Часть 1

Упражнение 1.

Создайте html-страницу, которая при загрузке запрашивает у пользователя его имя и фамилию, а затем выводит их в качестве приветствия (например, «Здравствуй, Петя Иванов!»). Оформите скрипт двумя способами: в html-документе и в отдельном файле js.

Упражнение 2

Напишите сценарий, который запрашивает у пользователя название месяца, и выводит на экран количество дней в нем (считать, что год – не високосный). Название нужно вводить по- русски. Подумайте о том, что разные пользователи могут по-разному печатать названия месяцев (Май, май, МАЙ или даже МаЙ). Подсказка: прочтите про методы работы со строками.

Упражнение 3

Присвойте переменной test значение «Это строка, которая формируется средствами JavaScript». Найдите:

- подстроки, ограниченные символами с индексами 12 и 18; 42 и 46
- индексы первого и последнего вхождения буквы «а» в данную строку; сочетания букв «ми»; сочетания букв «ст».

Результаты должны быть выведены на экран, каждый – в отдельной строчке, оформленной по следующему образцу:

> Первое вхождение букв «ст» в данную строку начинается с индекса ... (указать номер)

Упражнение 4

Предположим, что объект, представляющий сообщение, имеет свойства для даты и отправителя. Отсортируйте массив сообщений сначала по дате, а затем по отправителю. Убедитесь, что метод sort действительно устойчивый: сообщения с одинаковым отправителем остаются отсортированными по дате после второй сортировки.

Количество сообщений должен вводить пользователь, для каждого сообщения пользователь же вводит дату и отправителя.

Упражнение 5

Напишите функцию, которая возвращает все позиции заданного значения в массиве. Например, indexOf(arr, 0) возвращает все такие индексы i, что arr[i] равно нулю. Воспользуйтесь методами map и filter.

Упражнение 6

Вычислите размах (т. е. разность между максимальным и минимальным значениями) массива с помощью метода reduce.

Упражнение 7

Напишите функции map, filter, forEach для множеств.

Упражнение 8

Напишите функции union(set1, set2), intersection(set1, set2), difference(set1, set2), которые возвращают объединение, пересечение и разность множеств, не изменяя аргументов.

Часть 2

Упражнение 1

Дан список стран и городов каждой страны. Затем даны названия городов. Для каждого города укажите, в какой стране он находится.

Упражнение 2

Есть массив сообщений:

```
let messages = [
{text: "Hello", from: "John"},
{text: "How goes?", from: "John"}, {text: "See you soon", from: "Alice"}
];
```

У вас есть к ним доступ, но управление этим массивом происходит где-то ещё. Добавляются новые сообщения и удаляются старые, и вы не знаете в какой момент это может произойти.

Имея такую вводную информацию, решите, какую структуру данных вы могли бы использовать для ответа на вопрос «было ли сообщение прочитано?». Структура должна быть подходящей, чтобы можно было однозначно сказать, было ли прочитано это сообщение для каждого объекта сообщения.

P.S. Когда сообщение удаляется из массива `messages`, оно должно также исчезать из структуры данных.

P.P.S. Нам не следует модифицировать сами объекты сообщений, добавлять туда свойства. Если сообщения принадлежат какому-то другому коду, то это может привести к плохим последствиям.

Упражнение 3

Напишите функцию `sum`, которая работает таким образом: `sum(a)(b) = a+b`.

Например:

```
sum(1)(2) = 3 sum(5)(-1) = 4
```

Упражнение 4

Напишите функцию `printNumbers(from, to)`, которая выводит простые числа в диапазоне от `from` до `to` каждые две секунды.

Сделайте два варианта решения.

1. Используя `setInterval`.
2. Используя рекурсивный `setTimeout`.

Упражнение 5

Создайте декоратор delay(f, ms), который задерживает каждый вызов f на ms миллисекунд.

Например:

```
function f(x) {
  alert(x);

}

// создаём обёртки

let f1000 = delay(f, 1000); let f1500 = delay(f, 1500);

f1000("test");    //    показывает    "test"    после    1000    мс
f1500("test"); // показывает "test" после 1500 мс
```

Другими словами, `delay(f, ms)` возвращает вариант `f` с «задержкой на `ms` мс».

В приведённом выше коде `f` – функция с одним аргументом, но ваше решение должно передавать все аргументы и контекст `this`.

Упражнение 6

Напишите иерархию классов «Сказочный персонаж» -> «Дракон». Предусмотрите в каждом классе не менее двух полей и не менее двух собственных методов (кроме конструктора, геттеров и сеттеров). Опишите конструкторы, функции-сеттеры и функции-геттеры. Продемонстрируйте переопределение унаследованного метода. Продемонстрируйте работу ваших классов.

Упражнение 7

Создайте класс `FormatError`, который наследует от встроенного класса `SyntaxError`.

Класс должен поддерживать свойства `message`, `name` и `stack`.

Пример использования:

```
let err = new FormatError("ошибка форматирования");

alert(     err.message     );     //     ошибка     форматирования
alert( err.name ); // FormatError

alert( err.stack ); // stack

alert( err instanceof FormatError ); // true

alert( err instanceof SyntaxError ); // true (потому что наследует от SyntaxError)
```

Упражнение 8

Напишите функцию invokeAfterDelay, возвращающую обещание, которое вызывает заданную функцию с заданной задержкой. Продемонстрируйте ее работу, возвращая обещание, содержащее случайное число от 0 до 1. Полученный результат печатайте на консоли.