Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное
учреждениевысшего образования
Казанский государственный энергетический университет
Кафедра информационных технологий и интеллектуальных систем

AJAX и обработка данных

Методические указания для выполнения лабораторных работ

Казань, 2024

Лабораторная работа
Основы технологии AJAX

Цель работы: изучить основы технологии AJAX. Написать простейшие клиентское и серверное приложение, обменивающиеся ассинхронными данными.

Краткие теоретические сведения

AJAX (Asynchronous JavaScript and XML (и DHTML, и т.д.). – это набор техник разработки веб-интерфейсов, позволяющих делать динамические запросы к серверу без видимой перезагрузки веб-страницы: пользователь не замечает, когда его браузер запрашивает данные. AJAX обеспечивает динамичность и асинхронность web-разработок при отсутствии необходимости обновления страниц.

Например, при использовании Google или web-клиента Gmail с помощью AJAX обеспечивается динамическое и асинхронное поведение, где исключается обновление страниц. Посредством AJAX пользователь может взаимодействовать с web-страницами, подобно работе клиентов с более богатыми возможностями.

Возможности, предоставляемые AJAX:

- Стандартно-базированная презентация с использованием XHTML и CSS;
- Динамическое отображение и взаимодействие с использованием объектной модели документа;
- Взаимообмен данными и манипуляция с задействованием XML и XSLT;
- Асинхронное извлечение данных с использованием XMLHttpRequest;
- JavaScript, связывающий все вместе.

AJAX позволяет писать быстрореагирующие веб-приложения, в которых не нужно постоянно обновлять страницы. AJAX – простая технология, поддерживаемая всеми основными браузерами.

Как работает AJAX

В веб-клиенте Gmail или Google Maps имеется возможность проверки правописания и прокрутки по всему изображению, соответственно, без обновления страниц. AJAX – это технология, которая обрабатывает операциив JavaScript и асинхронно запускает на стороне сервера операции, предоставляющие желаемый результат.

В основе технологии AJAX лежит объект XMLHttpRequest. Изначально он появился в Internet Explorer, а затем – в Mozilla/Safari.

Объект XMLHttpRequest

Это объект JavaScript, создается как показано в примере 1.

Пример 1. Создание нового объекта XMLHttpRequest

```
<script language="javascript" type="text/javascript">
var request = new XMLHttpRequest();
</script>
```

Это объект, который управляет всем взаимодействием с сервером – это технология JavaScript в объекте XMLHttpRequest, который общается с сервером.

В нормальных Web-приложениях пользователи заполняют поля форм и нажимают кнопку *Submit* (подтвердить). Затем форма передается на сервер полностью, сервер обрабатывает сценарий (обычно PHP или Java, возможно, CGI-процесс и т.п.), а потом передает назад всю новую страницу. Эта страница может быть HTML-страницей с новой формой с некоторыми заполненными данными, либо страницей подтверждения, либо, возможно, страницей с какими-то выбранными вариантами, зависящими от введенных в оригинальную форму данных.

Пока сценарий или программа на сервере не обработается и не возвратится новая форма, пользователи должны ждать. Их экраны очистятся и будут перерисовываться по мере поступления новых данных от сервера. Ajax по существу помещает технологию JavaScript и объект XMLHttpRequest *между* вашей Web-формой и сервером. Когда пользователи заполняют формы,данные передаются в какой-то JavaScript-код, а не прямо на сервер. Вместо этого JavaScriptкод собирает данные формы и передает запрос на сервер. Пока это происходит, форма на экране пользователя не мелькает, не мигает, не исчезает и не блокируется. Код JavaScript передает запрос в фоновом режиме; пользователь даже не замечает, что происходит запрос на сервер. Более того, запрос передается асинхронно, это означает, что ваш JavaScript-код (и пользователь) не ожидают ответа сервера. То есть, пользователи могут продолжать вводить данные, прокручивать страницу и работать сприложением.

Затем сервер передает данные обратно в ваш JavaScript-код (все еще находящийся в вашей Web-форме), который решает, что делать с данными. Он может обновить поля формы «на лету», придавая свойство немедленности вашему приложению – пользователи получают новые данные без подтверждения или обновления их форм. JavaScript-код может даже получить данные, выполнить какие-либо вычисления и передать еще один запрос. XMLHttpRequest может общаться с сервером по своему желанию, а пользователь даже не догадывается о том, что происходит на самом деле. В результате получаем динамичность, чувствительность, высокую интерактивность настольного приложения вместе со всеми возможностями интернет.

Добавление JavaScript-кода

Фактически JavaScript-код используется для небольшого числа основных задач:

- Получить данные формы: JavaScript-код упрощает извлечение данных из вашей HTML-формы и передает их на сервер.
- Изменить значения в форме: Форма обновляется тоже легко, от установки значений полей до замены изображений «на лету».

• Выполнить анализ HTML и XML: JavaScript-код используется для управления DOM и для работы со структурой HTML-формы и всеми XML-данными, возвращаемыми сервером.

Для выполнения первых двух задач используется метод getElementById(), приведенный в примере 2.

Пример 2. Сбор и установка значений полей при помощи JavaScript-кода

```
// Получить значение поля «phone» и записать его в переменную phone
var phone = document.getElementById("phone").value;
// Установить значения в форме, используя массив response
document.getElementById("order").value = response[0];
document.getElementById("address").value = response[1];
```

Получение объекта Request

XMLHttpRequest является центральным для Ajax-приложений.

Работа с браузерами Microsoft

Браузер Microsoft Internet Explorer для обработки XML использует анализатор MSXML. Поэтому, когда вы пишете Ajax-приложения, которые должны работать в Internet Explorer, необходимо создать объект особым способом. Есть две различных версии MSXML. Версия MSXML зависит от версии технологии JavaScript, установленной в Internet Explorer, поэтому нужно написать код, подходящий для обеих версий.

В примере 3 приведен код для создания XMLHttpRequest в браузерах Microsoft.

Пример 3. Создание объекта XMLHttpRequest в браузерах Microsoft

```
var request  = null; function
createRequest () {try {
request = new ActiveXOblect("Msxml2.request");
} catch (othermicrosoft) {
try {
request = new ActiveXObject("Microsoft.request");
} catch (failed) {
request = null;
}
}
}
```

Этот код пытается создать объект, используя одну версию MSXML; если это не получится, создается объект для второй версии. Если ничего не сработало, переменная request устанавливается в false, для того чтобы указать коду, что что-то не так. В этом случае, возможно, идет работа с браузером не от Microsoft и нужно использовать другой код для выполнения работы.

Работа с Mozilla и браузерами не от Microsoft

var request = new XMLHttpRequest object;.

Эта строка создает объект XMLHttpRequest в Mozilla, Firefox, Safari, Opera и в большой степени в каждом браузере не от Microsoft, поддерживающем Ajax в любой форме или разновидности.

Объединение

В примере 4 приведен код для всех браузеров.

Пример 4. Создание объекта XMLHttpRequest для всех браузеров

```
var request  = null; function
createRequest () {try {
request = new XMLHttpRequest();
} catch (trymicrosoft) {
try {
request = new ActiveXOblect("Msxml2.request");
} catch (othermicrosoft) {
try {
request = new ActiveXObject("Microsoft.request");
} catch (failed) {
request = null;
}
}
}
if (request == null) alert("Ошибка создания XMLHttpRequest объекта!");
}
```

Основу этого кода можно разделить на три шага:

1. Создаем переменную request для ссылки на объект XMLHttpRequest, который создается.
2. Создаем объект для неMicrosoft браузеров, если это не получается, то переходим к следующему пункту.
3. В блоке try создайте объект в браузерах Microsoft:

• В блоке try создайте объект с использованием объекта Msxml2.request.
• Если это не получится, В блоке try создайте объект с использованием объекта Microsoft.request.

В конце этого процесса request должен ссылаться на корректный объект XMLHttpRequest, независимо от используемого пользователем браузера.

Запрос/ответ в Ajax

Выполнение запроса

1. Получить какие-либо данные из Web-формы.
2. Создать URL для подключения.

3. Открыть соединение с сервером.
4. Установить функцию для сервера, которая выполнится после его ответа.
5. Передать запрос.

В примере 5 приведен пример Ajax-метода, который выполняет эти операции и именно в этом порядке:

Пример 5. Выполнить запрос с Ajax

```
function getResult() {
//Создать URL для подключения
var url = "test.php";
//Создаем объект XMLHttpRequest
createRequest();
//Открыть соединение с сервером
request.open("GET", url, true);
//Установить функцию для сервера, которая будет выполнена после его ответа
request.onreadystatechange = updatePage;
//Отправить запрос
request.send(null);
}
```

Код устанавливает PHP-сценарий в качестве URL для подключения. Затем открывается соединение. Указывается метод соединения (GET) и URL. Последний параметр, когда установлен в true, запрашивает асинхронное соединение (то есть, делает это способом, соответствующим названию Ajax). При использовании false код ждал бы выполнения запроса и не продолжал бы работу до получения ответа. При использовании true пользователи могут работать с формой (и даже вызывать другие JavaScript-методы) пока сервер обрабатывает этот запрос в фоновом режиме.

Свойство onreadystatechange request (это экземпляр объекта XMLHttpRequest) позволяет информировать сервер о том, что следует делать после завершения работы (что может быть через пять минут или через пять часов).

Поскольку код не собирается ждать сервер, вы должны дать серверу знать, что делать, так чтобы вы смогли среагировать. В данном случае будет инициирован конкретный метод (называемый updatePage()) после завершения сервером обработки вашего запроса.

Наконец, вызывается send() со значением null. Поскольку вы добавили данные для передачи на сервер в URL запроса, вам не надо передавать что- либо в запросе. Таким образом, передается ваш запрос, и сервер может делатьто, что вы указали ему делать.

Данные в примере 5 являются простым текстом и могут быть включены как часть URL-запроса. GET посылает запрос вместо более сложного POST. Не добавляется XML, заголовки контента, не передаются данные в теле запроса.

На странице test.php, которую мы указываем в запросе производится либо обработка принятой информации, если таковая передавалась в запросе, например, как test.php?city=chernihiv, либо просто возвращается результат.

Пример 5.1 пример test.php

```
<?php
//Sending zipCode
echo rand(10000, 99999);
?>
```

Обработка ответа

Теперь вы должны разобраться с ответом сервера:

- Не делать ничего, пока свойство request.readyState не будет равно 4.
- Сервер будет записывать свой ответ в свойстве request.responseText.

Первый момент (состояния готовности), нужно уточнить какие бывают стадии HTTP-запроса. Пока просто проверяем на равенство определенному значению (4), и все будет работать. Второй момент (использование свойства request.responseText для получения ответа от сервера) является простым. В примере 6 приведен пример метода (который сервер может вызвать), основанного на значениях, переданных в примере 5.

Пример 6. Обработка ответа от сервера

```
function updatePage() {
if(request.readyState == 4) {
var response = request.responseText;
//Что-то делаем с пришедшими нам данными, например, присваиваем
их полю с id="zipCode"
document.getElementById("zipCode").value = response;
}
}
```

Код ожидает, пока сервер не вызовет его с нужным состоянием готовности, и затем использует значение, которое сервер возвращает, для установки другого поля формы. В результате поле zipCode неожиданно появляется с ZIP-кодом, но пользователь *ни разу не щелкнул по кнопке*! Поле zipCode является обычным текстовым полем. После возврата сервером ZIP-кода и установки этого поля методом updatePage() в значение ZIP-кода города/штата пользователи *могут* переопределить это значение. Это сделано умышленно по двум причинам: сохранить этот пример простым и показать, что иногда нужно, чтобы пользователи имели возможность переопределить значения, возвращенные сервером. Такие моменты важны при хорошем дизайне пользовательского интерфейса.

Перехват в Web-формах

Пример 7. Запуск Ajax-процесса

```
<form>
<p>City: <input type="text" name="city" id="city" size="25"
onChange="getResult();" /></p>
<p>Zip Code: <input type="text" name="zipCode" id="zipCode" size="5"
readonly /></p>
</form>
```

Для более легкой работы с HTML при помощи JavaScript были разработаны несколько фреймворков, которые упрощают повседневные рутинные функции и помогают программисту не задумываться о реализации примитивов, например, доступа к полям DOM-а, их изменение стандартными средствами JavaScript, а предоставляют удобные инструменты для данных функций. Из наиболее известных можно отметить два фреймворка: Ext JS и jQuery.

ExtJS – библиотека JavaScript для разработки веб-приложений и пользовательских интерфейсов, изначально задуманная как расширенная версия Yahoo! UI Library, однако преобразовавшаяся затем в отдельный фреймворк. Использует адаптеры для доступа к библиотекам Yahoo! UI Library, jQuery или Prototype/script.aculo.us. Поддерживает технологию AJAX, анимацию, работу с DOM, реализацию таблиц, вкладок, обработку событий и все остальные новшества «Web 2.0».

С версии 2.1 библиотека ExtJS распространяется по условиям трёх лицензий: Commercial License, Open Source License и OEM / Reseller License. Начиная с версии Ext JS 3.0 библиотека разбивается на две части: Ext Core (набор JavaScript функций, позволяющий создавать динамические веб-страницы и распространяемый по MIT-лицензии) и Ext JS (набор виджетов для создания пользовательских интерфейсов, распространяемый аналогично Ext JS 2.1 по условиям трёх лицензий).

jQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобныйAPI по работе с Ajax. Т.к. jQuery является более простой для изучения остановимся на ней подробнее.

Возможности:

- переход по дереву DOM, включая поддержку Xpath как плагина;
- события;
- визуальные эффекты;
- AJAX-дополнения;
- JavaScript-плагины

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки, управление передаётся JQuery, идентифицирующей кнопки и затем преобразовывающий его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функционал, полезный для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, большая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно тот JavaScript-функционал, который на нём был бы востребован.

Для использования jQuery достаточно скачать с сайта http://jquery.com/ последнюю версию фремворка и подключить его в свой проект.

Пример 8. Пример подключения jQuery как один внешний файл

```
<head>
<script type="text/javascript" src="путь/к/jQuery.js"></script>
</head>
```

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним – jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким можно писать его следующим образом:

Пример 9. Использование функции $

```
jQuery(function($) {
// Тут код скрипта, где в $ будет jQuery
})
```

Работу с jQuery можно разделить на 2 типа:

- Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта.
- Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора CSS, что возвращает объект jQuery, содержащий некоторое количество элементов HTML- страницы. Эти элементы затем обрабатываются методами jQuery. Например,

Пример 10.

```
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
```

находит все элементы с классом test, а также все элементы <p> с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Методы, начинающиеся с $., удобно применять для обработки глобальных объектов. Например:

Пример 11.

```
$.each([1,2,3], function() {
document.write(this + 1);
});
```

добавит на страницу 234.

Для примера немного изменим предыдущие файлы и добавим в них использование фреймворка jQuery. Подключаем фреймворк в файл index.html:

Пример 12. Подключение фрейморка jQuery

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

Добавим в нашу форму дополнительное поле, в котором будем изменять значение параметра учета количества сделанных изменений:

Пример 13. Добавление дополнительного поля в форму

```
<form>
<p>City: <input type="text" name="city" id="city" size="25"
onChange="getResult();" /></p>
<p>ZipCode: <input type="text" name="zipCode" id="zipCode" size="5"
readonly /> получен с помощью AJAX запроса</p>
<p>Значение было изменено <input type="text" name="jq" id="jq"
size="3" value="0" readonly /> раз с помощью jQuery</p>
</form>
```

Дополним функцию updatePage() вызовом дополнительной функции jq(), которая, используя механизмы jQuery, будет изменять данные в поле с id=”jq”,в котором указывается количество обновлений страницы:

Пример 14. Исходный код функции jq()

```
function jq() {
//Записываем в переменную X значение поля с id='jq'
$x = $("#jq").attr("value");
//Увеличиваем переменную на 1
$x++;
//Обновляем значение атрибута value поля с id='jq'
$("#jq").attr("value", $x);
}
```

Про дополнительные возможности jQuery можно узнать из документации и примеров использования на сайте разработчиков jquery.com.

Порядок выполнения работы

1. Ознакомьтесь с теоретическими сведениями.
2. Изучите работу скриптов, приведенных в теоретических сведениях.
3. Реализуйте задание согласно варианта.
4. Составьте отчет о выполнении работы.

Варианты заданий

1. Проверка логина при регистрации – проверка на существование вводимого логина в базе данных пользователей, при наборе логина в соответствующем поле.
2. Корзина товаров в онлайн магазине. Добавление/удаление.
3. Города и области – выборка области по выбранному городу.
4. Подсчет хеша при заполнении строки.
5. Транслитерация.
6. Простейший почтовый клиент – выборка темы письма по запросу и по клику на теме письма доставка его тела.
7. Мониторинг сервисов на ПК. Опрос списка запущенных процессов, и ассинхронное обновление.
8. Гостевая книга, Комментарии – добавление/удаление.
9. Чат.