下面这些简单的小技巧能够帮助你玩转jQuery，帮你提升 jQuery 技能。

* 返回顶部按钮
* 预加载图像
* 检查图像是否加载
* 自动修复破坏的图像
* 悬停切换类
* 禁用输入字段
* 停止加载链接
* 切换淡入/幻灯片
* 简单的手风琴
* 让两个div高度相同
* 在新标签页/窗口打开外部链接
* 通过文本查找元素
* 在改变Visibility时触发
* AJAX调用错误处理
* 链式插件调用

**返回顶部按钮**

通过使用jQuery中的 animate 和 scrollTop 方法，不用插件就可以创建一个滚动到顶部的简单动画：

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Back to top  $('.top').click(function (e) {    e.preventDefault();    $('html, body').animate({scrollTop: 0}, 800);  }); |

|  |  |
| --- | --- |
| 1  2 | <!-- Create an anchor tag -->  <a class="top" href="#">Back to top</a> |

改变 scrollTop 的值可以更改你想要放置滚动条的位置。所有你真正需要做的是在800毫秒的时间内设置文档主体的动画，直到它滚动到文档的顶部。

注：小心scrollTop的一些错误行为。

**预加载图像**

如果你的网页要使用大量开始不可见的（例如，悬停的）图像，那么可以预加载这些图像：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $.preloadImages = function () {      for (var i = 0; i < arguments.length; i++) {          $('<img>').attr('src', arguments[i]);      }  };    $.preloadImages('img/hover-on.png', 'img/hover-off.png'); |

**检查图像是否加载**

有时为了继续脚本，你可能需要检查图像是否全部加载完毕：

|  |  |
| --- | --- |
| 1  2  3 | $('img').load(function () {      console.log('image load successful');  }); |

你也可以用ID或类替换<img>标签来检查某个特定的图像是否被加载。

**自动修复破坏的图像**

逐个替换已经破坏的图像链接是非常痛苦的。不过，下面这段简单的代码可以帮助你：

|  |  |
| --- | --- |
| 1  2  3  4  5 | $('img').on('error', function () {      if(!$(this).hasClass('broken-image')) {          $(this).prop('src', 'img/broken.png').addClass('broken-image');      }  }); |

即使没有任何断掉的链接，加上这一段代码也不会让你有任何损失。

**悬停切换类**

假设你希望当用户将鼠标悬停在可点击的元素上时，它会改变颜色。那么你可以在用户悬停的时候添加类到元素中，反之则删除类：

|  |  |
| --- | --- |
| 1  2  3  4  5 | $('.btn').hover(function () {      $(this).addClass('hover');  }, function () {      $(this).removeClass('hover');  }); |

你只需要添加必要的CSS即可。更简单的方法是使用toggleClass 方法：

|  |  |
| --- | --- |
| 1  2  3 | $('.btn').hover(function () {      $(this).toggleClass('hover');  }); |

注：可能在这种情况下，CSS这种解决方案更快，不过了解这个方法很有必要。

**禁用输入字段**

有时候，你可能想要禁用表格的提交按钮或它的某一项文字输入直到用户执行了特定操作（例如，勾选“我已阅读相关条款”复选框）。添加 disabled属性到你的输入就可以在你想要的时候才启用它：

|  |  |
| --- | --- |
| 1 | $('input[type="submit"]').prop('disabled', true); |

然后你只需要运行输入的prop 方法就可以了，不过disabled 的值要设置为false：

|  |  |
| --- | --- |
| 1 | $('input[type="submit"]').prop('disabled', false); |

**停止加载链接**

有时候，你既不需要链接到某个特定的网页，也不想要重新加载页面——你可能希望链接做点别的事情，例如说触发一些其他脚本。这就要在阻止默认动作上做文章了：

|  |  |
| --- | --- |
| 1  2  3 | $('a.no-link').click(function (e) {      e.preventDefault();  }); |

**淡入/滑动切换**

滑动和淡入都是我们用jQuery做动画的时候大量运用的东西。如果你只是想在用户点击之后展示一个元素的话，那么用fadeIn 和slideDown 方法就很完美。但是，如果你想要元素在第一次点击的时候出现，然后在第二次点击的时候消失的话，那么可以试试下面的代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Fade  $('.btn').click(function () {      $('.element').fadeToggle('slow');  });    // Toggle  $('.btn').click(function () {      $('.element').slideToggle('slow');  }); |

**简单的手风琴**

这是一个可快速生成手风琴的简单方法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // Close all panels  $('#accordion').find('.content').hide();    // Accordion  $('#accordion').find('.accordion-header').click(function () {      var next = $(this).next();      next.slideToggle('fast');      $('.content').not(next).slideUp('fast');      return false;  }); |

通过添加这个脚本，你真正需要做的仅仅是在页面上添加必要的HTML元素，这样它就可以运行工作了。

**让两个div高度相同**

有时候，你需要让两个div无论包含什么内容都拥有相同的高度：

|  |  |
| --- | --- |
| 1 | $('.div').css('min-height', $('.main-div').height()); |

设置 min-height，这意味着它可以比主div大但绝对不能比主div小。不过，还有一种更灵活的方法是遍历一组元素，然后将高度设置为最高的那个元素的高度：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var $columns = $('.column');  var height = 0;  $columns.each(function () {      if ($(this).height() > height) {          height = $(this).height();      }  });  $columns.height(height); |

如果你希望所有列的高度相同：

|  |  |
| --- | --- |
| 1  2  3  4 | var $rows = $('.same-height-columns');  $rows.each(function () {      $(this).find('.column').height($(this).height());  }); |

**在新标签页/窗口打开外部链接**

在一个新的浏览器tab或窗口中打开外部链接，并确保同一个来源的链接能在同一个tab或者窗口中打开：

|  |  |
| --- | --- |
| 1  2  3 | $('a[href^="http"]').attr('target', '\_blank');  $('a[href^="//"]').attr('target', '\_blank');  $('a[href^="' + window.location.origin + '"]').attr('target', '\_self'); |

注意：window.location.origin 在IE10中无效。修复的时候要小心这个问题。

**通过文本查找元素**

通过使用jQuery中的contains() 选择器，你可以找到元素内容的文本。如果文本不存在，那就隐藏该元素：

|  |  |
| --- | --- |
| 1  2 | var search = $('#search').val();  $('div:not(:contains("' + search + '"))').hide(); |

**在改变Visibility时触发**

当用户不再关注某个tab，或重新聚焦原来的那个tab上时，触发JavaScript：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $(document).on('visibilitychange', function (e) {      if (e.target.visibilityState === "visible") {          console.log('Tab is now in view!');      } else if (e.target.visibilityState === "hidden") {          console.log('Tab is now hidden!');      }  }); |

**AJAX调用错误处理**

当Ajax调用返回404或500错误时，就执行错误处理程序。如果没有定义处理程序，其他的jQuery代码或会就此罢工。定义一个全局的Ajax错误处理程序：

|  |  |
| --- | --- |
| 1  2  3 | $(document).ajaxError(function (e, xhr, settings, error) {      console.log(error);  }); |

**链式插件调用**

jQuery允许“链式”插件的方法调用，以减轻反复查询DOM并创建多个jQuery对象的过程。比方说，下面的代码片段代表了你的插件方法调用：

|  |  |
| --- | --- |
| 1  2  3 | $('#elem').show();  $('#elem').html('bla');  $('#elem').otherStuff(); |

通过使用链式，可以大大改善：

|  |  |
| --- | --- |
| 1  2  3  4 | $('#elem')    .show()    .html('bla')    .otherStuff(); |

还有一种方法是在（前缀$）变量中高速缓存元素：

|  |  |
| --- | --- |
| 1  2  3  4 | var $elem = $('#elem');  $elem.hide();  $elem.html('bla');  $elem.otherStuff(); |

链式和高速缓存的方法都是jQuery中可以让代码变得更短和更快的代最佳做法。