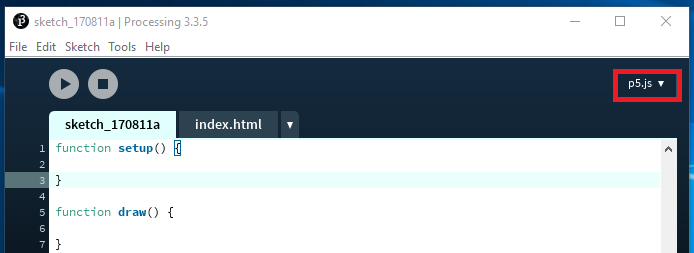
Pong

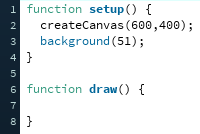
Introduction

Pong is one of the earliest arcade video games and the first sports arcade video game. It is a table tennis sports game featuring simple two-dimensional graphics. It was released in 1972.

1. Start Processing and make sure you have selected p5.js from the drop down menu. You’ll see two functions, one called **setup()** and one called **draw()**. setup() happens only ones – when press Play. draw() happens many times per second like the “Forever” block in Scratch.

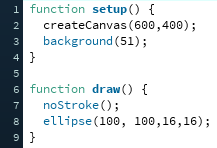


1. Start by creating a canvas in **setup()** and giving it a background color – this canvas is the area our game will occupy.

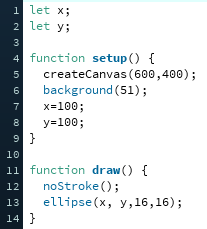


1. Next, let’s draw a circle in **draw()**. **noStroke()** means we don’t want a line around it. **ellipse()** takes 4 parameters the first two tell it the location, the second two tell it the size.

**CHALLENGE**: Try making an oval rather than a circle. Can you colour it using the **fill()** command?



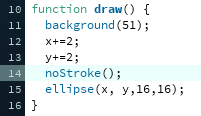
1. The circle we’ve just created will be our ball so we don’t want it to sit in one spot. Create two variables one called **x**, the other called **y** and set them both to 100 in **setup()**. Then use them as the location parameters for our ball.



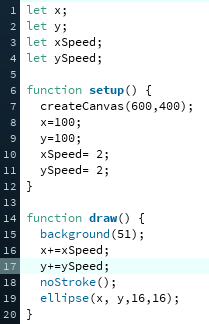
1. Now we have variables telling the circle where it is on the screen we can make it move. In the **draw()** function we want to increase **x** by 2 and **y** by 2.

NOTE: the notation x += 2 is the same as writing x = x + 2 – it means we want to increase the value of x by 2.

CHALLENGE: Can you make the ball go the opposite direction across the screen?



1. We want to be able to change the speed, so make two variables **xSpeed** and **xSpeed** then initialise them both to 2 in **setup()**. In **draw()** we now want to increase **x** by **xSpeed** and increase **y** by **ySpeed**.



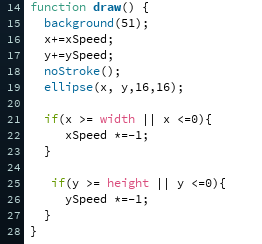
1. When our ball reaches the edge it keeps going out of sight, we want to make our ball bounce around the screen. To do this we need to check if the ball’s position is off the screen. To do this we can make use of the **width** and  **height** variables that are built in.

**width** and **height** are defined when you **createCanvas()** in setup in our case **width** = 600 and **height** = 400.

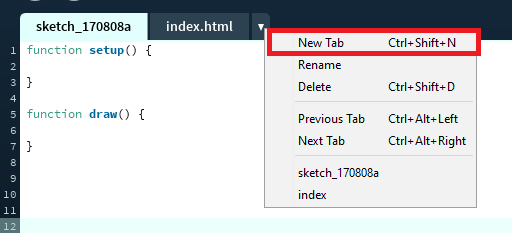
Once we’ve drawn our ball we want to check whether it is off the screen so we look to see:

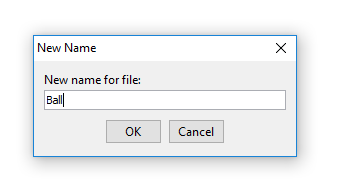
* If x is larger than the width: this means the ball is off the right side of the screen
* If x is less than 0: this means the ball is off the left side of the screen
* If y is larger than the height: this means the ball is off the bottom of the screen
* If y is less than 0: this means the ball is off the top of the screen

If any of these statements are true, we want to reverse the speed (so 2 becomes -2)



1. Our code is getting rather long now, and if we wanted to make another ball we’d have to duplicate a lot of code – developers don’t like duplicating code, it means much more typing for very little value. If you click on the arrow next to **index.html** you can create a ‘New Tab’ named “Ball”

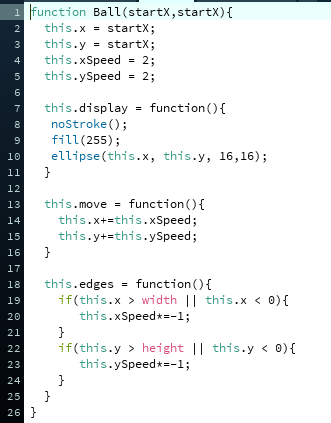




In this new tab we want to take all the code that affects the ball and create a series of functions and variables that will define the ball as an object. Notice that all the variables have “**this.**” In front of them – this is because we’re operating within an object so we need to make sure all the variables belong to **this** object.

The functions we need for the ball are:

1. display(): this will contain all the code that displays the ball
2. move(): this will contain all the code that moves the ball
3. edges(): this will contain all the code that checks where the ball is



Your main tab will now be rather empty and if you start the game your ball will have vanished. This is because we need to create a ball **object** to use the code in our new tab. Before **setup()** create a **var**iable called **b** and inside **setup()** make **b** a **new Ball()** at position 100,100. Inside **draw()** we need to call those functions we made inside **Ball()**.

