[Обучение Ext JS] :: Часть 2. Основы Ext

Фев 17 2010 этой главе познакомимся с приложениями **Ext**, создав несколько диалоговых окон которые будут взаимодействовать друг с другом, пользователем и страницей. Мы, используя, onReady, MessageBox, научимся создавать различные виды диалоговых окон изменять **HTML** и стили нашей страницы. Более того, в этой главе мы будем:

- Выяснять как легче всего настроить приложения **Ext**

- Ждать пока **DOM** (объектная модель документов) не будет доступна для взаимодействия
- Использовать диалоговые окна с целью выяснить намерения пользователя, и чем он хочет заняться
- Динамично изменять **HTML** и **CSS** настранице, в ответ на действия пользователя

Начнем с рассмотрения нескольких основных функций **Ext**. Мы изучим как работает приведенный в первой главе пример и доработаем его. Для этого мы воспользуемся теми функциями, которые нам будут нужны на протяжении всего курса:

- Ext.onReady: Эта функция удостоверяется в том, что документ готов к обработке
- Ext.Msg: Эта функция создает нам программно выглядящие сообщения
- configuration objects: Эта функция определяет как будут взаимодействовать приложения
- Ext.get: Эта функция работает с элементами DOM

На старт, внимание, марш!

В этой части мы внимательно рассмотрим событие onReady — первое с чем вам стоит разобраться, если вы решили работать в **Ext**. Мы научимся отображать различные типы диалоговых окон и как отвечать на пользовательские действия с ними. Прежде чем перейти к этому, нам нужно рассмотреть некоторые основные правила работы с **Ext**.

Разделительное изображение

Прежде чем мы перейдем к дальнейшим пунктам, нам стоит снабдить **Ext** необходимой ему вещью — разделительным изображением. **Ext** нужно прозрачное изображение формата GIF, размером пиксель на пиксель. Нам нужно прописать место этого изображения используя следующую строку:

```
Ext.onReady(function(){
   Ext.BLANK_IMAGE_URL = 'images/s.gif';
});
```

Вы наверное задаетесь вопросом "А зачем вообще это нужно?". Пользовательский интерфейс **Ext** создается при помощи CSS, скриптам же необходимо оформлять базовые элементы HTML. Единственным элементом с точным размером является наше изображение. Таким образом изображение используется для того чтобы определять каким образом отрисовываются компоненты **Ext**. Это, кстати, является частью ее кросс-браузерности.

Приложение

В Ext существует много "приложений". В их число входят такие элементы как окна сообщений, сетки и еще много всего, что помогает создавать удобный интерфейс. Я предпочитаю рассматривать многие компоненты, например onReady как основные, а приложениями считаю все, носящие более узкоспециализированный характер mdash; к примеру сетка, которая представляет табличные данные другому пользователю.

Время действовать

Давайте создадим новую страницу (или просто изменим ту, которую вы создали в прошлой главе) и добавим в нее код, взывающий диалоговое окно когда страница готова:

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
   Ext.BLANK_IMAGE_URL = 'images/s.gif';
   Ext.Msg.show({
      title: 'Milton',
      msg: 'Have you seen my stapler?',
      buttons: {
        yes: true,
        no: true,
        cancel: true
      }
   });
});
```

Как и в прошлой главе, мы разместили наш код в пределах функции *onReady*. После этого мы можем приступать к созданию диалогового окна, и последующему его изменению посредством объекта *конфигурации*. Для этого диалога объект использует ри элемента. Последний из которыз - вложенный объет для трех кнопок. Вот как наш коод теперь выглядит в браузере:

Это выглядит как очень простое диалоговое окно, но если мы начнем щелкать по нему мышкой, вот тут-то и проступит функционал **Ext**. Окно можно перетаскивать по экрану совсем как окно десктопного приложения. Так же есть встроенная кнопка закрытия, нажатие клавиши *Escape* при активном окне или кнопки **Cancel** закроет его.

Это что сейчас было?

Давайте рассмотрим две использованные нами функции поближе:

- *Ext.onReady:* Эта функция помогает коду подождать до тех пор, пока не будет активна служба **DOM**. Это необходимо по той простой причине, что **JavaScript** начинает исполнять код сразу же, как только открывается документ, в котором наши элементы **DOM** могут и не существовать.
- *Ext.Msg.show:* Это основная функция используемая для создания диалогового окна. Она позаботится обо всем, что необходимо для корректной его работы. Также есть способы, которые помогут вам сэкономить время. Их мы рассмотрим чуть позже.

Использование onReady

Пришло время детального рассмотрения использованного нами кода.

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
   Ext.Msg.show({
      title: 'Milton',
      msg: 'Have you seen my stapler?',
      buttons: {
        yes: true,
        no: true,
        cancel: true
      }
   });
});
```

Функция *onReady* использована нами для того, чтобы заставить код ждать до тех пор, пока страница не будет готова. Аргумент попадающий в *onReady* является функцией, и может быть передан как название функции, или создан потоково. Как мы и сделали в прмере кода. Результатом этого способа являются анонимные функции, которые используются один раз.
Если вы собираетесь выполнять функцию многократно, тогда ее можно определить и назвать как-нибудь так:

Исходный код примера:

Исходный код примера:

```
function stapler(){
   Ext.Msg.show({
      title: 'Milton',
      msg: 'Have you seen my stapler?',
      buttons: {
        yes: true,
        no: true,
        cancel: true
      }
   });
}
Ext.onReady(stapler);
```

Когда мы будем надстраивать наше приложение мы не будем использовать большое количество анонимных функций. Вместо этого мы создадим функции многократного использования.

ВАЖНО: *В коде кнопок так же можно изменять надписи на них. Вместо использования булевого значения, впишите желаемый текст, например: {yes: 'Maybe'}.*

Еще чудеса приложений

Вернемся к созданию нашего небольшого приложения и добавм иконку и кнопки. Это можно сделать добавив стиль или изменив конфигурацию, чтобы поставить код иконуи рядом с кодом кнопок.
Сначала мы обсудим какие **CSS** нам необходимы. Добавьте следующий код в документ заключив в тег стиля:

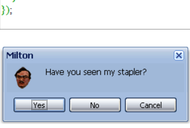
```
.milton-icon {
  background: url(milton-head-icon.png) no-repeat;
}
```

Также нам необходимо внести изменения в конфигурацию нашего приложения. Записи иконки необходимо название стиля как значение milton-icon. Мы также прописали функцию, которая будет выполняться если пользователь нажмет на любую из конопок. Эта функция является анонимной и используется чтобы передавать переменные:

Исходный код примера:

Исходный код примера:

```
Ext.Msg.show({
   title: 'Milton',
   msg: 'Have you seen my stapler?',
   buttons: {
      yes: true,
      no: true,
      cancel: true
   },
   icon: 'milton-icon',
   fn: function(btn) {
      Ext.Msg.alert('You Clicked', btn);
   }
});
```



В нашем случае, у функции всего один параметр, которым является название кнопки которую нажали. Таким образом, если пользователь нажал кнопку Yes, то переменной btn присвоиться значение yes. Используя пример кода мы берем названия кнопок и отправляем их в оповещение в качестве сообщения.

ВАЖНО: *Встроенный функционал заботится о том, чтобы кнопка Cancel, иконка close в верхем правом углу, и клавиша Esc связаны вместе для выполнения одного действия отмены. Это один из многих способов, которыми* ***Ext*** *делает процесс кодирования легче.*