Alex Samos Muñoz

1. Write the code, one line for each action:

a) Create an empty object user.

```
let user = {};
```

b) Add the property name with the value John.

```
user.name = "Jhon";
```

c) Add the property surname with the value Smith.

```
user.surname = "Smith";
```

d) Change the value of the name to Pete.

```
user.name = "Pete";
```

e) Remove the property name from the object.

```
delete user.name;
```

f) Write the function isEmpty(obj) which returns true if the object has no properties, false otherwise.

```
let obj = {}
function isEmpty(obj) {
  let longitud = 0;

  for (let elemento in obj) {
    longitud ++;
  }
  return (longitud == 0)? true : false;
}

console.log(isEmpty(obj));
```

2. We have an object storing salaries of our team. Write the code to sum all salaries and store in the variable sum.

let salaries = {
John: 100,
Ann: 160,
Pete: 130
}

```
let salaries = {
  John: 100,
  Ann: 160,
  Pete: 130
  }

let sumaSalarios = 0;
let valores = Object.values(salaries);
for (let elemento of valores){
    sumaSalarios += parseInt(elemento);
}
```

```
salaries.suma = sumaSalarios;
console.log(salaries);
```

3. Create a function multiplyNumeric(obj) that multiplies all numeric property values of obj by 2

```
let salaries = {
  John: 100,
  Ann: 160,
  Pete: 130
}

function multiplyNumeric(obj) {
  let aux = Object.fromEntries(
    Object.entries(obj).map(elemento => [elemento[0], elemento[1]*2])
  )
 return aux;
}

let nuevosSalatios = multiplyNumeric(salaries);
console.log(nuevosSalatios);
```

4. Here the function makeUser returns an object. What is the result of accessing its ref? Why?

```
function makeUser() {
 return {
 name: "John",
 ref: this
 };
}
let user = makeUser();
alert( user.ref.name ); // What's the result?
```

El resultado es undefined porque el this hace referencia al bloque de codigo en este caso a la función no al return del objeto

5. Create an object calculator with three methods:

- **read() prompts for two values and saves them as object properties with names a and b respectively.**
- **sum() returns the sum of saved values.**
- **mul() multiplies saved values and returns the result.**

```
let calculator = {
  a: 0,
  b: 0,
  read: function(){
    this.a = parseInt(prompt('Inserte el primer numero: '));
    this.b = parseInt(prompt('Inserte el segundo numero: '));
  },
  sum(){
    return this.a + this.b;
  },
  mul(){
    return this.a * this.b;
  }
};

calculator.read();
console.log(calculator.sum());
console.log(calculator.mul());
```

6. Having the following object, write a function that gets the total amount of kg that the fruit shop has. Create two pieces of code solving the problem. In one of them, Object.values must appear and in the other one, for...of must be present. Prepare the function for the case that there is no fruit at all

let frutas={
"manzanas golden": 25,
"manzanas fuji": 20,
"pera conferencia": 17,
"pera ercolina": 12, }

```
function kilosTotales(obj) {
  let sumaTotal = 0;

  let valoresFrutas = Object.values(obj);
  if (valoresFrutas.length == 0) {
    return console.log("El objeto esta vacio");
  }else{
    for (let elemento of valoresFrutas) {
      sumaTotal += elemento;
    }
     return console.log(sumaTotal);
  }
}
```

```
let frutas={
  "manzanas golden": 25,
   "manzanas fuji": 20,
  "pera conferencia": 17,
  "pera ercolina": 12,
  }

kilosTotales(frutas);

function kilosTotales(obj) {
  let sumaTotal = 0;

    for (let [clave, valor] of Object.entries(obj)) {
      sumaTotal += valor;
    }
    if (sumaTotal == 0) {
      return console.log("El objeto esta vacio");
    }else{
      return console.log(sumaTotal);
    }
  }

kilosTotales(frutas);
```

7. Take the last code and write a function that returns an object containing the name of the fruit (including all varieties) and the total number of kgs

```
function kilosTotales(obj) {
let frutasKilos = {
  manzanas: 0,
  peras: 0
};


for (let key in obj) {
  if (key.startsWith("manzana")) {
    frutasKilos.manzanas += obj[key];
  }else if(key.startsWith("pera")){
    frutasKilos.peras += obj[key];
  }
}
```

```
let frutas={
"manzanas golden": 25,
"manzanas fuji": 20,
"pera conferencia": 17,
"pera ercolina": 12,
}

let obj = kilosTotales(frutas);

console.log(obj);
```

8. There's a ladder object that allows to go up and down:
let ladder = {
step: 0,
up() {
this.step++;
},
down() {
this.step--;
},
showStep: function() { // shows the current step
alert(this.step);
}
};
Now, if we need to make several calls in sequence, can do it like this:
ladder.up();
ladder.up();
ladder.down();
ladder.showStep(); // 1
ladder.down();
ladder.showStep(); // 0
Modify the code of up, down and showStep to make the calls chainable, like
ladder.up().up().down().showStep().down().showStep(); // shows 1 then 0

```
let ladder = {
  step: 0,
  up() {
  this.step++;
  return this;
  },
  down() {
  this.step--;
```

```
  return this; // el this hace referencia al bloque de contenido al que
pertenece, por ende al estar dentro del objeto, hace referencia al
objeto
  },
  showStep: function() { // shows the current step
  alert( this.step );
  return this;
  }
  };
  ladder.up().up().down().showStep().down().showStep();
```

9. Create an object, fruits, with the properties name and kg. Once created, assign four methods to the object: sell, buy, outOfStockDate and buyingDate. As there is no date property, the last methods must be programmed but user should see no error

```
let frutas={
  nombre: "manzana",
  kilos: 200,
  sell(){
    let compra = parseInt(prompt("Cuantos kilos quieres: ")) || 0;

    if (compra > this.kilos) {
      console.log("No hay suficientes kilos");
    }else{
      console.log("Venta efectuada");
      this.kilos= this.kilos - compra;
    }
  },
  buy(){
    let compra = parseInt(prompt("Cuantos kilos se han comprado: ")) ||
0;

    if (compra == 0) {
      console.log("No se ha comprado nada");
    }else{
      console.log("Compra efectuada");
      this.kilos = this.kilos + compra;
    }
  },
  outOfStockDate(){
    return true;
  },
  buyingDate(){
```

```
    return true;
  }
 }

 console.log(frutas.kilos);
 frutas.sell();
 console.log(frutas.kilos);
 frutas.buy();
 console.log(frutas.kilos);
 frutas.buyingDate();
 frutas.outOfStockDate();
```

10. Create an object that stores information about a spare car parts sold by a car shop. It should contain 4 or more rows and, for each one, name and number of parts. Create a function that sum a number to every spare part.

```
let partesCoche = {
  parte1: {
    nombre: "Motor",
    cantidad: 100
  },
  parte2: {
    nombre: "Filtro de aceite",
    cantidad: 150
  },
  parte3: {
    nombre: "Amortiguador",
    cantidad: 200
  },
  parte4: {
    nombre: "Filtro de aire",
    cantidad: 120
  },
  sumaPartes(){
    return this.parte1.cantidad + this.parte2.cantidad +
this.parte3.cantidad + this.parte4.cantidad;
  }
};

console.log(partesCoche.sumaPartes());
```

11. Create a function that creates an object storing the following information about an user: name, address, body dimensions. Use as less number of primary properties as possible. Create an user “usuario1” and copy this object to “usuario2”. Both of them must be different objects.

```
function crearUsuario(nombre, direccion, altura, anchura) {
  let aux = {
    user:{
      nombre: nombre,
      direccion: direccion,
      dimensiones:{
        altura: altura,
        anchura: anchura
      }
    }
  }

  return aux;
}

let usuario1 = crearUsuario("usuario1", "Calle Merlo", 170, 90)
let usuario2 = structuredClone(usuario1);
console.log(usuario1);
```

12. Add functions to get user’s information to the previous object. Add a function to get user’s friends. Despite this property does not exist, it must give no error. Call a function to get user’s mate, which does not exist. Again it must give no error.

```
function crearUsuario(nombre, direccion, altura, anchura) {
  let aux = {
    user:{
      nombre: nombre,
      direccion: direccion,
      dimensiones:{
        altura: altura,
        anchura: anchura
      },
    },
      infoUsuario(){
        return this.user;
      },
      devolverAmigos(){
        return this.user?.amigos;
```

```
      },
      devolverPareja(){
        return this.user?.pareja;
      }
  }

  return aux;
}

let usuario1 = crearUsuario("usuario1", "Calle Merlo", 170, 90)
console.log(usuario1.infoUsuario());
console.log(usuario1.devolverAmigos());
console.log(usuario1.devolverPareja());
```