

# [Обучение Ext JS] :: Часть 9. Окна и диалоги

Мар 04 2010**Поведение**

До сих пор опции настроек для Ext.Msg.show обращались к внешнему виду, но существует несколько опций которые могут изменять поведение. Используя свойство progress, мы можем скопировать стандартный прогресс диалог **Ext JS**:

Ext.Msg.show({progress:true});

используя это в tandemе с другими опциями, такими как title и msg, можно создать свой собственный диалог хода выполнения. Скодным образом опции prompt и multiline позволяют создавать собственные всплывающие окна:

Ext.Msg.show({prompt:true, multiline:true});

Здесь мы создаем всплывающее окно которое принимает несколько строк ввода. Но опуская значение multiline или установив его на false, мы можем ограничить всплывающее окно до одной строки ввода. Снова, используя других опций настроек для Ext.Msg.show позволяет нам расширить приведенный пример когда до полноправного диалогового окна с возможностью ввода.

Другой способ, меняющий поведение всплывающего окна - модальный. Эта опция позволяет уточнять когда пользователь может взаимодействовать с элементами за всплывающим окном. Когда установлено значение true, полуразмытый оверлей не позволит взаимодействовать.

Как мы уже обсуждали ранее, всплывающие окна Ext.Msg не блокируют выполнение скрипта. Это значит, что нам понадобится обратный вызов активирующий код после того как диалог перестанет быть нужным. Это осуществимо используя опцию настройки show's fn, которая вызывается двумя аргументами: ID нажатой кнопки и текстом, введенным в поле текста диалогового окна. Очевидно что для простой подсказки вы не получите текста, но эта функция даст способы обработки обратных вызовов которые будут использоваться во всех диалогах, которые можно создать при помощи Ext.Msg.show.

Мы вкратце коснулись того факта, что окна сообщений Ext.Msg на самом деле измененный Ext.Windows. И если вы думаете, что мы смогли сильно улучшить Ext.Msg, то подождите до тех пор, пока не увидите что для нас может сделать Ext.Window.

## Окна

Любой пользователь компьютера знаком с концептом окон. Информационной панелью, которая появляется на экране для того чтобы предоставить больше данных по действию совершенному пользователем. Мы можем скопировать этот концепт используя класс Ext.Window, мощный компонент, поддерживающий большое количество сценариев.

## Начальные примеры

Открыть окно можно минимальным кодом:

```
var w = new Ext.Window({height:100, width: 200});
```

```
w.show();
```

Его исполнение дает пустое всплывающее окно которое само по себе совершенно бесполезно. Но в нем отображается несколько прописанных по умолчанию особенностей Ext. Window. Сразу установленная, без каких либо настроек ваше окно можно перетаскивать,

изменять размер и очень полезную иконку закрывающую это окно. Не очень внушительная демонстрация, впрочем, это потому, что на самом деле наше окно ничего не показывает.

Самый простой способ наполнить окно - старый добрый **HTML**. Расширенный пример

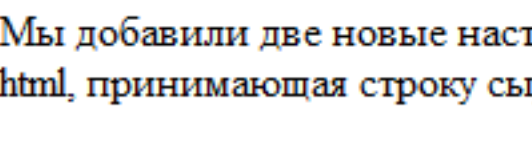
показывающий эту особенность:

Исходный код примера:

# Исходный код примера:

```
var w = new Ext.Window({
    height: 150, width: 200,
    title: 'A Window',
    html: '<h1>Oh</h1><p>HI THERE EVERYONE</p>'
});
w.show();
```

Мы добавили две новые настройки. Первая, опция названия - она позволяет установить текст полосы названия окна и вторая - опция html, принимающая строку сырого **HTML** которая будет отображаться в окне.



Использование такого подхода видно сразу. Мы можем вернуться к основам и внедрить нужный **HTML** непосредственно в зону содержания окна. Это позволит причесать окна так, как нам это требуется и даст кучу **CSS**-ловушек для стайлинга. Даже так, это не совсем то, что мы ожидали от **Ext JS**. Впрочем, другие опции настроек позволяют пойти уже знакомыми маршрутами.

## Потенциал панелей

Помните то, что Window является подклассом Panel, а у Panel есть много интересных штучек в запасе. Элементы опции настройки принимают массив объектов настройки или случаев компонентов:

Исходный код примера:

# Исходный код примера:

```
var w = new Ext.Window({
    items:[
        { xtype: 'textfield', fieldLabel: 'First Name'},
        new Ext.form.TextField({fieldLabel: 'Surname'})
    ]
});
w.show();
```

В вышеприведенном примере мы добавили два textfield, первое использует "ленивую" инициализацию xtype, а второй - стандартную инициализацию объекта. Эти два элемента будут добавлены во внутреннюю панель окна, но способом которым они будут отображаться можно управлять, основываясь на настройках схемы компоновки окна.

## Компоновка

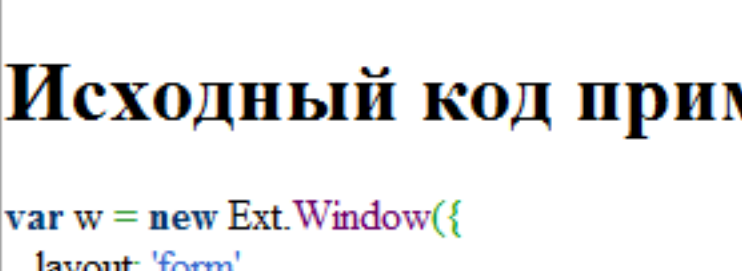
**Ext JS** содержит несколько моделей компоновки в наборе Ext.layout, и каждая из этих компоновок используется с панелью, позволяя размещать в ней элементы. В предыдущем примере мы показали как добавить пару textbox в Window, но мы можем улучшить внешний вид окна используя подходящую компоновку. В этом случае нам понадобится Ext.layout.FormLayout, который поддерживает врылки полей и общие интервалы и размещение:

Исходный код примера:

# Исходный код примера:

```
var w = new Ext.Window({
    layout: 'form',
    items:[
        { xtype: 'textfield', fieldLabel: 'First Name'},
        new Ext.form.TextField({fieldLabel: 'Surname'})
    ]
});
w.show();
```

Мы используем опции настройки компоновки для уточнения чего мы хотим от компоновки формы. И сразу же разница становится очевидной. В следующем рисунке первое диалоговое окно не использует компоновку. Второе же, напротив.



Это не особенность Ext.Window; она идет напрямую из суперкласса Panel. Но то, что Window поддерживает эту особенность очень важно для разработчиков приложений, особенно если вы понимаете как много времени займет создание насыщенной формы, используя технику вставки **HTML**. Другие компоновки, входящие в the Ext.layout дают еще больше возможностей для дизайна окон и расширяют возможный набор сценариев их использования.

## Настройка

В дополнение к различным способам наполнения зоны содержания окна у нас есть обширнейшие возможности для того, чтобы сделать всю систему гибче, настраивая поведение каждого всплывающего окна. Большинство опций настроек предоставлены иерархией суперкласса Ext.Windows.

## Когда я чищу окна

В нашем самом первом примере окна я продемонстрировал как можно получить большое количество особенностей — изменене размера, перетаскивание и кнопку закрытия. Окна с жесткой компоновкой не предназначены для того чтобы им изменяли размер, таким образом мы можем предотвратить такое поведение, выставив соответствующую опцию на false. Перетаскивание чаще является делом личного вкуса каждого, и в большинстве случаев оставить его работать не причинит вреда. Но многие говорят что времена этой опци прошли, да и сам я предпочитаю отключать ее.

Исходный код примера:

# Исходный код примера:

```
var w = new Ext.Window({
    height:50,
    width: 100,
    closable: false,
    draggable: false,
    resizable: false
});
w.show();
```

При использовании окон формы, очень полезно иметь подписи к кнопкам, объясняющие к чему приведет то или иное действие , например сохранит запись, или отменит все сделанные изменения. В таких случаях полезно отключить возможность закрытия окна.

Существует другая опция, дающая чуть больше контроля над этим поведением: closeAction можно выставить либо на hide или close.

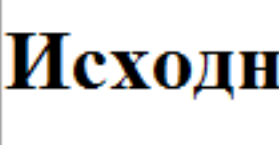
Используя hide окно просто исчезает, но не убивается. А close удалет окно из **DOM**. Это важное различие, в том случае, если вы позже планируете снова использовать это окно. Прятать гораздо более эффективно чем каждый раз пересоздавать окно.

## Дополнения

С подконтрольным функционалом, доступном по умолчанию можно рассмотреть способы дальнейшей оптимизации особенностей Ext.Window. Мы уже продемонстрировали использование опций ширины и высоты выставив фиксированные размеры окна и обрезали любое содержание выходявшее за границы этих размеров.

У нас есть другой способ. Настройки autoHeight и autoWidth, являющиеся логическими, позволяют наполнять окно элементами без необходимости беспокоиться о параметрах высоты и ширины. Проще говоря это очень полезный инструмент. Причем, можно задавать высоту, а ширина будет обесчичаться автоматически. Таким образом вам придется настраивать это самостоятельно. Ext

поскольку необходимо удостоверяться в том, что содержимое не растягивает окно за пределы экрана.




## Desktopping

Самый распространенный пример использования системы окон - компьютерный рабочий стол с множеством окон представляющими приложения или элементы файловой системы. В таких системах пользователям позволяет прятать окна для использования в дальнейшем, или сворачивать их. Так же есть возможность развернуть окно во весь экран. Это нам все хорошо знакомо и поддерживается Ext.Window логические значения настроек maximizable и minimizable.

Эти особенности по умолчанию заблокированы, но ни полностью готовы и работают точно так же как их десктопные эквиваленты. Когда они выставлены на true, в верхнем правом углу окна появляются всем знакомые иконки. Maximizable позволяет развернуть окно на все окно браузера, но с minimizable не все так просто. **Ext JS** не знает куда вы хотите деть окно. В ОС Windows это была бы панель задач.

Но в других операционных системах это совершенно другое место. Таким образом вам придется настраивать это самостоятельно. Ext дает только иконку и событие, которое необходимо обработать. Позже мы приведем краткий пример того, как этого можно добиться используя события доступные классу Ext.Window.

- « [первая](#)
- [← предыдущая](#)
- [1](#)
- [2](#)
- [3](#)
- [4](#)
- [следующая →](#)
- [последняя »](#)

-  [English](#)