Санкт-Петербургский государственный политехнический университет

Институт Прикладной математики и механики
Кафедра Прикладной математики

Д. В. Богданов

ИСПОЛЬЗОВАНИЕ JAVASCRIPT ДЛЯ МОДЕЛИРОВАНИЯ ФИГУР ЛИССАЖУ МЕТОДОМ ПРУЖИН

Курсовой проект

Направление подготовки бакалавров: 010800 Механика и математическое моделирование

Профиль ООП: 010800.62.01 Механика деформируемого твердого тела

Группа 23604/1

Руководитель проекта: Панченко А.Ю.

Допущен к защите:

«\_\_» 20\_\_ г.

Санкт-Петербург

2014

# **Оглавление**

[***Оглавление*** 2](#_Toc388226015)

[***Введение*** 3](#_Toc388226016)

[***Глава 1. Значение, виды и выбор фреймворка*** 4](#_Toc388226017)

[1.1 Что такое библиотеки JavaScript 4](#_Toc388226018)

[1.2 Виды JS фреймворков 5](#_Toc388226019)

[1.3 Выбор JS фреймворка 7](#_Toc388226020)

[***Глава 2. Создание приложения*** 8](#_Toc388226021)

[2.1 Начало работы с проектом и JavaScript 8](#_Toc388226022)

[2.2 Написание кода 9](#_Toc388226023)

[2.3 «Простое» и «сложное» движение 19](#_Toc388226024)

[2.4 Итог работы 20](#_Toc388226025)

[***Заключение*** 22](#_Toc388226026)

[***Литература*** 23](#_Toc388226027)

# **Введение**

Сегодня, как никогда прежде, JS-фреймворки играют большую роль в сайтостроении. Абсолютно все современные сайты построены посредством взаимодействия HTML и JS-фреймворков, а так же Flash, PHP, Java и так далее, но это уже не относится к теме данной курсовой работы. Анимация, красивые меню, фотогалереи, плавные переходы от страницы к странице, диалоговые окна, меняющиеся в зависимости от содержимого и многое другое, это все достигается при помощи JavaScript библиотек, которые активно используются при создании различных веб страниц. Поэтому тема данной курсовой работы является очень актуальной.

Целью данной курсовой работы является создание приложения с использованием одного из существующий JS-фреймворков. Поэтому в процессе выполнения курсовой работы необходимо решить следующие задачи:

* Дать общее определение JavaScript библиотеки и рассмотреть виды этих библиотек.
* Выбрать одну из библиотек для создания приложения.
* Создать приложение с использованием одного из фреймворков.

В первой главе приведено определение JavaScript библиотек, немного истории создания. Рассматриваются различные библиотеки, после чего выбирается один из этих фреймворков для создания клиентского приложения.

Во второй главе описывается то, как создается это приложение с примерами кодов страниц.

# **Глава 1. Значение, виды и выбор фреймворка**

## 1.1 Что такое библиотеки JavaScript

Библиотека JavaScript — сборник классов и/или функций на языке JavaScript.

Язык JavaScript, изначально разработанный Netscape (а затем развиваемый Mozilla), долгое время использовался в сети на многих сайтах, но широкую популярность получил с приходом Веб 2.0 — периода развития компьютерных систем, в котором JavaScript совместно с различными диалектами XML стал активно использоваться в разработке пользовательских интерфейсов как веб-приложений, так и настольных приложений. JavaScript в связке с CSS используется для создания динамических сайтов, более доступных, чем основанные на Flash альтернативы.

С увеличением популярности JavaScript, простота создания динамических элементов пользовательского интерфейса стала играть ключевую роль в веб-разработке. Этим обусловлен лавинообразный характер появления различных библиотек JavaScript, таких как Ext и Dojo. С другой стороны, одним из последствий войны браузеров стала разница в реализации объектной модели документа и это обусловило необходимость затрачивать дополнительные усилия для реализации корректной работы различных браузеров. Данное обстоятельство обусловило появление библиотек JavaScript, предоставляющих кроссбраузерный интерфейс к методам DOM, таких как Prototype, script.aculo.us или jQuery.

Удобство использования библиотек JavaScript привело к тому, что Microsoft, Yahoo! и другие крупные ИТ-компании разрабатывают свои собственные основанные на JavaScript библиотеки элементов пользовательского интерфейса, встраиваемые в веб-приложения, разрабатываемые этими компаниями.

Практически все библиотеки JavaScript выпускаются под лицензиями копицентр и копилефт, чтобы обеспечить свободное от лицензионных отчислений разработку, использование и модификацию.

Более того, некоторые библиотеки JavaScript позволяют упростить взаимодействие JavaScript с другими языками, такими как CSS, PHP, Ruby и Java. Это позволяет упростить запуск приложений JavaScript с приложениями, написанными на других языках программирования.

## 1.2 Виды JS фреймворков

Существует большое количество JavaScript библиотек, каждая из которых предназначена для упрощения программирования на JavaScript. Все библиотеки имеют свои особенности.

Рассмотрим несколько популярных библиотек:

**Dojo** — свободная модульная библиотека JavaScript. Разработана с целью упростить ускоренную разработку основанных на JavaScript или AJAX приложений и сайтов. Разработка библиотеки была начата Алексом Русселом в 2004 году. **Dojo Foundation** — некоммерческая организация, созданная для продвижения Dojo. Dojo используется в Zend Framework, начиная с версии 1.6.0. Библиотека достаточно глючная. Многие примеры с официального сайта не работают.

Элементы интерфейса dojo — это пакеты, сформированные из компонентов: JavaScript-кода, разметки HTML и CSS. Они могут быть использованы для добавления различных интерактивных возможностей к сайту:

* меню, закладок, всплывающих подсказок;
* селекторов даты, времени; часов;
* сортируемых таблиц, динамических диаграмм, векторной 2D графики;
* элементов интерфейса «дерево»;
* различных HTML-форм с возможностью проверки ввода пользователя;
* анимированных эффектов, и возможностей построения своих собственных эффектов.

**Ext JS** — библиотека JavaScript для разработки веб-приложений и пользовательских интерфейсов, изначально задуманная как расширенная версия Yahoo! UI Library, однако преобразовавшаяся затем в отдельный фреймворк. До версии 4.0 использовала адаптеры для доступа к библиотекам Yahoo! UI Library, jQuery или Prototype/script.aculo.us, начиная с 4-ой версии адаптеры отсутствуют. Поддерживает технологию AJAX, анимацию, работу с DOM, реализацию таблиц, вкладок, обработку событий и все остальные новшества Web 2.0.

**jQuery** — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Основные возможности **jQuery**:

* Движок кроссбраузерных CSS-селекторов ;
* Переход по дереву DOM;
* События;
* Визуальные эффекты;
* AJAX-дополнения;
* JavaScript-плагины.

**MooTools** — это свободный JavaScript-фреймворк для разработки кроссбраузерных веб-приложений и веб-сервисов. MooTools является модульным, объектно-ориентированным фреймворком, созданным для помощи разработчикам JavaScript. MooTools совместим и протестирован с браузерами: Safari 2+, Internet Explorer 6+, Firefox 2+ (и другими, основанными на движке Gecko), Opera 9+. Фреймворк MooTools используется в CMS Contao, Joomla 1.5+, ZoneMinder, MODx. MooTools содержит мощную коллекцию классов и продвинутую систему наследования, которая позволяет вторичное использование кода, а также его расширение.

Также предоставляет собственный набор классов, с помощью которых возможна, реализация различных эффектов. К ним относятся изменения размеров окна браузера, показ и гашение объектов, эффекты движения и Ajax.

**Prototype** — JavaScript фреймворк, упрощающий работу с Ajax и некоторыми другими функциями. Несмотря на его доступность в виде отдельной библиотеки, он обычно используется программистами вместе с Ruby on Rails, Tapestry, script.aculo.us и Rico.

Заявлено, что данный фреймворк совместим со следующими браузерами: Internet Explorer 6.0+, Mozilla Firefox 1.5+, Apple Safari 2.0.4+ и Opera 9.25+,Google Chrome 1.0+. В Prototype присутствуют самые разные способы упрощения создания JavaScript приложений, например, сокращённый вызов некоторых функций языка.

**Rico** — библиотека JavaScript для разработки насыщенных интернет приложений. Предоставляется открытым кодом. Rico предоставляет полную поддержку Ajax, управление посредством «drag&drop» и библиотеку кинематографических эффектов. Rico поддерживает JSON и Prototype.

**Особенности Rico:**

* **LiveGrid** — элемент управления, позволяющий автоматически обновлять HTML-таблицу через Ajax.
* **Animation Effects** — В Rico 2.0 присутствует инструмент для анимирования, утверждается, что он более гибок, чем другие веб-приложения для анимирования. Анимация в Rico может быть приостановлена или остановлена, к ней могут быть применены другие эффекты, что позволяет создавать анимацию, чутко реагирующую на действия пользователя.
* **Styling** — Rico предлагает несколько кинематографических и простых эффектов на своём очень простом интерфейсе.
* **Ajax Support** — В Rico имеется интерфейс для регистрирования Ajax-овых обработчиков запросов, а также регистрирования объектов HTML и JavaScript как Ajax-овых обработчиков.

## 1.3 Выбор JS фреймворка

Перед мной стоял выбор из огромного количества JavaScript библиотек. Не долго думая, я покопался в интернете, почитал информацию по каждому из фреймворков. Некоторые из них, по мнению пользователей, были «бажными» (**Dojo**), другие были сложны в освоении, либо просто не хватало необходимой информации для того, чтобы комфортно пользоваться ими.

По истечению некоторого времени поиска информации, я все же решил использовать JQuery, так как он является наиболее популярным и самым простым в освоении (по отзывам пользователей) среди себе подобных фреймворков, и информации с примерами использования было вполне достаточно.

# **Глава 2. Создание приложения**

## 2.1 Начало работы с проектом и JavaScript

Итак, выбрав необходимый нам фреймворк, встала задача: как написать код (тема была уже определена, т.к. требовалась моделирование грузика на пружинка).

По моим представлениям, данное приложение должно иметь такой функционал:

1. Корректное отображение исследуемого объекта, с плавной анимацией ;
2. Приятный, не режущий глаз, внешний вид (несколько слайдеров и text area);
3. График движения грузика
4. Объединить скрипт простого и сложного движения на одной странице (экономия ресурсов)
5. Очистка графика по нажатию кнопки

## 2.2 Написание кода

Для начала создадим стартовую страницу index.htm на которую и будем выводить наш фрейм для моделирования, график и слайдеры настройки, кнопку очистки и checkbox для выбора «простого» или «сложного» типа движений (о типе движения в пп 2.3).

<!DOCTYPE html>

<html>

<head>

 <meta charset="utf-8">

 <title>Пружинка</title>

 <script src="ocanvas.min.js"></script>

 <script src="script.js?<?= time() ?>"></script>

 <script src="jquery-1.6.1.min.js"></script>

 <script src="TM.js"></script>

 <script language="javascript" type="text/javascript" src="flot/jquery.flot.js"></script>

 <script language="javascript" type="text/javascript" src="jquery.flot.axislabels.js"></script>

</head>

<body>

 <div style="float:left; width: 607px">

 <canvas id="canvasMech" width="600" height="600" style="border:1px solid #000000; float: left;"></canvas>

 </div>

 <!--график-->

 <div id="vGraph" style="width:600px; height:600px; border: 1px solid black; margin-left: 700px "></div>

 <!--слайдеры и textarea, описывается их позиция, размеры и проверка на правильность ввода в textarea-->

 <div style="margin-left:20px; margin-top:20px; margin-bottom:20px;">

 <div> <input type="range" id="slider\_m" style="width: 150px;" oninput="app.setSlider\_m(this.value); document.getElementById('text\_m').value = this.value;">

 m =

 <input id="text\_m" style="width: 5ex;" required pattern="[-+]?([0-9]\*\.[0-9]+|[0-9]+)" oninput="

 // если введено не число - строка не пройдет валидацию по паттерну выше, и checkValidity() вернет false

 if (!this.checkValidity()) return;

 app.setSlider\_m(this.value);

 document.getElementById('slider\_m').value = this.value;">

 </div>

 <br>

 <div> <input type="range" id="slider\_Cx" style="width: 150px;" oninput="app.setSlider\_Cx(this.value); document.getElementById('text\_Cx').value = this.value;">

 Cx =

 <input id="text\_Cx" style="width: 5ex;" required pattern="[-+]?([0-9]\*\.[0-9]+|[0-9]+)" oninput="

 // если введено не число - строка не пройдет валидацию по паттерну выше, и checkValidity() вернет false

 if (!this.checkValidity()) return;

 app.setSlider\_Cx(this.value);

 document.getElementById('slider\_Cx').value = this.value;">

 <input type="button" style="width: 100px; margin-left:100px" name="" onclick="app.clearTrajectory();" value="Очистить">

 </div>

 <br>

 <div> <input type="range" id="slider\_Cy" style="width: 150px;" oninput="app.setSlider\_Cy(this.value); document.getElementById('text\_Cy').value = this.value;">

 Cy =

 <input id="text\_Cy" style="width: 5ex;" required pattern="[-+]?([0-9]\*\.[0-9]+|[0-9]+)" oninput="

 // если введено не число - строка не пройдет валидацию по паттерну выше, и checkValidity() вернет false

 if (!this.checkValidity()) return;

 app.setSlider\_Cy(this.value);

 document.getElementById('slider\_Cy').value = this.value;">

 <input type ="checkbox" id ="check" style="margin-left:100px;" value="0">Сложное движение

 </div>

 <br>

 <div> <input type="range" id="slider\_B" style="width: 150px;" oninput="app.setSlider\_B(this.value); document.getElementById('text\_B').value = this.value;">

 B =

 <input id="text\_B" style="width: 5ex;" required pattern="[-+]?([0-9]\*\.[0-9]+|[0-9]+)" oninput="

 // если введено не число - строка не пройдет валидацию по паттерну выше, и checkValidity() вернет false

 if (!this.checkValidity()) return;

 app.setSlider\_B(this.value);

 document.getElementById('slider\_B').value = this.value;">

 </div>

 </div>

 <script type="text/javascript">

 <!--Передаем в скрипт id слайдеров, textarea, canvas, checkbox-->

 var app = new MainMech(

 document.getElementById('canvasMech'),

 document.getElementById('slider\_m'),

 document.getElementById('text\_m'),

 document.getElementById('slider\_Cx'),

 document.getElementById('text\_Cx'),

 document.getElementById('slider\_Cy'),

 document.getElementById('text\_Cy'),

 document.getElementById('slider\_B'),

 document.getElementById('text\_B'),

 document.getElementById('check'));

 </script>

</body>

</html>

Для того чтобы создать меню, нам нужна библиотека jQuery. Для этого зайдем на официальный сайт http://jquery.com/ и скачиваем оттуда самую последнюю версию. Скачиваем с http://ocanvas.org/ файл ocanvas.min.js и помещаем в ту же папку. Используем oCanvas для вывода анимации пружины.

Далее создадим скрипт для заполнения массива координат графика.

function TM\_graph(htmlElement, xAxisLabel, yAxisLabel, yArrayLen){

 this.htmlElement = htmlElement;

 this.xAxisLabel = xAxisLabel;

 this.yAxisLabel = yAxisLabel;

 this.yArrayLen = yArrayLen;

 this.vArray = [];

}

TM\_graph.prototype.graphIter = function(x, y){

 this.vArray.push([x, y]); // добавляем значение в конец массива

 if (this.vArray.length > this.yArrayLen) this.vArray.shift(); // если в массиве больше yArrayLen значений - удаляем первое

 var xAxisLabel1 = this.xAxisLabel;

 var yAxisLabel1 = this.yAxisLabel;

 var htmlElement1 = this.htmlElement;

 var vArray1 = this.vArray;

 $(function() {

 var options = {

 xaxes: [{axisLabel: xAxisLabel1}],

 yaxes: [{axisLabel: yAxisLabel1}]

 };

 $.plot(htmlElement1, [vArray1], options); // рисуем график на элементе "vGraph"

 });

};

Теперь займемся главным скриптом, который будет отвечать за моделирование движение грузика на пружинках (просчет и отрисовка)

function MainMech(canvas, slider\_m, text\_m, slider\_Cx, text\_Cx, slider\_Cy, text\_Cy, slider\_B, text\_B, check) {

 // \*\*\* Задание физических параметров \*\*\*

 var m = 1; // масса

 var Cx = 1; // жесткость X

 var Cy = 1; // жесткость Y

 var B = 0; // вязкость

 // \*\*\* Задание вычислительных параметров \*\*\*

 const fps = 24; // frames per second - число кадров в секунду (качечтво отображения)

// const fps = 4 \* 29; // frames per second - число кадров в секунду (качечтво отображения)

 const spf = 10; // steps per frame - число шагов интегрирования между кадрами (edtkbxbdftn скорость расчета)

 const dt = 0.1 / fps; // шаг интегрирования (качество расчета)

 var steps = 0; // количество шагов интегрирования

 // Установка слайдеров значений

 // функции, запускающиеся при перемещении слайдера

 this.setSlider\_m = function(new\_m){m = new\_m;}; // new\_m - значение на слайдере

 this.setSlider\_Cx = function(new\_Cx){Cx = new\_Cx;};

 this.setSlider\_Cy = function(new\_Cy){Cy = new\_Cy;};

 this.setSlider\_B = function(new\_B){B = new\_B ;};

 //задаем параметры slider

 slider\_m.min = 1; slider\_m.max = 10;

 slider\_m.step = 0.10;

 slider\_m.value = m; // начальное значение ползунка должно задаваться после min и max

 text\_m.value = m.toFixed(2);

 slider\_Cx.min = 1.00; slider\_Cx.max = 10.00;

 slider\_Cx.step = 0.10;

 slider\_Cx.value = Cx; // начальное значение ползунка должно задаваться после min и max

 text\_Cx.value = Cx.toFixed(2);

 slider\_Cy.min = 1.00; slider\_Cy.max = 10.00;

 slider\_Cy.step = 0.10;

 slider\_Cy.value = Cy; // начальное значение ползунка должно задаваться после min и max

 text\_Cy.value = Cy.toFixed(2);

 slider\_B.min = 0.00; slider\_B.max = 10.00;

 slider\_B.step = 0.10;

 slider\_B.value = B; // начальное значение ползунка должно задаваться после min и max

 text\_B.value = B.toFixed(2);

 var count = true; // проводить ли расчет системы

 var vx = 0; // скорость тела X

 var vy = 0; // скорость тела Y

 // создаем объект, связанный с элементом canvas на html странице

 var ocanvas = oCanvas.create({

 canvas: "#canvasMech", // canvasMech - id объекта canvas на html странице

 fps: fps // сколько кадров в секунду

 });

 var rw = canvas.width / 30; var rh = canvas.height / 30;

 var x0 = 15 \* rw ; var y0 = 15 \* rh ;

 var Rx = y0;

 var Ry = x0;

 // создаем пружину

 const coil = 15; // количество витков

 const startX1 = 0; // закрепление пружины

 const startY1 = y0; // закрепление пружины

 const startX2 = canvas.width; // закрепление пружины

 const startY2 = y0; // закрепление пружины

 const startX3 = x0; // закрепление пружины

 const startY3 = 0; // закрепление пружины

 const startX4 = x0; // закрепление пружины

 const startY4 = canvas.height; // закрепление пружины

 var lines1 = []; // этот массив будет содержать ссылки на линии 1й пружины

 var lines2 = []; // этот массив будет содержать ссылки на линии 2й пружины

 var lines3 = []; // этот массив будет содержать ссылки на линии 3й пружины

 var lines4 = []; // этот массив будет содержать ссылки на линии 4й пружины

 for (var i = 0; i < coil; i++ ) {

 lines1[i] = ocanvas.display.line({

 start: { x: 0, y:0 },

 end: { x: 0, y: 0 },

 stroke: "1px #000000"

 }).add();

 lines2[i] = ocanvas.display.line({

 start: { x: 0, y:0 },

 end: { x: 0, y: 0 },

 stroke: "1px #000000"

 }).add();

 lines3[i] = ocanvas.display.line({

 start: { x: 0, y:0 },

 end: { x: 0, y: 0 },

 stroke: "1px #000000"

 }).add();

 lines4[i] = ocanvas.display.line({

 start: { x: 0, y:0 },

 end: { x: 0, y: 0 },

 stroke: "1px #000000"

 }).add();

 }

 // создаем круг

 var ellipse = ocanvas.display.ellipse({x: x0, y: y0, radius: 5 \* m, fill: "rgba(255, 0, 0, 1)"}).add();

 // захват круга мышью

 ellipse.dragAndDrop({

 changeZindex: true, // если много объектов - захваченный будет нарисован спереди

 start: function () { count = false; trajectory = []; this.fill = "rgba(255, 0, 0, 0.5)"; }, // отключаем расчет и делаем объект полупрозрачным

 move: function () { vx = 0; vy = 0; drawSpring();}, // перемещение

 end: function () { count = true; this.fill = "rgba(255, 0, 0, 1)"; } // включаем расчет и убираем полупрозрачность

 });

 ocanvas.bind("mousedown", function () {count = false;}); // заморозить фигуру при клике на поле

 var trajectory = [];

 this.clearTrajectory = function()

 {

 trajectory = [];

 };

 function getF(C , x0, y0 , x, y )

 {

 var R = Math.sqrt( (x-x0)\*(x-x0)+(y-y0)\*(y-y0) );

 var dx = ( Rx/R) \* (x0-x);

 var dy = (Ry/R) \* (y0-y);

 return {x: C\*dx, y:C\*dy};

 }

 function dynamics(){ // интегрирование по времени

 if (!count) return;

 if(check.checked){ //проверка нажатия checkbox

 ellipse.fill = "rgba(0, 0, 255, 1)"

 for (var s=1; s<=spf; s++) {

 var f1 = getF(Cx, 0, y0, ellipse.x, ellipse.y);

 var f2 = getF(Cx, ocanvas.width, y0, ellipse.x, ellipse.y);

 var f3 = getF(Cy, x0, 0, ellipse.x, ellipse.y);

 var f4 = getF(Cy, x0, ocanvas.height, ellipse.x, ellipse.y);

 var fx = f1.x + f2.x + f3.x + f4.x - B \* vx;

 var fy = f1.y + f2.y + f3.y + f4.y - B \* vy;

 vx += fx / m \* dt;

 vy += fy / m \* dt;

 ellipse.x += vx \* dt;

 ellipse.y += vy \* dt;

 ellipse.radius = 5 \* m;

 steps++;

 if (steps % 10 == 0)

 {

 trajectory.push([ellipse.x - x0, y0 - ellipse.y]);

 $.plot($('#vGraph'), [trajectory], {});

 }

 drawSpring();

 }

 }

 else {

 ellipse.fill = "rgba(255, 0, 0, 1)"

 for (var s=1; s<=spf; s++) {

 var fx = - 2 \* Cx \* (ellipse.x - x0) - B \* vx;

 var fy = - 2 \* Cy \* (ellipse.y - y0) - B \* vy;

 vx += fx / m \* dt;

 vy += fy / m \* dt;

 ellipse.x += vx \* dt;

 ellipse.y += vy \* dt;

 ellipse.radius = 5 \* m;

 steps++;

 // подать данные на график

 if (steps % 10 == 0)

 {

 trajectory.push([ellipse.x-x0, y0-ellipse.y]);

 $.plot($('#vGraph'), [trajectory], {});

 }

 drawSpring();

 }

 }

 }

 // Рисуем пружины

function drawSpring() {

 for (var i = 0; i < coil; i++ ) {

 lines1[i].start.x = startX1 + ((ellipse.x-startX1) )/coil\*i;

 lines1[i].end.x = startX1 + ((ellipse.x-startX1))/coil\*(i+1);

 lines1[i].start.y = y0 + ((i%2==0)?1:-1)\*7 + (ellipse.y-startY1)/coil\*i;

 lines1[i].end.y = y0 + ((i%2==0)?-1:1)\*7 + (ellipse.y-startY1)/coil\*(i+1);

 if (i==0) lines1[i].start.y = y0;

 if (i==(coil-1)) lines1[i].end.y = ellipse.y;

 lines2[i].start.x = startX2 + ((ellipse.x-startX2) )/coil\*i;

 lines2[i].end.x = startX2 + ((ellipse.x-startX2) )/coil\*(i+1);

 lines2[i].start.y = y0 + ((i%2==0)?-1:1)\*7 + (ellipse.y-startY1)/coil\*i;

 lines2[i].end.y = y0 + ((i%2==0)?1:-1)\*7 + (ellipse.y-startY1)/coil\*(i+1);

 if (i==0) lines2[i].start.y = y0;

 if (i==(coil-1)) lines2[i].end.y = ellipse.y;

 lines3[i].start.y = startY3 + ((ellipse.y-startY3) )/coil\*i;

 lines3[i].end.y = startY3 + ((ellipse.y-startY3))/coil\*(i+1);

 lines3[i].start.x = x0 + ((i%2==0)?-1:1)\*7 + (ellipse.x-startX3)/coil\*i;

 lines3[i].end.x = x0 + ((i%2==0)?1:-1)\*7 + (ellipse.x-startX3)/coil\*(i+1);

 if (i==0) lines3[i].start.x = x0;

 if (i==(coil-1)) lines3[i].end.x = ellipse.x;

 lines4[i].start.y = startY4 + ((ellipse.y-startY4) )/coil\*i;

 lines4[i].end.y = startY4 + ((ellipse.y-startY4) )/coil\*(i+1);

 lines4[i].start.x = x0 + ((i%2==0)?1:-1)\*7 + (ellipse.x-startX4)/coil\*i;

 lines4[i].end.x = x0 + ((i%2==0)?-1:1)\*7 + (ellipse.x-startX4)/coil\*(i+1);

 if (i==0) lines4[i].start.x = x0;

 if (i==(coil-1)) lines4[i].end.x = ellipse.x;

 }

 }

 ocanvas.setLoop(dynamics).start(); // функция, выполняющаяся на каждом шаге

}

## 2.3 «Простое» и «сложное» движение

Отличие двух типов движения заключается только в расчете сил:

|  |  |
| --- | --- |
|  |  |

****

Как видно из картинки сила должна действовать так, как на картинке, так оно и есть при сложном движении, рассматриваем проекции сил на оси каждой пружины. При простом же движении мы переходим к рассмотрению аналогичной задачи, состоящая из двух частей:

1. Грузик на горизонтальной пружине
2. Грузик на вертикальной пружине

## 2.4 Итог работы

Итогом данной работы получилось приложение «**Моделирование фигур Лиссажу методом пружин**» скриншоты которой я представляю ниже.

На данном скриншоте изображена главная страница после загрузки:



На этом скриншоте продемонстрированно простое движение с графиком движения



На этом скриншоте продемонстрировано сложное движение с графиком



***Заключение***

В ходе написания курсовой работы был прочитан и систематизирован теоретический материал по данной теме, а так же выполнены следующие поставленные во введении задачи:

* Описано что такое JS-фреймворки
* Представлены виды фреймворков с кратким описанием каждого из них
* Было разработано приложение с использованием фреймворка Jquery
* Приложение обладает простым и понятным интерфейсом, все элементы просто в использовании
* Приложение моделирует как само движение грузика, так и видна траектория движения

Исходя из изученного материала, я пришел к выводу, что для наглядного моделирования удобно использовать JavaScript, так как он обладает широкими возможностями и простым синтаксисом. Так же удобно использовать JavaScript для проектов, которые будут дорабатываться в дальнейшем, т.к. при помощи базовых знаний языка и комментариев можно быстро разбираться с кодом. Так же JavaScript является языком мультиплатформенным, мы можем просмотреть результат в любом браузере. JavaScript оптимальный язык для написания многих наглядных приложений.

# ***Литература***

1. http://ru.wikipedia.org/wiki/Библиотека\_JavaScript
2. http://habrahabr.ru/
3. http://jqbook.net.ru/
4. http://ru.html.net/
5. Антон Шевчук. Jquery. Учебник для начинающих.: 2013. – 123с.
6. Бенедетти Р., Крэнли Р. Изучаем работу с jQuery. – СПб.: Питер, 2012. – 528 с.
7. Бер Бибо, Иегуда Кац. jQuery. Подробное руководство по продвинутому JavaScript, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 624 с.
8. Каслдайн Э., Шарки К. Изучаем jQuery. 2-е изд. – СПб.: Питер, 2012. – 400 с.
9. Самков Г. А. jQuery. Сборник рецептов. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. – 416 с.

Размещено на Allbest.ru