

АJAX-чат на prototype.js



**RAD с помощью библиотеки
prototype.js на примере AJAX-чата**
мастер-класс

Александр Шуркаев
GooDoo IT / Newmedia Stars
www.prototypejs.ru
www.prototypejs.ru,
htmlcoder.visions.ru
alshur@ya.ru

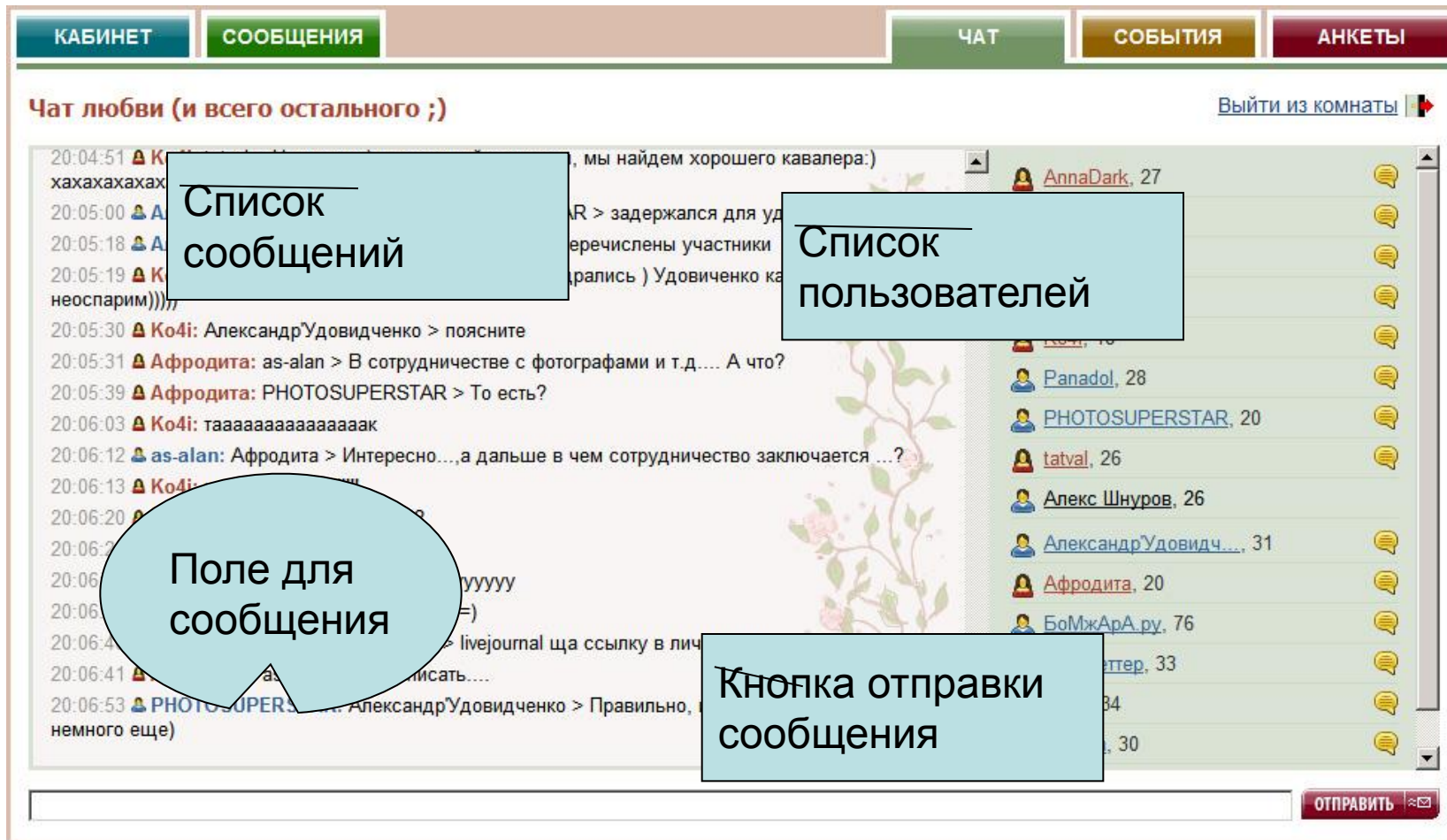
Почему чат на AJAX'е?

- Передача малого количества данных
- Быстрое обновление данных
- Нет необходимости в наличии Java-машины и дополнительного ПО
- Не используются дополнительные порты
- AJAX — это модно :-)

Почему prototype.js?

- Удобная и кроссбраузерная работа с AJAX'ом, DOM'ом
- Готовые расширения JavaScript-объектов
- ООП подход
- Самый популярный framework

Базовый интерфейс чата



The screenshot shows a web-based chat interface with a navigation bar at the top containing buttons for 'КАБИНЕТ', 'СООБЩЕНИЯ', 'ЧАТ', 'СОБЫТИЯ', and 'АНКЕТЫ'. The main chat area is titled 'Чат любви (и всего остального ;)'. The chat history on the left shows messages from users like 'Ko4i', 'Афродита', and 'as-alan'. On the right, a list of users is displayed, including 'AnnaDark, 27', 'Panadol, 28', 'PHOTOSUPERSTAR, 20', 'tatval, 26', 'Алекс Шнуров, 26', 'АлександрУдовидч..., 31', 'Афродита, 20', and 'BoMxApA.py, 76'. At the bottom, there is a text input field and a red 'ОТПРАВИТЬ' button.

Ключевые элементы интерфейса:

- Список сообщений
- Список пользователей
- Поле для сообщения
- Кнопка отправки сообщения

Передача данных в AJAX-чате

- **Что такое JSON и чем он хорош?**
 - Передача сериализованных объектов и массивов
 - Native-поддержка в prototype.js
 - JSON мало «весит»
- **Вообще, есть и другие способы**
 - XML
 - HTML-куски
 - Текст
 - HTTP-заголовки (реализация веб-сервисов)

AJAX-запросы к серверу

- Список новых сообщений
- Список всех пользователей
- Отправка сообщения

АJAX-запросы к серверу

- **Список новых сообщений**

Запрос (GET)	Ответ (JSON)
<pre>/ajax/chat/data.html?room_id=1&stamp=23&rand=0.04169033924254095</pre>	<pre>{"stamp": 24,"messages": [{"time": 1176209243, "text": "Превед!","user_id": 5005}, {"time": 1176209999,"text": "Ха-ха", "user_id": 10}]}</pre>

АJAX-запросы к серверу

■ Список всех пользователей

- Получение дополнительных пользователей, нужных для отрисовки списка сообщений

Запрос (POST)	Ответ (JSON)
<code>/ajax/chat/users.html?room_id=1&extra_user_ids=5005,15&rand=0.12987317360431827</code>	<pre>{{"video_count": 0, "is_extra_user": 1, "is_contact": 0, "sex": 1, "age": 34, "login": "petya", "username": "PPP", "is_friend": 0, "is_ignored": 0, "smallavatar": "/images/0000/g/o/petya/photo/album1/12345.27192_21-sm.jpg", "id": 5005}, {"video_count": 0, "is_extra_user": 1, "is_contact": 1, "sex": 0, "age": 18, "login": "ko4i", "username": "Ko4i-MO3Г", "is_friend": 0, "is_ignored": 0, "smallavatar": "/images/0000/k/o/ko4i/photo/album1/12345.76161_3221-sm.gif", "id": 10}}</pre>

АJAX-запросы к серверу

■ Отправка сообщения

- Объединение отправки и показа новых сообщений
- Буферизация сообщений

Запрос (POST)	Ответ (JSON)
<pre>/ajax/chat/msg.html? msg=Hi!&room_id=1&stamp=554&rand =0.9728927922458538</pre>	<pre>{"stamp": 555, "messages": [{"time": 1176221022, "text": "Hi!", "user_id": 99}]}</pre>

Структура JavaScript-модулей



chat.js (*класс Chat + вспомогательные объекты*)



init.js (*«константы» + Ajax.Responders*)



/lib/



prototype.js (*версия 1.5*)



utils.js (*некоторые вспомогательные функции*)

JS-классы и объекты чата

- **Ключевые используемые prototype.js-классы**
 - PeriodicalExecuter
 - Ajax.Request
 - Template
- **Класс Chat**
- **Вспомогательные объекты**
 - ChatHelpers
 - ChatAjaxHelpers
 - ChatStrings

JS-классы и объекты чата

■ PeriodicalExecuter

```
startListTimer: function(type){  
    if (typeof this[type + '_list_timer'] == 'undefined')  
        this[type + '_list_timer'] = new  
        PeriodicalExecuter(this[('get-' + type +  
        '-list').camelize()].bind(this), this.options[type +  
        '_updater_frequency']);  
    else this[type + '_list_timer'].registerCallback();  
},  
stopListTimer: function(type){  
    if (typeof this[type + '_list_timer'] != 'undefined')  
        this[type + '_list_timer'].stop();  
}
```

JS-классы и объекты чата

▪ Ajax.Request

```
getMsgsList: function() {  
    this.stopListTimer('msgs');  
    new Ajax.Request(  
        this.options.msgs_url,  
        {  
            method: 'get',  
            parameters: 'room_id=' + this.room_id + '&stamp=' +  
this.stamp + ChatAjaxHelpers.addExtraAjaxParams(),  
            onComplete: this.getMsgsListDone.bind(this),  
            onFailure: ChatAjaxHelpers.handleAjaxFailure.bind(this,  
this.options.frontend_base_url)  
        });  
}
```

JS-классы и объекты чата

▪ Template

```
msg_default_tpl: new Template('<p class="#{css_class}"><span  
class="time">#{time}</span> <a href="#{profile_link}"  
target="_blank" title="Посмотреть профиль (в новом  
окне)">#{user_ico}</a><span  
class="#{username_css_class}">#{username}:</span>  
#{text}</p>'),  
msg_system_tpl: new Template('<p class="system-msg"><span  
class="time">#{time}</span> #{text}</p>'),  
msgListItemTemplate: function(vars, type){  
    var type = type || 'default';  
    return ChatHelpers['msg_' + type + '_tpl'].evaluate(vars);  
}
```

JS-классы и объекты чата

- **Класс Chat**

```
var Chat = new Class.create();
```

```
Chat.prototype = {
```

```
  initialize: function(room_id, msg_form, msg_input,  
    msg_submit, msgs_list_div, users_list_div, user_login,  
    options){
```

```
    /* ... */
```

```
  }
```

```
};
```

JS-классы и объекты чата

■ Вспомогательные объекты

- ChatHelpers
- ChatAjaxHelpers
- ChatStrings

```
var ChatStrings = {  
    NA: 'N/A',  
    TO: ' > '  
}
```


Практические приемы

- **Инициализация настроек класса (options)**
- **Приватные данные в классе**
- **Борьба с кэшированием AJAX-запросов**
- **windows-1251 вместо UTF**

Тip: инициализация настроек

```
this.options = {  
  msgs_url: '/ajax/chat/data.html',  
  msg_url: '/ajax/chat/msg.html',  
  users_url: '/ajax/chat/users.html',  
  frontend_base_url: '/chat/?room_id=0',  
  msgs_updater_frequency: 5,  
  users_updater_frequency: 20,  
  max_msgs_in_list: 100,  
  max_chars_in_username: 20  
};  
Object.extend(this.options, options || {});
```

Tip: приватные данные

```
var _users_open_cache = [];  
this.isInUsersOpenCache = function(user_id){  
    return (_users_open_cache.indexOf(user_id) >= 0);  
}  
this.usersOpenCacheAdd = function(user_id){  
    if (!this.isInUsersOpenCache(user_id))  
        _users_open_cache.push(user_id);  
}  
this.usersOpenCacheDelete = function(user_id){  
    _users_open_cache =  
    _users_open_cache.without(user_id);  
}
```

Tip: борьба с кэшированием

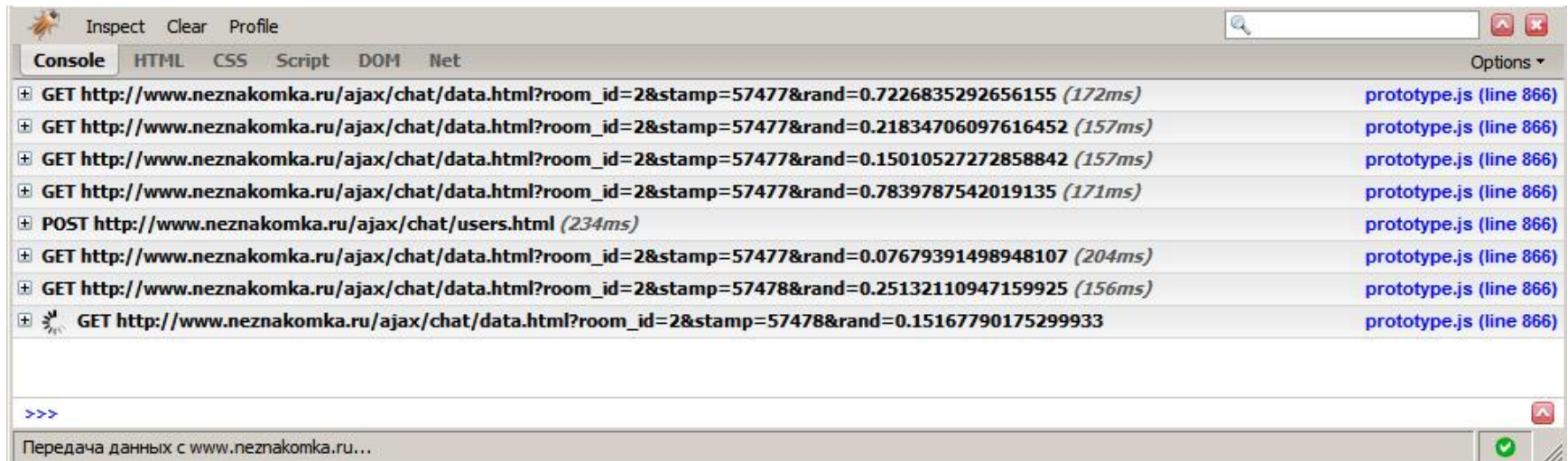
```
var ChatAjaxHelpers = {  
    /* ... */  
    addExtraAjaxParams: function() {  
        return '&rand=' + Math.random();  
    },  
    /* ... */  
}
```

Tip: windows-1251 вместо UTF

```
window.encodeURIComponent = function(str){  
    var trans = [];  
    for (var i = 0x410; i <= 0x44F; i++) trans[i] = i - 0x350; // А-Яа-я  
    trans[0x401] = 0xA8; // Ě  
    trans[0x451] = 0xB8; // ě  
    var ret = [];  
    for (var i = 0; i < str.length; i++){  
        var n = str.charCodeAt(i);  
        if (typeof trans[n] != 'undefined') n = trans[n];  
        if (n <= 0xFF) ret.push(n);  
    }  
    return escape(String.fromCharCode.apply(null, ret)).replace(/\\+/g,  
    '%2B'); // +  
}  
window.decodeURIComponent = function(str){  
    return unescape(str);  
}
```

Отладка чата

- **Обработка исключений и AJAX-ошибок**
 - Объект Ajax.Responders
 - Методы ChatAjaxHelpers.handleAjaxFailure и ChatAjaxHelpers.evalJsonResponse
- **Firebug! Firebug! Firebug!**



Ajax.Responders

```
var ajaxGlobalHandlers = {  
  onException: function(t, e){  
    if (typeof DEBUG !== 'undefined' && DEBUG){  
      var error_info = $H(e);  
      var s = "";  
      error_info.each(function(pair){  
        s += pair.key + ' = ' + pair.value + "\n";  
      });  
      alert(s);  
    }  
  }  
};  
Ajax.Responders.register(ajaxGlobalHandlers);
```

ChatAjaxHelpers.handleAjaxFailure

```
var ChatAjaxHelpers = {  
  /* ... */  
  handleAjaxFailure: function(frontend_base_url,  
    request){  
    if (typeof DEBUG !== 'undefined' && DEBUG)  
      alert('Error: ' + request.status + ' -- ' +  
        request.statusText);  
    else if (request.status == 412) location.reload(); //  
    Precondition Failed  
    else location.href = frontend_base_url;  
  }  
}
```


ChatAjaxHelpers.evalJsonResponse

```
var ChatAjaxHelpers = {  
    /* ... */  
    evalJsonResponse: function(json_str){  
        var data = null;  
        try{  
            data = eval('(' + json_str + ')');  
        }catch(e){  
            if (typeof DEBUG !== 'undefined' && DEBUG) throw  
            {message: 'JSON data is invalid!'};  
        }  
        return data;  
    }  
}
```

Вопросы



Что интересного на сервере?

- **Крутится Perl-демон**
 - Клиент делает запросы по HTTP, через AJAX
 - Демон «узнает» клиента по сессионной куке
 - В ответ на любой из запросов демон отдает текстовый ответ в формате JSON
- **JavaScript-модули gzip'уются**
 - Не стоит особо беспокоиться о большом размере модулей