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

Фев 17 2010**Познакомьтесь с JSON и конфигурацией объекта**
В нашем примере мы используем то, что называется конфигурацией объекта, и что является основным способом заставить **Ext** сделать то, что нам нужно. Это то что предоставляеят настройки различным опциям и доступно для используемых функций.

Старые методы
Мы привыкли называть функцией жестко заданный набор параметров. Это значит то, что нам приходится вспоминать их последовательность каждый раз, когда используется функция

```
var test = new TestFunction(
  'three',
  'fixed',
  'arguments'
);
```

использование функций по старинке может стать причиной многих проблем:
• Необходимо помнить последовательность параметров
• Они не описывают что из себя представляют параметры
• Меньшая гибкость при работе с опциональными параметрами

Новый метод —конфигурация объектов
Используя конфигурацию объектов мы можем выйти на новый уровень гибкости и описать наши переменные будут обычным текстом. Теперь порядок наших параметров не имеет какой-либо роли — firstWord может быть последним предметом, а thirdWord - первым. Или они вообще могут располагаться в случайном порядке. используя конфигурацию объектов параметрам функций больше не надо быть привязанным к определенному месту в коде.

```
var test = new TestFunction({
   firstWord: 'three',
   secondWord: 'fixed',
   thirdWord: 'arguments'
});
```
Этот способ допускает неограниченное расширение параметров нашей функции. Использование меньшего числа параметров или добавление новых стало гораздо проще. А еще здорово то, что порядок использования функций не будет нарушен сложением или вычитанием параметров на более поздних стадиях.

```
var test = new TestFunction({
   secondWord: 'three'
});
var test = new TestFunction({
   secondWord: 'three',
   fourthWord: 'wow'
});
```
Что такое конфигурация объекта?
Если вы знакомы с **CSS** или **JSON**, вы безусловно узнаете конфигурацию объекта, ведь в основном она представляет собой одно и то же. Конфигурация объектов всего лишь способ структурирования информации таким образом, чтобы она была легко прочитана языком программирования (в нашем случае **JavaScript**). Давайте взглянем на пример кода содержащий конфигурацию:

Исходный код примера:

Исходный код примера:

```
{
  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);
  }
}
```

Пример используемой конфигурации может сначала показаться сложным, но стоит только понять принцип ее работы и она становится чрезвычайно полезным инструментом изменения приложений. А поскольку почти каждое приложение **Ext** использует конфигурацию объекта, нам придется изучить это как можно скорее. И вскоре конфигурация станет нашим новым лучшим другом.

Несколько моментов, которые следует помнить при работе с конфигурацией объекта:
• В фигурные скобки заключается весь набор записей, и значит что все содержимое скобок является частью объекта —{records}.
• Каждая запись состоит из пары имя/значение, где имена и значения разделены двоеточиями, а пары - запятыми —{name0: value0, name1: value1}.
• Записи значений могут содержать любой другой тип информации, включая логическую, массивы, функции или даже другие объекты—{ name0: true, name1: { name2: value2 } }.
• Квадратными скобками обозначают массивы —{name: ['one', 'two', 'three'] }. Массив так же может содержать объекты с записями, значениями и многим другим.

JSON хорошо использовать потому, что при необходимости изменить приложения, если мы хотим больший выбор, мы просто добавляем их.

Как работает JSON?
Порой, можно услышать как люди говорят о вычислениях, которые вообще-то относятся к **JSON**. Функция вычисления это именно то, чем пользуется JavaScript для понимания строк **JSON**, и переводя ее в объекты, массивы и функции которые мы используем.

Время действовать
Хорошо! Мы заставили **Ext JS** работать и задали пользователю вопрос. Теперь давайте посмотрим что мы можем сделать с их ответами. Добавим кое-чего в нашу функцию диалогового окна чтобы мы могли решить что делать в ответ на каждое нажатие кнопки. Оператор-переключатель позаботится о выборе за нас:

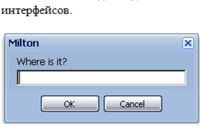
Исходный код примера:

Исходный код примера:

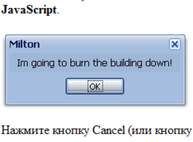
```
fn: function(btn) {
  switch(btn){
    case 'yes':
      Ext.Msg.prompt('Milton', 'Where is it?');
    break;
    case 'no':
      Ext.Msg.alert('Milton',
                'Im going to burn the building down!');
    break;
    case 'cancel':
      Ext.Msg.wait('Saving tables to disk...','File Copy');
    break;
  }
}
```

Помните некоторые встроенные типы диалогов о которых я говорил раньше? Так вот сейчас мы использовали некоторые из них. Они позволяют нам закончить простые задания без дополнительных временных затрат на написание конфигурации необходимого каждому стандартному сценарию.

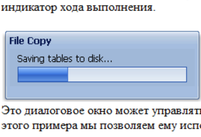
Нажмите OK и у вас появится строка ввода. Строкой ввода часто называют небольшое окно, которое позволяет вам вводить одно значение и это стандартный элемент большинства пользовательских интерфейсов.



Нажмите No и у вас появится оповещение. Я уверен что вы знакомы с его стандартным диалоговым окном в **JavaScript**.



Нажмите кнопку Cancel (или кнопку close или клавишу Escape) и вы увидите сообщение которое использует индикатор хода выполнения.



Это диалоговое окно может управляться **Ext** и ему можно сказать когда следует исчезнуть. но ради простоты этого примера мы позволяем ему использоваться постоянно.

ВАЖНО: *Кнопки focus и tab orders встроены в **Ext**. Обычно кнопка OK или Yes будет действием по умолчанию. Таким образом, нажимая клавишу Enter заставит кнопку сработать, а нажатие клавиши Tab будет перемещать вас между кнопками и элементами диалогового окна.*