Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 519.638.8

Телевич
Павел Леонидович

Использование Javascript технологии как средства для оптимизации серверных мощностей в Web-приложениях

АВТОРЕФЕРАТ
на соискание степени магистра технических наук
по специальности 1-40 81 01 «Информатика и технологии разработки программного обеспечения»

_____

Научный руководитель
Новиков Владимир Иванович
Доцент, к.т.н.

_____

Минск, 2017

Общая характеристика работы

Актуальность проблемы.Постоянное увеличение сложности создания современных Web-приложений выдвигает все более жесткие требования к системам их проектирования и к серверным мощностям. При этом, на сегодняшний день стало очевидным, что невозможно добиться существенного повышения производительности только за счет увеличения мощности сервера.

Более эффективным решением является оптимизация использования имеющихся инструментов, методов и ресурсов. Сектор Web-приложений является одной из тех областей, в которой остро ощущается нехватка серверных мощностей. Одним из наиболее перспективных подходов к обработке сложных задач вышеуказанного профиля и уменьшению нагрузок на сервер является применение систем асинхронных запросов.

В настоящее время асинхронность как подход получает все большее распространение, который характеризуются доступностью, высокой масштабируемостью, одним из лучших соотношений цена/производительность и являются совокупностью методов, объединенных определенной средой.

В век активно меняющихся технологий, разрабатывающихся и развивающихся новых приложений актуальность применения JavaScript заключается в их высоком потенциале применения для оптимизации серверных мощностей в Web-приложениях. Особым преимуществом использования JavaScript при оптимизации серверных мощностей является возможность его выполнения не только в браузере.

Цель работы состоит в следующем. На основании изучения использования JavaScript технологий, как средства для оптимизации серверных мощностей в Web-приложениях, представить наиболее перспективные алгоритмы применения JavaScript.

Задачи исследования:

1) Рассмотреть основные характеристики JavaScript технологий.

2) Проанализировать применение JavaScript технологий в оптимизации серверных мощностей в Web-приложениях.

3) Предложить пути, направленные на совершенствование применения JavaScript в процессе оптимизации серверных мощностей в Web-приложениях.

Научная новизна.Научная новизна работы состоит в получении характеристик, описывающие процессы, происходящие внутри среды исполнения Java script, а также предложение по оптимизации серверных мощностей в Web-приложениях.

Методы исследования.Методом исследования программных платформ является эксперимент, а также анализ исходных кодов программных платформ.

Достоверность научных положений и практических рекомендаций, полученных в диссертации, подтверждается корректным обоснованием постановок задач, точной формулировкой критериев, а также результатами внедрения предложенной технологии.

Практическое значение. JavaScript позволяет быть веб-страницам более динамичными, не перегружая при этом серверы. В настоящее время JavaScript предоставляет возможность для разработчиков создавать мощные приложения, которые ранее можно было написать только для определенной платформы. В то же время благодаря использованию JavaScript разработчики получают возможность создания целых приложений в браузере, что позволяет одностраничным приложениям сиюминутно реагировать на каждое действие пользователя. В силу данных обстоятельств JavaScript можно рассматривать как технологию, имеющую огромный потенциал в оптимизации серверных мощностей в Web-приложениях.

Структура работы. Данная работа состоит из введения, трех глав основной части. Каждая из которых разделена на подглавы, заключения, списка использованных источников приложения.

Содержание работы

Во введении обосновывается актуальность темы диссертационной работы. Ставятся цель и задачи исследования, решения которых выносятся на защиту.

В первой главе проведен обзор основных характеристик JavaScript технологий.

Рассмотрено понятие JavaScript технологий и их использование. Основное различие между языками заключается в том, что JavaScript является интерпретируемым, а Java – компилируемым. Имеется в виду, что код, созданный с помощью языка JavaScript, выполняется сразу же, без компиляции, тогда как код Java сначала должен пройти преобразование в байт-код, который будет в дальнейшем выполнен виртуальной машиной. Различия также заключаются в видах типизации – в JavaScript она динамическая, а в Java – статическая. Под динамической типизацией понимается процесс, при котором в различных участках программы переменная может принимать значения разных типов: если при объявлении переменной определить её как строковую, то по ходу выполнения ей может

быть присвоено числовое значение. При этом JavaScript интерпретирует подобный код безошибочно, а компилятор Java в этом случае выдаст ошибку.

Более того, JavaScript является прототипно объектно-ориентированным, а Java – классово объектно-ориентированным. В JavaScript основным элементом будут являться, таким образом, объекты-прототипы, объекты-прототипы, которые изменяются или могут меняться в ходе выполнения программы и могут быть клонированы с целью создания иных объектов, что и представляет собой принцип прототипного программирования. В рамках Java возможно выделить два ключевых понятия, класса и объекта. Перед тем как использовать объект, его следует описать в виде класса.

На данный момент JavaScript –это наиболее популярный клиентский язык программирования. Программы на данном языке называют скриптами; в браузере они подключаются напрямую к HTML и выполняются сразу же, как только загружается страница. Программы на языке JavaScript представляют собой, по сущности, обычный текст, написание которого не требует определенной продолжительной специальной подготовки.

JavaScript может выполняться не только в браузере, а и в других средах; для этого требуется особая программа – интерпретатор, тогда как сам процесс выполнения скрипта называется интерпретацией. Это означает, что исходный код созданной программы получает другой инструмент, называемый «интерпретатором», и выполняет его.

JavaScript постепенно становится всё более и более мощным, а возможности браузера растут в сторону десктопных приложений. Новая спецификация JavaScript ECMAScript 7 является значительным шагом вперёд в сторону улучшения синтаксиса языка. Браузеры нового поколения совершенствуют свои движки с целью увеличения скорости исполнения JavaScript, исправления багов и стараются следовать стандартам. JavaScript постепенно становится всё быстрее и стабильнее, а в язык добавляют новые возможности. Современные стандарты HTML5 и ECMAScript позволяют сохранить максимальную совместимость с предыдущими версиями, что, в свою очередь, позволяет избежать ошибок при работе с уже существующими приложениями.

Язык JavaScript является уникальным благодаря своей полной интеграции с HTML/CSS. Flash, Java и браузерные расширения обладают своими уникальными возможностями, которые можно использовать в сочетании с JavaScript. Полезны могут быть и надстройки для JavaScript – CoffeeScript, TypeScript и других языков.

Далее описаны свойства и методы структур данных в JavaScript.В JavaScript, в отличие от большинства языков, отсутствует концепция ввода (input) и вывода (output). Этот язык разработан таким образом, чтобы

запускаться в качестве языка сценариев, встроенного в среду исполнения. Наиболее популярная среда исполнения языка – это браузер, но интерпретаторы JavaScript представлены и в Adobe Acrobat, Photoshop, Widget engine (Yahoo!), и даже в серверном окружении, к примеру, node.js.

JavaScript представляет собой объектно-ориентированный язык, который имеет типы и операторы, а также встроенные объекты и методы. Очевидно, что синтаксис этого языка унаследован от других языков Java и C, множество готовых конструкций этих языков применяются и к JavaScript.

Ключевое отличие JavaScript от них – отсутствие классов, функциональность которых осуществляется посредством прототипов объектов. Еще одним важным отличием является то, что функции представляют собой объекты: в их рамках содержится исполняемый код.

В JavaScript существует целый свойств и методов структур данных, включая типы данных: [[Class]], instanceof и «утиная типизация», формат JSON, метод toJSON, setTimeout и setInterval, запуск кода из строки: eval, перехват ошибок «try..catch» и множество других. В настоящее время идет активная разработка инструментов и надстроек для JavaScript, которые помогут усовершенствовать данный язык и избежать множество ошибок, оптимизируя работу с кодом в целом.

В работе описаны особенности использования асинхронных запросов в JavaScript. Главная особенность AJAX состоит в том, что при его использовании не нужно обновлять всю страницу если необходимо обновить только конкретную часть. А асинхронность позволяет браузеру после отсылки запроса делать что угодно, прокрутить страницу, вывести сообщение об ожидании ответа.

Примерный алгоритм работы AJAX выглядит следующим образом:

1) Пользователь нажимает на элемент на веб-странице

2) Написанный на языке JavaScript скрипт находит нужную информацию необходимую для обновления страницы.

3) Браузер отправляет серверу запрос.

4) Сервер возвращает браузеру лишь ту часть документа которая была запрошена.

5) Скрипт меняет страницу исходя из полученной от сервера информации (не перезагружая всю страницу) [28, с. 91].

Исходя из всего вышесказанного можно выделить основные преимущества AJAX:

1) Дает возможность создать удобный web-интерфейс.
2) Взаимодействие с пользователем в «реальном времени».
3) Частичное обновление страницы вместо полного.
4) Простота и удобство при использовании.

Написанная на JavaScript клиентская часть должна обеспечить необходимый функционал для безопасного обмена данных с сервером и предоставить различные методы для обмена данными. Серверная часть, в свою очередь, должна корректно принимать и обрабатывать входящие данные и на их основе генерировать необходимую информацию. Ответ от сервера может быть получен не только в XML, как указывает название технологии.

Ключевыми особенностями использования асинхронных запросов в JavaScript являются: формирование и отправка асинхронных запросов к серверу в AJAX (Asynchronous JavaScript and XML), использование JSON, технологии XMLHttpRequest, реализованной на уровне браузеров.

Во второй главе проведен анализ применения JavaScript технологий в оптимизации серверных мощностей в web-приложениях.

Рассмотрены основные способы оптимизации серверных мощностей в web-приложениях и место JavaScript среди них.

Существует достаточно много путей оптимизации, таких как оптимальный порядок отображения контента в зависимости от его важности для пользователя, оптимизация производительности визуализации, а так же оптимизация контента.

1) Удаление ненужных ресурсов
2) Минификация: предварительная обработка и оптимизация на основе контекста
3) Оптимизация изображений
4) Http-кеширование
5) Использование CDN

Главное правило для достижения максимальной производительности JavaScript-приложений: везде, где это возможно, использовать внешние файлы JavaScript, вместо включения кода JavaScript непосредственно в файлы HTML. Иначе, мало того, что этот код JavaScript придется дублировать на множестве страниц, он не будет кэшироваться Web-браузером, и с каждой последующей загрузкой страницы будет загружаться заново. Первая загрузка страницы будет выполняться значительно медленнее, так как внешний файл требует передачи на сервер дополнительного HTTP-запроса. Однако в большинстве приложений связанный с этим удар по производительности более чем компенсируется экономией времени при последующих загрузках страницы.

Примером может служить разбиение результатов поиска на страницы. Встречаются приложения, в которых при первом Ajax-запросе записи результатов поиска вводятся в массив JSON, а второй используется для

получения общего количества результатов в базе данных для логики их разбиения на страницы. В листинге 1 и листинге 2 приведены примеры этих двух запросов (с использованием ИСР Prototype).

Листинг 1. Первый запрос: получение таблицы записей

```
varurl = "get_data.php";
var options = {
  method: "post",
  parameters: {"page":1,"rows":5},
  onSuccess: firstCallbackFunction,
  onFailure: firstCallbackFunction
}
newAjax.Request(url, options);
```

В листинге 2 показан второй запрос для получения количества записей.

Листинг 2. Второй запрос: получение общего количества записей

```
varurl = "get_count.php";
var options = {
  method: "post",
  parameters: {},
  onSuccess: secondCallbackFunction,
  onFailure: secondCallbackFunction
}
newAjax.Request(url, options);
```

В листинге 3 и листинге 4 приведены соответствующие HTTP-ответы в формате JSON.

Листинг 3. Первый ответ: массив записей

```
 {
  "records": [
            {"id":1,"name":"John","email":"john@example.com"},
            {"id":2,"name":"Mary","email":"mary@example.com"},
            {"id":3,"name":"Tony","email":"tony@example.com"},
            {"id":4,"name":"Emma","email":"emma@example.com"},
            {"id":5,"name":"Alan","email":"alan@example.com"}
  ]
}
```

В листинге 4 показан второй ответ, сообщающий общее количество записей.

Листинг 4. Второй ответ: общее количество записей

```
{"total_records": 95}
```

Разделение этих двух Ajax-запросов – лишняя трата ресурсов, так как их легко объединить в один запрос, который дает ответ, приведенный в листинге 5.

Листинг 5. Экономичный ответ: общее количество и массив записей

```
{
   "total_records": 95,
   "records": [
            {"id":1,"name":"John","email":"john@example.com"},
            {"id":2,"name":"Mary","email":"mary@example.com"},
            {"id":3,"name":"Tony","email":"tony@example.com"},
            {"id":4,"name":"Emma","email":"emma@example.com"},
            {"id":5,"name":"Alan","email":"alan@example.com"}
  ]
}
```

Это уменьшает не только число HTTP-запросов и ответов, но и число серверных сценариев, которые отвечают на AJAX-запросы.

Этот пример — очень простая демонстрация. Чем сложнее приложение, тем важнее уменьшить количество Ajax-запросов.

Далее рассмотрены основные преимущества и недостатки использования JavaScript в оптимизации серверных мощностей в Web-приложениях.

При написании веб-приложений, программирование на JavaScript используется наиболее часто. Если кратко перечислить ключевые особенности данного языка, то следует выделить следующее:

1) Объектно-ориентированность. Выполнение программы представляет собой взаимодействие объектов;

2) Приведение типов данных проводится автоматически;

3) Функции выступают объектами базового класса. Эта особенность делает JavaScript похожим на многие функциональные языки программирования, такие как Lisp и Haskell;

4) Автоматическая очистка памяти. Так называемая, сборка мусора делает JavaScript похожим на C# или Java.

Если говорить о сути применения JavaScript, то этот язык позволяет «*оживлять*» неподвижные страницы сайтов с помощью кода, который можно запустить на исполнение (*так называемые, скрипты*). То есть, можно провести аналогию с мультфильмами, где html и css– это прорисованные герои, а JavaScript – это то, что заставляет их двигаться.

Если говорить о синтаксисе JavaScript, то ему присущи следующие особенности:

1) Регистр важен. Функции с названиями func() и Func() – совершенно разные;

2) После операторов необходимо ставить точку с запятой;

3) Встроенные объекты и операции;

4) Пробелы не учитываются. Можно использовать сколько угодно отступов, а также переводов строки, чтобы оформить свой код.

Простейший код на JavaScript выглядит следующим образом:

```
01 <script type="text/javascript">
02 varsimple_method= function() {
03 alert('Всем Доброго дня!');
04    }();
05 </script>
```

Отметим недостаткиJavaScript.

1) Необходимость обеспечивать кроссбраузерность. Раз уж JavaScript выступает как интернет-технология, то приходится мириться с правилами, которые устанавливает всемирная паутина. Код должен корректно выполняться во всех, или хотя бы самых популярных, браузерах;

2) Система наследования в языке вызывает трудности в понимании происходящего. В JavaScript реализовано наследование, основанное на прототипах;

3) Отсутствует стандартная библиотека. JavaScript не предоставляет никаких возможностей для работы с файлами, потоками ввода-вывода и прочими полезными вещами;

4) Синтаксис в целом затрудняет понимание.

Рассмотрим преимущества JavaScript.

1) JavaScript предоставляет большое количество возможностей для решения самых разнообразных задач. Гибкость языка позволяет использовать множество шаблонов программирования применительно к конкретным условиям. Изобретательный ум получит настоящее удовольствие;

2) Популярность JavaScript открывает перед программистом немалое количество готовых библиотек, которые позволяют значительно упростить написание кода и нивелировать несовершенства синтаксиса;

3) Применение во многих областях. Широкие возможности JavaScript дают программистам шанс попробовать себя в качестве разработчика самых разнообразных приложений, а это, безусловно, подогревает интерес к профессиональной деятельности.

Таким образом, можно утверждать, что JavaScript получил широкое распространение в сфере веб-программирования, вобрав в себя возможности объектно-ориентированных и функциональных языков.

В третьей главе рассмотрено совершенствование применения JavaScript в процессе оптимизации серверных мощностей в web-приложениях.

Представлена система очередей как условие успешной асинхронной работы веб-приложения и средство оптимизации серверных мощностей.

Одна из сильных сторон JavaScript - обработка асинхронного кода. Вместо того, чтобы блокировать поток выполнения задачи, асинхронный код выстраивает события в очередь, которая выполняется после завершения других частей программы.

Основными функциями асинхронного кода JavaScript являются setTimeout и setInterval. Функция setTimeout выполняет заданную функцию после истечения определенного временного интервала. Она принимает возвратную функцию в качестве первого аргумента и время (в миллисекундах) в качестве второго аргумента. Вот пример использования:

```
01 console.log( "a" );
02 setTimeout(function()
03 {
04    console.log( "c" )
05 }, 500);
06 setTimeout(function()
07 {
08    console.log( "d" )
09 }, 500);
10 setTimeout(function()
11 {
12    console.log( "e" )
13 }, 500);
14 console.log( "b" );
```

Ожидается, что в консоли мы увидим “a”, “b”, а затем через примерно 500 мс - “c”, “d”, и “e”. Таймаут не будет выполняться до тех пор, пока весь остальной код в блоке не выполнится. То есть, если установлен таймаут, а затем какая-нибудь функция выполняется долго, то таймаут не начнет отсчитываться, пока функция не завершится. В реальности, асинхронные функции setTimeout и setInterval ставятся в очередь, известную как цикл событий.

Цикл событий очередь возвратных функций. Когда асинхронная функция выполняется, возвратная функция ставится в очередь. JavaScript не запускает обработку цикла событий, пока код, запущенный после асинхронной функции выполняется. Данный факт означает, что код JavaScript не является многопоточным, хотя и кажется таковым. Цикл

событий является очередью FIFO (первый пришел, первый вышел), что означает выполнение возвратных функций в порядке их поступления. JavaScript был выбран для платформы node.js именно по причине простого процесса разработки подобного кода.

Асинхронный JavaScript и XML (AJAX) навсегда изменил профиль JavaScript. Браузер может обновлять веб страницу без перезагрузки. Код реализации AJAX в разных браузерах может оказаться длинным и занудным. Но, благодаря jQuery (и другим библиотекам), AJAX стал очень простым и элегантным решением для обеспечения клиент-серверных коммуникаций.

JavaScript делает простым процесс создания асинхронных приложений. Использование обещаний, событий или именованных функций позволяет избежать "проблемы возвратных функций".

В последнем параграфе рассмотрены Node.js и Event loop: асинхронная работа для оптимизации серверных мощностей.

В основе Node.js лежит цикл событий, реализуемый библиотекой libev. На каждом витке цикла происходит следующее: в первую очередь идёт выполнение функций, установленных на предыдущем витке цикла с помощью process.nextTick(). Далее идёт обработка событий libev, в частности событий таймеров. В последнюю очередь идёт опрос libeio для завершения операций ввода/вывода и выполнения установленных для них коллбеков. В случае, если при прохождении цикла оказалось, что ни одна функция не установлена с помощью process.nextTick(), нет ни одного таймера и очереди запросов в libev и libeio пусты, то node завершает работу.

Благодаря тому, что Node.js имеет отличный инструмент, NPM — менеджер пакетов, с его помощью можно управлять модулями и зависимостями. Его легко использовать и масштабировать в рамках серверного окружения. Так, например, используя Node.js для нескольких проектов мы можем устанавливать пакеты/модули как глобально так и локально.

Так же есть ряд дополнительных инструментов для комфортной работы с Node.js. Так, например, для поддержания процессов используют утилиты: forever или supervisor. Первая устанавливается из менеджера пакетов NPM и служит только для поддержания процессов Node.js, в то время как второй умеет работать и с другими утилитами, такими как RabbitMQ, Bash Scripts и тем самым является более универсальной. Так же существует модуль supervisor который устанавливается как пакет для Node.js и играет роль наблюдателя за изменениями в скрипте и автоматического перезапуска без утечки памяти ОЗУ и без очистки межмодульных зависимостей.

Таким образом, если сервер содержит мало логики и занимается в основном малозатратной по времени обработкой приходящих данных, то цикл выполняется очень часто. Однако, если обработка данных происходит

долго, то целесообразно разделить этот процесс на отдельные части, между которыми возвращать управление в цикл событий для возможности запуска обслуживания нового соединения или обработки асинхронно читаемых с диска данных.

В заключении диссертационной работы решены следующие задачи:

1) Рассмотрены основные характеристики JavaScript технологий.

2) Проанализировано применение JavaScript технологий в оптимизации серверных мощностей в Web-приложениях.

3) Предложены пути, направленные на совершенствование применения JavaScript в процессе оптимизации серверных мощностей в Web-приложениях.

В приложении представлена авторская раработка расширения для Google Chrome, которая позволяет расширить функционал браузера. Данное расширение позволяет высчитывать значения без перезагрузки страницы, используя мощности клиентской стороны и технологию V8, которая способна наладить связь клиента с сервером (Front-end и Back-end).

Публикации

Телевич, П.Л. Использование Node.js технологии как платформу для оптимизации серверных мощностей [Электронный ресурс]. – 2017. – Режим доступа: https://habrahabr.ru/post/319368/ – Дата доступа: 12.01.2017.

Телевич, П.Л. Использование семантических технологий на платформе JavaScript[Электронный ресурс]. – 2017. – Режим доступа: http://conf.ostis.net/ – Дата доступа: 12.01.2017.