Week 9: Last week for 1010  
  
  
In week 9, we have learned about object-oriented designing (drawing with objects) and about “How to become a better programmer” using the help of Pseudo code. Object-oriented designing, as stated earlier, is essentially drawing using the help of what we have learned in week 8, objects. For this you essentially make objects with variables (Values that can change) and make a function that creates it. Here is what I mean.  
  
var personChildren = {

name: “Ken”,

age: “5”,

xPos: 20,

yPos: 50

}

var personChildren = {

name: “Ken”,

age: “30”,

xPos: 229,

yPos: 50

}

var drawPerson = function(person) {

fill(255, 0, 0);

var img = getImage("creatures/Winston");

image(img, person.xPos, winston.yPos);

var txt = person.name + ", " + person.age;

text(txt, winston.xPos+20, winston.yPos-10);

};  
  
drawPerson(personChildren);

drawPerson(personAdult);  
  
  
This now shows two “people” on the screen. What this does exactly is that you precise specific values in the object. After that, you send it to the function and the function gets the values it requires, you then run the function and voila.   
  
  
There is another way to do this though, you can also us things called methods. Essentially, this way you make a function instead of an object. Make so parameters etc… and you then draw it. Here is what this would look like.  
  
var Person = function(name, age, x, y){

this.name = name;

this.age = age;

this.x = x;

this.y = y;

};

Person.prototype.draw = function(){

fill(255, 0, 0);

var img = getImage("creatures/Winston");

image(img, person.xPos, winston.yPos);

var txt = person.name + ", " + person.age;

text(txt, winston.xPos+20, winston.yPos-10);

};

var personChildren = new Person(“Ken”, 5, 20, 50);

var personAdult = new Person(“Ken”, 30, 20, 229);

personChildren.draw();

personAdult.draw();

This is the same principle as before. Here we create a function caller Person. This function has parameters you can change to customize the person you will create. In this function though, you have lines such as “this.name = name”. These lines are essentially making an object with these specific “keys” or “sub variables” and change it to the value given.  
  
You then have a method which is essentially a function for an object, which draws the “Person”. You probably noticed that in the function name, it says “Person.prototype.draw”. What does the “prototype” mean? This means that the function “draw” can be used on any instance of “Person”. Essentially, if it is a person, it will draw them.  
  
After that, we only have 4 lines left. Those are what create the actual person. You create a variable from the function Person along with parameters to customize the person. You then call the draw function from these variables.  
  
  
Next thing we have learned is something called object inheritance. This is essentially a way to not duplicate functions with different names. Here is what I mean.

var person = function(name, age, x, y){

this.name = name;

this.age = age;

this.x = x;

this.y = y;

};

person.prototype.talk = function() {

text(txt, this.x+10, this.y-7);   
}

var animal = function(name, age, x, y){

this.name = name;

this.age = age;

this.x = x;

this.y = y;

};

animal.prototype.talk = function() {

text(txt, this.x+10, this.y-7);   
}

Hmmmmmm…. These functions are almost the same, the only difference is the name. You might need it to be different function because person is drawn a specific way but you might want animal to be drawn a different way. I cannot help with the actual method itself but you can notice that they both have a function called talk. What we can do instead of duplicating that function is do this instead.  
  
var Creature = function(name, age, x, y){

this.name = name;

this.age = age;

this.x = x;

this.y = y;

};

Creature.prototype.talk = function() {

text(txt, this.x+10, this.y-7);   
};

var Person = function(name, age, x, y) {

this.name = name;

this.age = age;

this.x = x;

this.y = y;

};

Person.prototype = Object.create(Creature.prototype);

Person.prototype.draw = function() {

fill(217, 90, 0);

var img = getImage("creatures/Hopper-Happy");

image(img, this.x, this.y);

var txt = this.name + ", " + this.age;

text(txt, this.x+10, this.y-7);

};

var Person1 = new Person(“Ken”, 5, 20, 50);

var Person2 = new Person(“Ken”, 30, 20, 229);

Person1.draw();

Person1.talk();

Person2.draw();  
  
  
Now you get access to the function “talk” to any object that is an instance of Creature! This is done thanks to the line:

Person.prototype = Object.create(Creature.prototype);  
  
This makes Person simply become an instance or child of Creature. This allows you to get the functions of the object Creature.

The last thing we have learned this week and as such for JavaScript is called Pseudo code. Pseudo code is essentially you writing exactly what you want to happen in plain text. You can do this using comment. Essentially it is simply a to do list that has absolutely every step written in English or any language you would like. This is just so you can understand what needs to happen to have your result.  
  
  
In recap, we learned how to use methods and objects to draw things multiple times. We can also use a preset for your methods. We have also learned about Pseudo code which is essentially plain text version of every step you need to do.