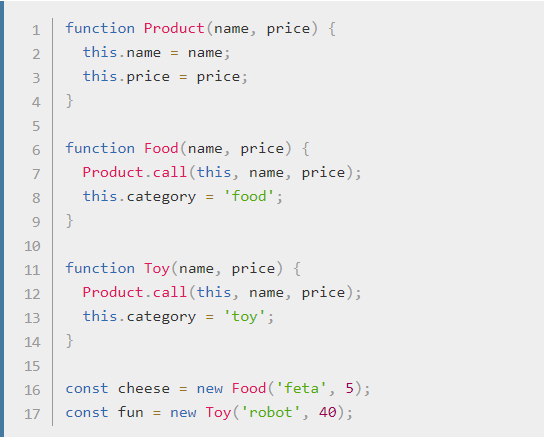
**Day 2 Lab Assignments**

1. Create factory function that returns an object of course [Use factory function pattern].
2. What’s the output of the following piece of code? Explain the code and the output.

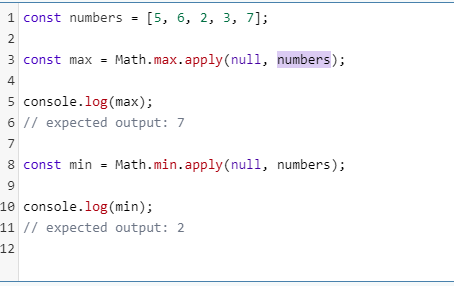


* + 1. Explain the use self-invoking functions in the previous piece of code.
    2. Explain the use of closures in the previous piece of code.

1. Explain the use of Function.call() in the following example:



1. Explain the use of Function.apply() in the following example:



* 1. What’s the difference between Function.call() and Function.apply().

1. Using ES6 new OOP syntax:
2. Implement Rectangle custom type:
   1. Constructor that width and height.
   2. CalcCircumference() function that returns the circumference of the rectangle.
   3. WhoAmI() Static function that prints text (“I am rectangle”).
3. Implement Square custom type, that inherits from rectangle, and change the following:
   1. Make the constructor of square takes one parameter only.
   2. In the square constructor, make width and height of the parent equal.
      1. You’ll need to call the super() constructor in the square constructor to access the parent object properties.
   3. Override circumference () to return circumference of the Square.
   4. Override WhoAmI() function to print text (“I am square”).
4. Create objects from rectangle and square and test them, and make sure the square inherited rectangle members.
5. Implement the following design patterns in JS: Module, Mixin and decorator and observer (use a full demo for implementation and using of each pattern):
   1. You can check the following links:
      1. <https://dev.to/shijiezhou/top-10-javascript-patterns-every-developers-like-168p>
      2. <https://scotch.io/bar-talk/4-javascript-design-patterns-you-should-know>
      3. <https://www.toptal.com/javascript/comprehensive-guide-javascript-design-patterns>
      4. <https://www.dofactory.com/javascript/design-patterns>

**Bonus:**

1. Explain Function.bind() using an example.
2. Implement other design patterns in Javascript.
3. Implement Stack custom type using object oriented in JS (pop and push member functions, and currentSize property) [Use ES6 new OOP syntax].
4. Display the area and perimeter of an object created from using Rectangle Constructor that has width and height properties and 2 extended methods for calculating area, perimeter .
   1. Override .toString() function to display a message declaring the width , height, area and perimeter of the created object.
   2. Create Class Property that counts numbers of created objects and Class method to retrieve it.
   3. Make proper updates in your code to make Rectangle inherits from Shape Constructor, and Square inherits from Rectangle.
   4. Make sure that any created object from Rectangle is instance of both Rectangle and Shape, while any created object from Square is instance of Square, Rectangle and Shape.
5. Implement linked list using [create custom object node, and custom object linkedList, to simulate linked list].
6. Make your own mathematical constructor that add, and multiply values of an array, based upon passing required mathematical operation. Let this constructor takes 3 parameters: 1st represents array 2nd parameter represents initiation of the collecting storage (i.e.1 for multiply, 0 for adding) 3rd parameter represents the required operator. Let this function contains an inner function that uses switch..case for handling different operations. Note: This constructor may have an IIFE if needed.

**<Script>document.write(“Thank YOU”) </Script>**