


[Обучение Ext JS] :: Часть 11. Drag-and-Drop

Мар 16 2010
 

мире разработки программного обеспечения, приложения рабочего стола все еще на первом месте. Хотя веб-сайты, а позднее и веб-приложения, развиваются быстрыми темпами используя такие основы как **Ext JS**, по уровню сложности им еще очень далеко до того, что мы видим в большинстве десктопных программ.

 Drag-and-drop, предмет обсуждения этой главы, является отличным примером вышесказанного. Перемещение предметов по экрану используя мышь, само-собой разумеющееся на стандартном рабочем столе компьютера. И кроме того существует почти во всех приложениях, пусть порой и весьма урезано.

Многое объясняется тем, что очень трудно реализовать drag-and-drop используя **JavaScript**. Очень трудно заставить правильно работать (и более того настроить!) перетаскиваемые виджеты даже для одного браузера. а корректное отображение во всех (или большинстве, наиболее популярных) превращается в настоящую мороку.

 В начале революции **Web 2.0**, drag-and-drop стал получать все большее распространение — обходя большую часть трудностей. Но даже тогда Сеть не начала использовать это так же легко и свободно как на рабочем столе. Возможно по причине плохого применения и демонстрации этой функции. В конце-концов, неужели кто-то действительно захочет работать с онлайн-магазином при помощи drag-and-drop?

Drop what you're doing

Введение drag-and-drop предоставляемого **Ext JS** является многоуровневым успехом. Он кросс-браузерный и легок в пользовании, но что более важно, он может использоваться для нескольких виджетов в границах основы, например таких как GridPanel и TreePanel, позволяя таким образом встраивать по настоящему полезные приложения.

 В дополнение, набор Ext.dd содержит рд классов, позволяющих с легкостью создавать простые особенности — продуктовые тележки входят в комплект. Существует достаточная степень разбиения для поддержки более продвинуты сценариев, как впрочем и предоставление средств для создания собственного решения drag-and-drop, которое может быть полностью встроено в ваше приложение **Ext**.

 В этой главе мы рассмотрим набор Ext.dd и то как он может расширить ваш инструментарий, но кроме этого мы посмотрим как расширить функциональность drag-and-drop предоставляемую различными компонентами **Ext**.

Life's a drag

Существуют две четкие части участвующие в создании рабочего drag-and-drop — перетаскивание и отпускание.Так как это две отдельные операции, для нашего простого случая мы обработаем их раздельно.

Находя решение

первая часть на которую нам стоит взглянуть, это действие перетаскивания. **Ext** включает ее при помощи Ext.dd.DragSource, который на самом деле заставляет все замереть. Убедитесь что на вашей странице есть элемент <div> с ID, dragMe:


```
new Ext.dd.DragSource("dragMe");
```

 Вот и все!



Запустите этот код и захватите элемент. Теперь вы можете перетаскивать его по странице, но надо сделать еще кое-что. **Ext JS** дает несколько дополнительных особенностей по умолчанию, они будут показаны в этом коде.

Приближение

Кода вы перетаскиваете элемент, вы видите что это сам элемент, а автоматически создаваемый **Ext** заместитель — легковесное представление любого перетаскиваемого элемента. Эта особенность предоставляется классом Ext.dd.StatusProxy.

 Если используется элемент с небольшим текстом внутри, вы увидите, что **Ext** использует его как содержание для статуса заместителя. Но это максимум из того, что можно добиться от **Ext** при создании копии элемента для визуализации перетаскивания. Размеры вашего элемента будут сброшены, и заместитель будет пытаться вставить содержимое по максимуму своих возможностей. Но если у вас в нем хранится много данных, то заместитель может оказаться беспомощным.

 каково же преимущество использования заместителя статуса, а не самого элемента? Ну, дело в том, что как видно из нашего демонстрационного кода, **Ext** создает маленькую запрещающую иконку когда вы перетаскиваете заместитель. Она меняется на зеленую галочку, если выбранное место подходит для размещения (это мы еще увидим). Но самое главное, что при использовании заместителя эти индикаторы не надо настраивать.



Другое преимущество использования заместителя статуса вместо настоящего элемента заключается в том, что так гораздо проще. Как в отношении ресурсов, так и практического применения. Представьте, что вы перетаскиваете таблицу на 10 столбцов и 200 рядов. Заместитель дает изящное и аккуратное решение таких проблем, а также предоставляет небольшой бонус, про который я расскажу чуть позже.

Щелк!

Другой встроенной особенностью Ext.dd.DragSource является действие, происходящее когда вы отпускаете элемент на место, которое ему не подходит. То самое когда у заместителя красный значок. Оно немедленно "отщелкивается" на свое прежнее место, выделяя оригинал светло-голубым для оповещения пользователя о возвращении элемента на старое место.

Брось меня

Теперь, когда мы смогли что-то такое там перетащить, можно переходить ко второй части нашего плана — отпусканию. Идея заключается в том, что у нас есть цель отпускания — элемент, включающий зеленую галочку.

 Для того чтобы это сделать, надо использовать класс Ext.dd.DropTarget, что почти то же самое, что и DragTarget:


```
new Ext.dd.DropTarget("dropHere");
```

Коструктор DropTarget принимает смешанный объект элемента — в этом случае ID элемента на нашей странице — и определяет, что элемент является верной целью для любой операции отпускания. Исполнение этих двух примеров вместе продемонстрирует изменения статуса иконки перетаскиваемого заместителя с красной на зеленую когда она будет над местом сброса.

Но стойте: Ничего не происходит!

Самое хитрое в этом виде простого drag-and-drop, то, что когда вы выполняете действие, ничего не происходит. Фактически перетаскиваемый элемент возвращается на старое место, как это было при красном значке.

 Трюк тут в том, что надо создать функцию, переписывающую метод DropTarget.notifyDrop. По умолчанию этот метод не имеет применения и просто возвращает перетаскиваемые элементы на изначальные места. Но дописав немного кода мы можем закончить перенос в любое доступное нам место.

ВАЖНО: Почему необходим этот шаг? Конечно ***Ext JS*** должен знать что мы хотим перетащить вот это вон туда. Существует множество вариаций drag-and-drop, таким образом ***Ext*** не может понять что именно вы собираетесь сделать. Поэтому он заставляет вас добавить действий в операцию drag-and-drop.

Мы собираемся рассмотреть некоторые примеры кода, которые цепляются за эти особенности поддержки drag-and-drop создавая простой пример заставляющий перетаскиваемый элемент вставать в границы места сброса. Возможно именно это вы хотели бы видеть по умолчанию. Хотя само по себе оно не так уж и полезно, но определенно заполнит какие-то пробелы.

 Учитывая, что у нас тот же самый код что и в прошлых примерах с DragSource и DropTarget, нам нужно добавить notifyDrop в DropTarget:


```
new Ext.dd.DragSource('drag');
new Ext.dd.DropTarget('drop', {
    notifyDrop: function() { return true; }
});
```

Это простое добавление, которое переписывает реализацию по умолчанию notifyDrop, возвращая true и делает подчеркивает, что мы не хотим чтобы перетаскиваемый элемент возвращался на старое место. Этого вполне хватает чтобы вполнину удовлетворить наши ожидания от того, что должен делать код drag-and-drop. Источник перетаскивания может быть опущен над целью сброса без возвращения. Тем не менее нам надо заставить элемент разместиться в конечном месте. Вскоре мы узнаем как это делается.

Interacting the fool

В то время как предыдущие примеры интересны в довольно отвлеченном смысле, — они дают нам понять как работает система **Ext JS** drag-and-drop— они не снизят ваши разработки. Для этого нам надо полагать как могут взаимодействовать несколько перетаскиваемых элементов и целей сброса. И то, как мы можем настроить более сложную систему перетаскивая данные.

- 1
- 2
- 3
- 4
- [следующая >](#)
- [последняя »](#)