**Питання AGAX**

1. Що таке AJAX?

це не сама по собі технологія, а термін, який описує «новий» підхід до спільного використання існуючих технологій. AJAX включає в себе: HTML або XHTML, CSS, JavaScript, DOM, XML, XSLT, і XMLHttpRequest об'єкт. Коли ці технології об'єднуються в модель AJAX, веб-додатки здатні робити швидкі, взаємодоповнюючі оновлення інтерфейсу користувача без необхідності повного перезавантаження сторінки браузером. Програми працюють швидше і стають більш чуйними до дій користувачів.

1. Що я можу зробити з AJAX?

Елементи інтерфейсу

В першу чергу AJAX корисний для форм і кнопок, пов'язаних з елементарними діями: додаванням в кошик, підпискою і т.д.

Зараз саме в порядку речей такі дії на сайтах здійснюються без перезавантаження сторінки.

Динамічне завантаження даних

Наприклад, дерево, яке при розширенні вузла запитує дані із сервера.

Live Search

Живий пошук - класичний приклад використання AJAX, прийнятий сучасними пошуковими системами.

Користувач починає вводити пошукову фразу, а JavaScript пропонує можливі варіанти, отримуючи список найбільш ймовірних доповнень з сервера.

1. COMET

це загальний термін, що описує різні методи отримання даних з ініціативи сервера.

Можна сказати, що AJAX "надсилає запит - отримав результат", а COMET - "безперервний канал, через який надходять дані".

1. Приклади програм COMET

* Чат – людина сидить і дивиться, що пишуть інші. При цьому нові повідомлення надходять «самі по собі», він не повинен натискати на кнопку для оновлення вікна чату.
* Аукціон – людина дивиться на екран і бачить, як оновлюється поточна ставка на товар.
* Інтерфейс редагування – коли один редактор починає редагувати документ, інші бачать інформацію про нього. Співавторство також можливе, коли редактори бачать зміни один одного.

На даний момент технологія COMET зручно реалізована у всіх браузерах.

1. Обмін даними

Для обміну даними на сторінці повинен бути створений об'єкт XMLHttpRequest, який є своєрідним посередником між Браузером Користувача і сервером (рис. 1). За допомогою XMLHttpRequest можна відправити запит на сервер, а також отримати відповідь у вигляді різного роду даних.

1. Зв’язки з сервером

Існує два способи зв'язку з сервером. Перший спосіб - це запит GET. У цьому запиті ви отримуєте доступ до документа на сервері, передаючи йому аргументи через саму URL-адресу. У цьому випадку на стороні клієнта логічно буде використовувати функцію втечі Javascript, щоб переконатися, що деякі дані не переривають запит.

Не рекомендується робити ЗАПИТИ GET на сервер з великими обсягами даних. Для цього є запит POST.

1. Обов’язки в зв’язку

Клієнтська частина, написана на Javascript, повинна забезпечити необхідний функціонал для безпечного обміну з сервером і забезпечити методи обміну даними будь-яким з перерахованих вище способів. Серверна частина повинна обробляти вхідні дані, а на їх основі генерувати нову інформацію (наприклад, працювати з базою даних), і віддавати її клієнту. Наприклад, для запиту інформації з сервера можна скористатися регулярним GET-запитом з передачею декількох і малих параметрів, а для оновлення інформації, або додавання нової інформації, потрібно буде використовувати запит POST, так як він дозволяє передавати великі обсяги даних.

1. Асинхронна передача даних

Як уже згадувалося, AJAX використовує асинхронну передачу даних. Це означає, що поки дані передаються, користувач може виконувати інші необхідні йому дії. У цей час необхідно повідомити користувача про те, що відбувається будь-який обмін даними, інакше користувач подумає, що щось не так і може покинути сайт, або повторно назвати «повішеним», на його думку, функцією. Індикація під час обміну даними в додатку Web 2.0 грає дуже важливу роль: відвідувачі можуть ще не звикнути до таких способів оновлення сторінки.

1. Відповідь від сервера

Відповідь від сервера може бути більше, ніж просто XML, як випливає з назви технології. На додаток до XML, ви можете отримати відповідь у звичайному тексті, або JSON (Javascript object Позначення). Якщо відповідь була отримана простим текстом, ви можете відразу ж відобразити її в контейнері на сторінці. Коли ви отримуєте відповідь як XML, ви зазвичай обробляєте отриманий XML-документ на клієнтському комп'ютері та перетворюєте дані на (X)HTML. При отриманні відповіді, відформатованої JSON, клієнту потрібно лише виконати отриманий код (javascript-функцію eval), щоб отримати повний об'єкт Javascript. Але тут потрібно бути обережним і враховувати той факт, що шкідливий код може передаватися за цією технологією, тому перед виконанням отриманого з сервера коду слід ретельно перевірити і обробити його. Існує така практика, як «пустий» запит, при якому відповіді від сервера не надходить, змінюються тільки дані на стороні сервера.

1. Методи об'єкта XMLHttpRequest

Зверніть увагу, що назви методів написані в тому ж стилі (camel-style), що і інші функції Javascript. Будьте обережні при їх використанні.

* abort() — скасовує поточний запит на сервер.
* getAllResponseHeaders() - Отримати всі заголовки відповідей із сервера.
* getResponseHeader ("header\_name") - Отримати вказаний заголовок.
* open ("request\_type", "URL", "асинхронний", "ім'я користувача", "пароль") — ініціалізувати запит на сервер, вказати метод запиту. Тип запиту та URL-адреса є обов'язковими параметрами. Третій аргумент - логічне значення. Як правило, це завжди вірно або не вказано взагалі (за замовчуванням вірно).

1. Аутентифікації за допомогою методів об'єкта XMLHttpRequest

Для аутентифікації використовуються четвертий і п'ятий аргументи (зберігати дані аутентифікації в скрипті дуже небезпечно, так як скрипт може переглядати будь-який користувач).

* send ("content") — надішліть HTTP-запит на сервер і отримаєте відповідь.
* setRequestHeader ("header\_name", "value") — встановіть значення заголовка запиту.

1. Властивості об'єкта XMLHttpRequest

onreadystatechange є однією з найважливіших властивостей об'єкта XMLHttpRequest. Ця властивість визначає обробник, який викликається щоразу, коли стан об'єкта змінюється

* readyState - це число, яке вказує на стан об'єкта
* responseText — представлення відповіді сервера у форматі звичайного тексту (рядка).
* responseXML - це об'єкт документа, сумісний з DOM, отриманим із сервера
* status — стан відповіді від сервера.
* statusText - це текстове представлення стану відповіді від сервера.

1. Властивість ReadyState:

0 - Об'єкт не ініціалізовано.

1 - Об'єкт завантажує дані.

2 – Об'єкт завантажив свої дані.

3 - Об'єкт не повністю завантажений, але може взаємодіяти з користувачем.

4 — Об'єкт повністю ініціалізований; З сервера надійшла відповідь.

Саме виходячи зі стану готовності об'єкта, можна надати відвідувачеві інформацію про те, на якому етапі знаходиться процес обміну даними з сервером і, можливо, повідомити його про це візуально.

1. Створення об'єкта XMLHttpRequest

Як вже було сказано вище, створення цього об'єкта для кожного типу браузера є унікальним процесом.

Наприклад, щоб створити об'єкт у браузерах, сумісних з Gecko, Konqueror і Safari, ви будете використовувати такий вираз:

var Request = new XMLHttpRequest();

A для Internet Explorer використовує наступне

var Request = new ActiveXObject("Microsoft.XMLHTTP");

Або

var Request = new ActiveXObject("Msxml2.XMLHTTP");

1. Запит на сервер

Алгоритм запиту на сервер виглядає так:

* Перевірте наявність XMLHttpRequest.
* Ініціалізувати підключення до сервера.
* Відправка запиту на сервер.
* Обробка отриманих даних.

Для створення запиту на сервер ми створимо невелику функцію, яка буде функціонально поєднувати функції для запитів GET і POST.

1. Обробка відповіді

У попередньому прикладі ми зробили запит на функцію сервера. Але це за своєю суттю небезпечно, тому що ми не обробляємо стани об'єктів і стани відповіді з сервера.

Давайте додамо до нашого коду, щоб він міг відображати візуальне сповіщення про процес завантаження.

Як ви вже знаєте, об'єкт XMLHttpRequest дає змогу дізнатися стан відповіді із сервера. Давайте скористаємося цією можливістю.

...

Request.onreadystatechange = function()

{

//Если обмен данными завершен

if (Request.readyState == 4)

{

//Передаем управление обработчику пользователя

r\_handler(Request);

}

else

{

//Оповещаем пользователя о загрузке

}

}

...

1. Параметри відповіді із сервера

З сервера можна отримати кілька типів даних:

Звичайний текст

XML

JSON

Якщо ви отримуєте звичайний текст, то ви можете відразу відправити його в контейнер, тобто на вихід. При отриманні даних у форматі XML необхідно обробити дані за допомогою функцій DOM і представити результат за допомогою HTML.

1. Використання JSON

JSON є об'єктом позначення Javascript. З її допомогою можна представити об'єкт у вигляді рядка (тут можна дати аналогію з функцією серіалізації). При отриманні даних JSON необхідно виконати його, щоб отримати повноцінний об'єкт Javascript і виконати на ньому необхідні операції. Майте на увазі, що ця передача та виконання даних не є безпечним. Ви повинні стежити за тим, що приходить для виконання.

{

"data":

{

"misc":

[

{

"name" : "JSON-элемент один",

"type" : "Подзаголовок 1"

},

{

"name" : "JSON-элемент два",

"type" : "Подзаголовок 2"

}

]

}

}

1. Робота з серверними мовами програмування

Така робота нічим не відрізняється від звичайної. Наприклад, я візьму PHP як мову на стороні сервера. У клієнтській частині нічого не змінилося, але серверна частина тепер представлена PHP-файлом.

при доступі до цього файлу клієнт поверне лінію Hello, World. Як ви розумієте, це являє собою найширші можливості для створення додатків. Передаючи аргументи при виклику сервера за допомогою XMLHttpRequest, можна параметризувати вихід, тим самим забезпечуючи широку функціональність веб-застосунку.

На додаток до PHP, ви можете використовувати будь-яку іншу мову програмування на стороні сервера.

echo "Hello, World!";

1. Основні переваги використання AJAX:

* скорочення трафіку (за рахунок зменшення обсягу переданих даних між клієнтом і сервером);
* зниження навантаження на сервер (потрібно не генерувати всю сторінку, а тільки ту частину, яку потрібно оновити);
* підвищення продуктивності і чуйності (немає необхідності в повному перезавантаженні сторінки, досить оновити зміст тільки окремих блоків);
* Підвищена інтерактивність (за допомогою AJAX можна відразу відобразити результати і зробити ресурс більш зручним для використання).

1. Створення асинхронних запитів за допомогою XHR

Що таке асинхронне прохання, ми вже обговорювали вище. Тобто він працює у фоновому режимі і не заважає користувачеві взаємодіяти зі сторінкою. А це означає, що при відправці такого запиту сторінка не «замерзла», з нею можна продовжувати взаємодіяти.

XHR є абревіатурою для об'єкта , яка просто дозволяє взаємодіяти з сервером через AJAX.XMLHttpRequest

1. Асинхронний drag'n'drop.

Іноді для асинхронних операцій доводиться робити різні «фінти з вухами». Наприклад, ви хочете перетягнути'n'drop в дерево, тобто перетягувати статті з одного розділу в інший за допомогою миші, і що вони змінюють батьківський на сервері в базі даних.

Drag'n'drop is "взяв мишачий об'єкт - покладіть його туди, де він повинен - готовий". Але в асинхронному візерунку ви не можете мати все прямо відразу.

Необхідно перевірити привілеї на сервері, перевірити, чи об'єкт все ще існує, раптом його видалив інший користувач.

Потрібно якось показати, що процес почався, але результат «буде ..». І як? В асинхронному візерунку вказівник миші не може просто навести курсор на об'єкт, ставши годинником.

В цьому випадку вони використовують або синхронні запити на сервер - а потім все дійсно зависає, або оригінальний вихід - ставлять об'єкт так, ніби він був переданий, і повідомляють анімовану іконку про очікування відповіді.

Якщо відповідь негативна, обробник відповіді переміщує об'єкт назад

1. Блокування та/або керування версіями

Контроль версій - кожен новий документ стає версією, так що зміни ніколи не губляться. Керування версіями тягне за собою конфлікти, коли Петро почав редагувати документ до Васі, а зберіг його пізніше. При цьому в останній версії зміни Васі були втрачені, хоча передостання (Васіна) версія в системі обов'язково є.

1. Контекст автоматичного оновлення

Проблему застарілого контенту можна на 99% вирішити за допомогою миттєвого автооновлення.

Браузер зберігає постійне з'єднання з сервером (або час від часу робить коригувальні запити) - і необхідні зміни відправляються через цей канал.

Наприклад, нові статті іноді завантажуються у гілку відкритого дерева, а нові літери завантажуються у відкритий поштовий інтерфейс.

Загалом, проблема застарілого контексту безпосередньо пов'язана з проблемою цілісності даних. Сервер несе відповідальність за остаточну перевірку цілісності, як у випадку перевірки форми.

1. Синхронна модель VS асинхронна модель

У звичайному програмуванні всі операції синхронні, тобто виконуються одна за одною.і

Умовно кажучи, ми діємо наступним чином:

кинути вудку

Чекаючи, поки він вкусить

клюють - увімкніть затягування прядильного стрижня

При асинхронному підході ми:

повісити на стрижень спеціальний детектор укусу, дайте йому витягнути спінінг, коли він вкусить

кинути вудку

робити інші речі

датчик укусу працює, спрацьовуючи обертовий ліфт

Тобто в синхронному випадку вудка постійно привертає нашу увагу. Риболовля є послідовним процесом

В асинхронному варіанті - ми спочатку встановили програму, що робити з укусом, а потім опустили вудку, щоб зловити і зробили інші речі.

Наприклад, ставимо ще 5 таких стрижнів.

Асинхронне програмування складніше, ніж синхронне програмування, і спочатку незвичайне, оскільки зумовлює те, що буде працювати після.

Тобто програму «що робити, коли вона кусається» необхідно встановити перед укусом, і взагалі невідомо, чи є в водоймі риба.

Є методи, які полегшують асинхронне програмування, такі як відкладений об'єкт відкладений (Twisted, Dojo, Mochikit), але про це в окремій статті.

1. Синхронна модель в AJAX

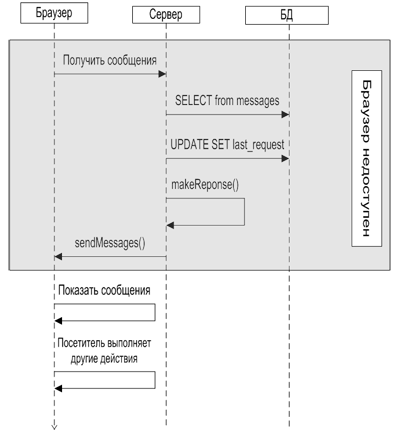
Поверніться до наших оперативних пам'яті: браузер, сервер і, скажімо, база даних.

У синхронній моделі браузер відправляє запит на сервер і зависає, чекаючи, поки він виконає всю необхідну роботу. Сервер виконує запити проти бази даних, обгортає відповідь у необхідному форматі та виводить її. Браузер. коли він отримує відповідь, він викликає функцію відображення.

Всі процеси виконуються послідовно, по одному.

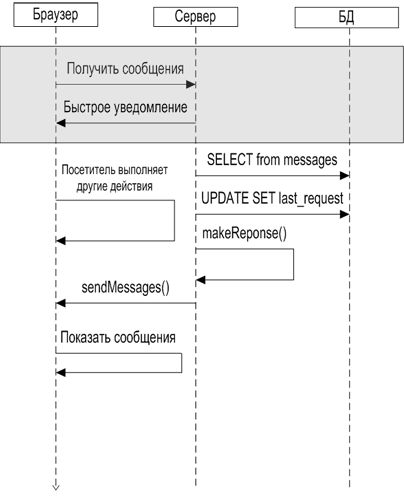
Мережні латки вмикаються під час тайм-ауту, зазначеного сірим кольором на схемі.

Користувач не може зробити щось інше на тій же сторінці, поки відбувається синхронне спілкування.



1. Асинхронна модель в AJAX

У асинхронному візерунку відправляється запит ("стрижень встановлений"), і можна зробити щось інше. При виконанні запиту ("клюють") - запускається заздалегідь  
підготовлена програмістом функція ( "затягнути спінінг") відображення повідомлення сервера.



Тут сервер відразу повідомляє браузер про те, що запит прийнятий до обробки і відпускає його для подальшої роботи. Коли відповідь буде готова, сервер перешле її, і відповідна функція відображення буде викликатися в браузері, але до тих пір, поки ця відповідь буде сформована і відправлена - браузер вільний.

1. Сенс AJAX

Сенс AJAX полягає в інтеграції технологій

Технологія AJAX використовує комбінацію:

* (X) HTML, CSS для представлення та стилізації інформації
* Модель DOM, якою маніпулює клієнтський JavaScript для забезпечення динамічного відображення та взаємодії з інформацією
* XMLHttpRequest для асинхронного зв'язку з веб-сервером. У деяких Фреймворках AJAX і в деяких ситуаціях xmlHttpRequest замінюється IFrame, тегом SCRIPT або іншими подібними .транспорт
* JSON часто використовується для обміну даними, але підійде будь-який формат, включаючи багатий HTML, текст, XML і навіть деякі EBML.

1. Метод $.ajax

Перш ніж використовувати метод $.ajax в JavaScript, давайте визначимо практичне завдання. Все досить просто - користувач повинен заповнити просту форму, а після цього ми відправимо її на сервер і отримаємо всі заповнені дані у вигляді таблиці. У формі потрібно буде вказати ім'я (логін), електронну пошту і набір знань у вигляді позначених прапорців (прапорців).

1. Оглядачі, які підтримують AJAX

Зауважте, що це загальний список і підтримка застосунків AJAX залежатиме від особливостей підтримки оглядача:

* Версія Microsoft Internet Explorer 5.0 і вище, а також оглядачі, засновані на ньому (версії ОС Mac не підтримуються)
* Оглядач Opera та програмне забезпечення для пристроїв від Opera Software.
* Засновані на Gecko подібно до Mozilla, Mozilla Firefox, SeaMonkey, Camino, Flock, Epiphany, Galeon та версія Netscape 7.1 і вище.
* Оглядачі з вбудованим KHTML API (WebKit) версії 3.2 і вище, зокрема Konqueror 3.2 і вище, версія Apple Safari 1.2 і вище, Google Chrome.