

МИНОБРНАУКИ РОССИИ

Федеральное государственное образовательное учреждение высшего профессионального образования

**«Московский технологический университет»**

Институт Информационных Технологий

Кафедра «ИНэУМ»

**ОТЧЕТ**

по лабораторной работе № 3

по дисциплине

«Проектирование сред программирования и их компонентов»

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент группы ИКМО-04-2017 | |  | Борзунов Н.С. |
| Принял | к.т.н., доцент | | Данилкин Ф.А. |
| Выполнено | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 | г. |  |
| Зачтено | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2018 | г. |  |

МОСКВА 2018

**Оглавление**

[Задания лабораторной работы 3](#_Toc515712047)

[Задание №1 3](#_Toc515712048)

[Задание №2 5](#_Toc515712049)

[Задание №3 7](#_Toc515712050)

[Задание №4 8](#_Toc515712051)

[Задание №5 10](#_Toc515712052)

[Задание №6 12](#_Toc515712053)

[Заключение 14](#_Toc515712054)

[Список использованной литературы 15](#_Toc515712055)

**Задания лабораторной работы**

**Задание №1**

Создайте объект calculator с тремя методами:

1. read() запрашивает prompt два значения и сохраняет их как свойства объекта
2. sum() возвращает сумму этих двух значений
3. mul() возвращает произведение этих двух значений

var calculator = {

...ваш код...

}

calculator.read();

alert( calculator.sum() );

alert( calculator.mul() );

**Код программы**

Листинг 1.1 - Файл 1.html

<html>

<body>

<script>

var calculator =

{

read: function()

{

this.a = +prompt('Введите первое число', '');

this.b = +prompt('Введите второе число', '');

},

sum: function()

{

return this.a + this.b;

},

mul: function()

{

return this.a \* this.b;

}

}

calculator.read();

alert('Сумма введенных чисел: ' + calculator.sum() );

alert('Произведение введенных чисел: ' + calculator.mul() );

</script>

</body>

</html>

**Демонстрация выполнения программы**

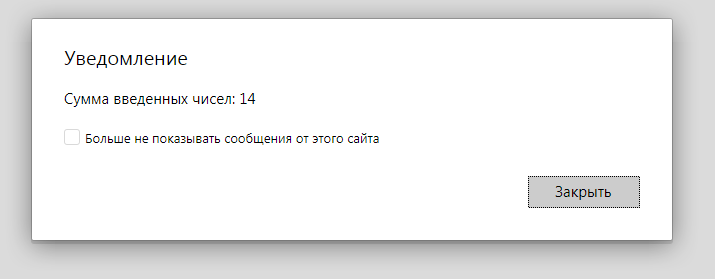


Рисунок 1.1 - Результат выполнения программы

**Задание №2**

Есть объект «лестница» ladder:

var ladder = {

step: 0,

up: function() { // вверх по лестнице

this.step++;

},

down: function() { // вниз по лестнице

this.step--;

},

showStep: function() { // вывести текущую ступеньку

alert( this.step );

}

};

Сейчас, если нужно последовательно вызвать несколько методов объекта, это можно сделать так:

ladder.up();

ladder.up();

ladder.down();

ladder.showStep(); // 1

Модифицируйте код методов объекта, чтобы вызовы можно было делать цепочкой, вот так:

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

Как видно, такая запись содержит «меньше букв» и может быть более наглядной.

**Код программы**

Листинг 2.1 - Файл 2.html

<html>

<body>

<script>

var ladder =

{

step: 0,

up: function()

{

this.step++;

return this;

},

down: function()

{

this.step--;

return this;

},

showStep: function()

{

alert( this.step );

return this;

}

}

ladder.up().up().down().up().down().showStep();

</script>

</body>

</html>

**Демонстрация выполнения программы**

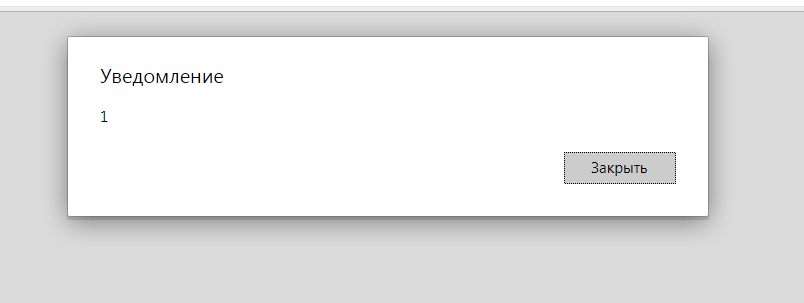


Рисунок 2.1 - Результат выполнения программы

**Задание №3**

Напишите функцию sum, которая будет работать так:

sum(1)(2) == 3; // 1 + 2

sum(1)(2)(3) == 6; // 1 + 2 + 3

sum(5)(-1)(2) == 6

sum(6)(-1)(-2)(-3) == 0

sum(0)(1)(2)(3)(4)(5) == 15

Количество скобок может быть любым.

**Код программы**

Листинг 3.1 - Файл 3.html

<html>

<body>

<script>

function sum(a)

{

var nNum = a;

function f(b)

{

nNum += b;

return f;

}

f.toString = function()

{

return nNum;

};

return f;

}

alert( sum(1)(2) );

alert( sum(5)(-1)(2) );

alert( sum(6)(-1)(-2)(-3) );

alert( sum(0)(1)(2)(3)(4)(5) );

</script>

</body>

</html>

**Демонстрация выполнения программы**

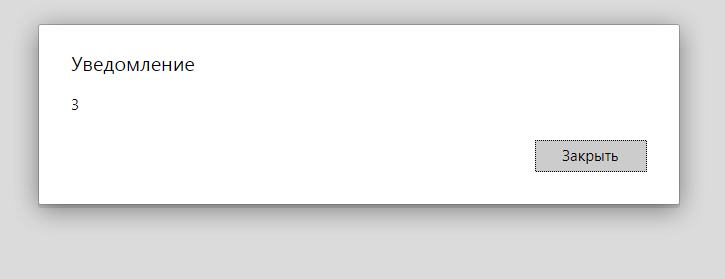


Рисунок 3.1 - Результат выполнения программы

**Задание №4**

Напишите функцию-конструктор Calculator, которая создает объект с тремя методами:

Метод read() запрашивает два значения при помощи prompt и запоминает их в свойствах объекта.

Метод sum() возвращает сумму запомненных свойств.

Метод mul() возвращает произведение запомненных свойств.

Пример использования:

var calculator = new Calculator();

calculator.read();

alert( "Сумма=" + calculator.sum() );

alert( "Произведение=" + calculator.mul() );

**Код программы**

Листинг 4.1 - Файл 4.html

<html>

<body>

<script>

function Calculator()

{

this.read = function()

{

this.Val1 = +prompt('Введите первое число', '');

this.Val2 = +prompt('Введите второе число', '');

};

this.sum = function()

{

return this.Val1 + this.Val2;

};

this.mul = function()

{

return this.Val1 \* this.Val2;

};

}

var calculator = new Calculator();

calculator.read();

alert("Сумма введенных чисел: " + calculator.sum());

alert("Произведение введенных чисел: " + calculator.mul());

</script>

</body>

</html>

**Демонстрация выполнения программы**

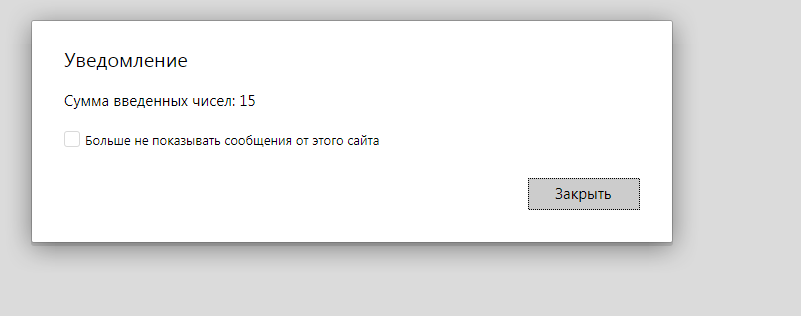


Рисунок 4.1 - Результат выполнения программы

**Задание №5**

Напишите конструктор Calculator, который создаёт расширяемые объекты-калькуляторы.

Эта задача состоит из двух частей, которые можно решать одна за другой.

Первый шаг задачи: вызов calculate(str) принимает строку, например «1 + 2», с жёстко заданным форматом «ЧИСЛО операция ЧИСЛО» (по одному пробелу вокруг операции), и возвращает результат. Понимает плюс + и минус -.

Пример использования:

var calc = new Calculator;

alert( calc.calculate("3 + 7") ); // 10

Второй шаг – добавить калькулятору метод addMethod(name, func), который учит калькулятор новой операции. Он получает имя операции name и функцию от двух аргументов func(a,b), которая должна её реализовывать.

Например, добавим операции умножить \*, поделить / и возвести в степень \*\*:

var powerCalc = new Calculator;

powerCalc.addMethod("\*", function(a, b) {

return a \* b;

});

powerCalc.addMethod("/", function(a, b) {

return a / b;

});

powerCalc.addMethod("\*\*", function(a, b) {

return Math.pow(a, b);

});

var result = powerCalc.calculate("2 \*\* 3");

alert( result ); // 8

Поддержка скобок и сложных математических выражений в этой задаче не требуется.

Числа и операции могут состоять из нескольких символов. Между ними ровно один пробел.

**Код программы**

Листинг 5.1 - Файл 5.html

<html>

<body>

<script>

function Calculator()

{

var method =

{

"-": function(a, b)

{

return a - b;

},

"+": function(a, b)

{

return a + b;

}

};

this.calculate = function(str)

{

var split = str.split(' '), a = +split[0], b = split[1], c = +split[2];

return method[b](a, c);

}

this.addMethod = function(name, func)

{

method[name] = func;

};

}

var calc = new Calculator;

calc.addMethod("\*", function(a, b)

{

return a \* b;

});

calc.addMethod("/", function(a, b) {

return a / b;

});

calc.addMethod("\*\*", function(a, b) {

return Math.pow(a, b);

});

var a = calc.calculate("6 \* 5");

alert(a);

</script>

</body>

</html>

**Демонстрация выполнения программы**



Рисунок 5.1 - Результат выполнения программы

**Задание №6**

Целое число, большее 1, называется простым, если оно не делится нацело ни на какое другое, кроме себя и 1.

Древний алгоритм «Решето Эратосфена» для поиска всех простых чисел до n выглядит так:

1. Создать список последовательных чисел от 2 до n: 2, 3, 4, ..., n.

2. Пусть p=2, это первое простое число.

3. Зачеркнуть все последующие числа в списке с разницей в p, т.е. 2\*p, 3\*p, 4\*p и т.д. В случае p=2это будут 4,6,8....

4. Поменять значение p на первое не зачеркнутое число после p.

5. Повторить шаги 3-4 пока p2 < n.

6. Все оставшиеся не зачеркнутыми числа – простые.

Реализуйте «Решето Эратосфена» в JavaScript, используя массив.

Найдите все простые числа до 100 и выведите их сумму.

**Код программы**

Листинг 5.1 - Файл 5.html

<html>

<body>

<script>

n = prompt('Введите число', '');

nN= parseInt(n);

var nArr = [];

for (var i = 2; i < nN;i++)

{

nArr[i] = true

}

var SVal = 2;

do

{

for (i=2\*SVal;i<nN;i+=SVal)

{

nArr[i] = false;

}

for (i = SVal+1; i < nN;i++)

{

if (nArr[i]) break;

}

SVal= i;

} while (SVal \* SVal < nN)

var nSum = 0;

for (i = 0; i < nArr.length;i++)

{

if (nArr[i])

{

nSum += i;

}

}

alert(nSum);

</script>

</body>

</html>

**Демонстрация выполнения программы**

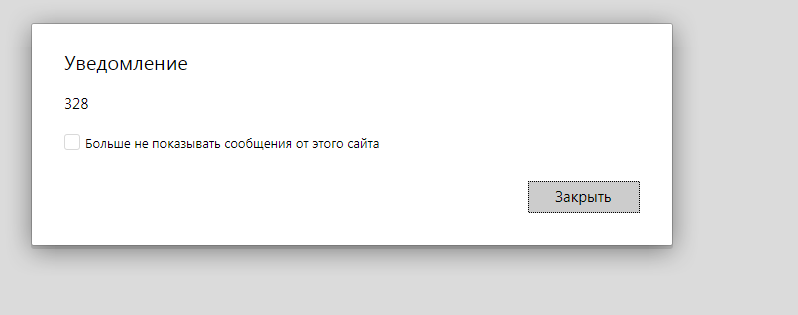


Рисунок 6.1 - Результат выполнения программы

**Заключение**

При выполнении данной лабораторной работы были реализованы задачи создания калькулятор, вызова цепочку вызовов, реализации суммы произвольного количества скобок, создания калькулятор при помощи конструктора.

**Список использованной литературы**

1. Дж. Роббинс HTML5, CSS3 и JavaScript. Исчерпывающее руководство (+ DVD), 2017

2. Э. Браун Изучаем JavaScript. Руководство по созданию современных веб-сайтов, 2017

3. М. Дэвид JavaScript и jQuery. Исчерпывающее руководство. 3-е издание, 2017