ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению**  
**09.03.04 Программная инженерия**

*Доронина Оксана Сергеевна*

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Использование визуализации при анализе экономических данных

|  |  |
| --- | --- |
| Рецензент  доцент, канд. техн. наук  А.В. Савченко | Научный руководитель  доцент, канд. техн. наук  М. В. Шишкин |

Нижний Новгород, 2016

Оглавление

Введение

Глава 1

Часть 1.1Визуализация данных……………………………5

Часть 1.2Государтсвенные закупки……………………….15

Часть 1.3JavaScript и React…………………………………18

Часть 1.4JSX………………………………………………....22

Часть 1.5 Архитектура Flux………………………………...24

Глава 2 Практическая часть

Вывод………………………………………………………...33

Введение

Проблема обработки данных наиболее актуальна в наше время. В современном мире существует огромное разнообразие данных и их систематизация, способы представления тоже становятся разнообразнее. Если в старину доступ к данным имел не каждый человек, а только избранные, то теперь каждый человек может найти интересующую его информацию, например, о бюджете страны, или о средствах, потраченных на ремонтные работы, найти данные об изменениях цен на продукты и услуги ЖКХ. Существует много видов обработки данных, но наиболее лучшим является визуализация.

Визуальная информация лучше воспринимается и позволяет быстро и эффективно донести до зрителя собственные мысли и идеи. Физиологически, восприятие визуальной информации является основной для человека. Есть многочисленные исследования, подтверждающие, что:

* 90% информации человек воспринимает через зрение
* 70% сенсорных рецепторов находятся в глазах
* около половины нейронов головного мозга человека задействованы в обработке визуальной информации
* на 19% меньше при работе с визуальными данными используется когнитивная функция мозга, отвечающая за обработку и анализ информации
* на 17% выше производительность человека, работающего с визуальной информацией
* на 4,5% лучше воспоминаются подробные детали визуальной информации

Именно по ряду вышеуказанных причин, для донесения информации и для ее лучшего усвоения лучше всего использовать визуализацию данных.

Цель данной работы- использование визуализации в анализе экономических данных.

Задачами данной работы являются:

* Поиск подходящих экономических данных;
* Анализ данных;
* Построение интерактивных графиков для выбранных данных

Работа состоит из введения, первая и вторая главы, заключения.  
В первой главе - теоретическая основа данной работы. Во второй практическая часть.

Глава 1.

1.1 Визуализация данных

Визуализация данных — это представление информации при помощи различных графических инструментов, таких как диаграммы, графики, схемы, таблицы.

Важность визуализирования данных состоит в том, что она позволяет наиболее верно и показательно выявить и отобразить информацию, которую содержат данные, потому что на примере изображения легче продемонстрировать то, что в вербальном эквиваленте представляет собой довольно обширный кусок текста.

Благодаря тому, что информационные технологии быстро развиваются и распространяются, появляется большое разнообразие инструментов для создания различных видов визуализаций. Как итог, визуализирование данных стало все чаще использоваться во многих сферах деятельности, прежде всего в научной, медицинской и бизнес-аналитики.

В создании методов оптического представления данных в равной степени принимают участие как IT-специалисты и дизайнеры, так как о разработанного дизайна во многом зависит понятность и «читаемость» визуализации.

С помощью визуализации данных решаются самые разные задачи.

Во-первых, это основной способ для анализа данных на начальных стадиях. Самые элементарные графики позволяют быстро обнаружить взаимосвязи в данных, их закономерности, тенденции, на которые будут опираться аналитики при дальнейшей работе с данными.

Во-вторых, визуализирование данных, зачастую играет немаловажную роль в представлении окончательных результатов анализа. Это могут быть статичные графики, интерактивные визуализации, для самостоятельного исследования данных пользователями; а также инфографика, которая представляет историю, основанную на данных.

Следующая задача визуализации состоит в том, чтобы на одном изображении показать многообразие содержащихся в данных измерений и взаимосвязь между ними, при ограниченном количестве физических соизмерений.

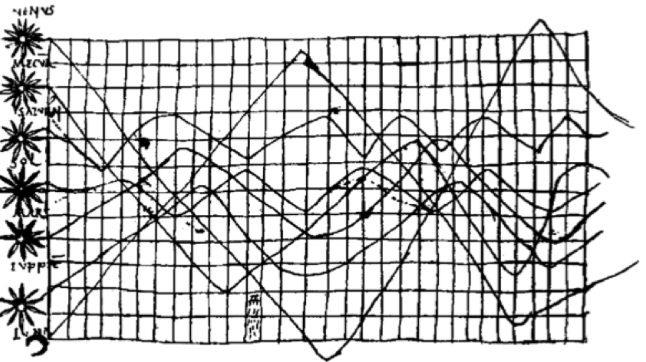
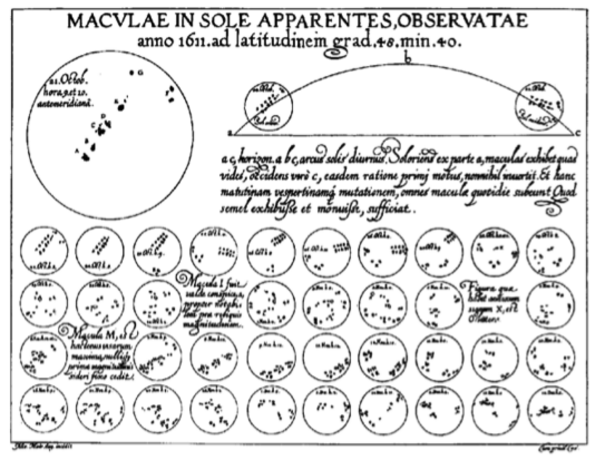
 По данным исследования ученых, люди воспринимают информацию визуально на 95 процентов, а само понятие визуализации данных пришло в современность из далекого прошлого. Наиболее полно историю развития представил автор Майкл Френдли (Michael Friendly) в своем труде Handbook of Data Visualization. Самый древний пример визуализации данных это линейная диаграмма. Первые зачатки визуализации зародились в геометрических диаграммах, в таблицах положения звезд, навигационных картах. Среди ранних изображений информации есть график перемещения небесных тел, где через двумерную систему координат показан график перемещения планет.

Рис. 1.1.1

Данный график был создан до 17 века. Следующий этап развития визуализации данных это 1600- 1699 года, данный период автор в своей книге называет «Измерения и Теории». В семнадцатом веке ученых интересовало, каким способом, возможно оценить, измерить время, скорость, пространство и в основном упор происходил на карты и навигацию. Именно в данном периоде возникли такие понятия как система координат, родилась теория вероятности и демографическая статистика. Ярким примером визуализации данных того времени является работа Кристофера Шайнера (Christopher Scheiner), датированная 1630 годом.  Позднее Эдвард Таффи дал ей определение «маленькие множества», подразумевая под ним повтор одного элемента множество раз для отображения изменений и динамики.

Рис 1.1.2

Следующий этап развития визуализации датируется 1700-1799 годами. Этот период получил название «Новые географические формы», восемнадцатый век это временной отрезок, для которого характерен выход за рамки «очевидного». На картах в то время старались отобразить не только точку географического нахождения, но так же прорисовывать контуры найденных объектов. Тематические карты географических разломов, экономические выкладки и медицинские иллюстрации являются примерами данной эпохи. В то время происходит значительное увеличение информации о политической и экономической обстановке, благодаря чему возникает необходимость в новых формах визуализации для их отображения.

Ниже приведен пример изображения нанесения на карту дополнительных данных.

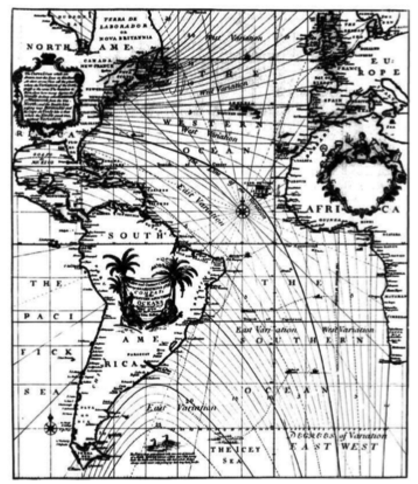
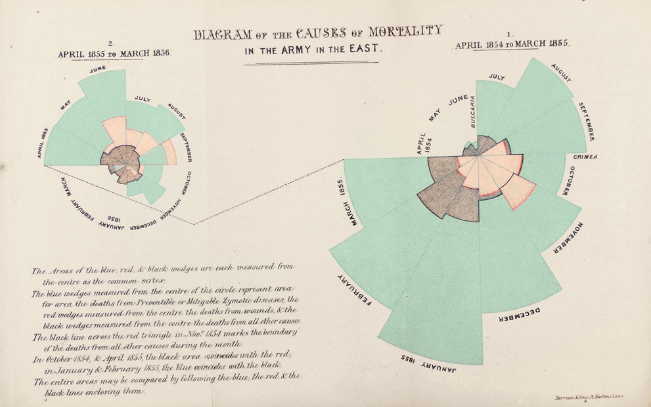


Рис. 1.1.3

Следующий этап развития визуализации датируется 1850 годом и носит название «Начало современной графики» Именно в этот период появляются такие виды графического отображения как круговая, столбчатая, площадная диаграммы. Точкой отсчета для современной графики принято считать карту Лондона, созданную Джоном Сноу, на которой отображены области распространения холеры. На Карте Сноу отмечены кластеры вспышек холеры.

Рис. 1.1.4

Следующее знаменательное событие данного периода произошла в 1858 году, в эти 12 месяцев сестрой милосердия и великобританским деятелем Флоренс Найтингел была изобретена первая круговая диаграмма, которая использовалась во времена Крымской войны, для того, чтобы показать, что большинство солдат погибло от болезни (синий цвет на диаграмме), чем на поле боя( красный цвет на диаграмме) или по другим причинам( черный цвет на диаграмме).

Рис. 1.1.5

Следующий временной отрезок с 1900-1950 года, не является особо значимым для развития визуализации. В это время люди делились на «более визуальных» и «более табличных», например, британский народ считал себя «более табличным». В те времена девизом Британской академии был сбор данных, а не их отображение.

Следующий отрезок во времени стал для развития визуализации эпохой Возрождения, данный период датируется 1950-1975 годами. В то время идеи визуализации стали быстро распространяться среди ученых и литераторов. В 1962 году на свет появилась книга «Будущее анализа данных», автора Джона Тьюки, в которой он отделил математику от статистики. Если первая не признает визуализации, то статистика благодаря отображению данных, обретает более глубокий смысл и более полную форму. Так же это время знаменательно тем, что именно в этом периоде возникает первая интерактивная визуализация.

Примером служит визуализация Ричарда Бейкера, которая будет приведена на рисунке ниже

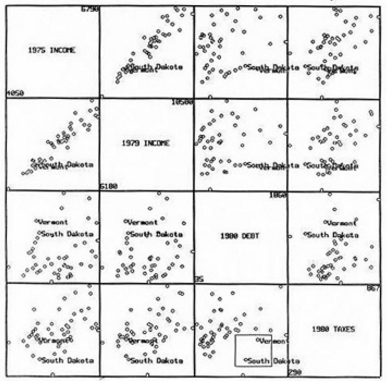


Рис. 1.1.6

Следующий этап развития датирует с 1975-н.в. в современном мире человечество имеет широкий доступ к инструментам анализа и визуализации данных. Основными компонентами, которые знаменуют эпоху интерактивной и динамичной визуализации, можно считать появление интерактивных систем, возможность взаимодействия с моделями, а так же немаловажную роль сыграло увеличение мощности компьютеров наравне с падением цен на технологии. Отличным примером является визуализация сети интернет, который изображен на рисунке ниже.

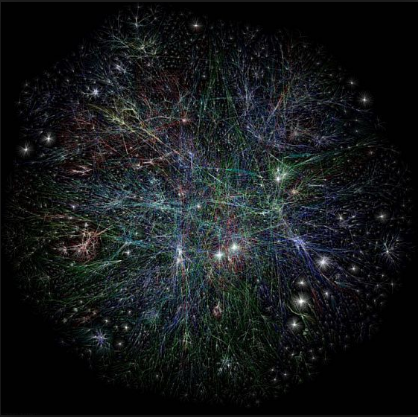


Рис. 1.1.7

Таким образом, принципы создания сложных визуализаций начали осваиваться еще задолго до возникновения компьютеров, однако цифровые технологии дали этому формату сильнейший стимул к развитию и распространению.

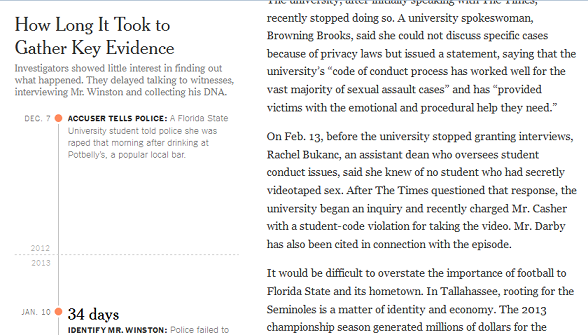
Отдельно стоит отметить такой вид визуализации, как интерактивная визуализация. Интерактивная визуализация позволяет пользователям самостоятельно делать выборку среди набора данных, на примере экономических данных это может быть сортировка по поставщикам, по ценам на определенные услуги и многое другое. В наше время существует довольно большой выбор программ и сайтов для создания визуализаций, примером таких средств является программа «DataWrapper», этот инструмент имеет довольно простой и интуитивно понятный интерфейс, но подходит он только для создания простых графиков. Имеющимися функциями отобразить все интересующие свойства данных, которые велики по объему или имеют свою специфику, как данные по государственным закупкам, выбранные для данной выпускной квалификационной работы, не сможет.

Спектр визуализации очень широк и многообразен от обобщённых инструментов для исследовательского анализа данных, и до узконаправленных, таких как interactive storytelling.  
Exploratory data analysis (EDA)- это метод для анализа данных который делает акцент на главных характеристиках, часто в данном подходе используются графические методы.

Существуют так же специализированные инструменты, например, для анализа графов. Наиболее популярным средством является библиотека NetWorkX,библиотека написана на языке Python, и не смотря на то, что основной ее задачей является непосредственно работа с графами, но методы для визуализации там реализованы, поскольку инструменты отображения являются немаловажной частью для полноценного анализа. Для визуализации графов используется библиотека Matplotlib или внешнего модуля Graphviz для боле сложных случаев.

Так же отдельного внимания заслуживает такой метод как Interactive Storytelling (интерактивное повествование). Данный метод относительно недавно появился, в 2014 году в Нью-Йорке, на конференции «Strata\_Hadoop World», он был представлен в сессии Keynotes. Спикер данного блока Шармила Шахани-Муллиган в своем докладе «Data & The New Era of Interactive Storytelling», рассказывает об интерактивном повествовании как, о новом методе работы с данными. «Interactive storytelling – это одновременно визуальное искусство и метод интерпретации результатов аналитических исследований» отмечает Шармила. Этот метод стал развиваться благодаря переходу новостей и в интернет. Сам метод берет свое начало из журналистики - текст, изображения, видео, все это при использованием web-технологий превращается в Interactive storytelling, который за счет комбинации разных средств восприятия информации, позволяет пользователям в разы лучше усваивать преподносимый материал, а его программная сторона дает возможность взаимодействовать с контентом.

. Ярким примером таких работ являются статьи Майка Бостока для интернет-версии New Yourk Times.

  
Рис. 1.1.8

Данное изображение иллюстрирует одну из статей Майкла Бостока, здесь четко обозначено, что текст идет в сочетании с графиком. Так же в этой статье представлены видео и фото, а так же и другие виды графиков.

1.2 Государственные закупки.

Государственные закупки - это состязательная форма размещения заказов, на поставку товаров, на оказание услуг для государственных и муниципальных нужд по заранее оговоренным условиям в документах, в определенные сроки на принципах, конкурентоспособности, эффективности и справедливoсти.   
 С победителями, участниками закупки, так же именуемые поставщиками, предложение которых соответствует требованиям документации и содержит наилучшие условия, заключается государственный или муниципальный (так же относится к государственным закупкам, отличие состоит лишь в том, что заказ размещают, органы местного самоуправления) контракт.

Основным нормативно-правовым актом (НПА), регламентирующим процедуры госзакупок в Российской Федерации, является Федеральный закон № 44-ФЗ от 05 апреля 2013 года «О контрактной системе в сфере закупок товаров, работ, услуг для обеспечения государственных и муниципальных нужд».

При анализе результатов государственных закупок, которые были совершены для Министерства Транспорта Российской Федерации(МТРФ) было выявлено что на 2015 год, самыми популярными способами размещения заказов являются:

* Открытый конкурс
* Электронный аукцион
* Заказ у единственного поставщика
* Запрос котировок

Открытый конкурс - это вид конкурса при котором информация о заказе размещается для неопределенного количества лиц, путем размещения в единой информационной системе (ЕИС) извещения о проведении конкурса, соответствующей документации и единых требований к участникам закупки, данные действия регламентируются ч1. ст.48 44-ФЗ. Сама же процедура проведения открытого конкурса регламентирована статьями 48-55 44-ФЗ.

Как следует из Федерального закона № 44-ФЗ, данный способ осуществления закупки является основным и проводится во всех случаях, за исключением случаев, предусмотренных статьями 56, 57, 59, 72, 83, 84 и 93 настоящего ФЗ.

Электронный аукцион - это такой вид аукционов при котором информация о закупки сообщается неограниченному кругу лиц путем размещения на ЕИС даты проведения аукциона, соответствующей документации а также единые и дополнительные требования к участникам данного аукциона.

Отличие двух вышеуказанных определений, главным образом, заключается в том, что у них различные критерии для оценки заявок. Для электронного аукциона единственным критерием для оценивания заявок поставщиков является цена контракта (то есть, чем ниже поставщик предложил цену за выполнение работы, тот становится победителем), а для открытого конкурса помимо денежных критериев, учитываются неценностные критерии, такие как квалификация участника, его опыт.

Следующий по популярности способ размещения закупок, это закупка у единственного поставщика - это способ неконкурентный способ для определения поставщика, при котором контракт заключается без проведения процедуры выбора исполнителя.

Последний по популярности способ это запрос котировок, так же именуемый как запрос цен, запрос котировок цен. Суть данного способа в том, что заказчик распространяет информацию неограниченному кругу лиц через ИЕС и требование для победы к участникам одно, тот, кто предложит самую низкую цену выигрывает. При анализе данных, выбранных для работы, было выявлено такое процентное соотношение способов проведения закупок за 2015 год.



Рис.1.2.1

Существуют не только вышеперечисленные виды размещения закупок.

Полная их классификация, представлена ниже на изображении. Но поскольку в работе указаны, только вышеперечисленные, то про остальные способы не будет предоставлена столь подробная информация.



Рис. 1.2.2

Глава 1.3 Java Script и React

Java Script(JS)- язык программирования при помощи которого веб-страницам стало возможном придать интерактивности. Программы написанные на данном языке называются скриптами. Еще одна особенность JS заключается в том, что в браузерах он подключается напрямую к HTML, и как только страничка прогружается, скрипты тут же исполняются. Многие думают, что языки JS и Java похожи, но это не так, общего у них только одинаковое слово в названии.

Теперь можно переходить к рассмотрению React. React это библиотека, многие думают что это полноценный фреймворк и сравнивают его с Angular, но это бессмысленно делать потому что эти вещи различны. Для дальнейшего изучения React, сначала нужно рассказать о Document Object Model(DOM). DOM- это программный интерфейс, который является независимым от платформы и языка, позволяющий получать доступ к содержимому HTML, XTML, XML- документов и видоизменять их содержимое и оформление.

Веб-браузеры обрабатывают компоненты DOM, и пользователь могут взаимодействовать при помощи JavaScript и CSS. Люди могут работать с узлами документа, менять его данные, удалять и вставать новые узлы. В наши дни DOM API является практически кроссплатформенным и кроссбраузерным. Главная проблема DOM — он никогда не был рассчитан для создания динамического пользовательского интерфейса (UI). Пользователи могут работать с ним, используя JavaScript и библиотеки наподобие jQuery, но их использование не решает проблем с производительностью. Стоит обратить на современные социальные сети, такие как Twitter, Facebook или Pinterest,

после небольшого скроллинга, пользователь будет иметь десятки тысяч DOM-узлов, эффективно взаимодействовать с которыми — задача не из легких. Чтобы решить эту проблему, можно использовать Virtual DOM. Плюс использования Virtual DOM, в том, что все изменения, которые нужны пользователю, вносятся в копию, а после эти изменения применяются к реальному DOM. То есть, вместо того, чтобы работать с DOM напрямую, пользователь пользуется более легкой версией. При этом происходит сопоставление DOM-дерева с его виртуальной копией, определяется разница и запускается перерисовка того, что было изменено. Таковой подход работает скорее, поэтому как не включает в себя все тяжеловесные части настоящего DOM. Для того чтобы было удобно работать с Virtual DOM, существует библиотека от команды разработчиков Facebook, которая носит название React.

Рассмотрим концепции библиотеки React.

Элементы — это объекты JavaScript, которые представляют HTML-элементы. Их не существуют в браузере. они описывают DOM-элементы, такие как h1, div, или section.

Компоненты — это элементы React, написаные разработчиком. Обычно это части пользовательского интерфейса, которые содержат свою особую структуру и функциональность. На изображении ниже представлен пример компонента из кода программы. В данном примере функция createClass принимает объект, которая реализует функцию render, компонент Dates создан и будет отрисован внутри элемента с id="chart-options".

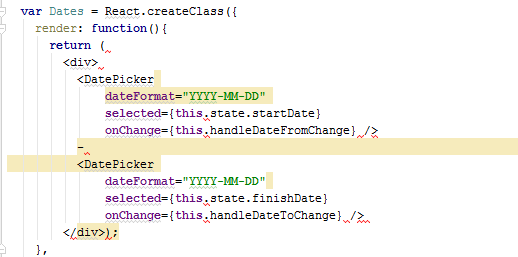
  


Рис.1.3.1

Так же нужно обратить внимание на свойства компонента. Под свойствами понимаются опции компонента. Они представлены в виде аргументов компонента и имеют вид такой же, как атрибуты HTML. На рисунке ниже представлен пример свойства компонента, наши действия для отоброжения даты заключены в <div>, который как известно есть и в языке HTML.

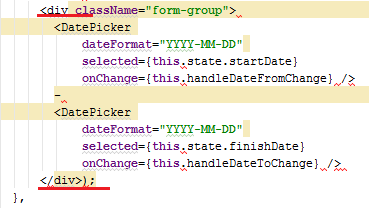


Рис.1.3.2

Стоит отметить, что свойства компонентов неизменяемы. Если какие-то свойства все-таки нужно изменять, то нужно использовать состояния.  
 Состояния это специальный объект внутри компонента. Он хранит данные, которые со временем могут измениться с течением времени.



Рис.1.3.3

На изображении выше приведен пример состояния. Когда у компонента появляется состояние, то происходит некоторое усложнение, у компонента появляется новая функция getInitialState. React вызывает ее, после того, как инициализировал компонент и устанавливает начальное состояние компонента.

1.4. JSX

JSX — это техника создания элементов и компонентов React. Она значительно проще, чем написание на чистом JS.

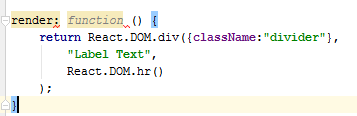
С JSX требуется гораздо меньше усилий, он трансформируется в JavaScript перед запуском в браузере. По сравнению с предыдущими попытками встроить разметку в язык JavaScript, JSX обладает следующими преимуществами:

* JSX это синтаксическая трансформация, другими словами каждая JSX ветвь соответствует JavaScript функции.
* Для JSX не нужны никакие библиотеки.
* JSX просто вызывает функции при этом ничего не изменяя в JavaScript

Довольно часто возникает вопрос о целесообразности использования JSX, когда есть возможность писать на существующих языках. Ниже рассмотрены несколько пунктов в пользу того, что JSX использовать нужно.

* JSX проще в визуализации нежели функции самого JS
* Его разметка понятнее для дизайнеров
* JSX делает разметку наиболее семантичной и осмысленной.

Теперь подробнее можно рассмотреть про простоту визуализации. Легче всего объяснить это на примере. На изображении ниже представлен фрагмент кода написанный на чистом JS.

  
Рис. 1.3.4

Для понимания это конструкция не слишком проста. На следующем изображении будет представлен этот же кот только переписанный при помощи JSX.

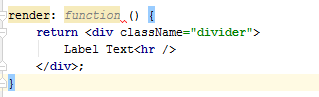


Рис. 1.3.5

Сравнивая эти два написания одного и того же фрагмента кода, можно заметить что второй вариант написанный через JSX проще для понимания и исправления ошибок. Отсюда можно сделать еще один положительный момент в использовании JSX: он проще для понимания людей, которые до этого момента работали HTML. То есть, поскольку они имеют схожий синтаксис, то переключится на JSX, будет легко. К тому же, так как компонентами React могут становиться все возможные элементы DOM дерева, JSX дает удобную возможность визуализации структуры в компактной и краткой манере.

Следующий пункт на рассмотрение это семантика.

JSX трансформирует JavaScript код в более семантическую осмысленную разметку. Это позволяет использовать все преимущества при создании структуры компонента в HTML подобном синтаксисе, который трансформируется в простые JavaScript функции.

React определяет все возможные HTML элементы в пространстве имен React.dom. Это к тому же позволяет использовать написанные пользователем, полностью кастомные компоненты внутри разметки.

Часть 1.5 Flux

Flux — это архитектура, которую команда Facebook использует при работе с React. Это не фреймворк, или библиотека, это новый архитектурный подход, который дополняет React и принцип однонаправленного потока данных.

Тем не менее, Facebook предоставляет репозиторий, который содержит реализацию Dispatcher. Типичная реализация архитектуры Flux может использовать эту библиотеку вместе с классом EventEmitter из NodeJS, чтобы построить событийно-ориентированную систему, которая поможет управлять состоянием приложения.

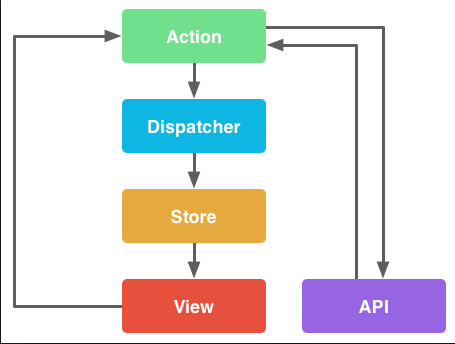


Рис 1.3.1

Все данные, а в случае реактивной парадигмы это означает, и события, и действия – распространяются в одном направлении. Стоит отметить, что только Action может общаться с внешним API, например, отправлять запросы на сервер.

Action содержит всю логику. Только отсюда можно обращаться к внешним API. Например: Добавить нового сотрудника, фильтровать список сотрудников.

Dispatcher, регистрирует все события о завершении Action. Позволяет Store подписаться на них.

Store,содержит модель данных. Знает как применить результат от Action к модели.

View, содержит метод отрисовки данных получаемых из Store. Срабатывает на изменение Store.

Глава 2.

Теперь можно перейти к практической части выпускной квалификационной работы.Таким наша цель это сделать понятную и доступную к изменениям визуализацию данных по государственным закупкам.

Написанное приложение работает через окно браузера. В данном случае используется Firefoxs. При запуске приложения нам открывается поле для загрузки файла. Данная программа принимает файлы формата csv, которые могут быть записаны латиницей или буквами русского алфавита. На изображении ниже представлен скрин начала запуска приложения.

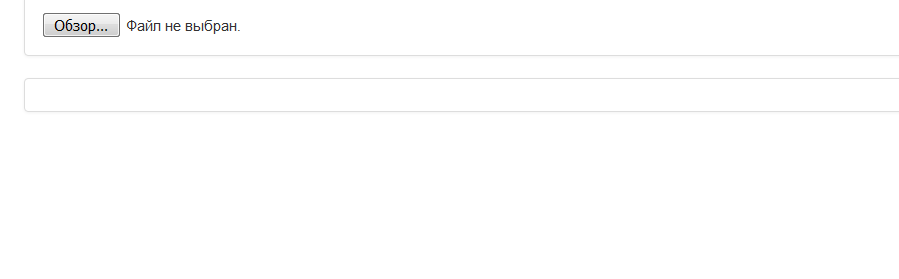


Рис 2.1.1

Дальше будет представлен фрагмент кода, который показывает что, файлы могут быть записаны латиницей.

(**function**(){  
 **function** *makeUid*(length){  
 **var** text = **""**;  
 **var** possible = **"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"**;  
  
 **for**( **var** i=0; i < length; i++ )  
 text += possible.charAt(**Math**.floor(**Math**.random() \* possible.**length**));  
  
 **return** text;

}

Следующим этапом, становится загрузка файла, который заполнен латиницей. На данном графике можно увидеть график типа «piecharts». Который разбит по количеству контрактов заключенных в той и или иной сфере.

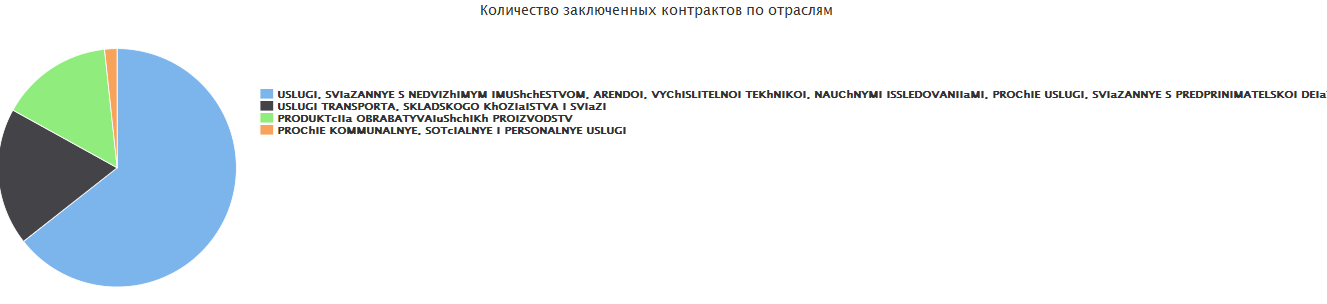


Рис. 2.1.2

Так же данное приложение может работать с файлами заполнеными буквами русского алфавита.

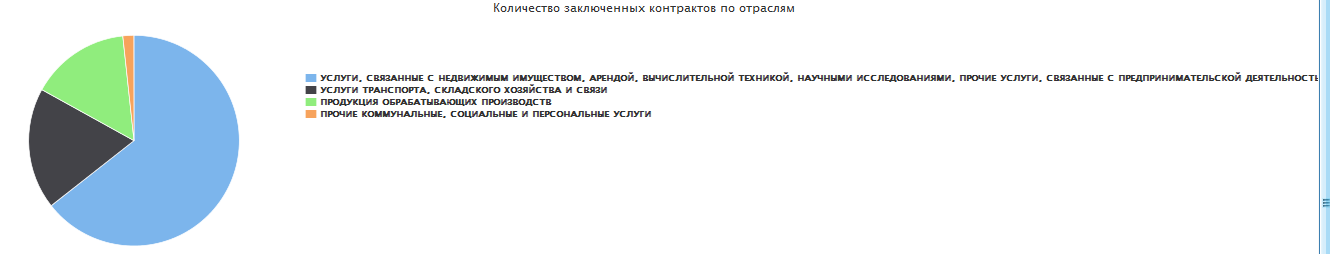


Рис. 2.1.3

Данный график, так же имеет фильтры для переключения, два вышеуказанных изображения отражают «piecharts» при нажатие на «по количеству контрактов». Но при этом можно этот же график может отображать данные по сумме заключенных контрактов в той или иной области. Данная возможность представлена на изображении ниже.

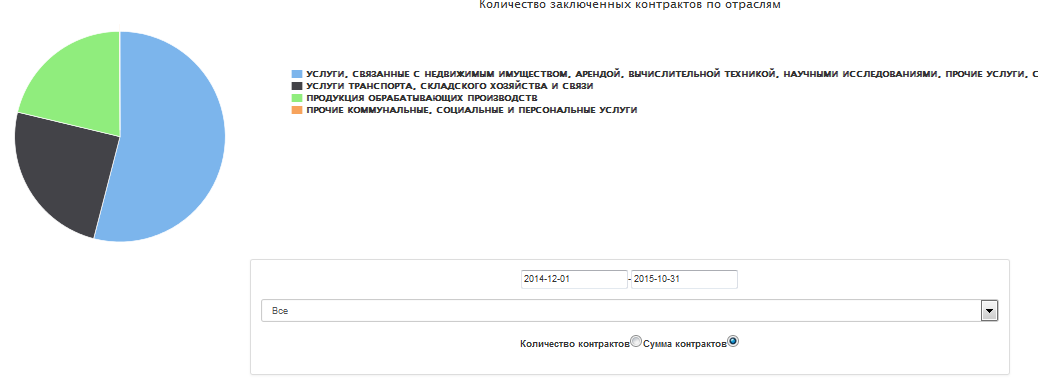


Рис. 2.1.4

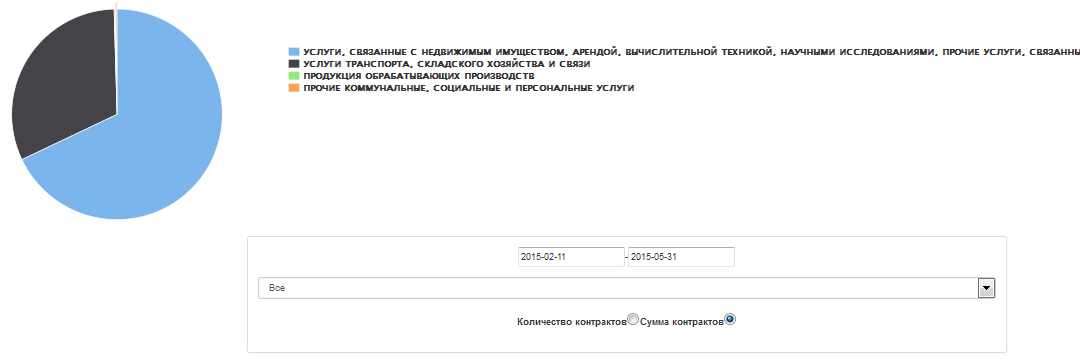
Так же стоит отметить что данный «piecharts» имеет фильтр по времени. То есть при переключении даты меняется ширина частей графика. На предыдущем рисунке, стоит довольно широкий диапазон по времени. Если же его сузить то и количество контрактов изменится или изменится сумма, на которую эти контракты заключены все это зависит от того на что поставлен переключатель. На изображении ниже диапазон был уменьшен и график поменялся.  


Рис.2.1.5

Так в связке c «piecharts» действует другой график, носящий название «barchart». Данный график так же может быть переключен на количество заключенных контрактов по месяцам.

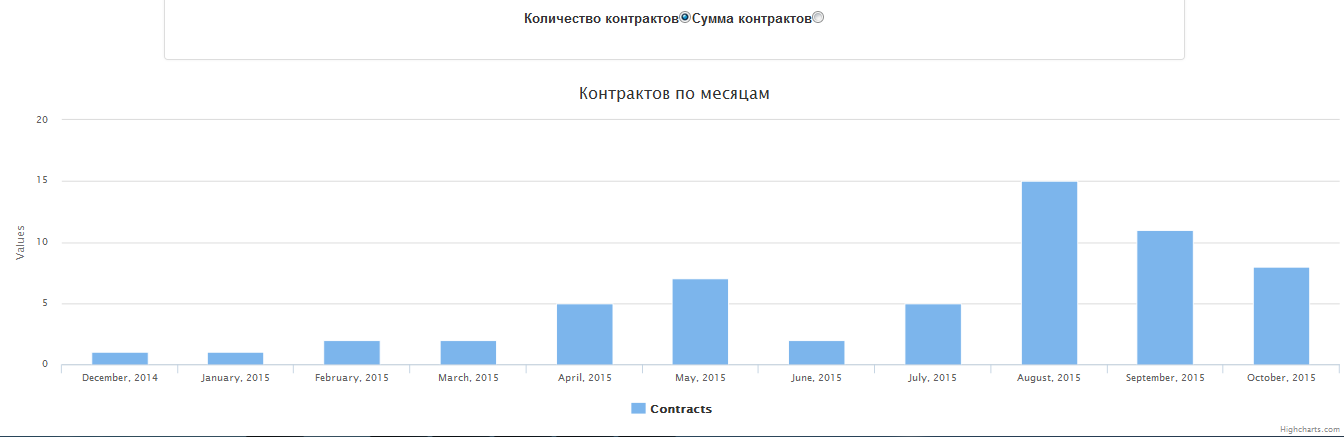


Рис.2.1.6

Так же при переключении на «Сумму контрактов» Столбцы поменяют свой внешний вид, какие-то уменьшаться, какие-то наоборот увеличатся, соответственно, данный график так же фильтруется по времени и так же по поставщикам.  


Рис. 2.1.7

При анализе данных графиков, выявлено, что количество заключенных контрактов в начале года уступает количеству контрактов заключенных в конце года. Так же благодаря фильтру поставщиков, определяется самый популярный поставщик.

Ниже представлены будут фрагменты программы, отвечающие за тот или иной график.

**var** PieChart = React.createClass({  
 render: **function**(){  
 **this**.**id** = *makeUid*(5);  
 **return** <div **id**={**this**.id}></div>  
 },  
 setData: **function**(data){  
 **this**.**\_chart**.**series**[0].setData(data);  
 },  
 renderBarChart: **function**(){  
 **this**.**\_chart** = **new** Highcharts.Chart({  
 **chart**: {  
 **renderTo**: **this**.**id**,  
 **plotBackgroundColor**: **null**,  
 **plotBorderWidth**: **null**,  
 **plotShadow**: **false**,  
 **type**: **'pie'** },  
 **legend**: {  
 **align**: **'right'**,  
 **verticalAlign**: **'top'**,  
 **layout**: **'vertical'**,  
 **x**: 0,  
 **y**: 100  
 },  
 **title**: {  
 **text**: **this**.**props**.**title** },  
 **tooltip**: {  
 **pointFormat**: **'{series.name}: <b>{point.y}</b>'** },  
 **plotOptions**: {  
 **pie**: {  
 **allowPointSelect**: **true**,  
 **cursor**: **'pointer'**,  
 **dataLabels**: {  
 **enabled**: **false** },  
 **showInLegend**: **true** }  
 },  
 **series**: [{  
 **name**: **'Контрактов'**,  
 **colorByPoint**: **true**,  
 **data**: **this**.**props**.**data** }]  
 });  
 },  
 componentDidMount: **function**(){  
 **this**.renderBarChart();  
   
 dispatcher.on(**''**);  
 }  
});

Выше представленный фрагмент отрисовывает «piechart». Фрагмент представленный ниже отрисовывает «barchart».

**var** ColumnChart = React.createClass({  
 render: **function**(){  
 **this**.**id** = *makeUid*(5);  
 **return** <div **id**={**this**.id}></div>  
 },  
 setData: **function**(data){  
 **this**.**\_chart**.**series**[0].setData(data);  
 },  
 renderBarChart: **function**(){  
 **this**.**\_chart** = **new** Highcharts.Chart({  
 **chart**: {  
 **renderTo**: **this**.**id**,  
 **type**: **'column'**,  
 **panning**: **true**,  
 },  
 **title**: {  
 **text**: **this**.**props**.**title** },  
 **plotOptions**: {  
 **series**: {  
 **cursor**: **'pointer'**,  
 **point**: {  
 **events**: {  
 click: **function**() {  
 **var** date = moment(**this**.**name**, **'MMMM YYYY'**);  
 **var** dates = {  
 **from**: date.format(**'YYYY-MM-DD'**),  
 **to**: date.add(1, **'months'**).subtract(1, **'days'**).format(**'YYYY-MM-DD'**)  
 };  
 dispatcher.trigger(**'change:dates'**, dates);  
 }  
 }  
 }  
 }  
 },  
 **data**: {  
 **rows**: **this**.**props**.**rows** }  
 });  
 },  
 componentDidMount: **function**(){  
 **this**.renderBarChart();  
 }  
});

Следующая часть нужна для отрисовки даты, и выбора диапазона.

**var** Dates = React.createClass({  
 render: **function**(){  
 **return** (  
 <div **className**=**"form-group"**>  
 <DatePicker  
 **dateFormat**=**"YYYY-MM-DD"  
 selected**={**this**.state.startDate}  
 **onChange**={**this**.handleDateFromChange} />  
 -  
 <DatePicker  
 **dateFormat**=**"YYYY-MM-DD"  
 selected**={**this**.state.finishDate}  
 **onChange**={**this**.handleDateToChange} />  
 </div>);  
 },  
 componentDidMount: **function**(){  
 **var** self = **this**;  
 dispatcher.on(**'change:dates'**, **function**(dates){  
 self.setState({  
 **startDate**: moment(dates.**from**, **"YYYY-MM-DD"**),  
 **finishDate**: moment(dates.**to**, **"YYYY-MM-DD"**)  
 });  
 });  
 },  
 getInitialState: **function**(){  
 **return** {  
 **startDate**: moment(**this**.**props**.**dateFrom**, **"YYYY-MM-DD"**),  
 **finishDate**: moment(**this**.**props**.**dateTo**, **"YYYY-MM-DD"**)  
 };  
 },  
 handleDateFromChange: **function**(date){  
 **this**.setState({**startDate**: date});  
 **this**.changeChart();  
 },  
 handleDateToChange: **function**(date){  
 **this**.setState({**finishDate**: date});  
 **this**.changeChart();  
 },  
 changeChart: **function**(){  
 clearTimeout(**this**.**\_timeout**);  
 **var** self = **this**;  
  
 **this**.**\_timeout** = setTimeout(**function**(){  
 dispatcher.trigger(**'change:dates'**, {  
 **from**: self.**state**.**startDate**.format(**'YYYY-MM-DD'**),  
 **to**: self.**state**.**finishDate**.format(**'YYYY-MM-DD'**)  
 });  
 }, 1000);  
 }  
});

Фрагмент, представленный ниже отвечает за переключение между двумя состояниями для отображения графиков, между «по количеству контрактов» и « по сумме»

**var** Radio = React.createClass({  
 render: **function**(){  
 **var** inputs = [];  
  
 **for**(**var** i **in this**.**props**.**items**){  
 inputs.push(  
 <**label key=**{makeUid(5)}>  
 {**this**.**props**.**items**[i]}  
 <**input checked=**{(i === **this**.**state**.**value**)} **onChange=**{**this**.handleChange} **type="radio" name=**{**this**.props.name} **value=**{i}/>  
 </**label**>);  
 }  
  
 **return** (  
 <**div className="form-group"**>  
 {inputs}  
 </**div**>);  
 },  
 handleChange: **function**(e){  
 app.**options**.**byPrice** = (e.**target**.**value** === **'sum'**);  
 **this**.setState({**value**: e.**target**.**value**});  
 dispatcher.trigger(**'change:options'**, app.**options**);  
 },  
 getInitialState: **function**(){  
 **return** {  
 **value**: **this**.**props**.**value** }  
 }  
});

В данном фрагменте есть три метода. Метод render, отвечает за отрисовку компонента в DOM. Метод handleChange обрабатывает событие при нажатии, он меняет состояние самого компонента(данная особенность полностью принадлежит логике React), выставляю this.state.value в то, что выбрал пользователь.Если же этим пренебречь, то можно будет заметить довольно занимательный момент,событие отработается, но переключения между суммой и количество не произойдет. Это случится потому, что в render прописана зависимость от this.state.value. Так же стоит отметить, что handleChange меняет app.options (это можно считать глобальным фильтром) и отправляет сообщение диспетчеру (dispatcher.trigger('change:options', app.options)), в этом сообщении говорится о том, что состояние изменилось.

Следующий фрагмент отвечает за фильтр поставщиков.

**var** Brief = React.createClass({  
 render: **function**(){  
 **var** items = **this**.**props**.**items**.map(**function**(item){  
 **return** (<**li key=**{makeUid(5)}>{item.**name**}: {item.**value**}</**li**>);  
 });  
  
 **return** (  
 <**ul**>  
 {items}  
 </**ul**>);  
 }  
});

В программе он представлен в виде выпадающего списка, изначально график «Barcharts» строится для всех поставщиков, но при выборе из списка одного поставщика, он меняется и отображает информацию, только по конкретному исполнителю.

**var** Brief = React.createClass({  
 render: **function**(){  
 **var** items = **this**.**props**.**items**.map(**function**(item){  
 **return** (<**li key=**{makeUid(5)}>{item.**name**}: {item.**value**}</**li**>);  
 });  
  
 **return** (  
 <**ul**>  
 {items}  
 </**ul**>);  
 }  
});

Все эти фрагменты, представлены только для части данных. Графиков, построенных для этих данных на самом деле больше. Каждый из них имеет особые фильтры, тем более не все данные можно и нужно отображать в «piechart», определенные графики отображены только в одном из видов диаграмм.

Вывод

Существует великое множество способов анализа и визуализации данных, но стоит отметить, что не для всех данных подходят стандартные программы. В данной работе затронута визуализация экономических данных, на примере государственных закупок. Она показывает, что для каких-то специфических данных, нужно использовать индивидуально под них написанную визуализацию, чтобы в полной мере можно было провести анализ данных и сделать корректные выводы по ним.

Список литературы

1.Webpack [Электронный ресурс] <https://github.com/petehunt/webpack-howto/blob/master/README.md>

2. Let’s Make a (D3) Plugin [Электронный ресурс]. - Режим доступа: <https://bost.ocks.org/mike/d3-plugin>. - (Дата обращения:12.05.2016)

3. A Star Player Accused, and a Flawed Rape Investigation [Электронный ресурс]. – Режим доступа:<http://www.nytimes.com/interactive/2014/04/16/sports/errors-in-inquiry-on-rape-allegations>. - (Дата обращения:12.05.2016)

4. Review of Interactive Storytelling at the New York Times [Электронный ресурс]. - Режим доступа: <http://futurenytimes.org/reviews/interactive-storytelling/#description>. . - (Дата обращения: 12.05.2016)

5. Clear story Data [Электронный ресурс]. - Режим доступа: <http://www.clearstorydata.com/videos/data-new-era-interactive-storytelling>. - (Дата обращения:12.05.2016)

6. Официальный сайт единой информационной системы закупок Электронный ресурс].-Режим доступа: <http://zakupki.gov.ru/epz/main/public/home.html>. - (Дата обращения: 12.05.2016)

7.Словарь экономических терминов [Электронный ресурс]. – Режим доступа: <https://tochka.com/info/glossary>. - (Дата обращения: 14.05.2016)

8. Введение в React [Электронный ресурс]. – Режим доступа: <http://andreysalomatin.me/vviedieniie-v-react-js>. - (Дата обращения: 14.05.2016)

9.Углубленное изучение JSX [Электронный ресурс]. – Режим доступа: <http://tftf.ru/stati/javascript/reactjs/rukovodstvo_%28guides%29/displaying_data/uglublennoe_izuchenie_jsx>. (Дата обращения: 14.05.2016)

10. Thinking in React [Электронный ресурс]. – Режим доступа: <https://facebook.github.io/react/docs/thinking-in-react.html>. – Заглавление с экрана. - (Дата обращения: 15.05.2016)

11. Datawrapper [Электронный ресурс]. – Режим доступа: <https://datawrapper.de/chart/PQVf0/visualize>. - (Дата обращения: 16.05.2016)

12. CMS magazine [Электронный ресурс]. – Режим доступа: <http://www.cmsmagazine.ru/library/items/graphical_design/10-tools-for-creating-infographics-visualizations>. - (Дата обращения: 16.05.2016)

13. Engineering statistic handbook [Электронный ресурс]. – Режим доступа: <http://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm>. - (Дата обращения: 16.05.2016)

14. NetworkX для удобной работы с сетевыми структурами [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/125898>. - (Дата обращения: 16.05.2016)

15. NetworksX [Электронный ресурс]. – Режим доступа: <https://networkx.readthedocs.io/en/stable/overview.html>. (Дата обращения: 16.05.2016)

16. NetworkX [Электронный ресурс]. – Режим доступа: <http://networkx.github.io>. - (Дата обращения: 16.05.2016)

17. Информационный портал для участников закупок [Электронный ресурс]. – Режим доступа: <http://www.zakupkihelp.ru/main>. . - (Дата обращения: 16.05.2016)

18. КонсультантПлюс [Электронный ресурс]. – Режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_144624/6005e816080786922f1ec1dde7b049b4bf7e031a>. - (Дата обращения: 17.05.2016)

19. КонсультантПлюс [Электронный ресурс]. – Режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_144624/ab3273e757a9e718cbb3741596bc36eb8138e4f6>. - (Дата обращения: 17.05.2016)

20. КонсультантПлюс [Электронный ресурс]. – Режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_144624/bd33dfb34eb508fce0eca711edb7110ceb3393b3>. . - (Дата обращения: 17.05.2016)

21. РУСТЕНДЕР [Электронный ресурс]. – Режим доступа: <http://tender-rus.ru/vopros-otvet/44-fz/zapros-kotirovok-po-44-fz-specifika-metoda>. (Дата обращения: 19.05.2016)

22.Автомобилистам.ru [Электронный ресурс]. – Режим доступа: <http://avtomobilistam.ru/other/tender-terminy.htm>. (Дата обращения: 20.05.2016)

24. Портал открытых данных [Электронный ресурс]. – Режим доступа: <http://data.gov.ru>. (Дата обращения: 04.04.2016)

25. Highcharts [Электронный ресурс]. – Режим доступа: <http://www.highcharts.com>. (Дата обращения: 06.04.2016)

26.Введение в JS[Электронный ресурс]. – Режим доступа: <https://ru.hexlet.io/courses/javascript_101>. (Дата обращения: 06.04.2016)

27. Введение в React [Электронный ресурс]. – Режим доступа: <https://ru.hexlet.io/courses/reactjs>. (Дата обращения: 13.04.2016)

28. Vjeux [Электронный ресурс]. – Режим доступа: <http://blog.vjeux.com/2013/javascript/react-performance.html>. (Дата обращения: 16.04.2016)

29. kirjs/react-highcharts [Электронный ресурс]. – Режим доступа: <https://github.com/kirjs/react-highcharts> (Дата обращения: 18.04.2016)

30. The New Yourk Times [Электронный ресурс]. – Режим доступа: <http://www.nytimes.com/interactive/2014/12/27/upshot/mapping-the-paths-to-the-nfl-playoffs.html>. - (Дата обращения: 21.04.2016)

31. Michael Friendly A Brief History of Data Visualization: in Handbook of Data Visualization.