**Design patterns**

**Singleton**

The singleton design pattern limits the number of instance to just one. It is a creational pattern (results in creating a new instance of a class/object)

Const singleton = (function() { // IIFE

Let instance; // undefined var

Function init() { // Public and private methods

Let hello =”hello world”;

Function printHello() {

Return hello;

}

Return { // private methods

callPrivateMethod: function() {

console.log(printHello());

}

}

}

Return {

getInstance: function() { // return getInstance from main function

if (!instance) {

instance = init();

}

Return instance;

}

}

})();

Const a = singleton.getInstance();

Const b = singleton.getInstance();

Console.log(a === b); // true

**Factory pattern**

The factory pattern is also a creational pattern as it creates new instances based on certain situations.

Function createCirle(radius) {

Return {

Radius,

Draw() {

Console.log(‘draw’);

}

}

}

Const circle1 = createCircle(5);

Const circle2 = createCircle(8);

**Strategy pattern**

The strategy pattern is a behavioural design pattern. Behavioural design patterns are all about class’s objects communication. These are more concerned about communication between objects.

Const Shipping = function() {

This.company; // undefined

}

Shipping.protoype = {

setStrategy: function(company) {

this.company = company;

}

Calculate: function(package) {

Return this.company.calculate(package);

}

}

// companies

Const UPS = function() {

This.calculate = function(package) {

// calculations

Return sum;

}

}

// create more companies like the one above if needed

Function run() {

Const package = 30;

// strategies

Const UPS = new UPS();

Const shipping = new Shipping();

Shipping.setStrategy(UPS); // pass object as argument

Console.log(shipping.calculate(package));

}