**1) Create an iffe that returns an object with three fields: variable value, function showValue() and function reverseValue(). Calling functions either logs the value or reverse it in an object. If value was not provided yet or is empty showValue function is to return information about that. Value type string or number if number to (\*(-1)) is string can be converted to umber do so.**

**2) Change the above iffe. So that returned object doesn’t publicly allow access to value but instead of that gives us a function setValue(). Purpose of this function is quite easy to grasp. *Closure pattern*.**

**3) Create a function definition (arrow ). This function is to take one input string/number and is to return object described in point above with value set to input.**

**4) Create four function definitions. One for every basic math operations and taking two input parameters. Create one more function. This function is to return calculation object. This object is to have parameters object field that holds two operation parameters inside (x and y). Function field that points to a function with math operation (defined at the beginning). A function setOperation() that sets the field from previous sentence and a Calculate function that makes calculation and returns its value.**

**5) Create an array (by hand) of objects and call sum() function in context of each one of them. Sum function is to come from this object  BaseObject = { X,y, sum: function (){ return this.X+this.y}}**

**6) Create an array (by hand) of objects and call sum() function in context of each one of them. Sum function is to come from this object BaseObject = { X, sum: function (y){ return this.X+y}}**

**7) Given an array of objects, for each object call operation() function in its context but from next object. If object is last get back to start of the array for operation function. In loop ☺ (for or while)**

[

{

x: 1,

y: 'object one value',

operation: () => 'object one prafix' + this.x + this.y

},

{

x: 2,

y: 'object two value',

operation: () => 'object two prafix' + this.x + this.y

},

{

x: 3,

y: 'object three value',

operation: () => 'object three prafix' + this.x + this.y

},

]

8) Scale riddle. With 8 balls ☺ EXAM