Язык JavaScript (Процедурное программирование)

ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ

При разработке больших и сложных программ логически независимые или повторяющиеся последовательности действий оформляют в виде вспомогательных подпрограмм.

Подпрограммы (процедуры) и функции позволяют улучшить читаемость кода. Разделение программы на функции позволяет также избежать избыточности кода, поскольку функцию записывают один раз, а вызывать ее на выполнение можно многократно из разных точек программы. Процесс отладки программы, содержащей функции, можно лучше структурировать. Часто используемые функции можно помещать в библиотеки. Таким образом создаются более простые в отладке и сопровождении программы.

МОДУЛИ

```
;(function(){
        window.myUtils = {
                cloneFirstLevel: function(obj){
                        var newObj = {};
                        for(var key in user1)
                        {
                                newObj[key] = obj[key];
                        }
                        return newObj;
                }
        };
}());
```

МОДУЛИ

```
var myUtils = (function(){
        return {
                cloneFirstLevel: function(obj){
                        var newObj = {};
                        for(var key in user1)
                        {
                                newObj[key] = obj[key];
                        }
                        return newObj;
                }
        };
}());
```

ЗАДАЧА

Разработать библиотеку включающую методы:

- метод remove(array, index) – возвращает массив array без удаленного элемента;
- метод repeat(str, count) – возвращает строку str повторенную count раз;
- метод pluck(array, property_name) – получает массив объектов array и возвращает массив значений определенного поля property_name.

ПОДКЛЮЧЕНИЕ СТОРОННИХ БИБЛИОТЕК

```
<script type="text/javascript" src="js/plotly-latest.min.js"></script>
<div id="placeholder" style="width: 480px; height: 400px;"></div>
<script type="text/javascript">
        var line1 = {
                x:[],
                y:[],
                type:'scatter'
        };
        for (var i = 0; i<20; i++)  {
                line1.x.push(i);
                line1.y.push(i);
        }
        var data = [line1];
        Plotly.newPlot('placeholder', data);
</script>
```

Построить график функции y = f(x)

Подключить стороннюю библиотеку для построения графиков.

Построить график функции y = f(x)

$$y = \begin{cases} 2x + 1, \text{ если } x < 0, \\ -1,5x + 1, \text{ если } 0 \le x < 2, \\ x - 4, \text{ если } x \ge 2 \end{cases}$$

шаг варьирования x равен 0.01 и интервал варьирования $-5 \le x \le 5$.

Расчёт функции y = f(x) реализовать в виде отдельной функции.

ТЕСТИРОВАНИЕ

Модульное тестирование, или **юнит-тестирование** (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

ТЕСТИРОВАНИЕ

SpecRunner.html (файл, который следует открыть в браузере для запуска тестов):

```
<html><head><meta charset="utf-8">
<title>Jasmine Spec Runner v2.6.2</title>
<link rel="shortcut icon" type="image/png" href="lib/jasmine-2.6.2/jasmine_favicon.png">
<link rel="stylesheet" href="lib/jasmine-2.6.2/jasmine.css">
<script src="lib/jasmine-2.6.2/jasmine.js"></script>
<script src="lib/jasmine-2.6.2/jasmine-html.js"></script>
<script src="lib/jasmine-2.6.2/boot.js"></script>
<!-- include source files here... -->
<script src="task01.js"></script>
</head><body></body></html>
```

task01.js (файл с тестами, часть 1)**:**

//Тестируемая функция

```
function calc_sum_numbers(str){
    var sum=0;
    for(var i=0; i<str.length; i++){
            x = parseInt(str[i]);
            if(x !== NaN)
                    sum += x;
    }
    return sum;
}
```

task01.js (файл с тестами, часть 2)**:**

```
//Определение набора тестов:
describe("task_01 calc_sum_numbers_from_string", function(){
    var input_str = '1111';
    var result = 4;
    var msg = "Вывести сумму цифр этого числа: " + input_str + "
результьтат 4";
    //Определение теста
    it(msg, function(){
        var rez = calc_sum_numbers(input_str);
        //определяет ожидания, которые проверяются в тесте
        expect(rez).toBe(4);
    });
});
```

ТЕСТИРОВАНИЕ

Примеры ожиданий для проверки результатов:

```
expect(2 + 2).toBe(4); //сравнение с использованием ===
expect(2 + 3).not.toBe(4); //сравнение с использованием !==
expect(a).toEqual(b); //сравнение переменных и объектов (включая содержимое)
expect(window.document).toBeDefined(); //значение должно быть определено
expect(a).toBeNull(); //значение должно быть null
expect(5 > 0).toBeTruthy(); //значение должно быть верно
expect(2 + 2).toBeLessThan(5); //значение должно быть меньше чем
expect("some string").toMatch(/string/); //значение должно соответствовать регулярному выражению
expect([1, 2, 3]).toContain(2); //значение должно содержать
…
```

Тестирование математической функции (оформить в отдельную в функцию):

$$e = \frac{1 + \sin^2(x + y)}{2 + \left| x - 2x / (1 - x^2 y^2) \right|} + x$$

- написать один позитивный тест;
- написать один негативный тест.