# [Обучение Ext JS] :: Часть 10. Эффекты в ExtJS

Мар 14 2010**Huffing and puffing**
Метод дыма Ext.Fx старается передать то, как выглядит медленно тающее облако дыма. В то время как некоторые из эффектов Ext.Fx используются для выделения областей, дым нужен для того чтобы убирать элементы с экрана. Для этого элемент расширяется, одновременно становясь все более прозрачным, и так до тех пор пока он совсем не исчезнет. У дыме нет собственных опций, так что его использование очевидно:

```
Ext.get('target').puff();
```

Дым можно использовать совместно с быстрым переходом, для того чтобы убирать окна с рабочего пространства, или более медленный переход может проиллюстрировать удаление элемента, возможно изображения из галереи. Тем не менее стоит обратить внимание на один нюанс: когда дым используется в обработке документов, он будет влиять на элементы вокруг себя.
Хотя абсолютно-позиционированные окне не влияют на окружение (в нашем примере галереи), существует вероятность того, что при использовании дыма изменение размеров вытолкнет изображения.
Разработчики знакомые с побочными эффектами связанными с использованием Ext.Fx.puff могут добавить его в свой набор инструментов, получая интересный способ создания динамического перехода в своих приложениях.

### Изменение масштаба высоты Ext JS

Изменение масштаба один из немногих эффектов **Ext JS** у которого есть особое назначение. Он изменяет размер элемента, используя гладкий переход, который есть у большинства приложений. Мы можем расширить текстовые области, скопировать эффекты увеличения и восстановления с которыми сталкивались в главе посвященной окнам, можно изменять фокус рабочего стола, уменьшая одну часть и увеличивая другую. В основном использование Ext.Fx.scale выглядит так:

```
Ext.get('target').scale(50, 150);
```

В этом примере целевому элементу будет изменен на 50 пикселей в ширину и 150 в высоту. Также эти значения могут быть нулевыми, и значит этот параметр меняться не будет. Пригодится если вы захотите изменить только одно измерение.

**ВАЖНО:** *Ext.Fx.scale работает только с теми элементами,у которых стиль отображения выставлен на блочный, позволяющий изменять высоту и ширину. В случае использования инлайн-элементов сообщение об ошибке не будет, но и изменений вы тоже не увидите.*

### Плавное движение

В рамках **Ext JS**, движением называют переход элементов пока они не скроются из виду. Очень похоже на эффект вызываемый методом Ext.Fx.ghost, который мы уже видели в действии. Но вызывается он Ext.Fx.slideIn и Ext.Fx.slideOut.
Мы можем использовать эти методы или для того чтобы избавиться от ставшего ненужным элемента или ввести новый элемент. Очевидным приложением для этого будет удаление записей, где новые элементы slideIn (вводятся) в список, а удаленные slideOut (выводятся).

```
Ext.get('target').slideOut();
window.setTimeout(function() {
   Ext.get('target').slideIn();
}, 3000);
```

В вышеприведенном примере мы убираем нашу цель из виду, и через три секунды возвращаем ее. По умолчанию элементы будут вводится и выводится через верхний край, но первый аргумент в обоих функциях позволяет использовать якоря для уточнения направления движения:

```
Ext.get('target').slideOut('tr');
```

Здесь мы указываем что цель будет выводится через верхний правый угол. Как и с Ext.Fx.puff нужно убедиться, что окружающие элементы могут быть сдвинуты со своих мест.

### переключение с видимого на невидимое

Метод switchOff по другому означает, удаление элемента с экрана.. Это совмещает два действия: сначала элемент становится прозрачным, обозначая начало эффекта, а затем он сворачивается сверху вниз до полного исчезновения. Этот двухэтапный эффект можно использовать для привлечения внимания пользователя прежде чем окно окно окончательно исчезнет. Можно использовать Ext.Fx.switchOff:

```
Ext.get('target').switchOff();
```

Как вы видите, здесь нет уникальных аргументов, таким образом кажется что очень простой в использовании эффект. Но позже мы обсудим некоторые дополнительные настройки, доступные для всех методов Ext.Fx — секрет позволяющий в совершенстве управлять эффектами.
SwitchOff становится довольно полезным в тех ситуациях, когда вы хотите заставить элементы исчезнуть без участия пользователя. Например, у вас есть программа отслеживающее количество посетителей на сайте, и когда пользователь выходит, его имя удаляется из списка. Не простой переход с экрана, которое сначала выделяется. При использовании switchOff следует короткая вспышка, обозначающее что сейчас что-то будет, и затем собственно переход с исчезновением.

### Сдвиг

У этого нет никаких эффектов. Вместо этого сдвиг надо доукомплектовать буквенным содержимым:
Одного или более значений из следующих:

- ширина
- высота
- x
- y

Установив одно из этих значений и отправив его в сдвиг мы заставим целевой
элемент переместиться на координаты x,y и изменить размер на указанные значения
— в той же гладкой манере присущей Ext.Fx. Вот
пример этого:

Исходный код примера:

## Исходный код примера:

```
Ext.get('target').shift({
   x: 5,
   y: 300,
   width: 300,
   height: 200
});
```

В этом примере наш целевой элемент переместится на пять пикселей от левого края, на 300 пикселей от верхнего. При этом размер изменится на 300x200 пикселей. Эти изменения происходят одновременно, по мере перемещения изменяется и размер.
В сущности, сдвиг это вариант Ext.Fx.scale добавляющий способность извлекать элемент. Это отличный механизм для переорганизации рабочего пространства — сдвиг панелей с данными с гладкой, динамичной манере.

### А теперь немного интересных штучек

Все рассмотренные методы имеют некоторые сходства. Но у каждого из них есть ощутимые различия, которые позволяют применять их для решения различных вопросов. Теперь мы рассмотрим как можно дополнительная способность Ext.Fx может улучшить уже рассмотренные эффекты.
Во многих вышеприведенных примерах функциональность слегка ограничена. Что было упущено до настоящего момента, заключается в том, что все рассмотренные методы могут принимать стандартный объект настройки, предоставляющую место для включения полного функционала.

### Fx

Мы слегка коснулись способа которым используются опции якоря в рамках Ext.Fx, например такие методы как призрак, принимающий строки, указывающие направление движения. Через несколько страниц мы не только досконально обсудим это, но и пройдемся по огромному количеству опций настройки, являющимися общими для всех методов Ext.Fx.

### Якоря в Ext

Указывая направление, якоря, выравнивание и многие другое зависит на схеме расположения якорей. Они используются в анимационной системой **Ext JS** для определения направления движения и у них довольно простые названия:

| Строка положения якоря | Описание |
|---|---|
| tl | Верхний левый угол |
| t | Центр верхнего края |
| tr | Правый верхний угол |
| l | Центр левого края |
| r | Центр правого края |
| bl | Нижний левый угол |
| b | Центр нижнего края |
| br | Нижний правый угол |

Эти опции позволяют восьмистороннее движение при использовании таких методов как Ext.Fx.ghost. Но точно такой же концепт можно заметить в таких методах как Ext.Element.alignTo и Ext.Element.anchorTo, где две якорные точки можно совмещать, и знак "?" используется для указания выравнивания по окну показа.
Вкратце, стоит ознакомиться с якорными позициями Ext JS, поскольку вы безусловно будете использовать их в своих приложениях — либо для того чтобы перемещать элементы, или согнуть их, чтобы они стали отвечать вашим требованиям. Более того в Ext.Components существуют основные элементы. Таким образом вы с большой долей вероятности эти особенности в разработке наполненных виджетов.
Теперь назад к теме — приручение класса Ext.Fx.

### Опции

как упоминалось ранее, каждый из рассмотренных методов Ext.Fx может взять опциональную буквенную настройку в качестве последнего параметра, что дает нам 11 настроек для улучшения эффектов. Самый простой это, наверное, длительность, которая указывает как долго должен длиться эффект. Полезно в случае, если вы хотите убедиться, что ваш переход длиться достаточно долго для того, чтобы пользователь мог его заметить. Хотя значение стоящее по умолчанию для каждого эффекта довольно чувствительно, существует много случаев для его перезаписи. Вот пример использования длительности и действительно, опция настройки аргумента в общем:

```
Ext.get('target').switchOff({
   duration: 10
});
```

В этом примере мы передали стандартный буквенный объект методу switchOff, и свойство длительности установлено на 10, обозначая что switchOff должно произойти через 10 секунд.
С поведением по умолчанию некоторых из методов Ext.Fx methods у вас получится что-то похожее на почти идеальный переход. Например многие их эффектов заставляют элемент исчезать, оставляя большую прореху на том месте где он располагался. Это происходит потому, что они завершают переход делая элемент невидимым, и не удаляют его из **DOM** или потока документов. Исправить это можно используя еще две опции: remove и useDisplay.
Первая, remove, означает, что мы хотим полностью уничтожить элемент по завершении перехода. Вторая, useDisplay, означает что элемент будет скрыт посредством display:none а не visibility:hidden, таким образом это не затронет узлы по соседству. Использовать совместно remove и useDisplay излишне. Если узел полностью удален, то нет никакой разницы является ли он скрытым. Если же вы хотите в последствии снова его использовать то ваш выбор определенно useDisplay. В другом случае, если вы хотите чтобы все было чисто (и избежать утечек памяти) используйте remove.
Опции настроек также указывают нам что еще происходит когда эффекты завершены. у нас есть две опции отвечающие за внешний вид элементов: afterCls и afterStyle, позволяющие классу **CSS** или сырым правилам **CSS** применяться к элементу в конце перехода. Это можно использовать для добавления в элемент постоянных изменений, возможно выделяя как новый предмет.
У нас есть еще две опции, которые позволяют выполнить дальнейшие действия с завершенными эффектами: callback, позволяющая нам указать вызываемую функции. в конце перехода, и scope, которая является приближением функции callback.
Например:

```
Ext.get('target').switchOff({
   callback: function() {
      alert('Effect complete!');
   }
});
```

Класс Ext.Fx обрабатывает запросы эффектов автоматически, так что не используйте это для запуска другого эффекта, вместо этого используйте действия запуска связанные с анимацией.

### Тише едешь - дальше будешь

Наша следующая опция настройки easing (смягчение). В терминологии анимации, смягчение относится к средствам, при помощи которых переход начинает и прекращает движение. По умолчанию эффекты **Ext JS** будут начинаться и заканчиваться весьма отрывисто, но используя эту опцию мы можем указать как это можно ускорить или замедлить.

**ВАЖНО:** *Эффекты смягчения поддерживаемые **Ext JS** весьма разнятся в зависимости от того, какой адаптер вы используете. Если вы используете адаптер **Ext JS** Base, то у вас будет все работать.*

простым примером этого действия является значение easeBoth. Оно заставляет анимацию постепенно ускоряться, а затем замедляться для того, чтобы обеспечить гладко выглядящий переход:

```
Ext.get('target').scale(50, 150, {easing: 'easeBoth'});
```

Здесь мы применяем опцию смягчения для эффекта изменения размера, и используя easeBoth вовремя выполнения этого у нас меньше рвущихся движений. Существует много смягчающих опций::

- easeNone
- easeIn
- easeOut
- easeBoth
- easeInStrong
- easeOutStrong
- easeBothStrong
- elasticIn
- elasticOut
- elasticBoth
- backIn
- backOut
- backBoth
- bounceIn
- bounceOut
- bounceBoth

Это список опций поддерживаемых **Ext JS Base**. Одни, вроде bounceInнеплохо оживляют эффекты, что возможно, было бы полезно в игровом приложении, а не вводе данных, но ведь в перспективе у нас - иметь большой выбор с возможностью найти решение для любой ситуации.
Последние несколько опций применяются к тому, как эффекты могут взаимодействовать друг с другом за время своего существования. Например блок укажет что никакие другие эффекты не запланированы, в то время как работает именно этот эффект. Полезно для случаев где дальнейшие изменения могут серьезно повлиять на состояние приложения. Опция блока говорит другим эффектам "сбавить темп" до тех пор пока все не успокоится.

```
Ext.get('target').shift({
   x: 5,
   y: 300,
   width: 300,
   height: 200,
   block: true
});
```

Здесь мы видим интересное использование метода Ext.Fx.shift при котором опции сдвига усилены обычными опциями Fx.
У наст так же есть stopFx и сопутствующие опции. StopFx позволяет нам отменить все задействованные эффекты после этого, в то время как сопутствующие укажут будут ли последующие запущены параллельно, или по очереди. По умолчанию стоит последнее, но порой случается так, что необходимо применить несколько эффектов одновременно, и в следующих разделах мы поговорим об этом подробнее.

### Множественные эффекты

Большую часть времени вы используете эффекты по одному, на разные элементы, которые вы просматриваете. Но при некоторых обстоятельствах может быть весьма полезно уметь использовать множественные эффекты. Существуют разные способы для обработки таких случаев, и в следующих разделах мы внимательно рассмотрим эти способы. А также узнаем как влиять на запущенные эффекты из кода.

### Сцепление

Простым способом запускать второй эффект после завершения первого, это использовать метод сцепления. Поскольку каждый из главных методов Ext.Fx возвращает Ext.Element который был целью эффекта, вы можете вызвать их дальнейшими методами Ext.Element, включая предоставляемые Ext.Fx:

```
Ext.get('target').slideIn().highlight();
```

как видно из примера, мы вызываем slideIn за которой следует выделение, а значит выделение будет добавлено в очередь эффектов и запущено после завершения slideIn.

### Очередность

очередность, это поведение по умолчанию для эффектов установленных либо методом сцепления, либо несколькими вызовами. Следующий пример когда по функционалу равен предыдущему:

```
Ext.get('target').slideIn();
Ext.get('target').highlight();
```

В обоих случаях эффект выделения был запланирован после slideIn. Это не вызывает необходимости сложных расчетов времени или настроек по обратному вызову для запуска эффекта.

### Одновременность

Хотя очередность и устанавливается по умолчанию, у нас есть возможность запускать эффекты параллельно используя уже обсуждавшуюся опцию настройки. Это дает возможность совместить эффекты, например выделение и изменение размера:

```
Ext.get('target').scale(300, 200 ,{
   concurrent:true
}).highlight();
```

Это очень сильный пример настройки.

- « первая
- ‹ предыдущая
- 1
- 2
- 3
- следующая ›
- последняя »