HTML5 Canvas and JavaScript

Lesson 3: Screen Coordinates and Colors

This week we are going to take a closer look at the way we specify locations and colors on the <**canvas**> drawing surface.

FillStyle Colors

Remember last week that we selected a color for our filled-in shapes using the **fillStyle** property:

```
        ctx.fillStyle = "blue";
```

The **fillStyle** property is very flexible, and can be used to choose a solid color or even a gradient or pattern! You can use several different formats to choose a color. Here are a few of the color options:

Color Format	Exmple	Description
rgb(RRR, GGG, BBB)	"rgb(0,255,0)"	This pattern can be used to set the red, green, and blue components of the color using regular decimal numbers between 0 and 255.
#RRGGBB	"#00FF00"	This pattern can be used to set the red, green, and blue components of the color using hexadecimal values between 00 and FF.
<color name>	"green"	You can also use one of over 140 pre-defined simple English names like “blue” or “aqua.”

➔ **Open your existing “index.html”. Try setting the fillStyle using the rgb() pattern with red, green, and blue components. What interesting colors can you make that don’t match any pre-defined name?**

If you want to experiment with other **fillStyle** attributes, take a look at the reference link below.

http://www.w3schools.com/tags/canvas_fillstyle.asp

Computer Screen Coordinates

Next, we used a **fillRect**() function to draw a rectangle on the screen. The function takes 4 number parameters, separated by commas. Here is an example:

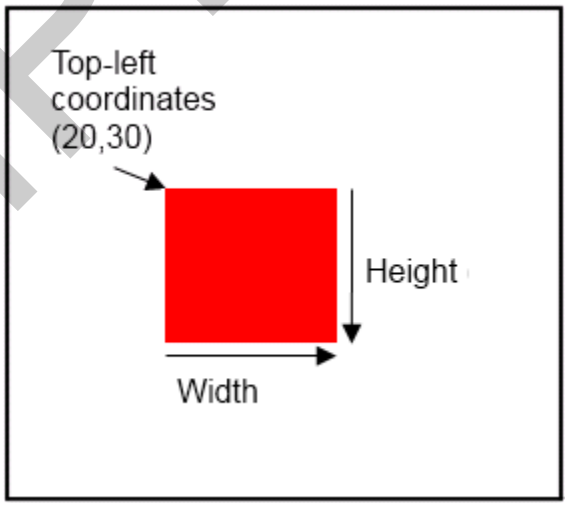
```
ctx.fillRect(20,30,50,50);
```

The first number is the **upper-left "x"** coordinate where the rectangle will start. The second number is the **upper-left "y"** coordinate. Then the third and fourth numbers set the rectangle **width** and **height**.

All this talk of "coordinates" may be a bit mysterious, so let's take a brief detour and explain how coordinates work on a computer screen.

Things like images, shapes, or even text are located in certain spots on a computer screen or within a window like our <**canvas**> element. The location is done by counting pixels (the little dots on your computer screen). You can set a location by using an "x" and a "y" coordinate.

The **"x" coordinate** counts a number of pixels **over from the left-hand edge** of your screen or window. The **"y" coordinate** counts a number of pixels **down from the top edge** of your screen or window.



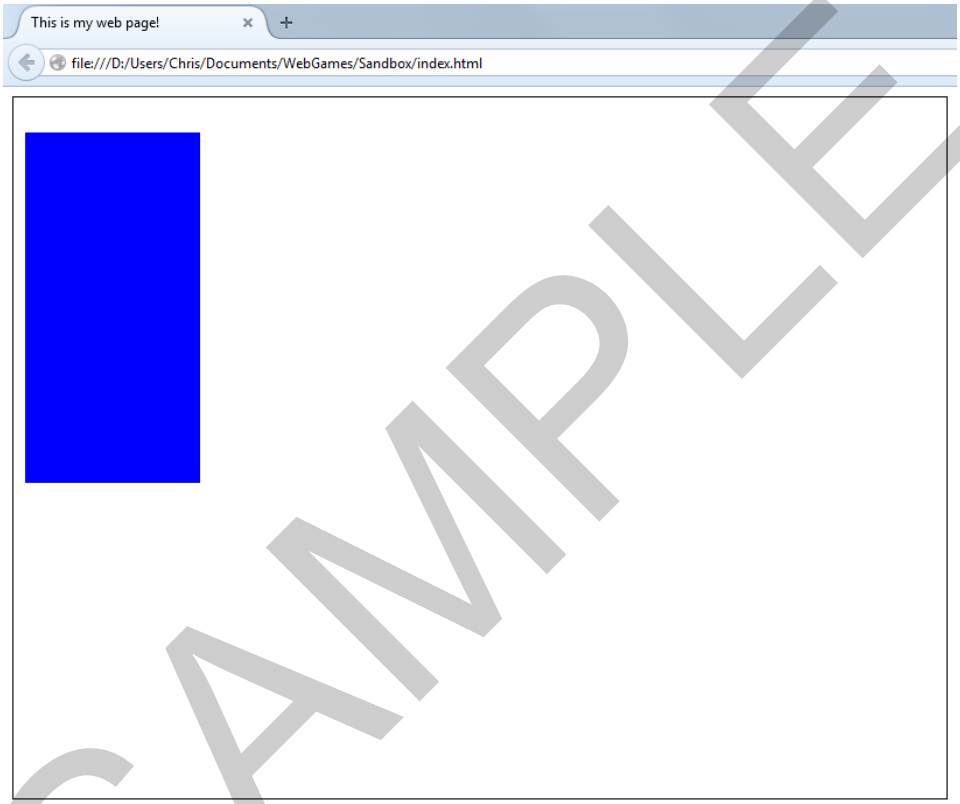
So a coordinate of (0, 0) represents the upper left-hand corner of the screen. A coordinate of (20, 30) means to start 20 pixels over and 30 pixels down from the top-left corner.

We have set the overall size of our <**canvas**> element to 800 pixels wide and 600 pixels high. This means that valid coordinates will range from (0, 0) in the top-left corner to (799, 599) in the bottom-right corner. Obviously, you can choose different sizes for your <**canvas**> element, and your valid pixel coordinates will need to stay within your chosen dimensions.

Now let's examine another **fillRect**() example.

```
        ctx.fillRect(10,30,150,300);
```

This example says to draw a filled-in rectangle starting at the coordinate (10, 30) and make the rectangle 150 x 300 pixels in size.



➔ **Go ahead and change your existing "index.html" page and replace your existing fillRect() line with the ctx.fillRect() example shown above. Do you get the same results? You might be using a different color based on your current fillStyle settings!**

Lesson 3 Activity: Block Diagrams

In this activity you are going to practice drawing rectangles of different sizes and colors!

Creating Multiple Blocks

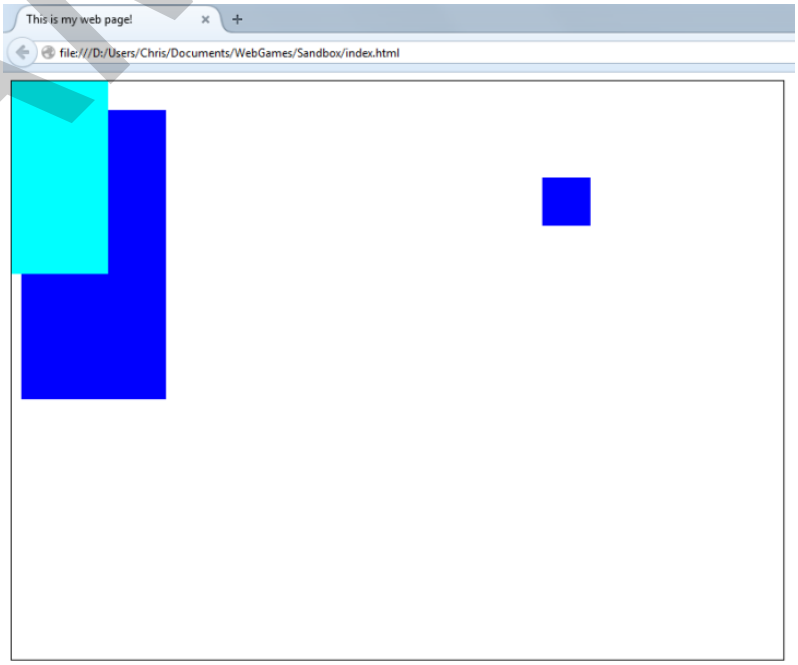
To create more than one rectangle, just add multiple JavaScript lines that use the **fillRect**() function call. For example:

```
ctx.fillStyle = "rgb(0, 0, 255)";
ctx.fillRect(10,30,150,300);
ctx.fillRect(550,100,50,50);
```

If you want to change colors, set the **ctx.fillStyle** property on a new line in your script as well.

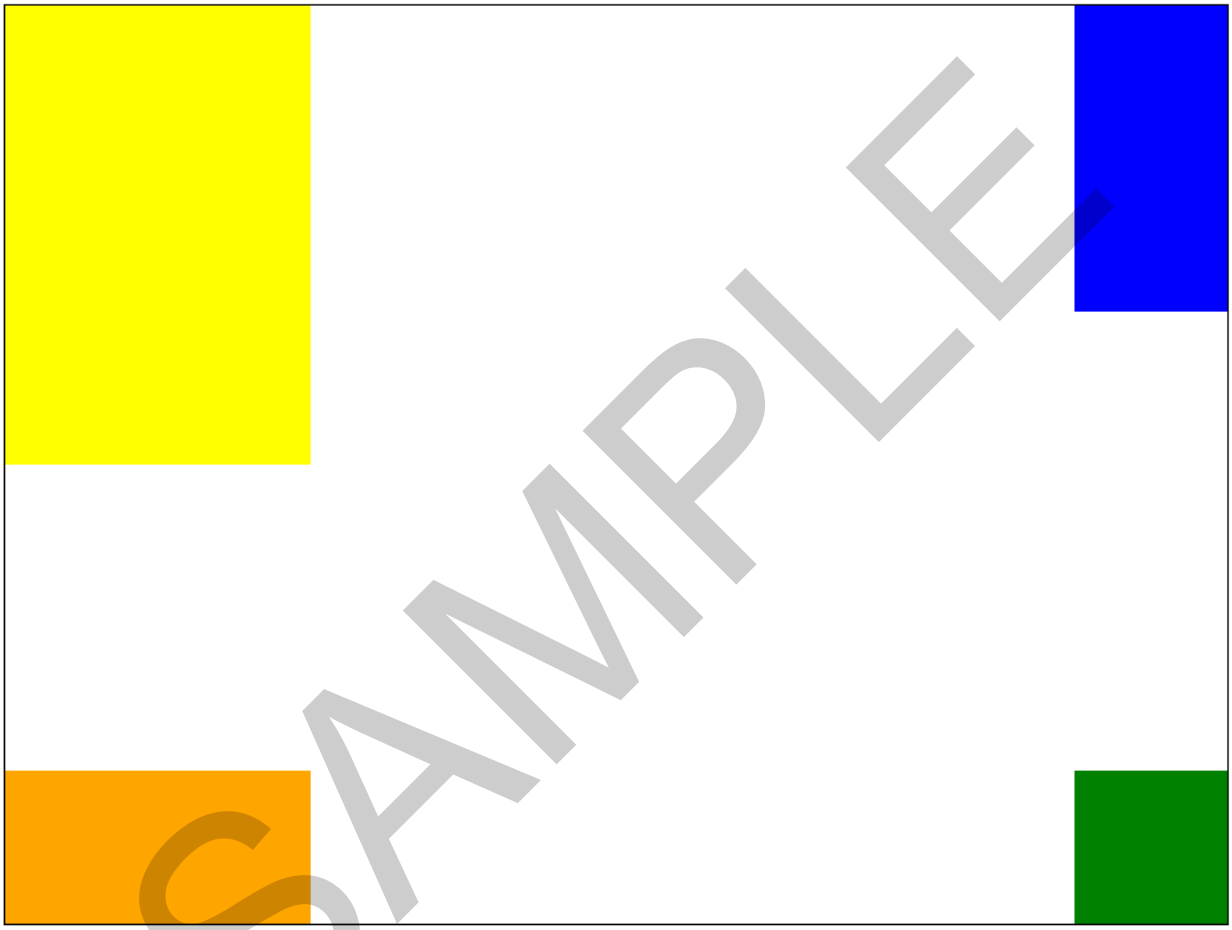
```
ctx.fillStyle = "rgb(0, 0, 255)";
ctx.fillRect(10,30,150,300);
ctx.fillRect(550,100,50,50);

ctx.fillStyle = "aqua";
ctx.fillRect(0,0,100,200);
```

Here is the result from these 5 lines of JavaScript code. Notice that the JavaScript lines run from top to bottom, and anything you draw on the surface later on will cover up things drawn earlier, if they share the same space on the screen.



Your Challenge

Your challenge is to create the JavaScript code to produce blocks in the arrangement and colors shown below.



Remember your overall <**canvas**> size is 800 pixels wide and 600 pixels tall. We have used the color names "yellow," "blue," "green," and "orange." The yellow block is 200 x 300 pixels, the blue is 100 x 200 pixels, the green is 100 x 100 pixels, and the orange is 200 x 100 pixels. We'll let you figure out the starting "X" and "Y" coordinates in each case. If you need a hint, turn to the solutions on the next page!

Activity Solution

The screen shot on the previous page was obtained by saving the following text content to "index.html", and then double-clicking on that file to load it in a web browser.

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is my web page!</title>
  </head>

  <body>

     <canvas style="border:1px solid black;" id="myCanvas"
             width="800" height="600"></canvas>
     <script>
        var canvas = document.getElementById("myCanvas");
        var ctx = canvas.getContext("2d");

        ctx.fillStyle = "yellow";
        ctx.fillRect(0,0,200,300);

        ctx.fillStyle = "blue";
        ctx.fillRect(700,0,100,200);

        ctx.fillStyle = "green";
        ctx.fillRect(700,500,100,100);

        ctx.fillStyle = "orange";
        ctx.fillRect(0,500,200,100);

     </script>
  </body>
</html>
```