Meet the Arc Function

WE'LL COVER THE FOLLOWING

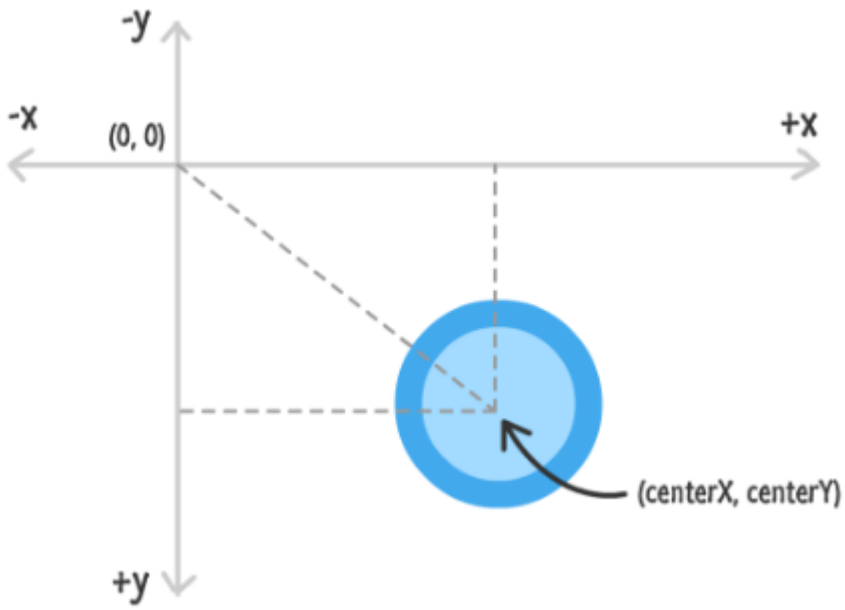
The way you draw a circle in your `canvas` by using the handy `arc` function. This function and the arguments you need to specify in order to use it look as follows:

```
arc(centerX, centerY, radius, startAngle, endAngle, isAntiClockwise);
```

These arguments are important in helping you draw the circle that you want, so let's look in detail what all of these arguments do.

centerX and centerY

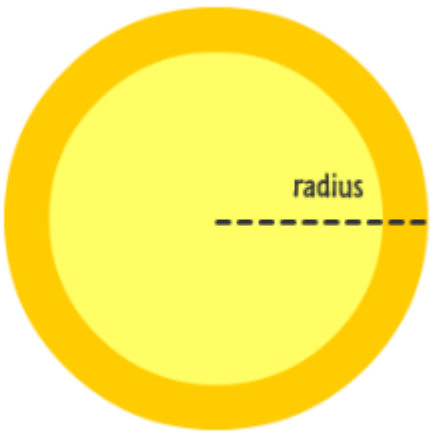
The `centerX` and `centerY` arguments are pretty straightforward to understand. They specify where the center of your circle will be positioned inside the canvas:



Remember, the canvas lives in an inverted Cartesian system. What this means is that the x value increases as you move right, and the y value increases as you go down. This might be a little different than what you may remember from graphing equations in school.

Radius

The radius specifies the straight line distance of your circle from its center to any edge:

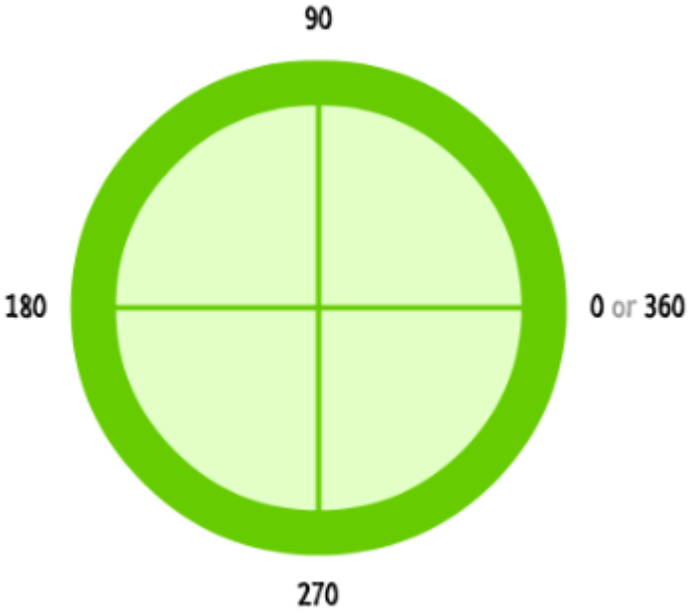


No Image selected

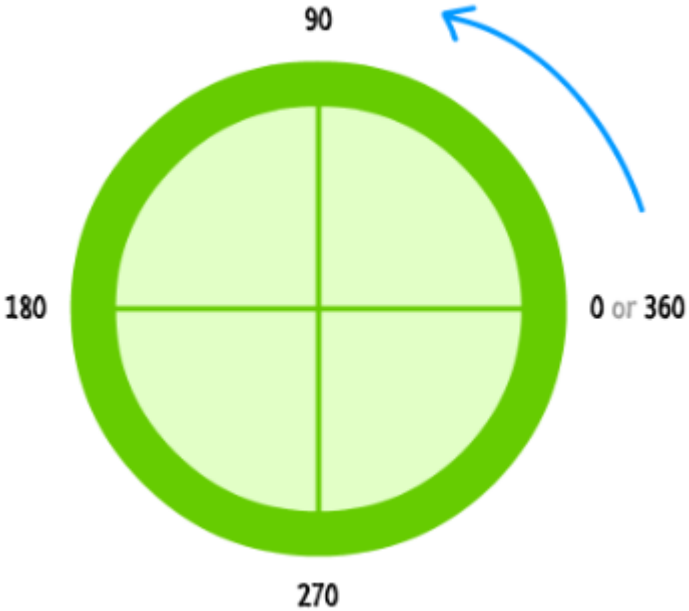
The larger your radius, the bigger your circle will be. The smaller your radius, the smaller your circle will be. If you provide a negative value, JavaScript will throw a nice `IndexSizeError` exception, so you don't want to do that.

startAngle, endAngle, and isAntiClockwise

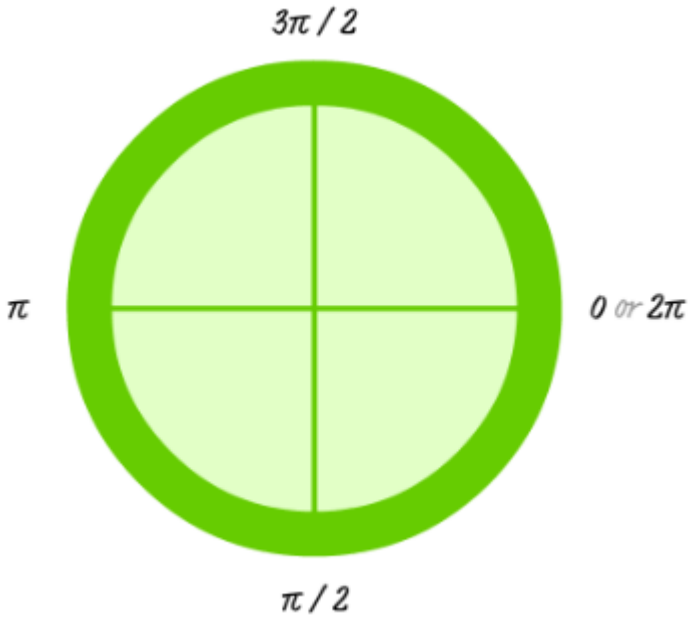
Now, we finally get to the interesting stuff. These three arguments are interesting and closely related to drawing your circle. As you probably know, a circle is made up of 360 degrees:



There are two important details to note - details that will probably shatter your belief in all that is good in this world. The first is that the angles increase clockwise for a circle when drawn in the `canvas`:



The second detail is that JavaScript doesn't work with degrees. In JavaScript land, you deal with everything in terms of radians:



Once you understand these two details, you crossed a big hurdle in mentally being able to visualize what your arc function will create.

> ### Note: Converting from Degrees to Radians
>
> To convert from degrees to radians, just use the following expression: `var radians = (Math.PI / 180) * degrees`.

Ok, let's now take a step forward and work through how the **startAngle**, **stopAngle**, and **isAntiClockwise** arguments play a role. There are three steps you need to follow:

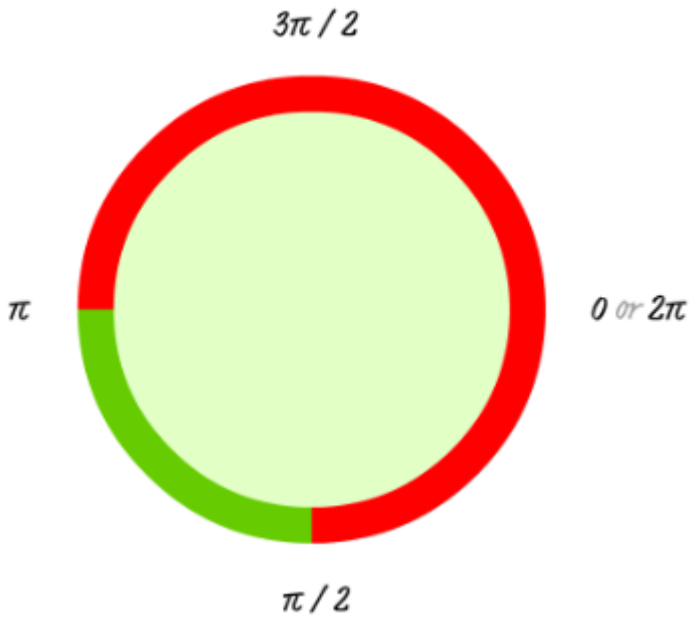
1. Mark your **startAngle**.
2. Mark your **stopAngle**.
3. Draw a line on the circumference either clockwise or anticlockwise depending on whether your value for **isAntiClockwise** is `true` or `false`.
4. If you are filling in your circle, fill in the region enclosed by the circumference and the straight line between the points referenced by **startAngle** and **stopAngle**.

Let's look through an example of this. Let's say your start angle is π / 2, and your end angle is π. You are also anticlockwise and centered at 200, 200 with a radius of 93.

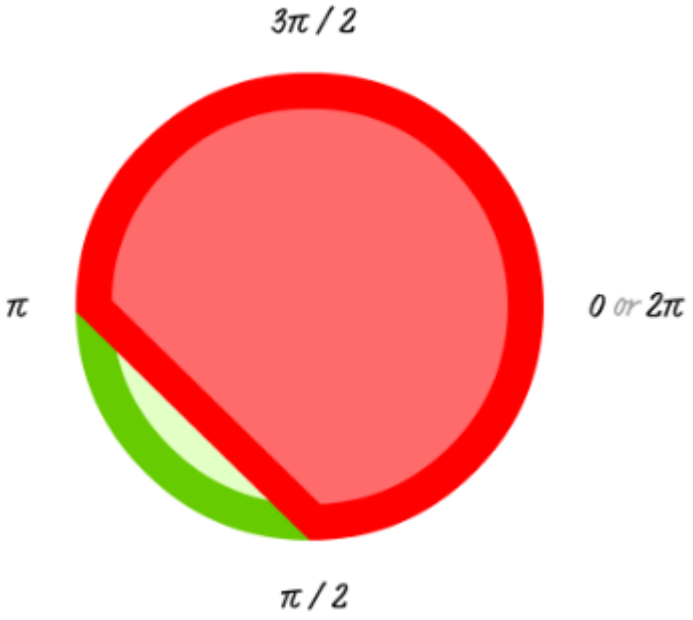
Given those values, the `arc` function would look as follows:

```
arc(200, 200, 93, Math.PI / 2, Math.PI, true);
```

If you had to visualize this, here is what your circle with just the stroke defined would look like:



If you filled in this circle, here is what you would see:

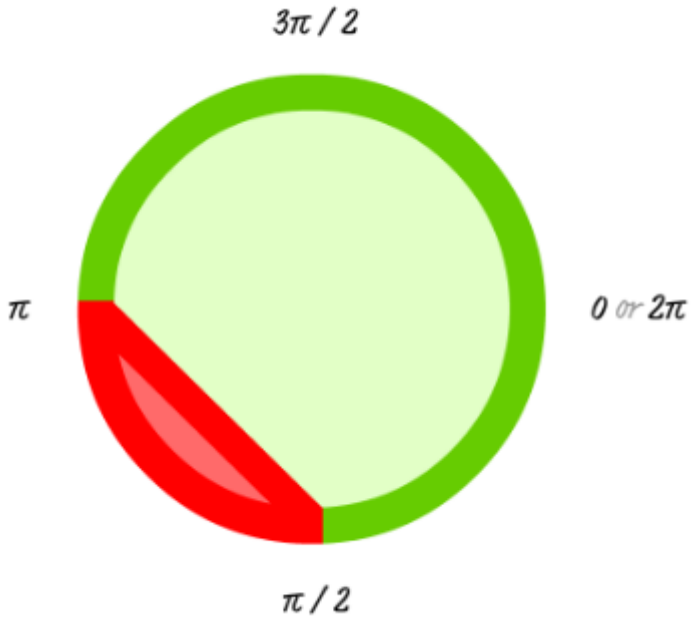


Notice that your circle's start and end points are defined by your **startAngle** (π / 2) and your **endAngle** (π). Because you are going anti-clockwise, notice that the outline and the colored region goes all the way around the circle on the long side.

If you switch from being anticlockwise to being clockwise but keep all of your other values the same, your `arc` function now looks as follows:

```
arc(200, 200, 93, Math.PI / 2, Math.PI, false);
```

As a result of the direction being changed, your circle takes a different turn (ha!):



Whenever you run into the `arc` function and need to visualize what the final circle looks like, use the four steps I described earlier. Those steps hold for whatever combination of **startAngle**, **endAngle**, and **true/false** you provide for the anti-clockwiseness of your circle.