code—ometry.

a practical guide to coding w/ the grade 6 mathematics curriculum

this book is property of:

designed+cared for by andrej babić ®

code—ometry.

Foreword:

Started as a thesis topic, ended up as a passion project; code-ometry was and is still probably the most spectacular use of my time. As the byline states, this is a practical guide to coding geared toward grade 6 classrooms based on their mathematics curriculum. This idea first came to me because I felt that there should be more interest in the STEAM fields through a child's life. Through the evolution of this project and my thesis, I realized that coding is not just something that people do as part of their jobs, it is a way of life. Rather, computational thinking is a way of life and this computational approach to problems does a lot more for childhood education than just being able to write a couple of lines of code. Computational thinking helps develop critical thinking and problem-solving, children with these skills score on average 16 points higher on standardized testing than those that do not have this knowledge, and it even increases confidence. So I poured most of my free time into this thesis turned passion project over the last 8 months, I hope you enjoy.

Thank you: {
Mama and Tata;
Jovana and Téa;
Ryanne Spies;
Michael Castledine;
Josh Peressotti;
}

Andrej Babić

intro:

There are so many things that come to mind when someone says code; it can mean speaking in piglatin, it can mean the name of a band or movie, but for our purposes, code is just the vehicle in which we will drive in order to show you what you need to know for your work in math.

Code is the basis of everything that we see on screens, from applications on your phone to the video games you play and every page on the internet. Code is all around us, and sometimes that can be confusing or scary, but code-ometry will make this simple for you.

Can you think of any more examples of code in your everyday life?

1:	
2:	
3:	
4:	
5:	

Great! Let's start by defining all that we need to know for this lesson.

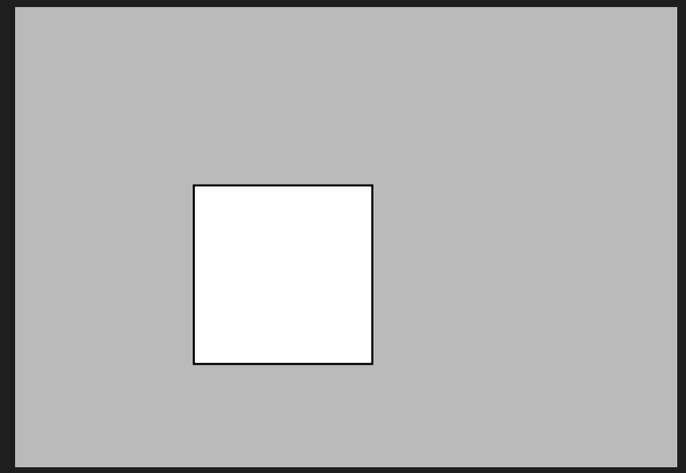
definitions:

Code: Simply put, computer code or just code is a set of instructions that you can give a computer, and order that computer to run those instructions. Without the code, your computer wouldn't work, and with it, your computer is able to do all those amazing things that it does now.

Geometry: A branch of mathematics concerned with questions of shape such as rectangles, the relative position of these shapes and the relationships between those shapes.

Editor: An editor allows you to write code in real-time and see what the results of your computer code are. The nice guy who created code-ometry has created an editor for you all to use to learn geometry through code.

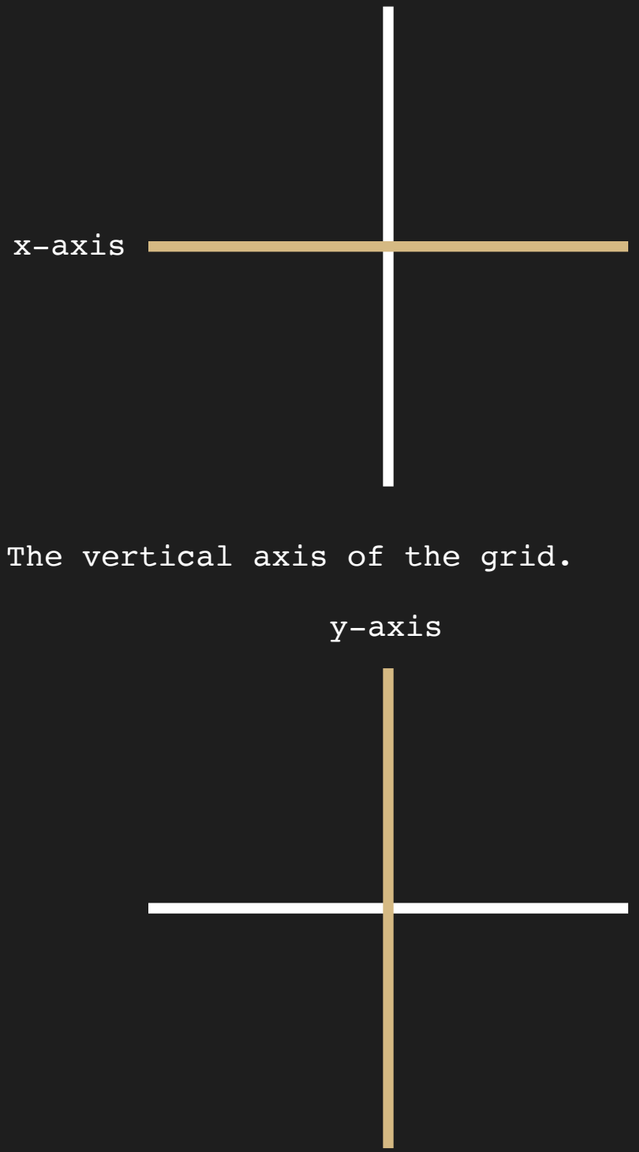
```
rect (100, 100, 100, 100)
```



Javascript (JS): Javascript is a language that is used with the editor. The same way that you use English to communicate with your friends and family, the JS is the way you communicate with the computer.

definition:

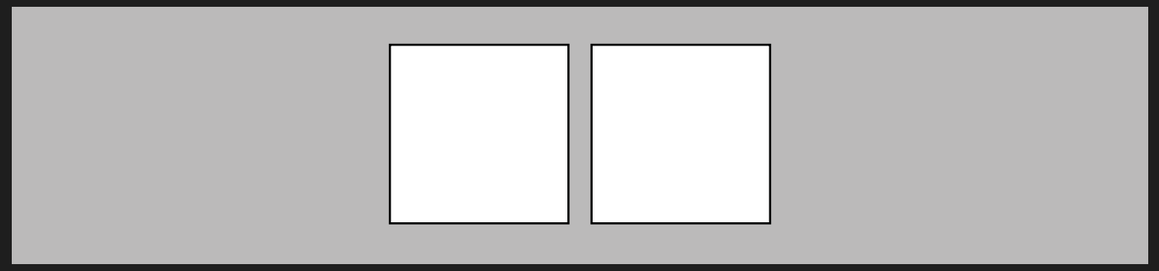
X-Axis: The horizontal axis of the grid.



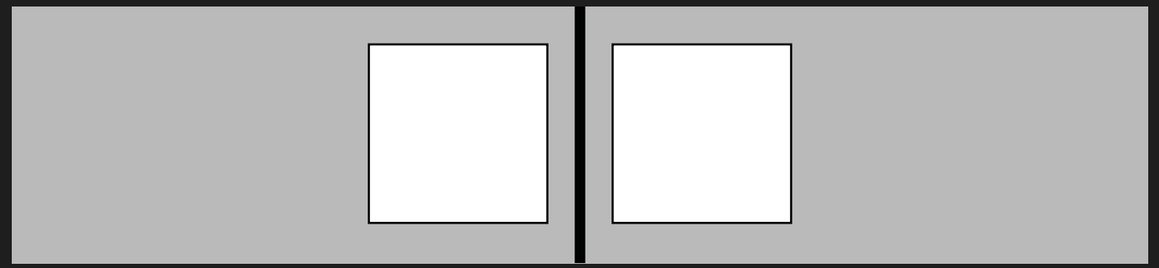
Y-Axis: The vertical axis of the grid.

definitions:

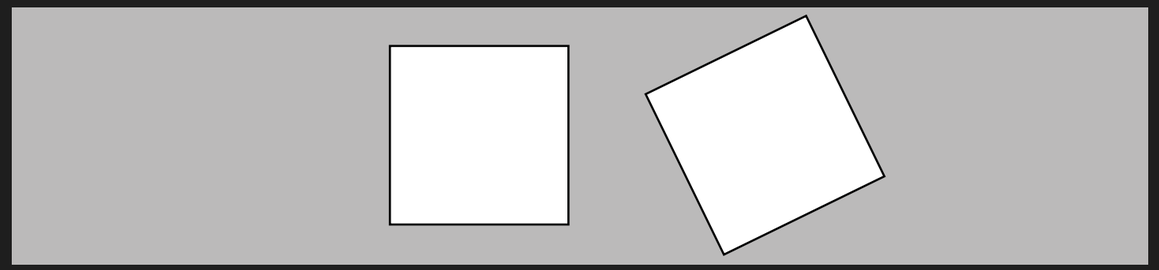
Translation: The moving of a point or figure, up and down (on the Y-Axis), or right to left (on the X-Axis).



Reflection: The flipping of a point or figure over a line of reflection which is called the mirror line.



Rotation: The turning of a figure or object around a fixed point, wherever it may be.



exercise 1:

Let’s keep our work on paper right now before we jump on over to the editor by doing a few geometry problems.

Can you tell what is going on here?

1:

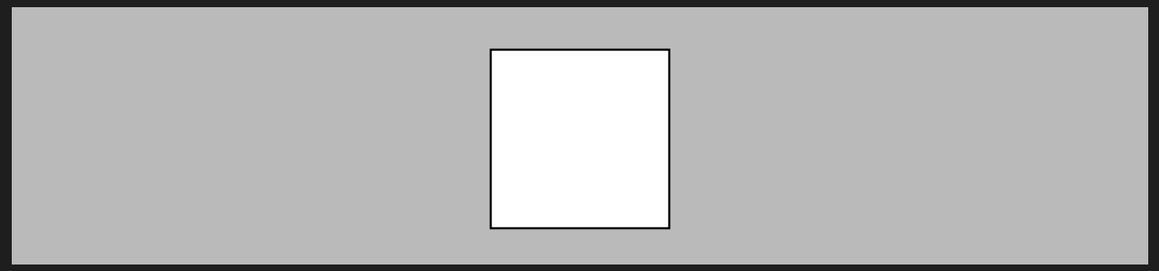
Let’s keep going and figure out what is going on here, soon we’ll be moving on to learning a few things about Javascript and the way that you can communicate with the editor.

2:

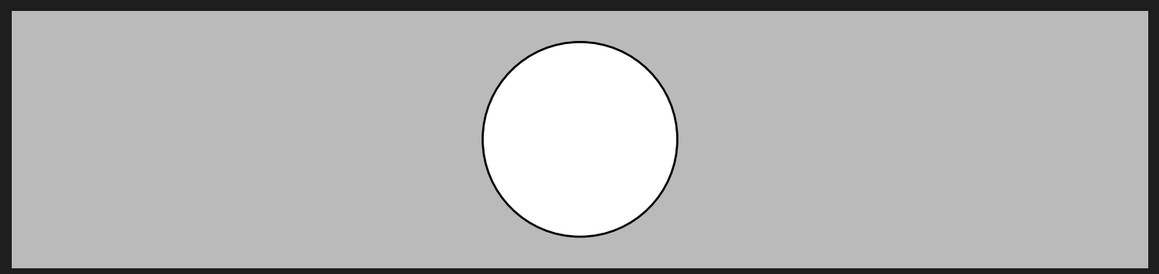
js definitions:

There are a couple of different ways to work with javascript, some more complicated than others. code-ometry has simplified all of these ways to work with javascript in order for you to learn geometry through this code.

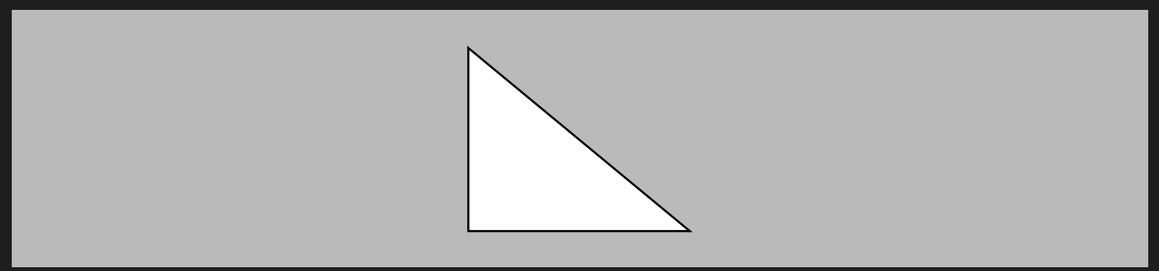
rect = rectangle



ellipse = circle

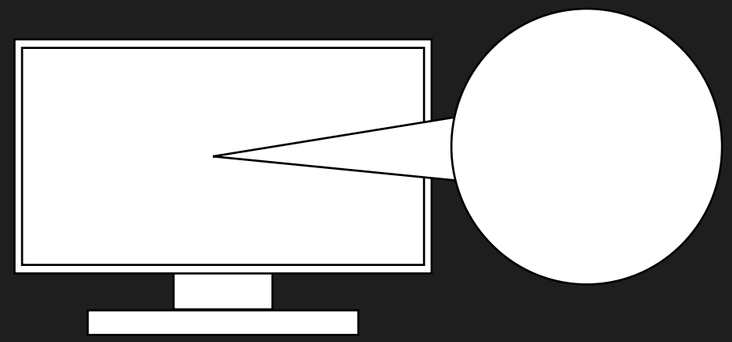


triangle = triangle :)

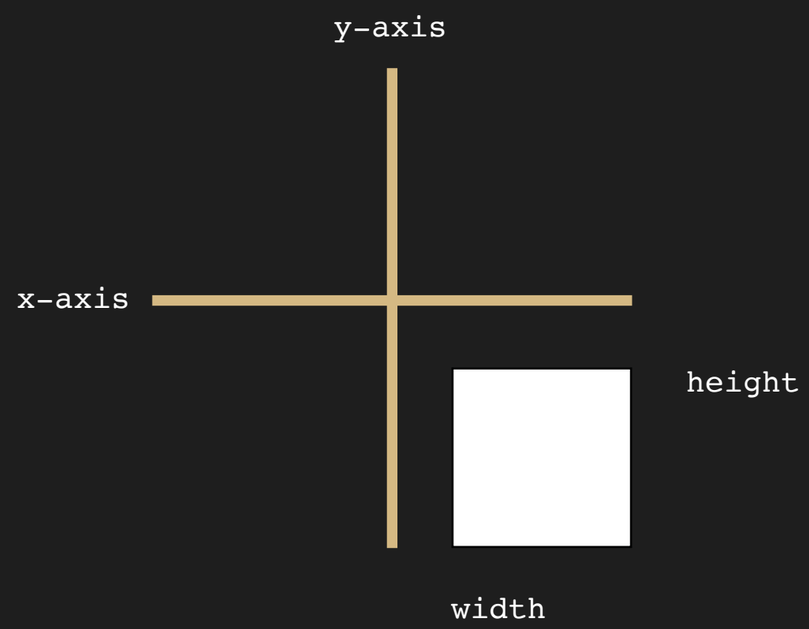


js definitions:

px = pixels (each pixel is a physical space when viewing images online)



(x, y, h, w) = (location on the x-axis with pixels, location on the y-axis with pixels, the height of the shape, the width of the shape)



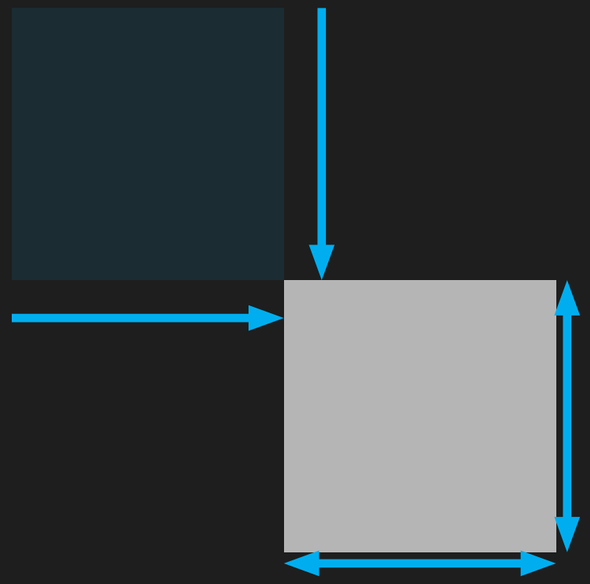
how code works:

Okay, so you've made it through the painful bit about learning all the definitions, now we can get to show you how they all work together. When writing code, we need to declare what it is that we're writing, this can be done by writing one of the shapes that we saw defined earlier such as: rect, ellipse, and triangle. Then we can add to it the (x, y, h, w) with numbers in each of these positions.

Let's look at a few examples.

```
rect (100, 100, 100, 100)
```

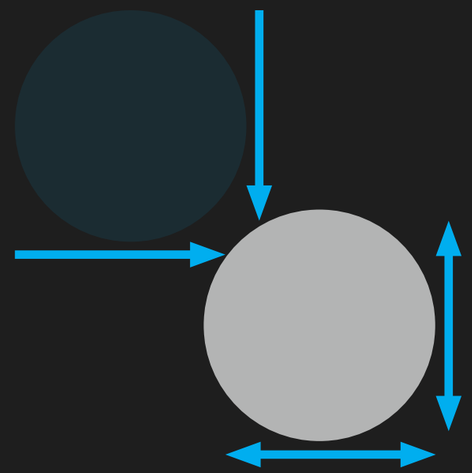
in the code gives us this outcome:



That wasn't so bad, now was it? Let's look at a few more:

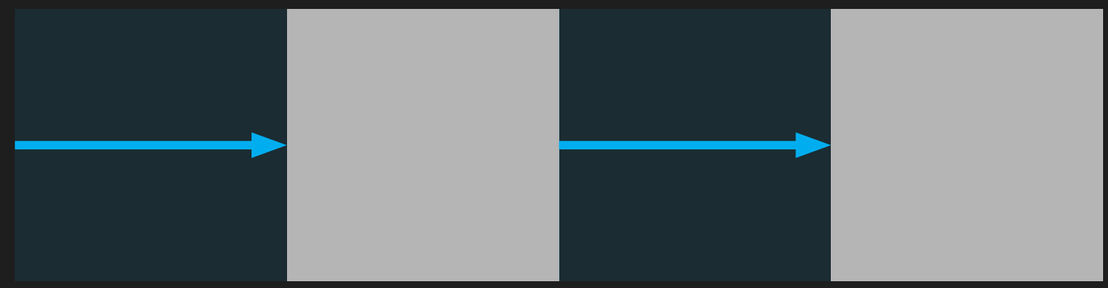
```
ellipse (100, 100, 100, 100)
```

In the code gives us this outcome:



```
rect (100, 100, 100, 100)
rect (300, 100, 100, 100)
```

In the code gives us this outcome:



exercise 2:

Continuing our work on paper, now that you know how to declare code and how the (x, y h, w) work, we can work with these shapes to see how they’re built.

See if you can tell what is being declared here, and what the (x, y h, w) are:

1:

Awesome, now can you tell what is happening here?

2:

define exercise:

We have went through and learned the basics of code, the basics of geometry, and their relationships to one another. We have even looked at some geometry problems, and looked at how to solve those geometry problems from a perspective of code. It's time for a little pop quiz (as if you haven't had one of those before), and when we get through this, we can begin coding!

Define code:

Define reflection:

Define: x-axis:
(here's a hint)

Define translation:

Define y-axis:
(here's a hint)

Define rotation:

Bonus Question! Define JavaScript:

let’s code:

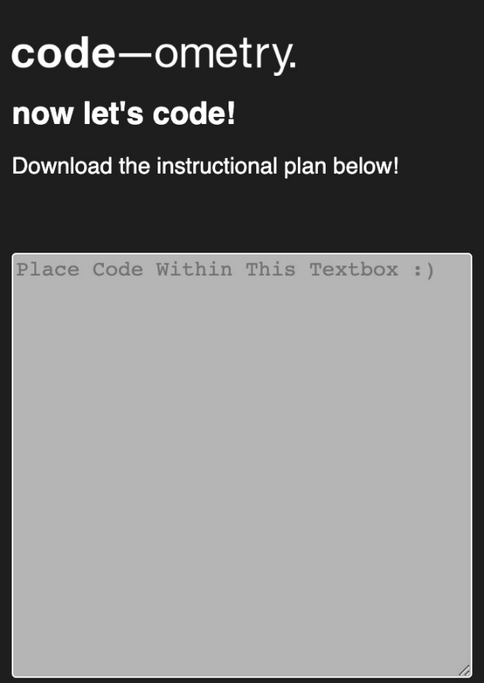
WOOHOO! Alright, I’ll try to contain my excitement... But you’re well on your way to becoming a programmer! Or a mathematician, I guess... Maybe both!

Lets visit the website _____ and make sure to to keep reading on.

Code-ometry has a web editor (which we’ve learned the definition of a little earlier), and we are going to use this editor to build some shapes.

It should look a little like this:

It should look a little like this:

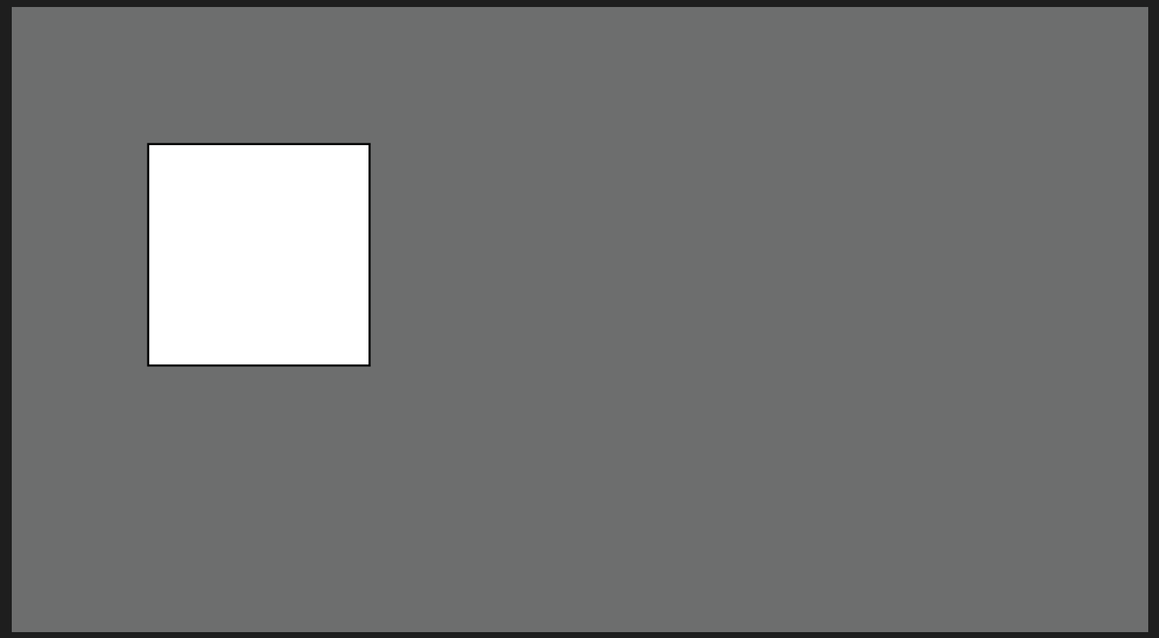


Let’s get to coding. We have defined rect as rectangle earlier. So we’ll make a nice and simple square on the web editor.

Try typing this into the textbox on your code-ometry site.

```
rect (100, 100, 100, 100)
```

It should look a little like this:



keep going:

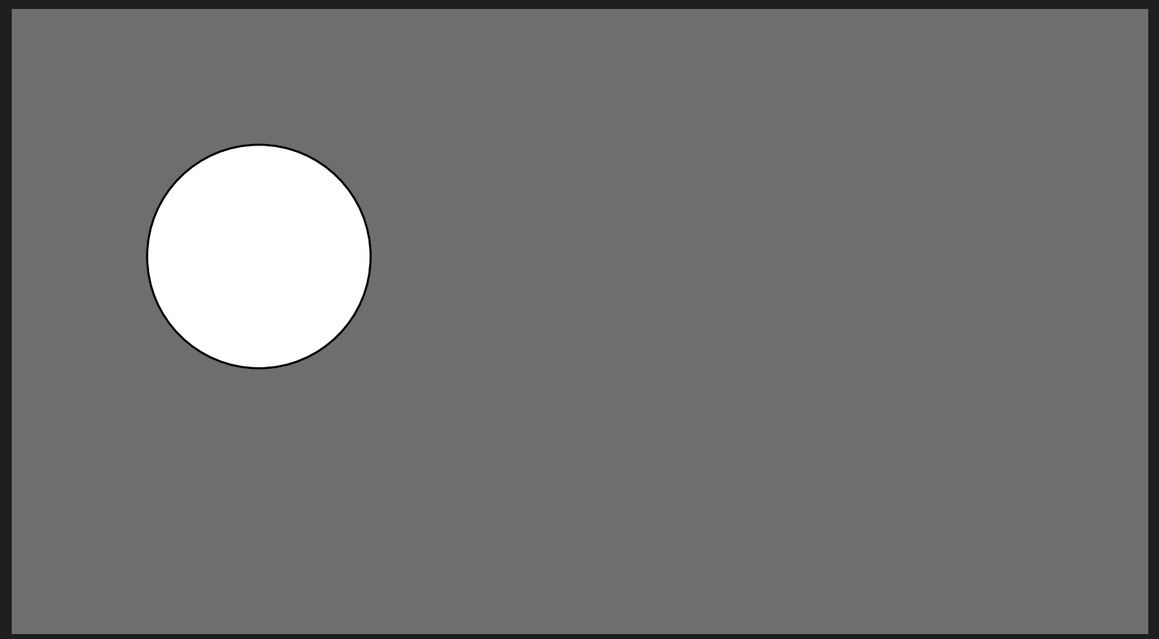
Well, I think you can now call yourself a coder.

What's that? You don't believe me? Then, let's reload the page and try another thing, and this one won't be so easy.

Let's make a circle with the same x-axis and y-axis, and the same height and width.

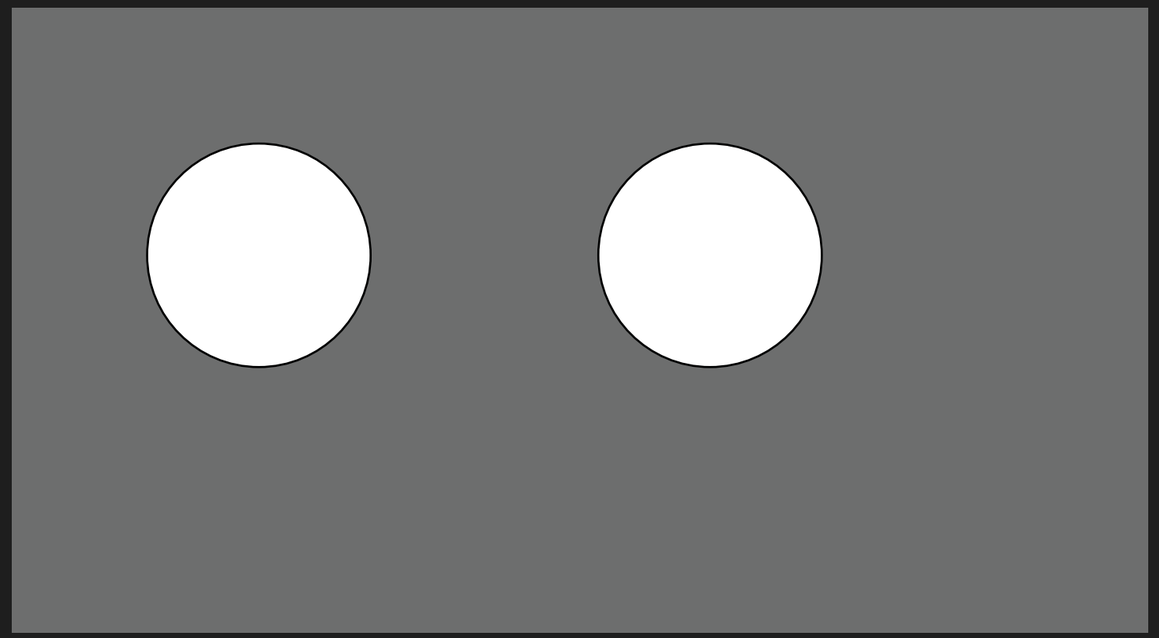
Remember, a circle isn't called a circ the way that a rectangle is declared as a rect.

When you're done, it should look a little like this:



Now let's get that circle moving.

With the code for a circle in the textbox, hit enter on your keyboard and start a new line of code. Take the x value of the circle and increase the number by 200.



There are two different things that could have happened to this circle to make it look this way, what are they?

1:	
2:	

exercise 3:

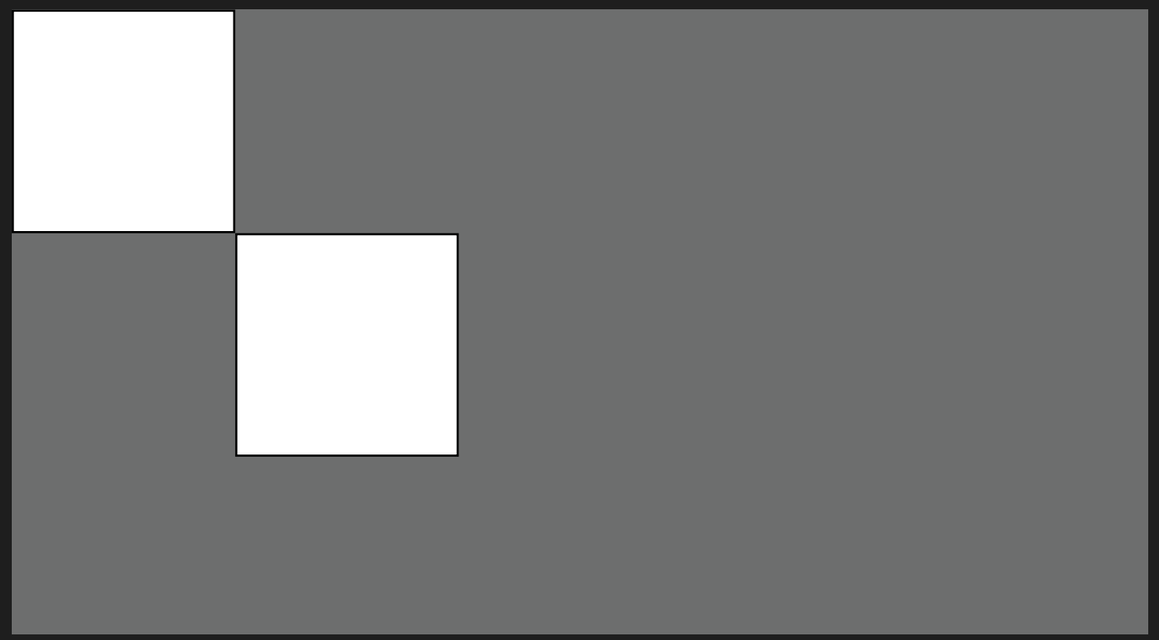
Now that you're a coder, we can focus on both the code and the geometry. code-ometry, hah, get it?

Draw a rectangle that has an x-axis value of 0, and a y-axis value of 0, height 100, and width 100.

Then hit enter and draw another rectangle with the same height and width values, but with an x-axis value of 100 and a y-axis value of 100.

Hint: Don't let the wording of the exercise mess you up, it is still always (x, y, h, w).

Here's what your code should look like:



Now there are three different things that could have happened to this circle to make it look this way, what are they?

1:

2:

3:

triangles:

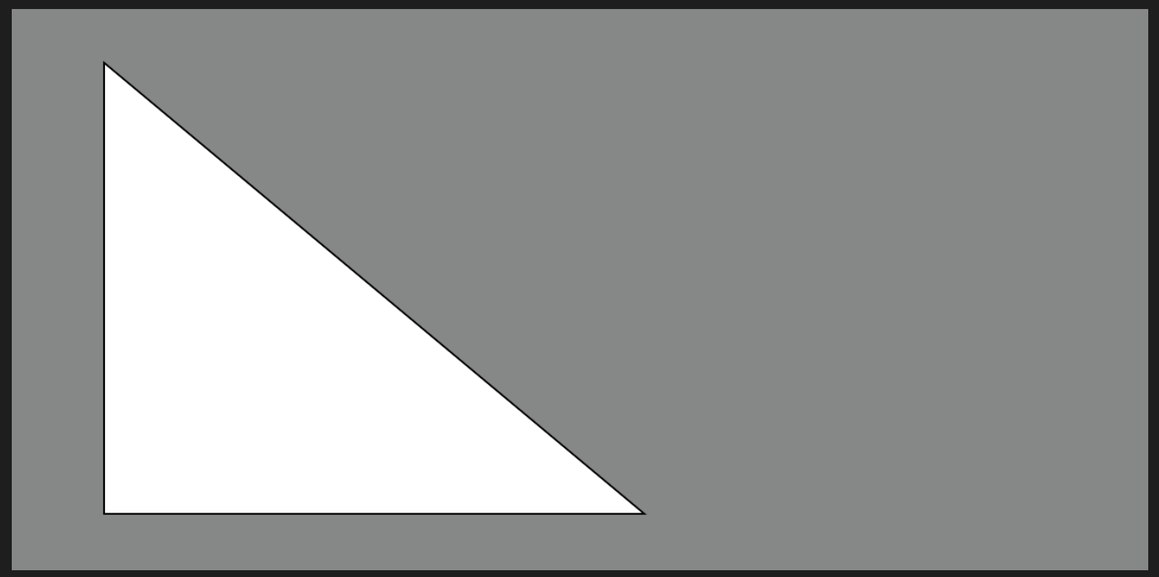
While rect and ellipse definitions call for four numbers (x, y, h, w), triangles don't play by those rules. There are three points on every triangle, no matter what. That's what makes it a triangle. While rectangles and circles can have two values for how they're made (the height and width of the shape), triangles have to have six values for how they're made, as each point has to have an x-axis value and a y-axis value.

It may be easier to remember if I help you out with the code.

Try reloading the web editor and typing in:

```
triangle (100, 100, 100, 350, 400, 350)
```

It should look a little like this:

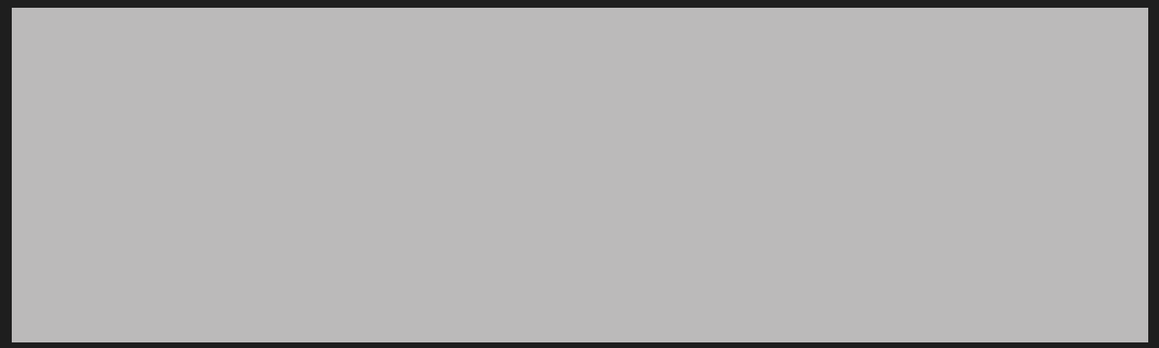


Let's look at this code and how it runs on the code editor:

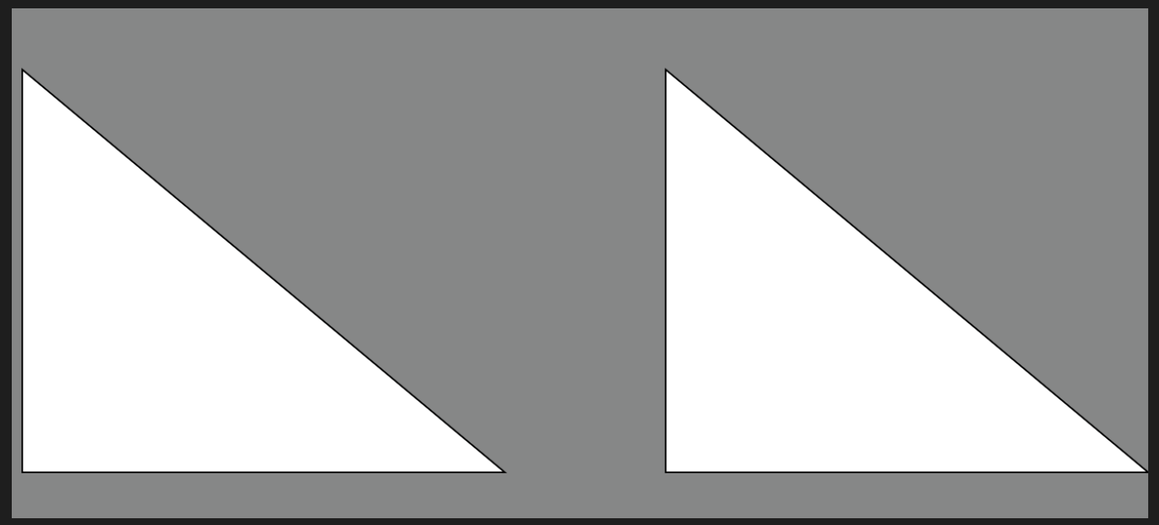
```
triangle (100, 100, 100, 350, 400, 350)

triangle (500, 100, 500, 350, 800, 350)
```

What's going on here?



Example:



triangles:

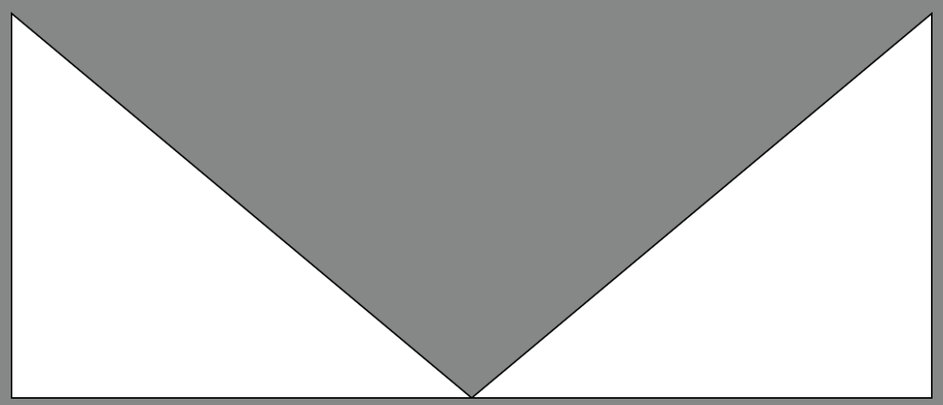
Now let’s look at this code and how it runs on the code editor:

```
triangle (100, 100, 100, 350, 400, 350)

triangle (700, 100, 700, 350, 400, 350)
```

What’s going on here?

Example:



If you’ve made it this far, I’d like to be the first to say congratulations!

Secondly, thank you for sticking through the code-ometry practical guide to coding in the grade 6 mathematics curriculum and learning a little bit about how code and mathematics work hand in hand.

Hopefully, you enjoyed learning a little bit of the javascript that was in this book and if you didn’t, at least it’s over now!

Thanks again,

Andrej Babić

code—ometry.

designed+cared for by andrej babić ®