Factory Pattern

It involves creating objects using factory function. Usually factory function is used to combine common logic in central location and implement individual respective logic

// creates vehicle using object literal

const vehicleA = {

    manufacturer: "Toyota",

    PlateNO: 12345,

    startEngine () {console.log("reving engine"},

    drive () {console.log("driving car...")}

}

// creates new vehicle using object literal

const vehicleB = {

    manufacturer: "Ford",

    PlateNO: 13345,

    startEngine () {console.log("reving engine"},

    drive () {console.log("driving car...")}

}

// here we have created multiple objects for vehicle but 'startEngine' and 'drive

// methods are common and redundant. In bigger projects these methods could be

// very big and can cause way too much redundancy

So we can combine common function and make factory function dynamic that depending on params it creates different objects.

// creates factory function

function vehicleFactory (manufacturer, plateNO) {

    return  {

        manufacturer,

        plateNO,

        startEngine () {

            console.log("reving engine")

        },

        drive () {

            console.log("driving car...")

        }

    }

}

const Vehicle1 = vehicleFactory("Toyota", 12345);

console.log(Vehicle1)

// prints

//{

//  manufacturer: 'Toyota',

//  plateNO: 12345,

//  startEngine: [Function: startEngine],

//  drive: [Function: drive]

//}

const Vehicle2 = vehicleFactory("Ford", 13345);

console.log(Vehicle2)

// prints

// {

//  manufacturer: 'Ford',

//  plateNO: 13345,

//  startEngine: [Function: startEngine],

//  drive: [Function: drive]

// }

This is very basic example in sample code we have implemented different examples of returning user object with different attribute of subscription types. Like user A, B and C can have subscription to online services but with different packages like premium, basic, regular etc..which provides them few common benefits as well as different specific benefits with individual packages.