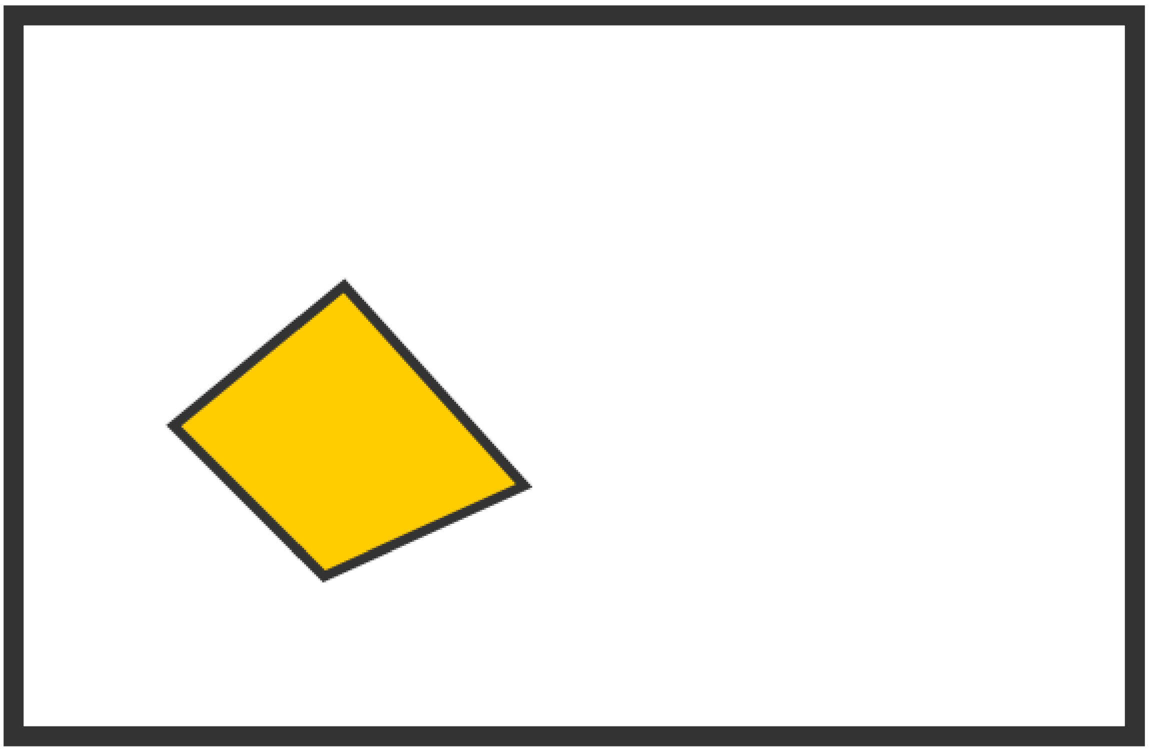
This Seems Simple, No?

Earlier, we looked at an example where we drew a simple four-sided shape. The `canvas` code for that looked as follows:

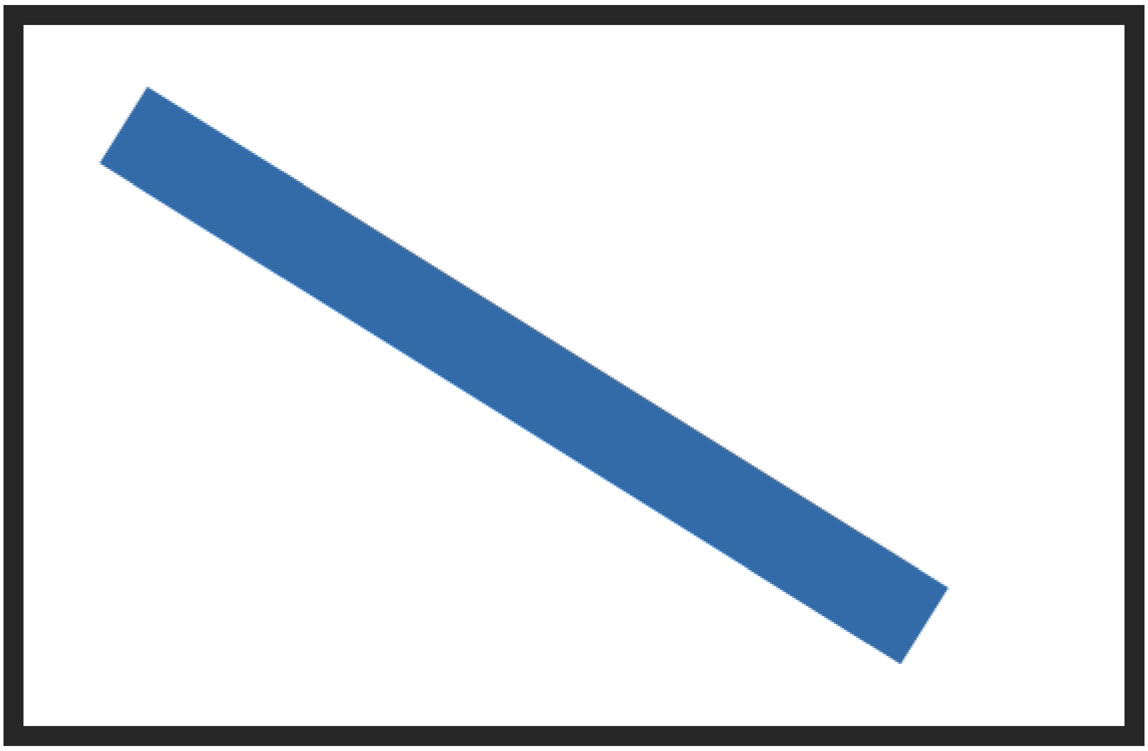
HTML JavaScript

```javascript
var canvas = document.querySelector("#myCanvas");
var context = canvas.getContext("2d");

context.moveTo(160, 130);
context.lineTo(75, 200);
context.lineTo(150, 275);
context.lineTo(250, 230);
context.closePath();

context.lineWidth = 5;
context.strokeStyle = "#333333";
context.fillStyle = "#FFCC00";

context.fill();
context.stroke();
```

output



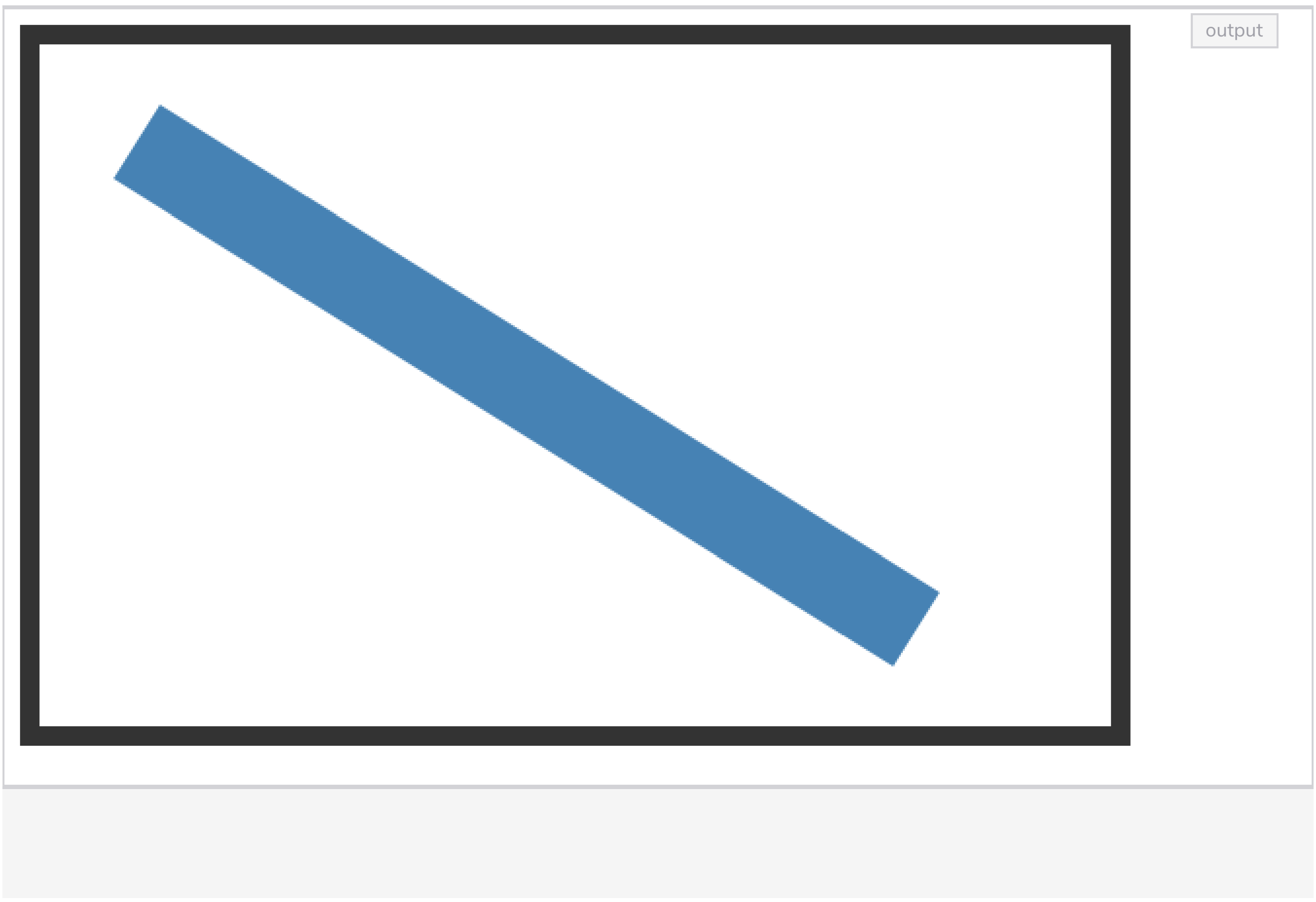
Now, let’s say that we want to add another shape to what we have here. The shape we want looks something like this:



In code, it is represented as follows:

HTML JavaScript

```javascript
var canvas = document.querySelector("#myCanvas");
var context = canvas.getContext("2d");

context.moveTo(50, 50);
context.lineTo(450, 300);
context.closePath();

context.lineWidth = 45;
context.strokeStyle = "steelblue";

context.fill();
context.stroke();

```



What do you think we should do to combine these shapes? One thing we learned from earlier is that the `stroke()` and `fill()` methods act as the big red button you push to get things displayed on our screen. One reasonable attempt might be to keep the common `stroke()` and `fill()` methods from earlier and merge in the lines of code that represent our diagonal line.

On the surface, this seems like a reasonable thing to do. The first part of the code deals with the four-sided shape and what it looks like. The second (and highlighted) part deals with our diagonal line and what it looks like. In my mind, this seems like a solid solution!

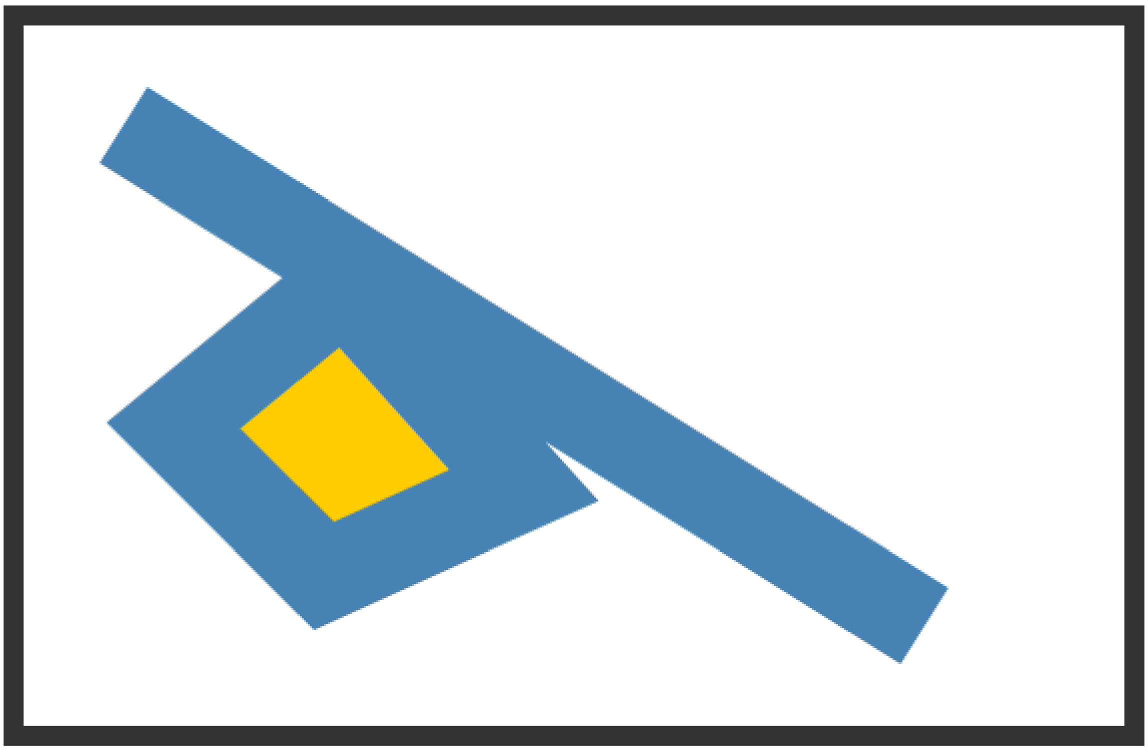
When we preview this in our browser, this is what you will see:

HTML JavaScript

javascript

```javascript
var canvas = document.querySelector("#myCanvas");
var context = canvas.getContext("2d");

context.moveTo(160, 130);
context.lineTo(75, 200);
context.lineTo(150, 275);
context.lineTo(250, 230);
context.closePath();

context.lineWidth = 5;
context.strokeStyle = "#333";
context.fillStyle = "#FFCC00";

context.moveTo(50, 50);
context.lineTo(450, 300);
context.closePath();

context.lineWidth = 45;
context.strokeStyle = "steelblue";

context.fill();
context.stroke();

```

output



Not quite what we had in mind, right? Ok, so it turns out merging the relevant lines of code under one single `stroke` and `fill` call didn't work out properly. What if we decide to keep the code for these shapes separate and have duplicate stroke/fill calls for each shape?

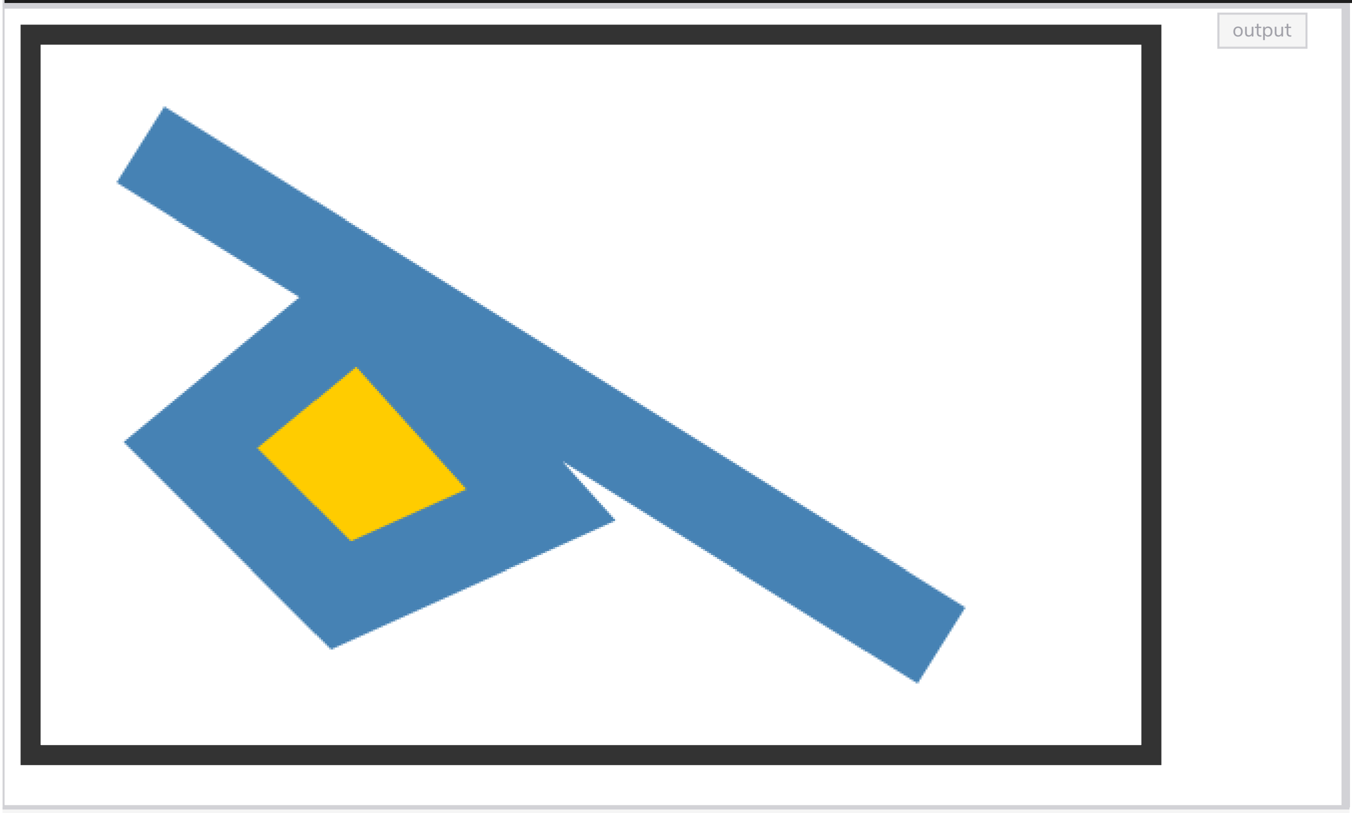
The code for that would look like this:

HTML JavaScript

```javascript
var canvas = document.querySelector("#myCanvas");
```

javascript

```
var context = canvas.getContext("2d");

// first shape
context.moveTo(160, 130);
context.lineTo(75, 200);
context.lineTo(150, 275);
context.lineTo(250, 230);
context.closePath();

context.lineWidth = 5;
context.strokeStyle = "#333";
context.fillStyle = "#FFCC00";

context.fill();
context.stroke();

// second shape
context.moveTo(50, 50);
context.lineTo(450, 300);
context.closePath();

context.lineWidth = 45;
context.strokeStyle = "steelblue";

context.stroke();

```



If you try this arrangement and preview in your browser, what do you see? **It's exactly the same thing as the weird jumble of shapes you saw earlier**. What do you think is going on? Let's figure this out in the next section.

What do you think is going on? Let's figure this out in the next section