jQuery UI 部件库 jQuery UI 小部件方法调用

jQuery UI 通过部件库（Widget Factory）扩展小部件

jQuery UI 的部件库（Widget Factory）使得创建小部件变得更加容易，这些小部件扩展了已有小部件的功能。这样子您就能在已有的基础上创建出功能强大的小部件，也可以在已有的小部件功能上做细微的调整。

注意：在学习本章节之前，需要明白什么是部件库（Widget Factory），及它是怎么工作的。如果您对这些知识还不熟悉，那么请先查看如何使用部件库（Widget Factory）章节。

创建小部件扩展

通过部件库（Widget Factory）创建小部件是通过向 `$.widget()` 传递小部件名称和一个原型对象来完成的。下面的实例是在 "custom" 命名空间中创建一个 "superDialog" 小部件。

```
$.widget( "custom.superDialog", {} );
```

为了支持扩展，`$.widget()` 可选性地接受作为父部件使用的小部件的构造函数。当指定一个父部件时，把它作为第二个参数进行传递，放在小部件名称后面，在小部件原型对象前面。

就像上面的实例，下面也要在 "custom" 命名空间中创建一个 "superDialog" 小部件。但是这次传递的是 jQuery UI 的 dialog（对话框）小部件 的构造函数（`$.ui.dialog`），表示 superDialog 小部件应该使用 jQuery UI 的 dialog（对话框）小部件作为父部件。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );
```

在这里，superDialog 和 dialog 两个小部件实质上是等价的，只是名称和命名空间不同而已。为了让我们新的小部件更具特点，我们可以添加一些方法到它的原型对象上。

小部件的原型对象是传递给 `$.widget()` 的最后一个参数。到目前为止，我们的实例使用的是一个空的对象。现在让我们给这个对象添加一个方法：

```
$.widget( "custom.superDialog", $.ui.dialog, {
    red: function() {
        this.element.css( "color", "red" );
    }
});

// Create a new <div>, convert it into a superDialog, and call the red() method.
$( "<div>I am red</div>" )
    .superDialog()
    .superDialog( "red" );
```

现在 superDialog 有一个 red() 方法，这会把它的文本颜色改为红色。请注意，部件库（Widget Factory）是如何自动设置 this 为小部件的实例对象。如需了解实例上所有可用的方法和属性列表，请访问 部件库（Widget Factory）API 文档。

扩展已有的方法

有时候，您需要调整或添加已有部件方法的行为。您可以把方法名称指定为原型对象上需要重载的方法名称。下面的实例重载了 dialog（对话框）的 open() 方法。由于对话框默认是打开的，当运行这段代码时，"open" 将会被记录。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
    }
});

// Create a new <div>, and convert it into a superDialog.
$( "<div>" ).superDialog();
```

当运行这段代码时，有一个问题。由于我们重载了 open() 的默认行为，所以 dialog（对话框）不再显示在屏幕上。

当我们在原型对象上使用方法，我们实际上是重载了原始的方法，在原型链中使用了一个新的方法。

为了让父部件方法可用，部件库（Widget Factory）提供了两个方法 - _super() 和 _superApply()。

使用 _super() 和 _superApply() 来访问父部件

_super() 和 _superApply() 在父部件中调用了同样的方法。请看下面的实例。就像上一个实例，这个实例也重载了 open() 方法来记录 "open"。然而，这次运行 _super() 是调用了 dialog（对话框）的 open()，并打开对话框。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );

        // Invoke the parent widget's open().
        return this._super();
    }
});

$( "<div>" ).superDialog();
```

_super() 和 _superApply() 实际上等同于最初的 Function.prototype.call() 和 Function.prototype.apply() 方法。因此，_super() 接受一个参数列表，_superApply() 接受一个数组作为参数。下面的实例演示了这二者之间的不同。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    _setOption: function( key, value ) {

        // Both invoke dialog's setOption() method. _super() requires the arguments
```

```
        // be passed as an argument list, _superApply() as a single array.
        this._super( key, value );
        this._superApply( arguments );
    }
});
```

重定义小部件

jQuery UI 1.9 添加了重定义小部件的功能。因此，可以不用创建一个新的小部件，我们只需要传递 `$.widget()` 这样一个已有的小部件名称和构造函数即可。下面的实例在 `open()` 中添加了相同的记录，但不是通过创建一个新的小部件来完成的。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

$( "<div>" ).dialog();
```

通过这个方法，我们可以扩展一个已有的小部件方法，但是仍然可以使用 `_super()` 来访问原始的方法 - 这些都不是通过创建一个新的小部件来完成的，而是直接重定义小部件即可。

小部件（Widgets）和多态性（Polymorphism）

当在小部件扩展及它们的插件之间进行交互时候，有一点值得注意，父部件的插件不能用来调用子部件元素上的方法。下面的实例演示了这一点。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );

var dialog = $( "<div>" ).superDialog();

// This works.
dialog.superDialog( "close" );

// This doesn't.
dialog.dialog( "close" );
```

上面的实例中，父部件的插件，`dialog()`，不能调用 superDialog 元素上的 `close()` 方法。如需了解更多调用小部件方法的知识，请查看 小部件（Widget）方法调用。

定制个性化实例

目前为止，我们看到的实例都有在小部件原型上扩展的方法。在原型上重载的方法影响了小部件的所有实例。

为了演示这一点，请看下面的实例。dialog（对话框）的两个势力都使用了相同的 `open()` 方法。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

// Create two dialogs, both use the same open(), therefore "open" is logged twice.
$( "<div>" ).dialog();
$( "<div>" ).dialog();
```

有时候，您只需要改变小部件的某个实例的行为。为了做到这点，您需要使用正常的 JavaScript 属性分配，获得对实例的引用，并重载该方法。具体如下面实例所示。

```
var dialogInstance = $( "<div>" )
    .dialog()
    // Retrieve the dialog's instance and store it.
    .data( "ui-dialog" );

// Override the close() method for this dialog
dialogInstance.close = function() {
    console.log( "close" );
};

// Create a second dialog
$( "<div>" ).dialog();

// Select both dialogs and call close() on each of them.
// "close" will only be logged once.
$( ":data(ui-dialog)" ).dialog( "close" );
```

个性化实例的重载方法技术是完美的一次性定制。

← jQuery UI 部件库 jQuery UI 小部件方法调用 →