Курсовой проект по предмету

программирование:

“Работа с JavaScript”

Выполнил: Александров С.Д. \_\_\_\_\_\_\_\_\_

Принял: Панченко А.Ю. \_\_\_\_\_\_\_\_\_

2014

**Оглавление**

1.Вступление………….………………………………………………………………….2

2.Возможности языка……………………………………………………………………2

3.Структура языка ………………………………………………………………………3

4.Встраивание в веб-страницы………………………………………………………….3

5.Пользовательские скрипты в браузере……………………………………………….3

6.Виджеты………………………………………………………………………………..4

7.Версии…………………………………………………………………………………..4

8.Библиотеки JavaScript………………………………………………………………….5

9.Средства тестирования………………………………………………………………...6

10.WebGL…………………………………………………………………………………7

11.Принципы WebGL……………………………………………………………………7

12.Реализации WebGL……………………………………………………………………7

13.Библиотеки WebGL……………………………………………………………………7

14.Пример создания и анимации объекта……………………………………………….8

15.Пример программы на JavaScript……………………………………………………..10

16.Вывод…………………………………………………………………………………..17

17.Список литературы……………………………………………………………………18

## Вступление

В современном мире интернет стал играть очень важную роль в жизни людей. Кто-то просто просматривает новости и видео, а для кого-то интернет это заработок. Так или иначе мы сталкиваемся с различными flash-приложениями или написанными на JavaScript(JS). Хотя раньше программирование на JS казалось ведомо только избранным, а сейчас же столь мощный пакет стал более доступным для обычных пользователей. Так же относительно недавно проводили сравнение вычислительных возможностей flash и JS,и оказалось, что JS значительно обгоняет в данном параметре своего конкурента. Поэтому автор считает, что наступает век JavaScript. Также автор надеется, что после прочтение ниже приведенной работы вы поддержите его мнение.

## Возможности языка

JavaScript является объектно-ориентированным языком, но используемое в языке [прототипирование](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих [функциональным языкам](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — функции как [объекты первого класса](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0), объекты как списки, [карринг](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3), [анонимные функции](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8), [замыкания](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком [Си](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29) имеет коренные отличия:

* [объекты](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), с возможностью [интроспекции](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29);
* функции как [объекты первого класса](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0);
* автоматическое [приведение типов](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2);
* автоматическая [сборка мусора](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0);
* [анонимные функции](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8).

В языке отсутствуют такие полезные вещи, как:

* [модульная система](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29): JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
* [стандартная библиотека](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0): в частности, отсутствует [интерфейс программирования приложений](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) по работе с файловой системой, управлению потоками ввода/вывода, базовых типов для [бинарных данных](http://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB);
* стандартные [интерфейсы](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) к [веб-серверам](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и [базам данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85);
* [система управления пакетами](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%B0%D0%BC%D0%B8), которая бы отслеживала зависимости и автоматически устанавливала их.

## 

## Структура языка

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей:

* ядро (ECMAScript),
* объектная модель браузера ([Browser Object Model или BOM](http://ru.wikipedia.org/w/index.php?title=Browser_Object_Model&action=edit&redlink=1) ([*de*](http://de.wikipedia.org/wiki/Browser_Object_Model))),
* объектная модель документа ([Document Object Model или DOM](http://ru.wikipedia.org/wiki/Document_Object_Model)).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

## Встраивание в веб-страницы

Для добавления JavaScript-кода на страницу, можно использовать теги <script></script>, которые рекомендуется, но не обязательно, помещать внутри контейнера <head>. Контейнеров <script> в одном документе может быть сколько угодно. Атрибут «type='text/javascript'» указывать необязательно, так как по умолчанию стоит javascript.

Скрипт, выводящий [модальное окно](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D0%BA%D0%BD%D0%BE) с классической надписью «Hello, World!» внутри браузера:

<script type="text/javascript">

alert('Hello, World!');

</script>

## Пользовательские скрипты в браузере

Пользовательские скрипты в браузере — это программы, написанные на JavaScript, выполняемые в браузере пользователя при загрузке страницы. Они позволяют автоматически заполнять формы, переформатировать страницы, скрывать нежелательное содержимое и встраивать желательное для отображения содержимое, изменять поведение клиентской части веб-приложений, добавлять элементы управления на страницу и т. д.

Для управления пользовательскими скриптами в [Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) используется расширение [Greasemonkey](http://ru.wikipedia.org/wiki/Greasemonkey); [Opera](http://ru.wikipedia.org/wiki/Opera) и [Google Chrome](http://ru.wikipedia.org/wiki/Google_Chrome) предоставляют средства поддержки пользовательских скриптов и возможности для выполнения ряда скриптов Greasemonkey.

## 

## Виджеты

Виджет — вспомогательная мини-программа, графический модуль которой размещается в рабочем пространстве соответствующей [родительской программы](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_%D0%B2%D0%B8%D0%B4%D0%B6%D0%B5%D1%82%D0%BE%D0%B2&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Widget_engine)), служащая для украшения рабочего пространства, развлечения, решения отдельных рабочих задач или быстрого получения информации из интернета без помощи веб-браузера. JavaScript используется как для реализации виджетов, так и для реализации движков виджетов. В частности, при помощи JavaScript реализованы [Apple Dashboard](http://ru.wikipedia.org/wiki/Apple_Dashboard), [Microsoft Gadgets](http://ru.wikipedia.org/w/index.php?title=Microsoft_Gadgets&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Microsoft_Gadgets)), [Yahoo!\_Widgets](http://ru.wikipedia.org/w/index.php?title=Yahoo%21_Widgets&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Yahoo%21_Widgets)), [Google Gadgets](http://ru.wikipedia.org/w/index.php?title=Google_Gadgets&action=edit&redlink=1), [Klipfolio Dashboard](http://ru.wikipedia.org/w/index.php?title=Klipfolio_Dashboard&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Klipfolio_Dashboard)).

## Версии

|  |  |  |
| --- | --- | --- |
| **JavaScript** | **Соответствующая версия JScript** | **Существенные изменения** |
| 1.0 ([Netscape](http://ru.wikipedia.org/wiki/Netscape_Navigator) 2.0, март 1996) | 1.0 (ранние версии [IE](http://ru.wikipedia.org/wiki/Microsoft_Internet_Explorer) 3.0, август 1996) | Оригинальная версия языка JavaScript. |
| 1.1 (Netscape 3.0, август 1996) | 2.0 (поздние версии IE 3.0, январь 1997) | В данной версии был реализован объект Array и устранены наиболее серьёзные ошибки. |
| 1.2 (Netscape 4.0, июнь 1997) |  | Реализован переключатель switch, регулярные выражения. Практически приведён в соответствии с первой редакцией спецификации ECMA-262. |
| 1.3 (Netscape 4.5, октябрь 1998) | 3.0 (IE 4.0, октябрь 1997) | Совместим с первой редакцией ECMA-262. |
| 1.4 (только Netscape Server) | 4.0 ([Visual Studio](http://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) 6, нет версии IE) | Применяется только в серверных продуктах Netscape. |
|  | 5.0 (IE 5.0, март 1999) |  |
|  | 5.1 (IE 5.01) |  |
| 1.5 (Netscape 6.0, ноябрь 2000; также поздние версии Netscape и [Mozilla](http://ru.wikipedia.org/wiki/Mozilla)) | 5.5 (IE 5.5, июль 2000) | Редакция 3 (декабрь 1999). Совместим с третьей редакцией спецификации ECMA-262. |
|  | 5.6 (IE 6.0, октябрь 2001) |  |
| 1.6 ([Gecko](http://ru.wikipedia.org/wiki/Gecko_%28%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA%29) 1.8, [Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) 1.5, ноябрь 2005) |  | Редакция 3 с некоторыми совместимыми улучшениями: [E4X](http://ru.wikipedia.org/wiki/ECMAScript_for_XML), дополнения к Array (например, Array.prototype.forEach), упрощения для Array и String |
| 1.7 (Gecko 1.8.1, Firefox 2.0, осень 2006), расширение JavaScript 1.6 |  | Редакция 3, с добавлением всех улучшений из JavaScript 1.6, генераторов и списочных выражений (list comprehensions, [a\*a for (a in iter)]) из [Python](http://ru.wikipedia.org/wiki/Python_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29), блоковых областей с использованием let и деструктурирующего присваивания (var [a, b] = [1, 2]). |
|  | [JScript .NET](http://ru.wikipedia.org/wiki/Microsoft_JScript_.NET) ([ASP.NET](http://ru.wikipedia.org/wiki/Microsoft_ASP.NET); нет версии IE) | (Считается, что JScript .NET разработан при участии других членов [ECMA](http://ru.wikipedia.org/wiki/ECMA)) |
| 1.8 (Gecko 1.9, Firefox 3.0, осень 2008), расширение JavaScript 1.7 |  | Новая форма записи для функций, сходная с типичными [лямбда-выражениями](http://ru.wikipedia.org/wiki/%D0%9B%D1%8F%D0%BC%D0%B1%D0%B4%D0%B0-%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F), [генераторы](http://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Generator_%28computer_science%29)), новые методы итеративной обработки массивов reduce() и reduceRight(). |
| 1.8.1 (Gecko 1.9.1, Firefox 3.5) |  | Встроенная поддержка JSON, метод getPrototypeOf() у Object, методы [trim()](http://ru.wikipedia.org/wiki/Trim_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), trimLeft(), trimRight() у String |
| 2.0 |  | Редакция 4 (разработка не закончена, название зарезервировано ECMA, но не было использовано для публикации) |
|  |  | Редакция 5 (ранее известная под названием ECMAScript 3.1. Финальная версия принята 3 декабря 2009 года.) |

## Библиотеки JavaScript

Для обеспечения высокого [уровня абстракции](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B9_%D0%B0%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и достижения приемлемой степени [кросс-браузерности](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C) при разработке веб-приложений используются библиотеки JavaScript. Они представляют собой набор многократно используемых объектов и функций.

Среди известных JavaScript библиотек можно отметить [Adobe life](http://ru.wikipedia.org/w/index.php?title=Adobe_life&action=edit&redlink=1), [Dojo](http://ru.wikipedia.org/wiki/Dojo) Toolkit, [Extjs](http://ru.wikipedia.org/wiki/Extjs), [jQuery](http://ru.wikipedia.org/wiki/JQuery), [Mootools](http://ru.wikipedia.org/wiki/Mootools), [Prototype](http://ru.wikipedia.org/wiki/Prototype_%28%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA%29), [Qooxdoo](http://ru.wikipedia.org/w/index.php?title=Qooxdoo&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Qooxdoo)), [Underscore](http://ru.wikipedia.org/wiki/Underscore).

1)**jQuery** — [библиотека JavaScript](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующаяся на взаимодействии [JavaScript](http://ru.wikipedia.org/wiki/JavaScript) и [HTML](http://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный [API](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) по работе с [AJAX](http://ru.wikipedia.org/wiki/AJAX).

Возможности:

* Движок кроссбраузерных CSS-селекторов [Sizzle](http://ru.wikipedia.org/w/index.php?title=Sizzle&action=edit&redlink=1), выделившийся в отдельный проект;
* Переход по дереву DOM, включая поддержку [XPath](http://ru.wikipedia.org/wiki/XPath) как плагина;
* События;
* Визуальные эффекты;
* [AJAX](http://ru.wikipedia.org/wiki/AJAX)-дополнения;
* JavaScript-[плагины](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD).

2) **Ext JS** — [библиотека JavaScript](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) для разработки [веб-приложений](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [пользовательских интерфейсов](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), изначально задуманная как расширенная версия [Yahoo! UI Library](http://ru.wikipedia.org/wiki/Yahoo!_UI_Library), однако преобразовавшаяся затем в отдельный фреймворк. До версии 4.0 использовала адаптеры для доступа к библиотекам [Yahoo! UI Library](http://ru.wikipedia.org/wiki/Yahoo!_UI_Library), [jQuery](http://ru.wikipedia.org/wiki/JQuery) или [Prototype](http://ru.wikipedia.org/wiki/Prototype_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA))/[script.aculo.us](http://ru.wikipedia.org/wiki/Script.aculo.us), начиная с 4-ой версии адаптеры отсутствуют. Поддерживает технологию [AJAX](http://ru.wikipedia.org/wiki/AJAX), анимацию, работу с [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model), реализацию таблиц, вкладок, обработку событий и все остальные новшества [Web 2.0](http://ru.wikipedia.org/wiki/Web_2.0).

## 

## Средства тестирования

Большинство [фреймворков автоматизированного тестирования](http://ru.wikipedia.org/w/index.php?title=%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Test_automation_framework)) JavaScript-кода предполагают запуск тестов в браузере. Это осуществляется при помощи HTML-страницы, являющейся [контекстом тестирования](http://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/Test_fixture)), которая, в свою очередь загружает всё необходимое для осуществления тестирования. Первыми такими фреймворками были [JsUnit](http://ru.wikipedia.org/w/index.php?title=JsUnit&action=edit&redlink=1) ([*англ.*](http://en.wikipedia.org/wiki/JSUnit)) (создан в 2001 году), [Selenium](http://ru.wikipedia.org/wiki/Selenium) (создан в 2004 году). Альтернатива — запуск тестов из командной строки. В этом случае используются окружения, отличные от браузера, например, Rhino. Одним из первых инструментов такого рода является Crosscheck, позволяющий тестировать код, эмулируя поведение Internet Explorer 6 и Firefox версий 1.0 и 1.5. Другой пример фреймворка автоматизированного тестирования JavaScript-кода, не использующего браузер для запуска тестов — библиотека env.js, созданная Джоном Резигом. Она использует Rhino и при этом содержит эмуляцию окружения браузера и DOM.

Blue Ridge, плагин к фреймворку веб-приложений [Ruby on Rails](http://ru.wikipedia.org/wiki/Ruby_on_Rails), позволяет осуществлять [модульное тестирование](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) JavaScript-кода как в браузере, так и вне его. Это достигается за счёт использования фреймворка автоматизированного тестирования Screw.Unit и Rhino с env.js[[101]](http://ru.wikipedia.org/wiki/JavaScript#cite_note-116).

Главная проблема систем тестирования, не использующих браузеры, в том, что они используют эмуляции, а не реальные окружения, в которых выполняется код. Это приводит к тому, что успешное прохождение тестов не гарантирует того, что код корректно отработает в браузере. Проблемой систем тестирования, использующих браузер, является сложность работы с ними, необходимость осуществления рутинных неавтоматизированных действий. Для решения этого JsTestDriver, фреймворк автоматизированного тестирования, разрабатываемый Google, использует сервер, взаимодействующий с браузерами для осуществления тестирования. Сходным образом ведёт себя Selenium Remote Control, входящий во фреймворк автоматизированного тестирования Selenium: он включает в себя сервер, запускающий и завершающий браузеры и действующий как HTTP-прокси для запросов к ним. Кроме того, в Selenium содержится Selenium Grid, позволяющий осуществлять одновременное тестирование JavaScript-кода на разных компьютерах с разными окружениями, уменьшая время выполнения тестов. Testswarm, имеющее поддержку фреймворков автоматизированного тестирования JavaScript-кода QUnit (библиотека [jQuery](http://ru.wikipedia.org/wiki/JQuery)), UnitTestJS (библиотека [Prototype](http://ru.wikipedia.org/wiki/Prototype_%28%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA%29)), JSSpec (библиотека [MooTools](http://ru.wikipedia.org/wiki/MooTools)), JsUnit, Selenium и Dojo Objective Harness, представляет собой распределённое средство поддержки [непрерывной интеграции](http://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F)[[108]](http://ru.wikipedia.org/wiki/JavaScript#cite_note-123).

Негативное свойство, которым может обладать фреймворк автоматизированного тестирования JavaScript-кода — наличие зависимостей. Это создаёт риск отказа в работе тестируемого кода, успешно проходящего тесты, в среде с отсутствием этих зависимостей. Например, исходная версия JsUnitTest, фреймворка, созданного и использовавшегося для тестирования библиотеки Prototype, зависела от самой Prototype, изменяющего свойства объектов в глобальной области видимости. Включение в [библиотеку JavaScript](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) инструмента тестирования — распространённая практика. Так YUI Test 3 является частью [Yahoo! UI Library](http://ru.wikipedia.org/wiki/Yahoo%21_UI_Library) и может быть безопасно использован для тестирования произвольного JavaScript-кода. QUnit — фреймворк автоматизированного тестирования, созданный разработчиками jQuery.

## 

## WebGL

**WebGL** (*Web-based Graphics Library*) — [программная библиотека](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) для языка программирования [JavaScript](http://ru.wikipedia.org/wiki/JavaScript), позволяющая создавать на JavaScript интерактивную 3D-графику, функционирующую в широком спектре совместимых с ней веб-браузерах. За счёт использования низкоуровневых средств поддержки [OpenGL](http://ru.wikipedia.org/wiki/OpenGL), часть кода на WebGL может выполняться непосредственно на [видеокартах](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%BA%D0%B0%D1%80%D1%82%D0%B0). Проект по созданию библиотеки управляется некоммерческой организацией [Khronos Group](http://ru.wikipedia.org/wiki/Khronos_Group).

## 

## Принципы WebGL

Библиотека построена на основе [OpenGL ES](http://ru.wikipedia.org/wiki/OpenGL_ES) 2.0 и обеспечивает [API](http://ru.wikipedia.org/wiki/API) для 3D-графики., использует [HTML5](http://ru.wikipedia.org/wiki/HTML5)-элемент [canvas](http://ru.wikipedia.org/wiki/Canvas), также оперирует с [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model). Автоматическое управление памятью предоставляется языком JavaScript.

## Реализация WebGL

* [Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) — WebGL был включён во все платформы, у которых есть нужная графическая карта с актуальными драйверами, начиная с версии 4.0.
* [Google Chrome](http://ru.wikipedia.org/wiki/Google_Chrome) — WebGL включён по умолчанию во все версии начиная с 9.
* [Safari](http://ru.wikipedia.org/wiki/Safari) — поддерживает WebGL, но поддержка отключена по умолчанию.
* [Opera](http://ru.wikipedia.org/wiki/Opera) — WebGL реализован в версии Opera 12.00, но отключена по умолчанию.
* [Internet Explorer](http://ru.wikipedia.org/wiki/Internet_Explorer) — Microsoft не заявляла об официальной поддержке WebGL, независимыми разработчиками выпущены плагины Chrome Frame и IEWebGL, предусматриваюющие опции, необходимые для поддержки WebGL в Internet Explorer. По-видимому, поддержка WebGL появится в IE 11.

## Библиотеки WebGL

* Для разработки WebGL используется несколько библиотек. Первой общедоступной стала библиотека WebGLU. Среди других библиотек для WebGL - [GLGE](http://ru.wikipedia.org/w/index.php?title=GLGE&action=edit&redlink=1), [C3DL](http://ru.wikipedia.org/w/index.php?title=C3DL&action=edit&redlink=1), Copperlicht, SpiderGL, gwt-g3d (обёртка для [GWT](http://ru.wikipedia.org/wiki/GWT)), [SceneJS](http://ru.wikipedia.org/w/index.php?title=SceneJS&action=edit&redlink=1), [X3DOM](http://ru.wikipedia.org/w/index.php?title=X3DOM&action=edit&redlink=1), [Processing.js](http://ru.wikipedia.org/w/index.php?title=Processing.js&action=edit&redlink=1), [Three.js](http://ru.wikipedia.org/w/index.php?title=Three.js&action=edit&redlink=1), Turbulenz, OSGJS, XB PointStream и CubicVR.js.
* ANGLE (Almost Native Graphics Layer Engine) — программа, выпущенная под [лицензией BSD](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD), которая позволяет переводить содержимое WebGL в OpenGL ES 2.0, вызывать API DirectX 9, которые взаимодействуют с платформами [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows) без необходимости в дополнительных драйверах [OpenGL](http://ru.wikipedia.org/wiki/OpenGL).

## 

## 

## Пример создания и анимации объекта.

Подключаем библиотеку **Three.js** и **RequestAnimationFrame.js**

1. <script src="Three.js"></script>
2. <script src="RequestAnimationFrame.js"></script>

Теперь можно приступить к написанию кода. Вначале нам нужно создать объект WebGLRenderer с режимом сглаживания.

1. var renderer = new THREE.WebGLRenderer({antialias: true});
2. renderer.setSize(document.body.clientWidth, document.body.clientHeight);

Далее применяем его BODY

1. document.body.appendChild(renderer.domElement);

ставим цвет для очистки рендера и очищаем его

1. renderer.setClearColorHex(0xEEEEEE, 1.0);
2. renderer.clear();

Теперь станем определять необходимые нам переменные:

1. var fov = 45; // угол обзора камеры
2. var width = renderer.domElement.width; // ширина сцены
3. var height = renderer.domElement.height; // высота сцены
4. var aspect = width / height; // соотношение сторон экрана
5. var near = 1; // минимальная видимость
6. var far = 10000; // максимальная видимость

Далее создаем объект камеры и отдаляем ее назад на 300 поинтов

1. var camera = new THREE.PerspectiveCamera( fov, aspect, near, far );
2. camera.position.z = 300;

Теперь создаем сцену, меш (трехмерный объект, в который добавляется другая геометрия), создаем простой куб с вершинами в 50 поинтов, и задаем черный цвет. После чего добавляем созданный нами меш в сцену, и рендерим сцену через камеру.

1. var scene = new THREE.Scene();
2. var cube = new THREE.Mesh(
3. new THREE.CubeGeometry(50,50,50),
4. new THREE.MeshBasicMaterial({color: 0x000000, opacity: 1})
5. );
6. scene.add(cube);
7. renderer.render(scene, camera);

Если сейчас посмотреть - можно увидеть просто черный квадрат. Создадим для него анимацию.

1. function animate(t) {
2. // задаем круговое движение камеры
3. camera.position.set(
4. Math.sin(t/1000)\*300, 150, Math.cos(t/1000)\*300);
5. // очищаем рендер и обновляем lookAt каждый фрейм
6. renderer.clear();
7. camera.lookAt(scene.position);
8. renderer.render(scene, camera);
9. window.requestAnimationFrame(animate, renderer.domElement);
10. animate(new Date().getTime());

Здесь мы создаем функцию вращения камеры по окружности вокруг куба. Очищаем каждый фрейм рендер окна, и добавляем функцию в цикличное выполнение через **requestAnimationFrame**, которая является аналогом функции **setInterval()**, но в отличии от нее первая дает браузеру знать, когда нужно прорисовать следующий фрейм, и выполняется только тогда, когда нужный элемент видимый на экране.

Добавив источник света.

1. var light = new THREE.SpotLight();
2. light.position.set( 170, 330, -160 );
3. scene.add(light);

И поверх уже созданного нами куба, добавим еще один.

1. var litCube = new THREE.Mesh(
2. new THREE.CubeGeometry(50, 50, 50),
3. new THREE.MeshLambertMaterial({color: 0xffffff}));
4. litCube.position.y = 50;
5. scene.add(litCube);

**Пример программы на JavaScript**

Файл №1 “index.html” (его мы и будем запускать)

<!DOCTYPE html>

<html>

<head>

<title>Three.js - dice</title>

<meta charset="utf-8">

<style>

body {

margin: 0;

padding: 0;

overflow: hidden;

}

</style>

**<!—подключаемые библиотеки -->**

<script type="text/javascript" src="Three.js"></script>

<script type="text/javascript" src="RequestAnimationFrame.js"></script>

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

<!-- app -->

<script type="text/javascript" src="demoapp.js"></script>

</head>

<body>

**//Создание музыки на заднем фоне**

<script type="text/javascript" src="http://scriptjava.net/source/scriptjava/scriptjava.js"></script>

<script type="text/javascript">

$$r(function() {

$$i({

create:'script',

attribute: {

'type':'text/javascript',

'src':'sound.js'

},

insert:$$().body,

onready:function() { **//выполняю только после загрузки скрипта**

**//создаю экземпляр обьекта**

var s=new sound();

s.start({

'music':'http://goldenland.onfind.net/2/1/5.mp3',

'player':'http://nagon.net/flash/player\_mod0.swf'

});

}

});

});

</script>

<script Language="JavaScript" type="text/javascript">

</script>

**//Создание кнопки**

<input type="button" value="Add 5" onClick="freshStart(5)">

**//Создание ползунков**

<br>

Axis X: -1500<input type="range" name= "Axis X" min="-1500" max="1500" oninput="wallMakerX(this.value)" >1500

<br>

Axis Y: -1500<input type="range" name= "Axis Y" min="-1500" max="1500" oninput="wallMakerY(this.value)" >1500

<br>

Axis Z: -1500<input type="range" name= "Axis Z" min="-1500" max="1500" oninput="wallMakerZ(this.value)" >1500

<br>

<div></div>

</body>

</html>

Файл №2 “jquery.js” (это библиотека; подключаем сами)

Файл №3 “Three.js” (это библиотека; подключаем сами)

Файл №4 “RequestAnimationFrame.js ”:

if ( !window.requestAnimationFrame ) {

window.requestAnimationFrame = ( function() {

return window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.oRequestAnimationFrame ||

window.msRequestAnimationFrame ||

function( /\* function FrameRequestCallback \*/ callback, /\* DOMElement Element \*/ element ) {

window.setTimeout( callback, 1000 / 60 );

};

} )();

}

Файл №5 “demoapp.js” (это тело нашей программы):

**//Задание переменных**

var container, camera, scene, renderer, floormesh,cubeNew,cubeMeshNew,cubeMesh,cubeMesh1,wallNew,wallMeshNew,i,wall,wallMesh, phi = 0;

var m = 10; **//Начальное количество кубиков**

var old\_m = 0;

var addX=0;

var addY=0;

var addZ=0;

var d; **// Количество добавленных кубиков**

var cubes = new Array(m);

var wayOut = new Array (m);

var walls = new Array(10);

var cubesMesh = new Array (m);

var wallsMesh = new Array(10);

var startPosition = new Array (m);

var wallsPosition = new Array(10);

var xPosition = new Array(m);

var zPosition = new Array(m);

var randi,randiNew; **//Рандомное число**

**//Задание цветов сторон кубика**

var materials = [

new THREE.MeshBasicMaterial( { color: 0xE01B4C }), **// правая сторона**

new THREE.MeshBasicMaterial( { color: 0x34609E }), **// левая сторона**

new THREE.MeshBasicMaterial( { color: 0x7CAD18 }), **//верх**

new THREE.MeshBasicMaterial( { color: 0x00EDB2 }), **// низ**

new THREE.MeshBasicMaterial( { color: 0xED7700 }), **// лицевая** **сторона**

new THREE.MeshBasicMaterial( { color: 0xB5B1AE }) **// задняя сторона**

];

**//Функция перемещения стены при помощи слайдера по X**

function wallMakerX(r)

{

wallMesh.position.x = 300 + Number(r);

render(m);

}

**//Функция перемещения стены при помощи слайдера по Y**

function wallMakerY(r)

{

wallMesh.position.y = 400 + Number(r);

render(m);

}

**//Функция перемещения стены при помощи слайдера по Z**

function wallMakerZ(r)

{

wallMesh.position.z = 450 + Number(r);

render(m);

}

**//Функция добавления новых кубиков в смерч**

function freshStart(d)

{

old\_m = m;

m += d;

init(m);

render(m);

}

**//Функция создания кубиков и задания их начального положения**

function init(m)

{

container = $( 'div' ).attr('id','cardfield');

$('body').append( container );

for ( i = 0 + old\_m; i < m; i++)

{

cubeNew = new THREE.CubeGeometry( 50, 50, 50, 1, 1, 1, materials );

cubes[i] = cubeNew;

cubeMeshNew = new THREE.Mesh( cubes[i], new THREE.MeshFaceMaterial() );

cubesMesh[i] = cubeMeshNew;

randiNew = (Math.random()\*50 + 1)

cubesMesh[i].position.x = Math.sin(randiNew \* 10) ;

cubesMesh[i].position.y = randiNew ;

cubesMesh[i].position.z = Math.cos(randiNew \* 10) ;

scene.addChild( cubesMesh[i] );

randi = Math.floor((Math.random()\*200)+10);

startPosition[i] = randi \* 50;

xPosition[i] = startPosition[i];

zPosition[i] = startPosition[i];

wayOut[i] = 1;

new THREE.ShadowVolume( cubesMesh[i] );

}

}

**//Функция необходимая для анимации кубиков**

function animate()

{

requestAnimationFrame(animate);

render(m);

}

**//Функция изменения положения кубиков в пространстве**

function render(m)

{

for (i = 0; i < m; i++)

{

**//Вращене кубиков смерча**

cubesMesh[i].rotation.x += (i / 20 + 0.5) \* Math.PI / 90;

cubesMesh[i].rotation.y += (i / 20 + 1.0) \* Math.PI / 90;

cubesMesh[i].rotation.z += (i / 20 + 1.5) \* Math.PI / 90;

**//Условие встречи со стеной**

if ((cubesMesh[i].position.x < (wallMesh.position.x + 100))&&(cubesMesh[i].position.x > (wallMesh.position.x - 100))&&(cubesMesh[i].position.y > (wallMesh.position.y - 600))&&(cubesMesh[i].position.y < (wallMesh.position.y + 600))&&(cubesMesh[i].position.z < (wallMesh.position.z + 300))&&(cubesMesh[i].position.z > (wallMesh.position.z - 300)))

{wayOut[i] = wayOut[i] \* (-1)}

**//Изменяет направление при встречи со стеной**

xPosition[i] += phi \* wayOut[i] ;

zPosition[i] += phi \* wayOut[i] ;

**//Изменение координат кубиков**

cubesMesh[i].position.x = Math.sin( xPosition[i] ) \* i \* 25;

cubesMesh[i].position.y = phi + i \* 50;

cubesMesh[i].position.z = Math.cos( zPosition[i] ) \* i \* 25;

}

phi = 0.1 + (Math.random())/5;

renderer.render(scene, camera);

}

$(document).ready(function() {

**//Создание неподвижной камеры**

camera = new THREE.Camera(45, window.innerWidth/window.innerHeight,1, 10000);

**//Задание положения камеры**

camera.position.z = 3000;

camera.position.y = 5000;

camera.target.position.y = -1000;

**//Создание сцены**

scene = new THREE.Scene();

**//Создание "пола"**

var floorgeo = new THREE.CubeGeometry(1500,1500,5);

floormesh = new THREE.Mesh(floorgeo, new THREE.MeshBasicMaterial({color: 0x248C0F, opacity:0.9}));

floormesh.position.y = -200;

floormesh.rotation.x = 90 \* Math.PI / 180;

scene.addChild(floormesh);

**//Создание стены, изменяющей направление**

wall = new THREE.CubeGeometry( 100, 600, 300, 1, 1, 1, materials );

wallMesh = new THREE.Mesh( wall, new THREE.MeshFaceMaterial() );

wallMesh.position.y = 400;

wallMesh.position.x = 300;

wallMesh.position.z = 250;

scene.addChild( wallMesh );

new THREE.ShadowVolume( wallMesh );

**//Задание источника света**

light = new THREE.DirectionalLight( 0xffffff );

light.castShadow = true;

light.position.set( 0, 1, 0 );

scene.addChild( light );

**//Обновление анимации**

init(m);

renderer = new THREE.WebGLRenderer();

renderer.setSize( window.innerWidth, window.innerHeight );

container.append( renderer.domElement );

animate();

});

**Вывод**

Стремительное развитие интернета создаёт благоприятные условия для развития своего дела в виртуальном пространстве. Постоянно расширяющаяся аудитория пользователей всемирной паутины является самым широким рынком на котором можно реализовывать разнообразные товары: как реальные, так и виртуальные

В основе любого бизнеса, который строится в интернет, лежит веб-сайт. Причём на успех проекта влияет очень много факторов. Но первое, что видит посетитель любого сайта — это его интерфейс.

От внешнего вида и удобства использования зависит, останется ли случайный веб-сёрфер почитать, что такого ценного имеет сайт или просто нажмёт кнопку «Закрыть» в своём браузере. Это уже потом, когда пользователь перейдёт в разряд постоянных читателей, он будет готов прощать ошибки и огрехи в интерфейсе сайта. Но при первой встрече интерфейс будет играть ключевую роль в завязывании тесных отношений с вашим клиентом.

Сделать оригинальную, стильную и качественно работающую одёжку сайта невозможно без использования JavaScript. Как мы уже показали ранее на примере программы, всего лишь несколько строк кода оживят ваш сайт.

Тем самым использование JavaScript на страницах веб-сайта откроет новую веху в развитии вашего веб-проекта.

**Список литературы**

1. Баррет Д. **JavaScript. Web-профессионалам.** - Киев: БХВ - Киев, 2001.

2. Бранденбау Д. **JavaScript: сборник рецептов.** - СПб.: Питер, 2000.

3. Вайк А. **JavaScript в примерах.** - Киев: ДиаСофт, 2000.

4. Вандер Вер Э. **JavaScript для "чайников".** - Диалектика, 2001.

5. Гарнаев А. **Web-программирование на Java и JavaScript.** - СПб.: БХВ Санкт-Перебург, 2002.

6. Дарнел Р. **JavaScript. Справочник.** - СПб.: Питер, 2000.

7. Дмитриева М. **Самоучитель JavaScript.** - СПб.: БХВ Санкт-Перебург, 2001.

8. Мэрдок К. **JavaScript: наглядный курс создания динамических Web-страниц.** - Диалектика, 2001.