Лабораторная работа 5

Асинхронность – 2. Promise, async/await

(15 баллов)

Выполните самостоятельно следующие задания и оформите отчет.

Требования по отчету:

Наличие титульного листа. Размер страницы должен соответствовать формату А4 (210х297), размеры полей: левое – 30 мм, правое – 10 мм, верхнее – 15 мм, нижнее – 20 мм. Шрифт Times new Roman, размер 14 pt полуторный междустрочный интервал. Выравнивание текста – по ширине, красная строка – 1,25 см, отступ слева и справа – 0 мм.

1. **(1 балл)** Запустите следующий код:

```
let promise = new Promise(function(resolve, reject) {
    resolve(1);

    setTimeout(() => resolve(2), 1000);
  });

promise.then(console.log);
```

   Что выводится на экран? Почему?
2. **(2 балла)** Проведите промисификацию функций (надо поменять функцию, которая принимает колбэк, чтобы она вместо этого возвращала промис) в задание №2 из лабораторной №4. Выполните задание при помощи промисов.
3. **(2 балла)** Выполните задание №3 из лабораторной №4, используя промисы.
4. **(2 балла)** Необходимо написать функцию в виде промис объекта, которая складывает 2 числа и имеет следующие условности:
   a. суммирование происходит каждые 2 секунды, сначала в функцию попадает произвольных числа, а в следующих вызовах отложенной функции в первый аргумент попадает сумма чисел с предыдущей итерации суммирования, а второй аргумент остаётся прежним;
   b. сумму и количество вызов отложенной функции выводить в консоль на каждой итерации;
   c. как только отложенная функция будет выполнена 5 раз прекратить периодическое выполнение.
   d. если хотя бы один из аргументов функции не определён или имеют не number тип, то отклонить выполнение reject, иначе выполнить сложение этих чисел;

   Продемонстрируйте оба варианта вызова таких функций как с успешным выполнением сложения этих чисел, так и с вызовом ошибки.
5. **(3 балла)** Перепишите функции из заданий 2, 3, 4 через async/await. Продемонстрируйте работу программ.

6. **(2 балла)** Есть «обычная» функция f. Как можно внутри неё получить результат выполнения async–функции wait?

```
async function wait() {
    await new Promise(resolve => setTimeout(resolve, 1000));

    return 10;
  }

  function f() {
    // ...что здесь написать?
    // чтобы вызвать wait() и дождаться результата "10" от async-функции
    // не забывайте, здесь нельзя использовать "await"
  }
```

Подсказка: это тот случай, когда понимание внутреннего устройства работы async/await очень кстати. Здесь нужно думать о вызове функции async, как о промисе.

7. **(3 балла)** Используя async/await напишите программу, которая моделирует проведение собеседования. Собеседование включает 2 этапа: претенденту выдается задание, затем он идет его выполнять, представляет результат и защищает его; потом претенденту дается 5 единиц времени отдохнуть; затем такие же действия проводятся для второго задания, кроме отдыха, разумеется.

   Реализуйте асинхронную функцию interviews, которая принимает произвольное число претендентов – массивов вида:
   [имя, время на подготовку 1 задания, время на защиту 1 задания, время подготовки 2 задания, время на защиту второго задания]

   Функция должна для каждого задания каждого претендента вывести строки:
   при начале выполнения задания – <имя> started the <N> task.
   при переходе к защите – <имя> moved on to the defense of the <N> task.
   при окончании выполнения задания – <имя> completed the <N> task.
   при начале отдыха перед вторым заданием – <имя> is resting.

   Второе задание каждый претендент может получить только после выполнения первого.

   Пример входных данных: [['Ivan', 5, 2, 7, 2], ['John', 3, 4, 5, 1], ['Sophia', 4, 2, 5, 1]]
   Пример вывода:

```
Ivan started the 1 task.
John started the 1 task.
Sophia started the 1 task.
John moved on to the defense of the 1 task.
Sophia moved on to the defense of the 1 task.
Ivan moved on to the defense of the 1 task.
Sophia completed the 1 task.
Sophia is resting.
Ivan completed the 1 task.
Ivan is resting.
John completed the 1 task.
John is resting.
```

```
Sophia started the 2 task.
Ivan started the 2 task.
John started the 2 task.
Sophia moved on to the defense of the 2 task.
Ivan moved on to the defense of the 2 task.
Sophia completed the 2 task.
John moved on to the defense of the 2 task.
Ivan completed the 2 task.
John completed the 2 task.
```