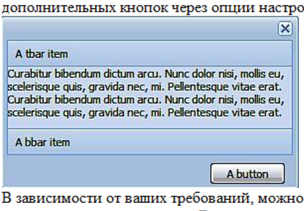
[Обучение Ext JS] :: Часть 9. Окна и диалоги

Мар 04 2010**Дальнейшие настройки**
Окона **Ext JS** поддерживают другой способ сворачивания. И он уже встроен в основу. Логическое collapsible добавляет другую кнопку в верхний правый угол, которая будет сворачивать окно так, что будет видно только полоса названия. Второе нажатие вернет окно на прежнее место.

Также можно использовать настройку expandOnShow для уточнения того, что скрытое окно всегда будет расширено точно на то же самое место. Это полезно в случае если необходимо обратить внимание пользователя на некогда свернутое им окно.

Оформляем наше окно
Наравне со стандартной полосой названия и областью тела содержимого, у нас есть возможность добавить в окно еще больше содержимого. Некоторые из этих областей могут быть полностью изменены, а некоторые все же имеют некоторые ограничения. Но вместе они предоставляют другой способ создания функционального окна.
каждое окно имеет встроенную возможность добавлять панели инструментов вверх и вниз. Эти панели могут включать собственно элементы панели инструментов, кнопки меню или текст. В дополнение мы можем использовать колонтитулы для размещения дополнительных кнопок через опции настройки кнопок.



В зависимости от ваших требований, можно выбрать одну или несколько зон содержания и разместить там все необходимые пользователю инструменты. Типичным примером будет создание окна с компоновкой формы, которая будет включать в колонтитул кнопки Save и Cancel. Это отражает типичный стиль формы ввода данных и будет автоматически, без необходимости вносить серьезные изменения, размещено **Ext JS**.

```
myWin.show('animTarget', function() { alert('Now Showing'); }, this);
```

Сначала мы можем указать элемент, или ID элемента для создания отправной тчки с которой должно анимироваться окно при открытии. Этот косметический эффект может быть так же указан при помощи animateTarget. Второй аргумент - функция обратного вызова, запускаемая когда завершен рендеринг окна. А третий аргумент просто предел обратного вызова.
Очевидным напарником show является hide. В самом деле, он использует те же аргументы, и заставляет окно исчезать с экрана. Если вы уверены, что позже вы не хотите использовать это окно, то возможно лучше будет воспользоваться close, который удалит его из **DOM**, тем самым уничтожив.
Функционал предоставляемый close и hide уже был продемонстрирован. Существуют другие методы, позволяющие программно управлять элементами которые мы уже рассмотрели (например minimize и maximize). Их функционал расширен restore, возвращающим окно в те размеры которые у него были до сворачивания и toggleMaximize, являющийся простым ярыком на переключение между maximize и restore. А говоря о возвращении окна к прежнему состоянию, следует упомянуть метод center, устанавливающий окно ровно посередине окна просмотра браузера.
Мы и дальше можем управлять расположением наших окон: alignTo позволяет разработчику программно перемещать окно к определенному элементу
У этого метода есть три параметра:

- Элемент align to— как строковый или полный элемент
- Позиция align with— описывается в документации Element.alignTo, и кратко обсуждается в Главе 10 —Эффекты
- Смещение позиционирования, определенное как массив [x,y]

Этот метод полезен в том случае, когда есть приложение с динамическим рабочим местом и вам надо удостовериться, что все окна появляются в правильном месте по отношению к другим отображаемым элементам. В таком случае пригодится метод anchorTo, который принимает те же самые аргументы и позволяет окну оставаться привязанным к другим элементам. Даже когда окно браузера меняет размер или прокручивается.
В то время как большинство методов Ext.Windowдают разработчику доступ к существующему функционалу через свой код, существуют и такие, которые предоставляеют улучшенные сценарии или особенности, которые будут полезны для кода.

События
У большинства из рассмотренных нами действий есть собственные события, которые одновременно служат как ловушки для пользовательского кода. Minimize является тем, который эксплицитно упоминался ранее, поскольку необходимо обрабатывать это событие, если вы хотите чтобы иконка уменьшения делала хоть что-то. В идеале вам следует ожидать того, что окно будет храниться в подобии панели задач.

Исходный код примера:

Исходный код примера:

```
var w = new Ext.Window({
    height: 50,
    width: 100,
    minimizable: true
});
w.on('minimize', doMin);
w.show();
```

В вышеприведенном примере мы создаем новое окно, которое определили как уменьшаемое, а затем слушатель события для уменьшения. Затем мы выводим окно на экране. Наша функция обработки события doMin фыглядит так:

```
function doMin() {
   w.collapse(false);
   w.alignTo(document.body, 'bl-bl');
}
```

Мы просто говорим нашему окну сворачиваться в панель названия, а затем используем метод alignTo, который мы обсуждали ранее. С выбранными параметрами нижний левы угол окна будет выравнен по нижнему левому краю документа.
Разумеется, со следующими окнами, весь нижний девый угол будет забить перекрывающимися окнами, что в настоящем приложении не приветствуется. Тем не менее это показывает работу события minimize.

Обработка состояния
Сокнами обработкасостояний довольно простое дело. Окна полностью обеспечены стандартными средствами управления состоянием **Ext JS** через флаг настроек состояний. Принимая во внимание, что мы хотим отследить позицию, статус минимизации, размер и упорядоченность окон по мере обновления страницы и пользовательской сессии. Код будет следующим:

```
var w = new Ext.Window({
    var w = new Ext.Window({
    stateful: true,
    stateevents:['resize'] // track the resize event
});
```

Мы используем опцию stateevents для установки массива событий окна, которые заставят компонент сохранить текущее состояние. В отличии от обработки состояний с TreeNodes, нам не надо использовать для ловушки события beforestatesave. Компонент окна позаботится о себе и автоматически установит настройки для отображения предыдущего состояния.

Управление окнами
В наполненном Internet-приложении может понадобиться разрешить пользователю иметь несколько открытых окон, просматривая различную информацию без постраничной навигации. **Ext JS** допускает это: можно создать любое количество немодальных окон и управлять ими как вам удобно. Тем не менее мы сталкиваемся с проблемой, когда пользуемся множеством окон — как управлять ими в виде группы? Например пользователю может захотеться очистить рабочее место свернув все открытые окна. Этого, и даже большего), можно достигнуть используя группу окон.

Поведение диспетчера окна по умолчанию
При создании Ext.Window, оно присоединяется к группе по умолчанию, на которую (опять же по умолчанию) можно сослаться используя класс Ext.WindowMgr. Тем не менее, можно создать столько дополнительных WindowGroups, сколько потребует приложение и назначить в них окна через опции настроек диспетчера.
Почему вашему приложению может понадобиться метод группировки окон? Ну, несколько окон упорядочены и могут быть сложены одно поверх другого. Применяя WindowMgr.bringToFront мы выводим одно окно на первый план, к примеру если оно было обновлено, и мы хотим привлечь к нему внимание пользователя. В некоторых случаях это будет почти то же самое что и использовать Ext.Window.toFront. Тем не менее, подход WindowMgr уважает упорядоченность индивидуальных WindowGroups. И потому, это намного более безопасный способ если вы создаете большое приложение с огромным количеством окон.
Группировка и упорядоченность окон часто вводит в заблуждение, так что давайте используем жти особенности в настоящем контексте.