# jQuery UI 教程

jQuery UI 是建立在 jQuery JavaScript 库上的一组用户界面交互、特效、小部件及主题。无论您是创建高度交互的 Web 应用程序还是仅仅向窗体控件添加一个日期选择器，jQuery UI 都是一个完美的选择。

jQuery UI 包含了许多维持状态的小部件（Widget），因此，它与典型的 jQuery 插件使用模式略有不同。所有的 jQuery UI 小部件（Widget）使用相同的模式，所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。

# jQuery UI 简介

jQuery UI 是一个建立在 jQuery JavaScript 库上的小部件和交互库，您可以使用它创建高度交互的 Web 应用程序。本教程将向您讲解 jQuery UI 是如何工作的。

## jQuery UI 特性

#### 简单易用

继承 jQuery 简易使用特性，提供高度抽象接口，短期改善网站易用性。

#### 开源免费

采用 MIT & GPL 双协议授权，轻松满足自由产品至企业产品各种授权需求。

#### 广泛兼容

兼容各主流桌面浏览器。包括IE 6+、Firefox 2+、Safari 3+、Opera 9+、Chrome 1+。

#### 轻便快捷

组件间相对独立，可按需加载，避免浪费带宽拖慢网页打开速度。

#### 标准先进

支持 WAI-ARIA，通过标准 XHTML 代码提供渐进增强，保证低端环境可访问性。

#### 美观多变

提供近 20 种预设主题，并可自定义多达 60 项可配置样式规则，提供 24 种背景纹理选择。

#### 开放公开

从结构规划到代码编写，全程开放，文档、代码、讨论，人人均可参与。

#### 强力支持

Google 为发布代码提供 CDN 内容分发网络支持。

#### 完整汉化

开发包内置包含中文在内的 40 多种语言包。

#### 缺点、不足

* 1.代码不够健壮：缺乏全面的测试用例，部分组件 Bugs 较多，不能达到企业级产品开发要求。
* 2.构架规划不足：组件间 API 缺乏协调，缺乏配合使用帮助。
* 3.控件较少：相对于 Dojo、YUI、Ext JS 等成熟产品，可用控件较少，无法满足复杂界面功能要求。

# jQuery UI 使用

一旦您下载了 jQuery UI，您将得到一个 zip 压缩包，包含下列文件：

* /css/
* /development-bundle/
* /js/
* index.html

## 在网页上使用 jQuery UI

在文本编辑器中打开 index.html，您将看到引用了一些外部文件：主题、jQuery 和 jQuery UI。通常情况下，您需要在页面中引用这三个文件，以便使用 jQuery UI 的窗体小部件和交互部件：

<link rel="stylesheet" href="css/themename/jquery-ui.custom.css" /><script src="js/jquery.min.js"></script><script src="js/jquery-ui.custom.min.js"></script>

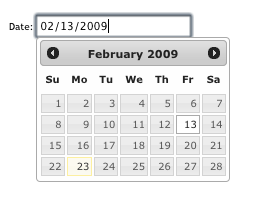
一旦您引用了这些必要的文件，您就能向您的页面添加一些 jQuery 小部件。比如，要制作一个日期选择器（datepicker）小部件，您需要向页面添加一个文本输入框，然后再调用 .datepicker()，如下所示：

**HTML:**

<input type="text" name="date" id="date" />

**JavaScript:**

$( "#date" ).datepicker();



# jQuery UI 定制

jQuery UI 提供了多种定制方式。您已经看到下载生成器（Download Builder）如何定制一个值包含您所需选项的自定义版本，这里还提供了其他定制方式。

## jQuery UI 基础：使用选项

jQuery UI 中的每个插件都有一个默认配置，默认配置值一般是根据最基本最常见的使用情况设置的。如果您想要让某个插件设置成非默认值，您可以使用 "options" 重写它的默认设置。选项是一组属性，作为参数传递给 jQuery UI 小部件。例如，滑块（slider）小部件具有 orientation 选项，该选项允许您指定滑块是水平的还是垂直的。为了设置滑块的该选项，您只需将它作为一个参数传递，如下所示：

$( "#mySliderDiv" ).slider({

orientation: "vertical"});

您可以传递更多不同的选项，每个选项之间用逗号分隔：

$( "#mySliderDiv" ).slider({

orientation: "vertical",

min: 0,

max: 150,

value: 50});

请记得选项需放在大括号 { } 内。

## 视觉定制：设计一个 jQuery UI 主题

如果您想要设计自己的主题，jQuery UI 提供了一个非常完美的用于主题定制的应用程序，这就是 ThemeRoller。具体定制请访问 [jQuery UI ThemeRoller](http://jqueryui.com/themeroller/" \t "https://www.runoob.com/jqueryui/_blank)。

ThemeRoller 为所有使用 jQuery UI 小部件设计的元素提供了一个自定义接口。当您调整左栏中的"levers"，右栏中的小部件将根据您的设计进行显示。ThemeRoller 的 Gallery 选项卡提供了一些与设计主题，与下载生成器（Download Builder）页面提供的一样。您可以基于这些主题做调整，或者直接下载。

## 下载主题

当您点击 ThemeRoller 页面中的 "Download theme" 按钮，将跳转到下载生成器（Download Builder）页面，您的自定义主题会在主体下拉菜单中自动选中。您可以进一步配置下载包。一旦下载完成，您将看到 example.html 页面使用了您自定义的主题。

**提示：** 如果您需要编辑您的主题，只需打开 CSS 文件，找到第 43 行，"To view and modify this theme, visit ..."，该 url 即为在 ThemeRoller 中打开主题进行编辑的链接。

# jQuery UI 工作原理

jQuery UI 包含了许多维持状态的小部件（Widget），因此，它与典型的 jQuery 插件使用模式略有不同。其安装方式与大部分 jQuery 插件的安装方式类似，jQuery UI 的小部件是基于 [部件库（Widget Factory）](https://www.runoob.com/jqueryui/jqueryui-widget-factory.html) 创建的，小部件库提供了通用的 API。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。本教程将通过 [进度条（progressbar）](https://www.runoob.com/jqueryui/example-progressbar.html) 小部件代码实例介绍常见的功能。

## 安装

为了跟踪部件的状态，我们首先介绍一下小部件的全生命周期。当小部件安装时，生命周期开始。我们只需要在一个或多个元素上调用插件，即安装了小部件。

$( "#elem" ).progressbar();

这将会初始化 jQuery 对象中的每个元素，在本例中，元素 id 为 "elem"。因为我们调用无参数的 .progressbar() 方法，小部件则会按照它的默认选项进行初始化。我们可以在安装时传递一组选项，这样既可重写默认选项。

$( "#elem" ).progressbar({ value: 20 });

安装时传递的选项数目多少可根据我们的需要而定。任何我们未传递的选项则都使用它们的默认值。

选项是小部件状态的组成部分，所以我们也可以在安装后再进行设置选项。我们将在后续的 option 方法中介绍这部分内容。

## 方法

既然小部件已经初始化，我们就可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都以方法调用的形式进行。为了在小部件上调用一个方法，我们可以向 jQuery 插件传递方法的名称。例如，为了在进度条（progressbar）小部件上调用 value方法，我们应该使用：

$( "#elem" ).progressbar( "value" );

如果方法接受参数，我们可以在方法名后传递参数。例如，为了传递参数 40 给 value 方法，我们可以使用：

$( "#elem" ).progressbar( "value", 40 );

就像 jQuery 中的其他方法一样，大部分的小部件方法为链接返回 jQuery 对象。

$( "#elem" )

.progressbar( "value", 90 )

.addClass( "almost-done" );

### **公共的方法**

每个小部件都有它自己的一套基于小部件所提供功能的方法。然而，有一些方法是所有小部件都共同具有的。

#### option

正如我们前面所提到的，我们可以在初始化之后通过 option 方法改变选项。例如，我们可以通过调用 option 方法改变 progressbar（进度条）的 value 为 30。

$( "#elem" ).progressbar( "option", "value", 30 );

请注意，这与之前我们调用 value 方法的实例有所不同。在本实例中，我们调用 option 方法，改变 value 选项为 30。

我们也可以为某个选项获取当前的值。

$( "#elem" ).progressbar( "option", "value" );

另外，我们可以通过给 option 方法传递一个对象，一次更新多个选项。

$( "#elem" ).progressbar( "option", {

value: 100,

disabled: true});

您也许注意到 option 方法有着与 jQuery 代码中取值器和设置器相同的标志，就像 .css() 和 .attr()。唯一的不同就是您必须传递字符串 "option" 作为第一个参数。

#### disable

disable 方法禁用小部件。在进度条（progressbar）实例中，这会改变样式让进度条显示为禁用状态。

$( "#elem" ).progressbar( "disable" );

调用 disable 方法等同于设置 disabled 选项为 true。

#### enable

enable 方法是 disable 方法的对立面。

$( "#elem" ).progressbar( "enable" );

调用 enable 方法等同于设置 disabled 选项为 false。

#### destroy

如果您不再需要小部件，那么可以销毁它，返回到最初的标记。这意味着小部件生命周期的终止。

$( "#elem" ).progressbar( "destroy" );

一旦您销毁了一个小部件，您就不能在该部件上调用任何方法，除非您再次初始化这个小部件。如果您要移除元素，可以直接通过 .remove()，也可以通过 .html() 或 .empty() 来实现，小部件会自动销毁。

#### widget

一些小部件生成包装器元素，或与原始元素断开连接的元素。在下面的实例中，widget 将返回生成的元素。在进度条（progressbar）实例中，没有生成的包装器，widget 方法返回原始的元素。

$( "#elem" ).progressbar( "widget" );

## 事件

所有的小部件都有跟他们各种行为相关的事件，用于在状态改变时通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称为前缀。例如，我们可以绑定进度条（）的 change 事件，一旦值发生变化时就触发。

$( "#elem" ).bind( "progressbarchange", function() {

alert( "The value has changed!" );});

每个事件都有一个相对应的回调，作为选项进行呈现。我们可以使用进度条（progressbar）的 change 回调，这等同于绑定 progressbarchange 事件。

$( "#elem" ).progressbar({

change: function() {

alert( "The value has changed!" );

}});

### **公共的事件**

大多数事件是针对特定的小部件，所有的小部件都有一个公共的 create 事件。该事件在小部件被创建时即被触发。

# jQuery UI 部件库（Widget Factory）

jQuery UI 部件库（Widget Factory）是一个可扩展的基础，所有的 jQuery UI 小部件都是在上面进行创建的。使用部件库（Widget Factory）来创建插件，提供了方便的状态管理，同时也为一些常见的任务提供了便捷，比如暴露插件方法，实例化后改变选项等。

* [通过部件库（Widget Factory）扩展小部件（Widget）](https://www.runoob.com/jqueryui/jqueryui-widget-extend.html)
* [小部件（Widget）方法调用](https://www.runoob.com/jqueryui/jqueryui-widget-method.html)
* [为什么使用部件库（Widget Factory）](https://www.runoob.com/jqueryui/jqueryui-widget-factory-why.html)
* [如何使用部件库（Widget Factory）](https://www.runoob.com/jqueryui/jqueryui-widget-factory-how.html)

# jQuery UI API - 进度条部件（Progressbar Widget）

## 所属类别

[小部件（Widgets）](https://www.runoob.com/jqueryui/ref-widgets.html)

## 用法

**描述：**显示一个确定的或不确定的进程状态。

**版本新增：**1.6

进度条被设计来显示进度的当前完成百分比。进度条通过 CSS 编码灵活调整大小，默认会缩放到适应父容器的大小。

一个确定的进度条只能在系统可以准确更新当前状态的情况下使用。一个确定的进度条不会从左向右填充，然后循环回到空 - 如果不能计算实际状态，则使用不确定的进度条以便提供用户反馈。

### **主题化**

进度条部件（Progressbar Widget）使用 [jQuery UI CSS 框架](https://www.runoob.com/jqueryui/api-css-framework.html) 来定义它的外观和感观的样式。如果需要使用进度条指定的样式，则可以使用下面的 CSS class 名称：

* ui-progressbar：进度条的外层容器。该元素会为不确定的进度条另外添加一个 ui-progressbar-indeterminate class。
  + ui-progressbar-value：该元素代表进度条的填充部分。
    - ui-progressbar-overlay：用于为不确定的进度条显示动画的覆盖层。

### **依赖**

* [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html)
* [部件库（Widget Factory）](https://www.runoob.com/jqueryui/api-jQuery-widget.html)

### **附加说明**

* 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

# jQuery UI 主题

所有的 jQuery UI 插件都允许开发人员无缝集成 UI 小部件到他们网站或应用程序的外观和感观。每个插件通过 CSS 定义样式，且包含了两层样式信息：标准的 [jQuery UI CSS 框架](https://www.runoob.com/jqueryui/jqueryui-theme-api.html) 样式和具体的插件样式。

jQuery UI CSS 框架提供了语义表示的类，用来表明小部件内元素的角色，比如标题、内容或可点击区域。这些在所有的小部件中都是一致的，一个可点击的 tab（标签页）、accordion（折叠面板）或 button（按钮）都有相同的 ui-state-default class，用来表明它们是可点击的。当用户鼠标悬浮在这些元素上面时，这个 class 就变成 ui-state-hover，当选中这些元素时则变成 ui-state-active。这些 class 的一致性使得所有部件中具有相似角色或交互状态的元素在外观表现上一致。

CSS 框架样式封装在一个单独的文件中，名为 ui.theme.css。这个文件时通过 [ThemeRoller](https://www.runoob.com/jqueryui/jqueryui-theme-themeroller.html) 应用程序来修改的。框架样式只包含影响外观和感观的属性，只要是颜色、背景图像、图标等。所以这些是 "安全的" 样式，不会影响到插件的功能。这种分隔意味着开发人员可以通过在 theme.css 文件中修改颜色和图像来创建一个自定义的外观和感观。由于未来的插件或者 bug 修复将是可用的，这些不通过修改即可与主题一起使用。

由于框架样式只覆盖了外观和感观，所以还需要包含具体的插件样式表，这些样式表包括了所有额外的让小部件具有功能性的结构样式规则，比如尺寸、内边距、外边距、定位、浮动。每个插件的样式表位于 themes/base 文件夹内，且配合插件进行命名，比如 "jquery.ui.accordion.css"。这些样式必须认真编辑，因为它们与脚本一起提供了框架样式的覆盖。

我们鼓励所有的开发人员创建 jQuery 插件，jQuery UI CSS 框架使得最终用户更容易定制主题和使用插件。

### **主题化**

下面列出了三种主题化 jQuery UI 插件的一般方法：

* **下载 ThemeRoller 主题**：最早的创建主题的方式是使用 [ThemeRoller](https://www.runoob.com/jqueryui/jqueryui-theme-themeroller.html) 来生成和下载一个主题。这个应用程序将创建一个新的 ui.theme.css 文件和一个包含了所有必需的背景图像及图标精灵的 images 文件夹。这个方法是最早的创建和维护主题的方式，但是它对 ThemeRoller 中提供的选项限制了自定义。
* **修改 CSS 文件**：为了对外观和感观做进一步的控制，您可以选择从默认主题（Smoothness）开始，或者从一个由 ThemeRoller 生成的主题开始，然后调整 ui.theme.css 文件，或者任意一个独立插件的样式表。例如，您可以很容易地调整所有按钮的角半径为不同于其他 UI 组件的值，或者使用自定义设置为图标精灵改变路径。通过一点点的样式范围，您甚至可以在一个 UI 中同时使用多个主题。为了易于维护，建议只更改 ui.theme.css 文件和图像。
* **重新编写自定义的 CSS**：为了最大程度地控制外观和感观，可以重新开始编写每个插件的 CSS，而不使用框架类或者特定的插件样式表。如果想要的外观和感观不能通过修改 CSS 来实现或者使用高度自定义的标记，那么就可以采用这个方法。这个方法要求在 CSS 方面有深厚的专业知识，且要求手动更新未来的插件。

### **使用 ThemeRoller、jQuery UI CSS 框架，以及设计自定义主题**

* [jQuery UI ThemeRoller](https://www.runoob.com/jqueryui/jqueryui-theme-themeroller.html)
* [jQuery UI CSS 框架 API](https://www.runoob.com/jqueryui/jqueryui-theme-api.html)
* [设计主题](https://www.runoob.com/jqueryui/jqueryui-theme-design.html)

# jQuery UI ThemeRoller

### **ThemeRoller 简介**

ThemeRoller 是一个 Web 应用程序，为 jQuery UI 设计和下载自定义主题提供了直观的界面。您可以访问 [jQuery UI ThemeRoller](http://jqueryui.com/themeroller" \t "https://www.runoob.com/jqueryui/_blank)进行主题定制。

jQuery UI ThemeRoller 是由波士顿的 [Filament Group, Inc](http://www.filamentgroup.com/" \t "https://www.runoob.com/jqueryui/_blank) 设计和开发的。



### **ThemeRoller 界面**

ThemeRoller 的界面分为不同面板，各面板分别是全局字体和圆角半径设置、小部件容器样式、可点击元素的互动状态，及覆盖和阴影的各种样式。这些面板允许配置各种 CSS 属性，比如字体尺寸、颜色、粗细、背景颜色和纹理、边框颜色、文本颜色、图标颜色、圆角半径，等等。

### **主题馆（Gallery）：预先设计主题**

ThemeRoller 主题可以通过永久链接 URL 进行查看，它包含一些预先设计的主题可供选择。主题馆（Gallery）可以通过位于应用程序界面顶端的标签栏进行访问。从主题馆（Gallery）中，您可以预览和下载主题，甚至可以选择一个主题，然后切换到 "Roll Your Own" 标签页进行调整。

### **下载主题**

当您设计完主题后，您可以下载主题以便在项目中进行使用。ThemeRoller 在顶部有一个 "Download theme（下载主题）" 按钮，可以生成一个 zip 压缩包，包含了所有的主题相关文件。下载文件中的图像是按照您的规格进行生成的，并保存为高质量的 PNG 文件。

您的主题将包含图像和 CSS，组成了 jQuery UI CSS 框架的自定义版本，包含了所有插件的图像和 CSS。

### **在项目中按照下载的主题**

一旦下载了主题，解压缩后，您将看到一个名为 themes 的文件夹。该文件夹包括了主题所需的 CSS 和图像。复制主题文件夹到您的项目中，并在页面中链接 themes/all.css 文件。

### **创建自定义的 "ThemeRoller-Ready" 组件**

IMG_257 IMG_258

ThemeRoller 生成了一个 jQuery UI CSS 框架的自定义版本，用于开发您自己的 ThemeRoller-ready jQuery 组件。通过这个框架生成的类被设计来提供通用的用户界面的设计情况，包括状态、图标以及各种辅助类。

如需了解更多 jQuery UI CSS 框架的开发信息，请查看 [主题化 API 文档](https://www.runoob.com/jqueryui/jqueryui-theme-api.html)。

### **ThemeRoller 链接**

* [教程：开发您自己的 jQuery "ThemeRoller-Ready" 组件，Filament Group](http://www.filamentgroup.com/lab/developer_your_own_jquery_themeroller_ready_components/" \t "https://www.runoob.com/jqueryui/_blank)
* [ThemeRoller 简介：设计 & 下载 jQuery UI 自定义主题，Filament Group](http://www.filamentgroup.com/lab/introducing_themeroller_design_download_custom_themes_for_jquery_ui/" \t "https://www.runoob.com/jqueryui/_blank)
* [视频：设计 & 下载 jQuery UI 自定义主题，Filament Group](http://www.dizi-izle-film-izle.net/" \t "https://www.runoob.com/jqueryui/_blank)

# jQuery UI CSS 框架 API

### **jQuery UI CSS 框架**

jQuery UI 包含了一个强大的 CSS 框架，为了创建自定义 jQuery 小部件而设计的。框架包含了通用的用户界面所需的类，且可使用 jQuery UI ThemeRoller 进行维护。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件。您需采用共享标记公约，以便在插件社区的代码集成。

### **框架类**

下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 ui.core.css 和 ui.theme.css 两个文件中。这些类被设计来用于用户界面元素，以便获得整个应用程序的视觉一致性，可通过 jQuery UI ThemeRoller 对组件进行主题化。

### **布局助手**

* .ui-helper-hidden：对元素应用 display: none。
* .ui-helper-hidden-accessible：对元素应用访问隐藏（通过页面绝对定位）。
* .ui-helper-reset：UI 元素的基本样式重置。重置的元素比如：padding、margin、text-decoration、list-style，等等。
* .ui-helper-clearfix：对父元素应用浮动包装属性。
* .ui-helper-zfix：对 <iframe> 元素应用 iframe "fix" CSS。

### **小部件容器**

* .ui-widget：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
* .ui-widget-header：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
* .ui-widget-content：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

### **交互状态**

* .ui-state-default：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
* .ui-state-hover：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
* .ui-state-focus：当键盘聚焦在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
* .ui-state-active：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

### **交互提示 Cues**

* .ui-state-highlight：对高亮或者选中元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "highlight" 容器样式。
* .ui-state-error：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
* .ui-state-error-text：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
* .ui-state-disabled：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
* .ui-priority-primary：对第一优先权的按钮应用的 Class。应用粗体文本。
* .ui-priority-secondary：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

### **图标**

#### 状态和图像

* .ui-icon：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。**注意：** .ui-icon class 将根据它的父容器得到一个不同的精灵背景图像。例如，ui-state-default 容器内的 ui-icon 元素将根据 ui-state-default 的图标颜色进行着色。

#### 图标类型

在声明 .ui-icon class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 .ui-icon-{icon type}-{icon sub description}-{direction}。

例如，一个指向右侧的三角形图标，如下所示： .ui-icon-triangle-1-e

jQuery UI 的 [ThemeRoller](http://jqueryui.com/themeroller) 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

### **其他视觉效果**

#### 圆角半径助手

* .ui-corner-tl：对元素的左上角应用圆角半径。
* .ui-corner-tr：对元素的右上角应用圆角半径。
* .ui-corner-bl：对元素的左下角应用圆角半径。
* .ui-corner-br：对元素的右下角应用圆角半径。
* .ui-corner-top：对元素上边的左右角应用圆角半径。
* .ui-corner-bottom：对元素下边的左右角应用圆角半径。
* .ui-corner-right：对元素右边的上下角应用圆角半径。
* .ui-corner-left：对元素左边的上下角应用圆角半径。
* .ui-corner-all：对元素的所有四个角应用圆角半径。

#### 覆盖 & 阴影

* .ui-widget-overlay：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
* .ui-widget-shadow：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

# jQuery UI 设计主题

### **文件结构**

主题是以特定的方式来增加他们的易用性。通常，文件目录结构如下所示：

* themename/ – 您的主题必须完全包含在一个单独的以主题名称命名的文件夹内。
* themename/themename.css – 这是基本的 CSS 文件。无论使用了哪个插件，该文件都必须在每个使用主题的页面中引用。该文件应该是轻量级的，只包括要点。
* themename/themename.pluginname.css – 您支持的每个插件都需要一个 CSS 文件。插件的名称应直接包含在文件名中。例如，如果您为 tabs（标签页）插件进行主题化，则有：themename.tabs.js。
* themename/img.png – 您的主题可以包含图像。它们可以根据您的喜好进行命名，这里没有特定的命名惯例。

如需了解主题文件结构是如何完成的实例，请访问 [jQuery UI 基本主题](https://github.com/jquery/jquery-ui/tree/master/themes/base" \t "https://www.runoob.com/jqueryui/_blank)。

### **定义样式**

为主题编写样式是非常简单的，这是因为主题的灵活性。

所有的主题都应该有一个基本的 CSS class。这个主要的 class 允许用户启用禁用主题。您的根 class 的格式应该是 .ui-themename。且它在 HTML 文件中的用法如下所示：

<html>

<head>

<title>My Site</title>

<link rel="stylesheet" href="themename/themename.css" />

<link rel="stylesheet" href="othertheme/othertheme.css" />

<link rel="stylesheet" href="othertheme/othertheme.dialog.css" />

</head>

<body class="ui-themename">

<div class="ui-othertheme">

<div class="ui-dialog">This is a modal dialog.</div>

</div>

</body>

</html>

在上面的实例中，发生了一些重要的事情：

* 我们同时向文档中加载两个主题。
* 整个页面机器所有内容，是根据 themename 的样式进行主题化的。
* 然而，带有 ui-othertheme class 的 <div>及其中的每个元素（包括模态对话框）都是根据 othertheme 的样式进行主题化的。

如果我们打开 themename.css 文件进行查看，我们可以看到如下代码：

body.ui-themename { background:#111; color:snow; }

.ui-themename a, a.ui-themename { color:#68D; outline:none; }

.ui-themename a:visited, a.ui-themename:visited { color:#D66; }

.ui-themename a:hover, a.ui-themename:hover { color:#FFF; }

请注意，themename.css 文件只包括全局通用的样式信息，特定插件的样式信息不在这里进行定义。这里的样式对所有主题都是适用的。不用担心一个主题会占据多个文件 - 这些会在创建和下载的过程被简化。

# 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）](https://www.runoob.com/jqueryui/jqueryui-widget-factory-how.html)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](https://www.runoob.com/jqueryui/example-dialog.html) 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](https://www.runoob.com/jqueryui/api-jQuery-widget.html)。

### **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](https://www.runoob.com/jqueryui/api-dialog.html" \l "method-open)。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()](https://www.runoob.com/jqueryui/api-jQuery-widget.html" \l "method-_super) 和 [\_superApply()](https://www.runoob.com/jqueryui/api-jQuery-widget.html" \l "method-_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](https://www.runoob.com/jqueryui/jqueryui-widget-method.html)。

### **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 小部件（Widget）方法调用

小部件（Widget）是通过 [部件库（Widget Factory）](https://www.runoob.com/jqueryui/jqueryui-widget-factory.html) 使用方法来改变他们初始化后的状态和执行动作而被创建的。有两种调用小部件方法的方式 - 通过部件库（Widget Factory）创建的插件，或者通过调用元素实例对象上的方法。

### **插件调用**

使用小部件的插件调用方法，把方法名称以字符串形式进行传递。例如，点击这里查看，如何调用 [dialog（对话框）小部件的 close() 方法](https://www.runoob.com/jqueryui/api-dialog.html" \l "method-close)。

$( ".selector" ).dialog( "close" );

如果方法要求参数 ，请作为额外的参数传递给插件。点击这里查看，如何调用 [dialog（对话框）的 option() 方法](https://www.runoob.com/jqueryui/api-dialog.html" \l "method-option)。

$( ".selector" ).dialog( "option", "height" );

这会返回 [dialog（对话框）的 height 选项](https://www.runoob.com/jqueryui/api-dialog.html" \l "option-height) 的值。

### **实例调用**

每个小部件的每个实例都是使用 [jQuery.data()](http://api.jquery.com/jQuery.data/" \t "https://www.runoob.com/jqueryui/_blank) 存储在元素上。为了检索实例对象，请使用小部件的全称作为键名调用 jQuery.data()。具体如下面实例所示。

var dialog = $( ".selector" ).data( "ui-dialog" );

在您引用实例对象之后，可以直接在上面调用方法。

var dialog = $( ".selector" ).data( "ui-dialog" );

dialog.close();

在 jQuery UI 1.11 中，新的 instance() 方法会使得这个过程变得更简单。

$( ".selector" ).dialog( "instance" ).close();

### **返回类型**

大多数通过小部件的插件调用的方法将返回一个 jQuery 对象，所以方法调用可以通过额外的 jQuery 方法链接。当在实例上进行调用时，则会返回 undefined。具体如下面实例所示。

var dialog = $( ".selector" ).dialog();

// Instance invocation - returns undefined

dialog.data( "ui-dialog" ).close();

// Plugin invocation - returns a jQuery object

dialog.dialog( "close" );

// Therefore, plugin method invocation makes it possible to// chain method calls with other jQuery functions

dialog.dialog( "close" )

.css( "color", "red" );

例外的是，返回小部件相关信息的那些方法。例如 [dialog（对话框）的 isOpen() 方法](https://www.runoob.com/jqueryui/api-dialog.html" \l "method-isOpen)。

$( ".selector" )

.dialog( "isOpen" )

// This will throw a TypeError

.css( "color", "red" );

这会产生一个 TypeError 错误，因为 isOpen() 返回的是一个布尔值，而不是一个 jQuery 对象。

# jQuery UI 为什么使用部件库（Widget Factory）

编写 jQuery 插件与向 jQuery.prototype（通常显示为 $.fn）添加方法一样简单，且需要遵循一些简单的规则，比如返回 this。所以为什么会存在部件库（Widget Factory）？

在本章节中，我们将讲解部件库（Widget Factory）的好处，并了解何时使用它，以及为什么要使用它。

## 无状态 vs. 有状态插件

大多数 jQuery 插件是无状态的，它们执行一些动作即完成了它们的任务。例如，如果您使用 .text( "hello" ) 设置元素的文本，没有安装阶段，结果都是一样的。对于这种类型的插件，它只是扩展了 jQuery 的原型。

然而，一些插件是有状态的，它们有全生命周期、维持状态以及对变化的反应。这些插件需要大量专门的代码来初始化和状态管理（有时是销毁）。这就导致出现了用于创建有状态插件的模板。更糟糕的是，每个插件的作者按照不同的方式进行管理插件的生命周期和状态，这就导致了不同的插件有不同的 API 样式。部件库（Widget Factory）旨在解决这些问题，它移除了模板，并为插件创建了一个一致的 API。

## 一致的 API

部件库（Widget Factory）定义了如何创建和销毁小部件，获取和设置选项，调用方法，以及监听小部件触发的事件。通过使用部件库（Widget Factory）来创建有状态的插件，会自动符合定义的标准，让新用户更容易使用您的插件。另外，部件库（Widget Factory）还能实现定义接口的功能。如果您对部件库（Widget Factory）提供的 API 还不熟悉，请查看 [如何使用部件库（Widget Factory）](https://www.runoob.com/jqueryui/jqueryui-widget-factory-how.html)。

## 在初始化时设置选项

当您创建一个接受选项的插件时，您应该为尽可能多的选项定义 defaults。然后在初始化时，把用户提供的选项与 defaults 进行合并。您也可以暴露 defaults，这样用户就可以更改默认值。

# jQuery UI 如何使用部件库（Widget Factory）

我们将创建一个进度条。正如下面实例所示，这可以通过调用 jQuery.widget() 来完成，它带有两个参数：一个是要创建的插件名称，一个是包含支持插件的函数的对象文字。当插件被调用时，它将创建一个新的插件实例，所有的函数都将在该实例的语境中被执行。这与两种重要方式的标准 jQuery 插件不同。首先，语境是一个对象，不是 DOM 元素。其次，语境总是一个单一的对象，不是一个集合。

$.widget( "custom.progressbar", {

\_create: function() {

var progress = this.options.value + "%";

this.element

.addClass( "progressbar" )

.text( progress );

}});

插件的名称必须包含命名空间，在这个实例中，我们使用了 custom 命名空间。您只能创建一层深的命名空间，因此，custom.progressbar 是一个有效的插件名称，而 very.custom.progressbar 不是一个有效的插件名称。

我们看到部件库（Widget Factory）为我们提供了两个属性。this.element 是一个包含一个元素的 jQuery 对象。如果我们的插件在包含多个元素的 jQuery 对象上调用，则会为每个元素创建一个单独的插件实例，且每个实例都会有自己的 this.element。第二个属性，this.options，是一个包含所有插件选项的键名/键值对的哈希（hash）。这些选项可以被传给插件，如下所示：

$( "<div></div>" )

.appendTo( "body" )

.progressbar({ value: 20 });

## 关闭注释

部件库（Widget Factory）只是创建有状态插件的一种方式。这里还有一些其他不同的模型可以使用，且每个都有各自的优势和劣势。部件库（Widget Factory）解决了很多常见的问题，且大大提高了效率，同时也大大提高了代码的重用性，使它适合于 jQuery UI 及其他有状态的插件。

请注意，在本章节中我们使用了 custom 命名空间。ui 命名空间被官方的 jQuery UI 插件保留。当创建您自己的插件时，您应该创建自己的命名空间。这样才能更清楚插件来自哪里，属于哪个范围。

### **清理**

在某些情况下，允许用户应用插件，然后再取消应用。您可以通过 \_destroy() 方法做到这一点。在 \_destroy() 方法内，您应该撤销在初始化和后期使用期间插件所做的一切动作。\_destroy() 是通过 .destroy() 方法被调用的，.destroy() 方法是在插件实例绑定的元素从 DOM 上移除时被自动调用的，所以这可被用于垃圾回收。基本的 .destroy() 方法也处理一些常用的清理操作，比如从小部件的 DOM 元素上移除实例引用，从元素上解除绑定小部件命名空间中的所有事件，解除绑定所有使用 \_bind() 添加的事件。

回调函数本质上只是附加选项，所以您可以像其他选项一样获取并设置它们。无论何时执行回调，都会有一个相对应的事件被触发。事件类型是通过连接插件的名称和回调函数名称确定的。回调和事件都接受两个相同的参数：一个事件对象和一个与事件相关的数据哈希，具体如下面实例所示。

您的插件可能需要包含防止用户使用的功能，为了做到这点，最好的方法就是创建一个可撤销的回调。用户可以撤销回调或者相关的事件，与他们撤销任何一个原生事件一样，都是通过调用 event.preventDefault() 或返回 false 来实现的。如果用户撤销回调，\_trigger() 方法将返回 false，这样您就能在插件内实现合适的功能。

# jQuery UI 1.10 API 文档

jQuery UI 是建立在 jQuery JavaScript 库上的一组用户界面交互、特效、小部件及主题。

如果您是 jQuery 新手，您可以先查看 [jQuery 教程](https://www.runoob.com/jquery/jquery-tutorial.html)。

本教程提供了 jQuery UI 1.10 API 文档。如果您使用的是旧版本，我们建议您更新到 jQuery UI 1.10，以便获取最好的支持，充分利用最新的 bug 修复和增强功能。如需了解更多有关 jQuery UI 1.10 的信息，请访问 [发布公告](http://blog.jqueryui.com/2013/01/jquery-ui-1-10-0/" \t "https://www.runoob.com/jqueryui/_blank) 和 [升级指南](http://jqueryui.com/upgrade-guide/1.10/" \t "https://www.runoob.com/jqueryui/_blank)。

jQuery UI 1.10 支持 jQuery 1.6 及以上的版本。

# jQuery UI API 类别 - 特效（Effects）

jQuery UI 在jQuery 内置的特效上添加了一些功能。jQuery UI 支持颜色动画和 Class 转换，同时也提供了一些额外的 [Easings](https://www.runoob.com/jqueryui/api-easings.html)。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

|  |
| --- |
|  |
| **API** | **描述** | **也属于类别** |
| [.addClass()](https://www.runoob.com/jqueryui/api-addClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |
| [百叶窗特效（Blind Effect）](https://www.runoob.com/jqueryui/api-blind-effect.html) | 百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。 |  |
| [反弹特效（Bounce Effect）](https://www.runoob.com/jqueryui/api-bounce-effect.html) | 反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。 |  |
| [剪辑特效（Clip Effect）](https://www.runoob.com/jqueryui/api-clip-effect.html) | 剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。 |  |
| [颜色动画（Color Animation）](https://www.runoob.com/jqueryui/api-color-animation.html) | 使用 .animate() 实现颜色动画效果。 |  |
| [降落特效（Drop Effect）](https://www.runoob.com/jqueryui/api-drop-effect.html) | 降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。 |  |
| [Easings](https://www.runoob.com/jqueryui/api-easings.html) | Easing 函数指定动画在不同点上的行进速度。 |  |
| [.effect()](https://www.runoob.com/jqueryui/api-effect.html) | 对一个元素应用动画特效。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [爆炸特效（Explode Effect）](https://www.runoob.com/jqueryui/api-explode-effect.html) | 爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。 |  |
| [淡入淡出特效（Fade Effect）](https://www.runoob.com/jqueryui/api-fade-effect.html) | 淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。 |  |
| [折叠特效（Fold Effect）](https://www.runoob.com/jqueryui/api-fold-effect.html) | 折叠特效（Fold Effect）通过折叠元素来隐藏或显示一个元素。 |  |
| [.hide()](https://www.runoob.com/jqueryui/api-hide.html) | 使用自定义效果来隐藏匹配的元素。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [突出特效（Highlight Effect）](https://www.runoob.com/jqueryui/api-highlight-effect.html) | 突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。 |  |
| [膨胀特效（Puff Effect）](https://www.runoob.com/jqueryui/api-puff-effect.html) | 通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。 |  |
| [跳动特效（Pulsate Effect）](https://www.runoob.com/jqueryui/api-pulsate-effect.html) | 跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。 |  |
| [.removeClass()](https://www.runoob.com/jqueryui/api-removeClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |
| [缩放特效（Scale Effect）](https://www.runoob.com/jqueryui/api-scale-effect.html) | 按照某个百分比缩放元素。 |  |
| [震动特效（Shake Effect）](https://www.runoob.com/jqueryui/api-shake-effect.html) | 垂直或水平方向多次震动元素。 |  |
| [.show()](https://www.runoob.com/jqueryui/api-show.html) | 使用自定义效果来显示匹配的元素。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [尺寸特效（Size Effect）](https://www.runoob.com/jqueryui/api-size-effect.html) | 调整元素尺寸到指定宽度和高度。 |  |
| [滑动特效（Slide Effect）](https://www.runoob.com/jqueryui/api-slide-effect.html) | 把元素滑动出视区。 |  |
| [.switchClass()](https://www.runoob.com/jqueryui/api-switchClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) |
| [.toggle()](https://www.runoob.com/jqueryui/api-toggle.html) | 使用自定义效果来显示或隐藏匹配的元素。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.toggleClass()](https://www.runoob.com/jqueryui/api-toggleClass.html) | 当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。 | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |
| [转移特效（Transfer Effect）](https://www.runoob.com/jqueryui/api-transfer-effect.html) | 把一个元素的轮廓转移到另一个元素。 |  |

# jQuery UI API - 百叶窗特效（Blind Effect）

## 所属类别

[特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html)

## 用法

**描述：**百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。

blind

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **类型** | **描述** | **默认值** |
| direction | String | 隐藏元素时拉动百叶窗的方向，显示元素时拉动百叶窗的方向。 可能的值：up、down、left、right、vertical、horizontal。 | "up" |

容器应用 overflow: hidden 时，高度的变化会影响到元素的可见性。

# jQuery UI API - 爆炸特效（Explode Effect）

## 所属类别

[特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html)

## 用法

**描述：**爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。

explode

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **类型** | **描述** | **默认值** |
| pieces | Integer | 爆炸裂开的碎片数目，应该是个平方数，任何其他值被舍入到最近的平方数。 | 9 |

# jQuery UI API - 跳动特效（Pulsate Effect）

## 所属类别

[特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html)

## 用法

**描述：**跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。

pulsate

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **类型** | **描述** | **默认值** |
| times | Integer | 元素跳动的次数。当隐藏或显示时，会添加半个跳动。 | 5 |

# jQuery UI API 类别 - 特效核心（Effects Core）

由 jquery.ui.effect.js 提供的功能。除了下面列出的方法，jquery.ui.effect.js 还包括一些 [Easings](https://www.runoob.com/jqueryui/api-easings.html)。

|  |  |  |
| --- | --- | --- |
| **API** | **描述** | **也属于类别** |
| [.addClass()](https://www.runoob.com/jqueryui/api-addClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |
| [颜色动画（Color Animation）](https://www.runoob.com/jqueryui/api-color-animation.html) | 使用 .animate() 实现颜色动画效果。 |  |
| [.effect()](https://www.runoob.com/jqueryui/api-effect.html) | 对一个元素应用动画特效。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.hide()](https://www.runoob.com/jqueryui/api-hide.html) | 使用自定义效果来隐藏匹配的元素。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.removeClass()](https://www.runoob.com/jqueryui/api-removeClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |
| [.show()](https://www.runoob.com/jqueryui/api-show.html) | 使用自定义效果来显示匹配的元素。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.switchClass()](https://www.runoob.com/jqueryui/api-switchClass.html) | 当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) |
| [.toggle()](https://www.runoob.com/jqueryui/api-toggle.html) | 使用自定义效果来显示或隐藏匹配的元素。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.toggleClass()](https://www.runoob.com/jqueryui/api-toggleClass.html) | 当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。 | [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) |

# jQuery UI API - .toggleClass()

## 所属类别

[特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) | [特效核心（Effects Core）](https://www.runoob.com/jqueryui/ref-effects-core.html) | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html)

## 用法

**描述：**当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

**返回：**[jQuery](http://api.jquery.com/Types/" \l "jQuery" \t "https://www.runoob.com/jqueryui/_blank)

.toggleClass( className [, switch ] [, duration ] [, easing ] [, complete ] )

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **类型** | **描述** | **默认值** |
| ClassName | String | 为匹配的元素集合中的每个元素要切换的一个或多个 class 名称，多个 class 名称用空格分隔 。 |  |
| switch | Boolean | 一个布尔值，指定 class 应被添加还是被移除。 |  |
| duration | Number 或 String | 一个字符串或一个数字，指定动画将运行多久。 | 400 |
| easing | String | 一个字符串，指示要使用的 [easing](https://www.runoob.com/jqueryui/api-easings.html) 函数。 | swing |
| complete | Function() | 一旦动画完成时要调用的函数。 |  |

.toggleClass( className [, switch ] [, options ] )

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **描述** |
| ClassName | String | 为匹配的元素集合中的每个元素要切换的一个或多个 class 名称，多个 class 名称用空格分隔 。 |
| switch | Boolean | 一个布尔值，指定 class 应被添加还是被移除。 |
| options | Object | 所有的动画设置。所有的属性是可选的。   * **duration**（默认值：400） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。 * **easing**（默认值：swing） 类型：String 描述：一个字符串，指示要使用的 [easing](https://www.runoob.com/jqueryui/api-easings.html) 函数。 * **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。 * **children**（默认值：false） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。 * **queue**（默认值：true） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。**自 jQuery 1.7 起**，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。 |

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 .toggleClass()，允许自定义动画持续时间和 [easing](https://www.runoob.com/jqueryui/api-easings.html) 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画结束时改变。

该插件扩展自 jQuery 内置的 [.toggleClass()](https://www.runoob.com/jquery/html-toggleClass.html) 方法。如果 jQuery UI 未加载，调用 .toggleClass() 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

# jQuery UI API 类别 - 交互（Interactions）

jQuery UI 提供了一套基于鼠标的交互。

|  |  |  |
| --- | --- | --- |
| **API** | **描述** | **也属于类别** |
| [可拖拽小部件（Draggable Widget）](https://www.runoob.com/jqueryui/api-draggable.html) | 允许使用鼠标移动元素。 |  |
| [可放置小部件（Droppable Widget）](https://www.runoob.com/jqueryui/api-droppable.html) | 为可拖拽小部件创建目标。 |  |
| [鼠标交互（Mouse Interaction）](https://www.runoob.com/jqueryui/api-mouse.html) | 基本交互层。 | [实用工具（Utilities）](https://www.runoob.com/jqueryui/ref-utilities.html) |
| [可调整尺寸小部件（Resizable Widget）](https://www.runoob.com/jqueryui/api-resizable.html) | 使用鼠标改变元素的尺寸。 |  |
| [可选择小部件（Selectable Widget）](https://www.runoob.com/jqueryui/api-selectable.html) | 使用鼠标选择单个元素或一组元素。 |  |
| [可排序小部件（Sortable Widget）](https://www.runoob.com/jqueryui/api-sortable.html) | 使用鼠标调整列表中或者网格中元素的排序。 |  |

# jQuery UI API - 可拖拽小部件（Draggable Widget）

## 所属类别

[交互（Interactions）](https://www.runoob.com/jqueryui/ref-interactions.html)

## 用法

**描述：**允许使用鼠标移动元素。

**版本新增：**1.0

**依赖：**

* [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html)
* [部件库（Widget Factory）](https://www.runoob.com/jqueryui/api-jQuery-widget.html)
* [鼠标交互（Mouse Interaction）](https://www.runoob.com/jqueryui/api-mouse.html)

**注释：**让被选元素可被鼠标拖拽。如果您不只是拖拽，而是拖拽 & 放置，请查看 [jQuery UI 可放置（Droppable）插件](https://www.runoob.com/jqueryui/api-droppable.html)，为可拖拽元素提供了一个放置目标。

### **快速导航**

|  |  |  |
| --- | --- | --- |
| **选项** | **方法** | **事件** |
| [addClasses](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-addClasses)  [appendTo](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-appendTo)  [axis](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-axis)  [cancel](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-cancel)  [connectToSortable](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-connectToSortable)  [containment](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-containment)  [cursor](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-cursor)  [cursorAt](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-cursorAt)  [delay](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-delay)  [disabled](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-disabled)  [distance](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-distance)  [grid](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-grid)  [handle](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-handle)  [helper](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-helper)  [iframeFix](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-iframeFix)  [opacity](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-opacity)  [refreshPositions](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-refreshPositions)  [revert](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-revert)  [revertDuration](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-revertDuration)  [scope](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-scope)  [scroll](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-scroll)  [scrollSensitivity](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-scrollSensitivity)  [scrollSpeed](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-scrollSpeed)  [snap](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-snap)  [snapMode](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-snapMode)  [snapTolerance](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-snapTolerance)  [stack](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-stack)  [zIndex](https://www.runoob.com/jqueryui/api-draggable.html" \l "option-zIndex) | [destroy](https://www.runoob.com/jqueryui/api-draggable.html" \l "method-destroy)  [disable](https://www.runoob.com/jqueryui/api-draggable.html" \l "method-disable)  [enable](https://www.runoob.com/jqueryui/api-draggable.html" \l "method-enable)  [option](https://www.runoob.com/jqueryui/api-draggable.html" \l "method-option)  [widget](https://www.runoob.com/jqueryui/api-draggable.html" \l "method-widget) | [create](https://www.runoob.com/jqueryui/api-draggable.html" \l "event-create)  [drag](https://www.runoob.com/jqueryui/api-draggable.html" \l "event-drag)  [start](https://www.runoob.com/jqueryui/api-draggable.html" \l "event-start)  [stop](https://www.runoob.com/jqueryui/api-draggable.html" \l "event-stop) |

|  |  |  |  |
| --- | --- | --- | --- |
| **选项** | **类型** | **描述** | **默认值** |
| addClasses | Boolean | 如果设置为 false，将阻止 ui-draggable class 被添加。当在数百个元素上调用 .draggable() 时，这么设置有利于性能优化。  **代码实例：**  初始化带有指定 addClasses 选项的 draggable：  $( ".selector" ).draggable({ addClasses: false });    在初始化后，获取或设置 addClasses 选项：  // gettervar addClasses = $( ".selector" ).draggable( "option", "addClasses" );  // setter  $( ".selector" ).draggable( "option", "addClasses", false ); | true |
| appendTo | jQuery 或 Element 或 Selector 或 String | 当拖拽时，draggable 助手（helper）要追加到哪一个元素。  **支持多个类型：**   * **jQuery**：一个 jQuery 对象，包含助手（helper）要追加到的元素。 * **Element**：要追加助手（helper）的元素。 * **Selector**：一个选择器，指定哪一个元素要追加助手（helper）。 * **String**：字符串 "parent" 将促使助手（helper）成为 draggable 的同级。   **代码实例：**  初始化带有指定 appendTo 选项的 draggable：  $( ".selector" ).draggable({ appendTo: "body" });    在初始化后，获取或设置 appendTo 选项：  // gettervar appendTo = $( ".selector" ).draggable( "option", "appendTo" );  // setter  $( ".selector" ).draggable( "option", "appendTo", "body" ); | "parent" |
| axis | String | 约束在水平轴 (x) 或垂直轴 (y) 上拖拽。可能的值："x", "y"。  **代码实例：**  初始化带有指定 axis 选项的 draggable：  $( ".selector" ).draggable({ axis: "x" });    在初始化后，获取或设置 axis 选项：  // gettervar axis = $( ".selector" ).draggable( "option", "axis" );  // setter  $( ".selector" ).draggable( "option", "axis", "x" ); | false |
| cancel | Selector | 防止从指定的元素上开始拖拽。  **代码实例：**  初始化带有指定 cancel 选项的 draggable：  $( ".selector" ).draggable({ cancel: ".title" });    在初始化后，获取或设置 cancel 选项：  // gettervar cancel = $( ".selector" ).draggable( "option", "cancel" );  // setter  $( ".selector" ).draggable( "option", "cancel", ".title" ); | "input, textarea, button, select, option" |
| connectToSortable | Selector | 允许 draggable 放置在指定的 sortable 上。如果使用了该选项，一个 draggable 可被放置在一个 sortable 列表上，然后成为列表的一部分。注意：helper 选项必须设置为 "clone"，以便更好地工作。必须包含 [可排序小部件（Sortable Widget）](https://www.runoob.com/jqueryui/api-sortable.html)。  **代码实例：**  初始化带有指定 connectToSortable 选项的 draggable：  $( ".selector" ).draggable({ connectToSortable: "#my-sortable" });    在初始化后，获取或设置 connectToSortable 选项：  // gettervar connectToSortable = $( ".selector" ).draggable( "option", "connectToSortable" );  // setter  $( ".selector" ).draggable( "option", "connectToSortable", "#my-sortable" ); | false |
| containment | Selector 或 Element 或 String 或 Array | 约束在指定元素或区域的边界内拖拽。  **支持多个类型：**   * **Selector**：可拖拽元素将被包含在 selector 第一个元素的边界内。如果未找到元素，则不设置 containment。 * **Element**：可拖拽元素将被包含在元素的边界。 * **String**：可能的值："parent"、"document"、"window"。 * **Array**：一个数组， 以形式 [ x1, y1, x2, y2 ] 定义元素的边界。   **代码实例：**  初始化带有指定 containment 选项的 draggable：  $( ".selector" ).draggable({ containment: "parent" });    在初始化后，获取或设置 containment 选项：  // gettervar containment = $( ".selector" ).draggable( "option", "containment" );  // setter  $( ".selector" ).draggable( "option", "containment", "parent" ); | false |
| cursor | String | 拖拽操作期间的 CSS 光标。  **代码实例：**  初始化带有指定 cursor 选项的 draggable：  $( ".selector" ).draggable({ cursor: "crosshair" });    在初始化后，获取或设置 cursor 选项：  // gettervar cursor = $( ".selector" ).draggable( "option", "cursor" );  // setter  $( ".selector" ).draggable( "option", "cursor", "crosshair" ); | "auto" |
| cursorAt | Object | 设置拖拽助手（helper）相对于鼠标光标的偏移。坐标可通过一个或两个键的组合成一个哈希给出：{ top, left, right, bottom }。  **代码实例：**  初始化带有指定 cursorAt 选项的 draggable：  $( ".selector" ).draggable({ cursorAt: { left: 5 } });    在初始化后，获取或设置 cursorAt 选项：  // gettervar cursorAt = $( ".selector" ).draggable( "option", "cursorAt" );  // setter  $( ".selector" ).draggable( "option", "cursorAt", { left: 5 } ); | false |
| delay | Number | 鼠标按下后直到拖拽开始为止的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的拖拽。  **代码实例：**  初始化带有指定 delay 选项的 draggable：  $( ".selector" ).draggable({ delay: 300 });    在初始化后，获取或设置 delay 选项：  // gettervar delay = $( ".selector" ).draggable( "option", "delay" );  // setter  $( ".selector" ).draggable( "option", "delay", 300 ); | 0 |
| disabled | Boolean | 如果设置为 true，则禁用该 draggable。  **代码实例：**  初始化带有指定 disabled 选项的 draggable：  $( ".selector" ).draggable({ disabled: true });    在初始化后，获取或设置 disabled 选项：  // gettervar disabled = $( ".selector" ).draggable( "option", "disabled" );  // setter  $( ".selector" ).draggable( "option", "disabled", true ); | false |
| distance | Number | 鼠标按下后拖拽开始前必须移动的距离，以像素计。该选项可以防止点击在某个元素上时不必要的拖拽。  **代码实例：**  初始化带有指定 distance 选项的 draggable：  $( ".selector" ).draggable({ distance: 10 });    在初始化后，获取或设置 distance 选项：  // gettervar distance = $( ".selector" ).draggable( "option", "distance" );  // setter  $( ".selector" ).draggable( "option", "distance", 10 ); | 1 |
| grid | Array | 对齐拖拽助手（helper）到网格，每个 x 和 y 像素。数组形式必须是 [ x, y ]。  **代码实例：**  初始化带有指定 grid 选项的 draggable：  $( ".selector" ).draggable({ grid: [ 50, 20 ] });    在初始化后，获取或设置 grid 选项：  // gettervar grid = $( ".selector" ).draggable( "option", "grid" );  // setter  $( ".selector" ).draggable( "option", "grid", [ 50, 20 ] ); | false |
| handle | Selector 或 Element | 如果指定了该选项，则限制开始拖拽，除非鼠标在指定的元素上按下。只有可拖拽（draggable）元素的后代元素才允许被拖拽。  **代码实例：**  初始化带有指定 handle 选项的 draggable：  $( ".selector" ).draggable({ handle: "h2" });    在初始化后，获取或设置 handle 选项：  // gettervar handle = $( ".selector" ).draggable( "option", "handle" );  // setter  $( ".selector" ).draggable( "option", "handle", "h2" ); | false |
| helper | String 或 Function() | 允许一个 helper 元素用于拖拽显示。  **支持多个类型：**   * **String**：如果设置为 "clone"，元素将被克隆，且克隆将被拖拽。 * **Function**：一个函数，将返回拖拽时要使用的 DOMElement。   **代码实例：**  初始化带有指定 helper 选项的 draggable：  $( ".selector" ).draggable({ helper: "clone" });    在初始化后，获取或设置 helper 选项：  // gettervar helper = $( ".selector" ).draggable( "option", "helper" );  // setter  $( ".selector" ).draggable( "option", "helper", "clone" ); | "original" |
| iframeFix | Boolean 或 Selector | 防止拖拽期间 iframes 捕捉鼠标移动（mousemove ）事件。在与 cursorAt选项结合使用时，或鼠标光标未覆盖在助手（helper）上时，非常有用。  **支持多个类型：**   * **Boolean**：当设置为 true 时，透明遮罩将被放置在页面上所有 iframes 上。 * **Selector**：匹配 selector 的任意 iframes 将被透明遮罩覆盖。   **代码实例：**  初始化带有指定 iframeFix 选项的 draggable：  $( ".selector" ).draggable({ iframeFix: true });    在初始化后，获取或设置 iframeFix 选项：  // gettervar iframeFix = $( ".selector" ).draggable( "option", "iframeFix" );  // setter  $( ".selector" ).draggable( "option", "iframeFix", true ); | false |
| opacity | Number | 当被拖拽时助手（helper）的不透明度。  **代码实例：**  初始化带有指定 opacity 选项的 draggable：  $( ".selector" ).draggable({ opacity: 0.35 });    在初始化后，获取或设置 opacity 选项：  // gettervar opacity = $( ".selector" ).draggable( "option", "opacity" );  // setter  $( ".selector" ).draggable( "option", "opacity", 0.35 ); | false |
| refreshPositions | Boolean | 如果设置为 true，在每次鼠标移动（mousemove）时都会计算所有可放置的位置。注意：这解决了高度动态的问题，但是明显降低了性能。  **代码实例：**  初始化带有指定 refreshPositions 选项的 draggable：  $( ".selector" ).draggable({ refreshPositions: true });    在初始化后，获取或设置 refreshPositions 选项：  // gettervar refreshPositions = $( ".selector" ).draggable( "option", "refreshPositions" );  // setter  $( ".selector" ).draggable( "option", "refreshPositions", true ); | false |
| revert | Boolean 或 String 或 Function() | 当拖拽停止时，元素是否还原到它的开始位置。  **支持多个类型：**   * **Boolean**：如果设置为 true，元素总会还原。 * **String**：如果设置为 "invalid"，还原仅在 draggable 未放置在 droppable 上时发生，如果设置为 "valid" 则相反。 * **Function**：一个函数，确定元素是否还原到它的开始位置。该函数必须返回 true 才能还原元素。   **代码实例：**  初始化带有指定 revert 选项的 draggable：  $( ".selector" ).draggable({ revert: true });    在初始化后，获取或设置 revert 选项：  // gettervar revert = $( ".selector" ).draggable( "option", "revert" );  // setter  $( ".selector" ).draggable( "option", "revert", true ); | false |
| revertDuration | Number | 还原（revert）动画的持续时间，以毫秒计。如果 revert 选项是 false 则忽略。  **代码实例：**  初始化带有指定 revertDuration 选项的 draggable：  $( ".selector" ).draggable({ revertDuration: 200 });    在初始化后，获取或设置 revertDuration 选项：  // gettervar revertDuration = $( ".selector" ).draggable( "option", "revertDuration" );  // setter  $( ".selector" ).draggable( "option", "revertDuration", 200 ); | 500 |
| scope | String | 用于组合配套 draggable 和 droppable 项，除了 droppable 的 accept 选项之外。一个与 droppable 带有相同的 scope 值的 draggable 会被该 droppable 接受。  **代码实例：**  初始化带有指定 scope 选项的 draggable：  $( ".selector" ).draggable({ scope: "tasks" });    在初始化后，获取或设置 scope 选项：  // gettervar scope = $( ".selector" ).draggable( "option", "scope" );  // setter  $( ".selector" ).draggable( "option", "scope", "tasks" ); | "default" |
| scroll | Boolean | 如果设置为 true，当拖拽时容器会自动滚动。  **代码实例：**  初始化带有指定 scroll 选项的 draggable：  $( ".selector" ).draggable({ scroll: false });    在初始化后，获取或设置 scroll 选项：  // gettervar scroll = $( ".selector" ).draggable( "option", "scroll" );  // setter  $( ".selector" ).draggable( "option", "scroll", false ); | true |
| scrollSensitivity | Number | 从要滚动的视区边缘起的距离，以像素计。距离是相对于指针的，不是相对于 draggable。如果 scroll 选项是 false 则忽略。  **代码实例：**  初始化带有指定 scrollSensitivity 选项的 draggable：  $( ".selector" ).draggable({ scrollSensitivity: 100 });    在初始化后，获取或设置 scrollSensitivity 选项：  // gettervar scrollSensitivity = $( ".selector" ).draggable( "option", "scrollSensitivity" );  // setter  $( ".selector" ).draggable( "option", "scrollSensitivity", 100 ); | 20 |
| scrollSpeed | Number | 当鼠标指针获取到在 scrollSensitivity 距离内时，窗体滚动的速度。如果 scroll 选项是 false 则忽略。  **代码实例：**  初始化带有指定 scrollSpeed 选项的 draggable：  $( ".selector" ).draggable({ scrollSpeed: 100 });    在初始化后，获取或设置 scrollSpeed 选项：  // gettervar scrollSpeed = $( ".selector" ).draggable( "option", "scrollSpeed" );  // setter  $( ".selector" ).draggable( "option", "scrollSpeed", 100 ); | 20 |
| snap | Boolean 或 Selector | 元素是否对齐到其他元素。  **支持多个类型：**   * **Boolean**：当设置为 true 时，元素会对齐到其它可拖拽（draggable ）元素。 * **Selector**：一个选择器，指定要对齐到哪个元素。   **代码实例：**  初始化带有指定 snap 选项的 draggable：  $( ".selector" ).draggable({ snap: true });    在初始化后，获取或设置 snap 选项：  // gettervar snap = $( ".selector" ).draggable( "option", "snap" );  // setter  $( ".selector" ).draggable( "option", "snap", true ); | false |
| snapMode | String | 决定 draggable 将对齐到对齐元素的哪个边缘。如果 snap 选项是 false 则忽略。可能的值："inner"、"outer"、"both"。  **代码实例：**  初始化带有指定 snapMode 选项的 draggable：  $( ".selector" ).draggable({ snapMode: "inner" });    在初始化后，获取或设置 snapMode 选项：  // gettervar snapMode = $( ".selector" ).draggable( "option", "snapMode" );  // setter  $( ".selector" ).draggable( "option", "snapMode", "inner" ); | "both" |
| snapTolerance | Number | 从要发生对齐的对齐元素边缘起的距离，以像素计。如果 snap 选项是 false 则忽略。  **代码实例：**  初始化带有指定 snapTolerance 选项的 draggable：  $( ".selector" ).draggable({ snapTolerance: 30 });    在初始化后，获取或设置 snapTolerance 选项：  // gettervar snapTolerance = $( ".selector" ).draggable( "option", "snapTolerance" );  // setter  $( ".selector" ).draggable( "option", "snapTolerance", 30 ); | 20 |
| stack | Selector | 控制匹配选择器（selector）的元素集合的 z-index，总是在当前拖拽项的前面，在类似窗口管理器这样的事物中非常有用。  **代码实例：**  初始化带有指定 stack 选项的 draggable：  $( ".selector" ).draggable({ stack: ".products" });    在初始化后，获取或设置 stack 选项：  // gettervar stack = $( ".selector" ).draggable( "option", "stack" );  // setter  $( ".selector" ).draggable( "option", "stack", ".products" ); | false |
| zIndex | Number | 当被拖拽时，助手（helper）的 Z-index。  **代码实例：**  初始化带有指定 zIndex 选项的 draggable：  $( ".selector" ).draggable({ zIndex: 100 });    在初始化后，获取或设置 zIndex 选项：  // gettervar zIndex = $( ".selector" ).draggable( "option", "zIndex" );  // setter  $( ".selector" ).draggable( "option", "zIndex", 100 ); | false |

|  |  |  |
| --- | --- | --- |
| **方法** | **返回** | **描述** |
| destroy() | jQuery (plugin only) | 完全移除 draggable 功能。这会把元素返回到它的预初始化状态。   * 该方法不接受任何参数。   **代码实例：**  调用 destroy 方法：  $( ".selector" ).draggable( "destroy" ); |
| disable() | jQuery (plugin only) | 禁用 draggable。   * 该方法不接受任何参数。   **代码实例：**  调用 disable 方法：  $( ".selector" ).draggable( "disable" ); |
| enable() | jQuery (plugin only) | 启用 draggable。   * 该方法不接受任何参数。   **代码实例：**  调用 enable 方法：  $( ".selector" ).draggable( "enable" ); |
| option( optionName ) | Object | 获取当前与指定的 optionName 关联的值。   * **optionName** 类型：String 描述：要获取的选项的名称。   **代码实例：**  调用该方法：  var isDisabled = $( ".selector" ).draggable( "option", "disabled" ); |
| option() | PlainObject | 获取一个包含键/值对的对象，键/值对表示当前 draggable 选项哈希。   * 该方法不接受任何参数。   **代码实例：**  调用该方法：  var options = $( ".selector" ).draggable( "option" ); |
| option( optionName, value ) | jQuery (plugin only) | 设置与指定的 optionName 关联的 draggable 选项的值。   * **optionName** 类型：String 描述：要设置的选项的名称。 * **value** 类型：Object 描述：要为选项设置的值。   **代码实例：**  调用该方法：  $( ".selector" ).draggable( "option", "disabled", true ); |
| option( options ) | jQuery (plugin only) | 为 draggable 设置一个或多个选项。   * **options** 类型：Object 描述：要设置的 option-value 对。   **代码实例：**  调用该方法：  $( ".selector" ).draggable( "option", { disabled: true } ); |
| widget() | jQuery | 返回一个包含 draggable 元素的 jQuery 对象。   * 该方法不接受任何参数。   **代码实例：**  调用 widget 方法：  var widget = $( ".selector" ).draggable( "widget" ); |

|  |  |  |
| --- | --- | --- |
| **事件** | **类型** | **描述** |
| create( event, ui ) | dragcreate | 当 draggable 被创建时触发。   * **event** 类型：Event * **ui** 类型：Object   注意：ui 对象是空的，这里包含它是为了与其他事件保持一致性。  **代码实例：**  初始化带有指定 create 回调的 draggable：  $( ".selector" ).draggable({  create: function( event, ui ) {}});    绑定一个事件监听器到 dragcreate 事件：  $( ".selector" ).on( "dragcreate", function( event, ui ) {} ); |
| drag( event, ui ) | drag | 在拖拽期间当鼠标移动时触发。   * **event** 类型：Event * **ui** 类型：Object   + **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。   + **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 { top, left } 对象。   + **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 { top, left } 对象。   **代码实例：**  初始化带有指定 drag 回调的 draggable：  $( ".selector" ).draggable({  drag: function( event, ui ) {}});    绑定一个事件监听器到 drag 事件：  $( ".selector" ).on( "drag", function( event, ui ) {} ); |
| start( event, ui ) | dragstart | 当拖拽开始时触发。   * **event** 类型：Event * **ui** 类型：Object   + **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。   + **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 { top, left } 对象。   + **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 { top, left } 对象。   **代码实例：**  初始化带有指定 start 回调的 draggable：  $( ".selector" ).draggable({  start: function( event, ui ) {}});    绑定一个事件监听器到 dragstart 事件：  $( ".selector" ).on( "dragstart", function( event, ui ) {} ); |
| stop( event, ui ) | dragstop | 当拖拽停止时触发。   * **event** 类型：Event * **ui** 类型：Object   + **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。   + **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 { top, left } 对象。   + **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 { top, left } 对象。   **代码实例：**  初始化带有指定 stop 回调的 draggable：  $( ".selector" ).draggable({  stop: function( event, ui ) {}});    绑定一个事件监听器到 dragstop 事件：  $( ".selector" ).on( "dragstop", function( event, ui ) {} ); |

# Query UI API 类别 - 方法重载（Method Overrides）

jQuery UI 重载了几个内置的 jQuery 方法，以提供额外的功能。当使用这些重载时，确保加载 jQuery UI 是很重要的。如果 jQuery UI 未加载，方法依然存在，但预期的功能将不可用，这会导致难以追踪的错误。

# jQuery UI API 类别 - 方法（Methods）

尽管 jQuery UI 主要由 [小部件（Widgets）](https://www.runoob.com/jqueryui/ref-widgets.html)、[交互（Interactions）](https://www.runoob.com/jqueryui/ref-interactions.html) 和 [特效（Effects）](https://www.runoob.com/jqueryui/ref-effects.html) 组成，但是还有一些简单的方法，增加了便利性。

# jQuery UI API 类别 - 主题（Theming）

jQuery UI 包括一个强大的 CSS 框架，用于创建自定义的 jQuery 小部件。该框架包括涵盖广泛的公共用户界面需求的 Class，并且可以使用 [jQuery UI ThemeRoller](http://jqueryui.com/themeroller/) 进行操纵。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件，您应采用共享标记公约，便于插件社区的代码集成。您可以查看 [jQuery UI 主题](https://www.runoob.com/jqueryui/jqueryui-theme.html) 了解更多详情。

|  |  |
| --- | --- |
| **API** | **描述** |
| [CSS 框架（CSS Framework）](https://www.runoob.com/jqueryui/api-css-framework.html) | jQuery UI 使用的允许组件主题化的 Class 名称列表。 |
| [图标（Icons）](https://www.runoob.com/jqueryui/api-icons.html) | jQuery UI 提供的图标列表。 |
| [堆叠元素（Stacking Elements）](https://www.runoob.com/jqueryui/api-stacking-elements.html) | 一种处理 z-index 和堆叠元素的模式。 |

# jQuery UI API 类别 - UI 核心（UI Core）

由 jquery.ui.core.js 提供的功能。

|  |  |  |
| --- | --- | --- |
| **API** | **描述** | **也属于类别** |
| [:data() Selector](https://www.runoob.com/jqueryui/api-data-selector.html) | 选择数据已存储在指定的键下的元素。 | [选择器（Selectors）](https://www.runoob.com/jqueryui/ref-selectors.html) |
| [.disableSelection()](https://www.runoob.com/jqueryui/api-disableSelection.html) | 禁用选择匹配的元素集合内的文本内容。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.enableSelection()](https://www.runoob.com/jqueryui/api-enableSelection.html) | 启用选择匹配的元素集合内的文本内容。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.focus()](https://www.runoob.com/jqueryui/api-focus.html) | 异步聚焦到一个元素。 | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [:focusable Selector](https://www.runoob.com/jqueryui/api-focusable-selector.html) | 选择可被聚焦的元素。 | [选择器（Selectors）](https://www.runoob.com/jqueryui/ref-selectors.html) |
| [jQuery.ui.keyCode](https://www.runoob.com/jqueryui/api-jQuery-ui-keyCode.html) | 一个相对于数字值的关键代码描述的映射。 |  |
| [.removeUniqueId()](https://www.runoob.com/jqueryui/api-removeUniqueId.html) | 为匹配的元素集合移除由 .uniqueId() 设置的 Id。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.scrollParent()](https://www.runoob.com/jqueryui/api-scrollParent.html) | 获取最近的可滚动的祖先。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [:tabbable Selector](https://www.runoob.com/jqueryui/api-tabbable-selector.html) | 选择用户可通过 tab 键聚焦的元素。 | [选择器（Selectors）](https://www.runoob.com/jqueryui/ref-selectors.html) |
| [.uniqueId()](https://www.runoob.com/jqueryui/api-uniqueId.html) | 为匹配的元素集合生成并申请一个唯一的 Id。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |
| [.zIndex()](https://www.runoob.com/jqueryui/api-zIndex.html) | 为元素获取 z-index。 | [方法（Methods）](https://www.runoob.com/jqueryui/ref-methods.html) |

# jQuery UI API 类别 - 选择器（Selectors）

|  |  |  |
| --- | --- | --- |
| **API** | **描述** | **也属于类别** |
| [:data() Selector](https://www.runoob.com/jqueryui/api-data-selector.html) | 选择数据已存储在指定的键下的元素。 | [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html) |
| [:focusable Selector](https://www.runoob.com/jqueryui/api-focusable-selector.html) | 选择可被聚焦的元素。 | [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html) |
| [:tabbable Selector](https://www.runoob.com/jqueryui/api-tabbable-selector.html) | 选择用户可通过 tab 键聚焦的元素。 | [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html) |

# jQuery UI API 类别 - 实用工具（Utilities）

|  |  |  |
| --- | --- | --- |
| **API** | **描述** | **也属于类别** |
| [Easings](https://www.runoob.com/jqueryui/api-easings.html) | Easing 函数指定动画在不同点上的行进速度。 |  |
| [部件库（Widget Factory）](https://www.runoob.com/jqueryui/api-jQuery-widget.html) | 使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。 | [小部件（Widgets）](https://www.runoob.com/jqueryui/ref-widgets.html) |
| [插件桥（Widget Plugin Bridge）](https://www.runoob.com/jqueryui/api-jQuery-widget-bridge.html) | jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。 | [小部件（Widgets）](https://www.runoob.com/jqueryui/ref-widgets.html) |
| [鼠标交互（Mouse Interaction）](https://www.runoob.com/jqueryui/api-mouse.html) | 基本交互层。 | [交互（Interactions）](https://www.runoob.com/jqueryui/ref-interactions.html) |
| [.position()](https://www.runoob.com/jqueryui/api-position.html) | 相对另一个元素定位一个元素。 | [方法重载（Method Overrides）](https://www.runoob.com/jqueryui/ref-overrides.html) | [方法（Method）](https://www.runoob.com/jqueryui/ref-methods.html) |

# jQuery UI API - 滑块部件（Slider Widget）

## 所属类别

[小部件（Widgets）](https://www.runoob.com/jqueryui/ref-widgets.html)

## 用法

**描述：**拖动手柄来选择一个数值。

**版本新增：**1.5

jQuery UI 滑块（Slider）插件允许通过滑块进行选择。有各种不同的选项，比如多个手柄和范围。手柄可通过鼠标或箭头按键进行移动。

滑块部件（Slider Widget）会在初始化时创建带有 class ui-slider-handle 的手柄元素。您可以通过在初始化之前创建并追加元素，同时向元素添加 ui-slider-handle class 来指定自定义的手柄元素。它只会创建匹配 [value](https://www.runoob.com/jqueryui/api-slider.html" \l "option-value)/[values](https://www.runoob.com/jqueryui/api-slider.html" \l "option-values) 长度所需的数量的手柄。例如，如果您指定 values: [ 1, 5, 18 ]，且创建一个自定义手柄，插件将创建其他两个。

### **主题化**

滑块部件（Slider Widget）使用 [jQuery UI CSS 框架](https://www.runoob.com/jqueryui/api-css-framework.html) 来定义它的外观和感观的样式。如果需要使用滑块指定的样式，则可以使用下面的 CSS class 名称：

* ui-slider：滑块控件的轨道。该元素会根据滑块的 [orientation](https://www.runoob.com/jqueryui/api-slider.html" \l "option-orientation) 另外带有一个 ui-slider-horizontal 或 ui-slider-vertical class。
  + ui-slider-handle：滑块手柄。
  + ui-slider-range：当设置 [range](https://www.runoob.com/jqueryui/api-slider.html" \l "option-range) 选项时使用的已选范围。如果 range 选项设置为 "min" 或 "max"，则该元素会分别另外带有一个 ui-slider-range-min 或 ui-slider-range-max class。

### **依赖**

* [UI 核心（UI Core）](https://www.runoob.com/jqueryui/ref-ui-core.html)
* [部件库（Widget Factory）](https://www.runoob.com/jqueryui/api-jQuery-widget.html)
* [鼠标交互（Mouse Interaction）](https://www.runoob.com/jqueryui/api-mouse.html)

### **附加说明**

* 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

# jQuery UI 实例

jQuery UI 提供了一组用户界面交互、特效、小部件、实用工具及主题。通过实例演示、查看源代码、[创建主题](http://jqueryui.com/themeroller/" \t "https://www.runoob.com/jqueryui/_blank)、阅读[API 文档](https://www.runoob.com/jqueryui/jqueryui-api.html)，开始学习使用 jQuery UI。

## 交互（Interactions）

交互（Interactions）向任何元素添加基本的基于鼠标的行为。您可以只通过短短的几行代码即可创建排序列表、缩放元素、拖放行为等等。交互（Interactions）也作用于更复杂的小部件和应用程序。

* [拖动（Draggable）](https://www.runoob.com/jqueryui/example-draggable.html)
* [放置（Droppable）](https://www.runoob.com/jqueryui/example-droppable.html)
* [缩放（Resizable）](https://www.runoob.com/jqueryui/example-resizable.html)
* [选择（Selectable）](https://www.runoob.com/jqueryui/example-selectable.html)
* [排序（Sortable）](https://www.runoob.com/jqueryui/example-sortable.html)

## 小部件（Widgets）

小部件（Widgets）有功能齐全的 UI 控件，使桌面应用程序也具备 Web 应用程序一样丰富的功能。所有的小部件（Widgets）提供了一个核心，带有定制行为的大量扩展以及完整的主题支持。

* [折叠面板（Accordion）](https://www.runoob.com/jqueryui/example-accordion.html)
* [自动完成（Autocomplete）](https://www.runoob.com/jqueryui/example-autocomplete.html)
* [按钮（Button）](https://www.runoob.com/jqueryui/example-button.html)
* [日期选择器（Datepicker）](https://www.runoob.com/jqueryui/example-datepicker.html)
* [对话框（Dialog）](https://www.runoob.com/jqueryui/example-dialog.html)
* [菜单（Menu）](https://www.runoob.com/jqueryui/example-menu.html)
* [进度条（Progressbar）](https://www.runoob.com/jqueryui/example-progressbar.html)
* [滑块（Slider）](https://www.runoob.com/jqueryui/example-slider.html)
* [旋转器（Spinner）](https://www.runoob.com/jqueryui/example-spinner.html)
* [标签页（Tabs）](https://www.runoob.com/jqueryui/example-tabs.html)
* [工具提示框（Tooltip）](https://www.runoob.com/jqueryui/example-tooltip.html)

## 特效（Effects）

特效（Effects）支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

* [特效（Effect）](https://www.runoob.com/jqueryui/example-effect.html)
* 可见性（Visibility）
  + [显示（Show）](https://www.runoob.com/jqueryui/example-show.html)
  + [隐藏（Hide）](https://www.runoob.com/jqueryui/example-hide.html)
  + [切换（Toggle）](https://www.runoob.com/jqueryui/example-toggle.html)
* Class 动画（Class Animation）
  + [添加 Class（Add Class）](https://www.runoob.com/jqueryui/example-addClass.html)
  + [移除 Class（Remove Class）](https://www.runoob.com/jqueryui/example-removeClass.html)
  + [切换 Class（Toggle Class）](https://www.runoob.com/jqueryui/example-toggleClass.html)
  + [转换 Class（Switch Class）](https://www.runoob.com/jqueryui/example-switchClass.html)
* [颜色动画（Color Animation）](https://www.runoob.com/jqueryui/example-animate.html)

## 实用工具（Utilities）

jQuery UI 使用实体工具（Utilities）来创建交互（interactions）和小部件（widgets）。

* [定位（Position）](https://www.runoob.com/jqueryui/example-position.html)
* [部件库（Widget Factory）](https://www.runoob.com/jqueryui/example-widget.html)

# jQuery UI 实例 - 按钮（Button）

用带有适当的悬停（hover）和激活（active）的样式的可主题化按钮来加强标准表单元素（比如按钮、输入框、锚）的功能。

# jQuery UI 实例 - 拖动（Draggable）

允许使用鼠标移动元素。

如需了解更多有关 draggable 交互的细节，请查看 API 文档 [可拖拽小部件（Draggable Widget）](https://www.runoob.com/jqueryui/api-draggable.html)。