**Object oriented Programming (OOP)**

we are going to learn a different way of writing programs that is used by professional programmers worldwide, and it can make our code cleaner, and we can avoid writing the same type of code multiple times. This is called object-oriented programming. In OOP we have different terms which are very important to understand. These are classes and objects.

**class**

A **class** contains information about a category. A **class** can be understood as a **blueprint**.

For example, before creating a car engineers create a blueprint of the car, and then create multiple cars which can be modified based on different users from that blueprint.

Another example, consider a class of dogs, but as we know all dogs are not the same. We have multiple breeds such as labrador, pug, German shepherd, etc. But if you observe carefully, all the dogs have some things which are common in all the breeds, such as all dogs have 4 legs, 2 ears, 2 eyes, and they are furry, they wave their tail, they bark, etc. But depending upon their breed they have very specific features which make them different from other breeds, but they also have a lot of common features which make them look like dogs. So in this example, Dog is our class, which has all the information about how to create a new dog and what features to add to that dog. This information is called attribute and functionality. Attributes of a dog are height, length, fur color, and the functionalities are run, bark, howl.

When we define a **class** in code, we can choose what attributes and functionalities we want to add to the class. Once we have the class ready then we can create multiple objects with the class.

The benefit of OOP is that we can keep our code in multiple files which makes our code more manageable.

**Define a class.**

To define a class, first, we write the keyword **class** in lowercase then we write the name of the class which has the first letter in uppercase in our case, Box. Then we begin with the parentheses, all the code related to the class will be kept inside these parentheses.

class Box { ... }

**constructor**

To add attributes we create a **constructor**() function. It's just like our normal function, but we don't need to write the keyword function. When our code will run, the constructor is the function to be executed. In this constructor, we will define the properties. This is interesting since when we define properties inside a class, we will use this keyword.

So to set the position of the box, we will write this.x = 100;

**this** keyword means that we are referring to this particular constructor.

And similar for y, this.y = 200, this.w = 50—width of the box, In class,

whenever we need to use any property such as x, y position, we have to use the **this** keyword, otherwise the program will have an error that it does not understand the property.

**Use the class**

In the sketch.js file first, we will create a variable for the box, let’s call it var box1. This variable will become the **object** of the class. To create the object in the setup() function we write

box1 = new Box().

The **new** keyword is used to create the object and after the new keyword, we add the name of the class. Now, the name of the class, its first letter is capital just like when we created the class.

We have the object of the class, now we can use this object to call the function of the class.

We need to **call** these functions so that we can observe the effects on the object. This we will do in the **draw**() function of the code. To call the function from the class, write the name of the object followed by a ***dot*** then the function name.

So it will be

**box1.show()**