Solar System Simulator

For this project you will practice your skills making objects and classes by creating an animated solar system simulator. The following are the required celestial bodies that your simulation must contain.

- Sun (Sol)
- Mercury
- Venus
- Earth
    - Moon (Luna)
- Mars
- Jupiter
    - Lo
    - Europa
    - Ganymede
    - Callisto
- Saturn
    - Titan
- Uranus
- Neptune
    - Triton
- Pluto

Don't make any of the other bodies unless you have the sun, earth, and moon working
If you need help with classes here are two videos.

6.2: Classes in JavaScript with ES6 - p5.js Tutorial
6.3: Constructor Arguments with Classes in JavaScript - p5.js Tutorial

- Your Celestial class should at a minimum contain properties and/or parameters to represent the following.
    - constructor

```
▼ Celestial {x: 200, y: 200, radius: 50, angle: 0, constructor: Object}
   x: 200
   y: 200
   radius: 50
   angle: 0
  ▶ <constructor>: "Celestial"
```

        - radius
            - Does not have to be to scale but should be sized from largest to smallest, excluding the sun.

```
let sun;
let earth;
let luna;

function setup() {
  createCanvas(400, 400);

  sun = new Celestial(50);
  earth = new Celestial(25);
```

```
  luna = new Celestial(5);
}
```

- display
  - color
    - Pass a Hex string as the color

```
sun.display("#F7AF14");
```

- orbit

```
earth.orbit(sun, 100, year);
```

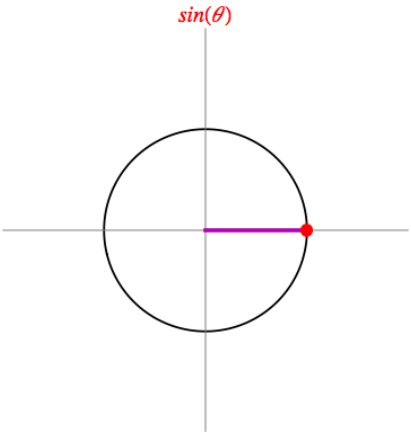
  - body
    - The other celestial body to orbit
  - distance
    - Orbital distance *Not to scale*
    - This is the distance between the bodies perimeters.
    -
  - period
    - Orbital Period: The time taken for the planet to orbit the Sun, given in Earth years for clarity
  - retrograde (optional)
    - A boolean that represents whether the body is moving in retrograde or prograde motion

You will also need two constant variables in the global scope to represent a day and a year. Experiment by changing the day variable to get a good speed for your simulation.

```
const day = 1;
const year = day * 365.25;
```

- Optional: Do some research to track down actual planetary positions for a given date, and use this information to initialise each body's orbit angle.

- Use of both the sine and cosine functions to calculate and animate orbital motion.



- See the Wagon Wheel example for help

- Issues of scale. Set the orbital distances to a fixed interval but the radii should be scaled with the exception of the sun.

Expert Challenges

- Create a more accurate elliptical or based on Kepler's Laws
- Create an asteroid Belt.
- Make the planet's rotate on their axis

For fun see this painfully accurate simulator