Федеральное государственное автономное

Образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №8**

Работа с объектами JavaScript

тема

Преподаватель \_\_\_\_\_\_\_\_\_ А.М. Скрипачев

подпись, дата инициалы, фамилия

Студент КИ16-16Б, №031620303 \_\_\_\_\_\_\_\_\_ В.В. Кокташев

Номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

# Цель работы

# Изучение основ работы с объектами JavaScript

# Задание

1. Продемонстрировать несколько способов создания объектов, несколько способов доступа к их свойствам (на чтение и запись);
2. создать конструктор для собственного объекта, добавить в него несколько свойств и методов, продемонстрировать работу с этими объектами;
3. расширить встроенный тип (например, Array или Date) функцией, оперирующей данными расширяемого объекта (например, вывод среднего арифметического для массива, или подсчет количества секунд со дня вашего рождения для даты).

# Краткий теоретический материал

## Объекты

Объекты – это составной тип данных, они объединяют множество значений в единый модуль и позволяют сохранять и извлекать значения по их именам. Говоря другими словами, объекты – это неупорядоченные коллекции свойств, каждое из которых имеет свои имя и значение. Именованные значения, хранящиеся в объекте, могут быть данными элемента.

## Создание объекта

Самый простой способ создать объект заключается во включении в программу литерала объекта. Литерал объекта – это заключенный в фигурные скобки список свойств (пар имя/значение), разделенных запятыми. Именем свойства может быть идентификатор или строковый литерал (допускается использовать пустую строку). Значением свойства может быть любое выражение, допустимое в JavaScript, – значение выражения (это может быть простое значение или объект) станет значением свойства.

Литерал объекта – это выражение, которое создает и инициализирует новый объект всякий раз, когда производится вычисление этого выражения. Значение каждого свойства вычисляется заново, когда вычисляется значение литерала. Это означает, что с помощью единственного литерала объекта можно создать множество новых объектов, если этот литерал поместить в тело цикла или функции, которая будет вызываться многократно, и что значения свойств этих объектов могут отличаться друг от друга. С помощью оператора new можно создать другую разновидность объектов. За этим оператором должно быть указано имя функции- конструктора, выполняющей инициализацию свойств объекта.

## Свойства объектов

Обычно для доступа к значениям свойств объекта используется оператор . (точка). Значение в левой части оператора должно быть ссылкой на объект, к свойствам которого требуется получить доступ. Обычно это просто имя переменной, содержащей ссылку на объект, но это может быть любое допустимое в JavaScript выражение, являющееся объектом. Значение в правой части оператора должно быть именем свойства. Это должен быть идентификатор, а не строка или выражение. Так, обратиться к свойству p объекта o можно посредством выражения o.p, а к свойству radius объекта circle – посредством выражения circle.radius. Свойства объекта работают как переменные: в них можно сохранять значения и считывать их.

## Объекты как ассоциативные массивы

Как мы знаем, доступ к свойствам объекта осуществляется посредством оператора «точка». Доступ к свойствам объекта возможен также при помощи оператора [], который обычно применяется при работе с массивами.

Важное различие между этими двумя синтаксисами, на которое следует обратить внимание, состоит в том, что в первом варианте имя свойства представляет собой идентификатор, а во втором – строку.

## Конструкторы

Чтобы определить собственный конструктор, достаточно написать функцию, добавляющую новые свойства к объекту, на который ссылается ключевое слово this.

В следующем фрагменте приводится определение конструктора, с помощью которого затем создаются два новых объекта:

// Определяем конструктор.

// Обратите внимание, как инициализируется объект с помощью "this".

function Rectangle(w, h)

{ this.width = w; this.height = h; }

// Вызываем конструктор для создания двух объектов Rectangle. Мы передаем ширину и высоту конструктору, чтобы можно было правильно проинициализировать оба новых объекта.

var rect1 = new Rectangle(2, 4); // rect1 = { width:2, height:4 };

var rect2 = new Rectangle(8.5, 11); // rect2 = { width:8.5, height:11 };

# Ход работы

1. Создано несколько объектов, причем часть из них при помощи включения в программу литерала объекта, а другая часть – при помощи оператора new (рисунок 1).

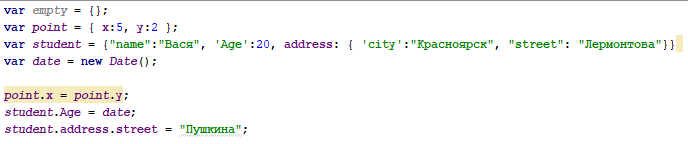


Рисунок 1 – Создание объектов различными способами

1. Создан конструктор для собственного объекта, включающий в себя 3 свойства, 1 функцию и принимающий 3 входных параметра (рисунок 2).

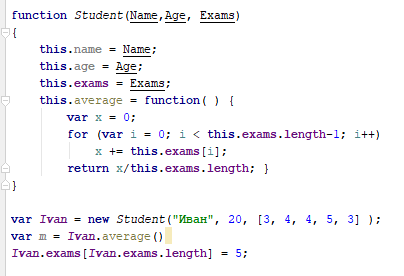


Рисунок 2 – Создание объекта при помощи конструктора

1. Встроенный тип Array расширен функцией sum, вычисляющей сумму всех элементов (рисунок 3).

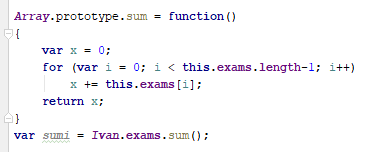


Рисунок 3 – Расширение встроенного типа Array

# Вывод

В ходе лабораторной работы были изучены основы работы с объектами JavaScript и расширения встроенных типов.

# ПРИЛОЖЕНИЕ А

Листинг 1 – файл script.js

var empty = {};

var point = { x:5, y:2 };

var student = {"name":"Вася", 'Age':20, address: { 'city':"Красноярск", "street": "Лермонтова"}}

var date = new Date();

point.x = point.y;

student.Age = date;

student.address.street = "Пушкина";

function Student(Name,Age, Exams)

{

this.name = Name;

this.age = Age;

this.exams = Exams;

this.average = function( ) {

var x = 0;

for (var i = 0; i < this.exams.length-1; i++)

x += this.exams[i];

return x/this.exams.length; }

}

var Ivan = new Student("Иван", 20, [3, 4, 4, 5, 3] );

var m = Ivan.average()

Ivan.exams[Ivan.exams.length] = 5;

Array.prototype.sum = function()

{

var x = 0;

for (var i = 0; i < this.exams.length-1; i++)

x += this.exams[i];

return x;

}

var sumi = Ivan.exams.sum();