changing the coordinates

fill(255, 0, 0, 128);

rect(20 + 60, 20 + 80, 40, 40);


// draw a translucent blue rectangle by translating the grid

fill(0, 0, 255, 128);

pushMatrix();

translate(60, 80);

rect(20, 20, 40, 40);

popMatrix();
```

Let's look at the translation code in more detail. `pushMatrix()` is a built-in function that saves the current position of the coordinate system. The `translate(60, 80)` moves the coordinate system 60 units right and 80 units down. The `rect(20, 20, 40, 40)` draws the rectangle at the same place it was originally. Remember, the things you draw don't move—the grid moves instead. Finally, `popMatrix()` restores the coordinate system to the way it was before you did the translate.

Why use `pushMatrix()` and `popMatrix()`? You could have done a `translate(-60, -80)` to move the grid back to its original position. However, when you start doing more sophisticated operations with the coordinate system, it's easier to use `pushMatrix()` and `popMatrix()` to save and restore the status rather than having to undo all your operations. Later on in this section, you will find out why those functions seem to have such strange names.

What's the advantage?

You may be thinking that picking up the coordinate system and moving it is a lot more trouble than just adding to coordinates. For a simple example like the rectangle, you are correct. But let's take an example of where `translate()` can make life easier.

Here is a program that draws a row of houses. It uses a loop that calls function named `drawHouse()`, which takes the `x` and `y` location of the house's upper-left corner as its parameters. Notice that the `drawHouse` function has to do a lot of parameter manipulation to get the house drawn at the given coordinates:

```
var drawHouse = function(x, y) {
  triangle(x + 15, y, x, y + 15, x + 30, y + 15);
  rect(x, y + 15, 30, 30);
  rect(x + 12, y + 30, 10, 15);
};



background(255, 255, 255);
for (var i = 10; i < 350; i = i + 50) {
   drawHouse(i, 20);
}
```

What if we used the `translate()` function instead of calculating new coordinates? In this case, the code draws the house in the same place every time, with its upper left corner at (0, 0), and lets translation do all the work instead.

```
var drawHouse = function(x, y) {
   pushMatrix();
   translate(x, y);
   triangle(15, 0, 0, 15, 30, 15);
```

```
    rect(0, 15, 30, 30);
    rect(12, 30, 10, 15);
    popMatrix();
};


background(255, 255, 255);
for (var i = 10; i < 350; i = i + 50) {
    drawHouse(i, 20);
}
```

It doesn't mean that you should always favor `translate()` instead of calculating new coordinates. Like much of what we teach, it is another tool in your toolbox, and it will be up to you to figure out when it makes sense to use this new `translate()` tool.