JavaScript Assignment-3

1. Hello, object

Write the code, one line for each action:

1. Create an empty object `user`.
2. Add the property `name` with the value `John`.
3. Add the property `surname` with the value `Smith`.
4. Change the value of the `name` to `Pete`.
5. Remove the property `name` from the object.

2. Check for emptiness

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

Should work like that:

```
let schedule = {};

alert( isEmpty(schedule) ); // true

schedule["8:30"] = "get up";

alert( isEmpty(schedule) ); // false
```

3. Constant objects?

Is it possible to change an object declared with `const`? What do you think?

```
const user = {
  name: "John"
};

// does it work?
user.name = "Pete";
```

4. Sum object properties

We have an object storing salaries of our team:

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

Write the code to sum all salaries and store in the variable `sum`. Should be `390` in the example above.

If `salaries` is empty, then the result must be `0`.

JavaScript Assignment-3

5. Multiply numeric properties by 2

Create a function `multiplyNumeric(obj)` that multiplies all numeric properties of `obj` by `2`.

For instance:

```
// before the call
let menu = {
  width: 200,
  height: 300,
  title: "My menu"
};

multiplyNumeric(menu);

// after the call
menu = {
  width: 400,
  height: 600,
  title: "My menu"
};
```

6. Create an object `calculator` with three methods:

- `read()` prompts for two values and saves them as object properties.
- `sum()` returns the sum of saved values.
- `mul()` multiplies saved values and returns the result.

```
let calculator = {
  // ... your code ...
};


calculator.read();
alert( calculator.sum() );
alert( calculator.mul() );
```

7. 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
```

Modify the code of `up` and `down` to make the calls chainable, like this:

```
ladder.up().up().down().showStep(); // 1
```

Such approach is widely used across JavaScript libraries.