**事件**

**本节内容**

* 页面加载事件
* 简单事件
* toggleClass
* hover
* 事件的传播
* 移除事件
* 模仿用户操作
* 键盘事件

**页面加载事件——$(document).ready()**

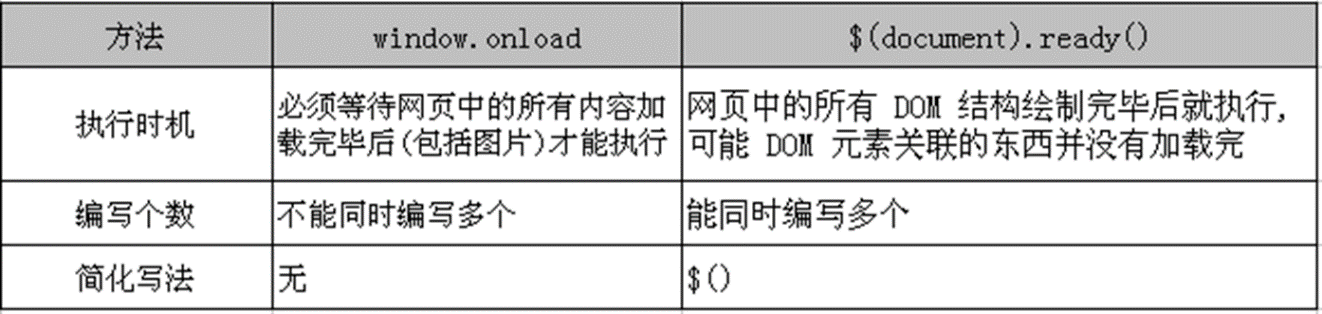
1、代码执行的时机

* $(document).ready()是jQuery基于页面加载执行任务的一种主要方式。
* 通过$(document).ready()注册的事件处理程序，则会在DOM完全就绪并可以使用时调用。这时关联文件不一定下载完毕。
* 而原生的window.onload事件是将页面上的全部元素下载完毕才会调用。

2. $(document).ready()内部存在一个行为队列，每次调用这个方法都 会向队列中添加一个新函数，当期页面加载完成后，所有函数都会被执行。而且，这些函数会按照注册它们的顺序依次执行。

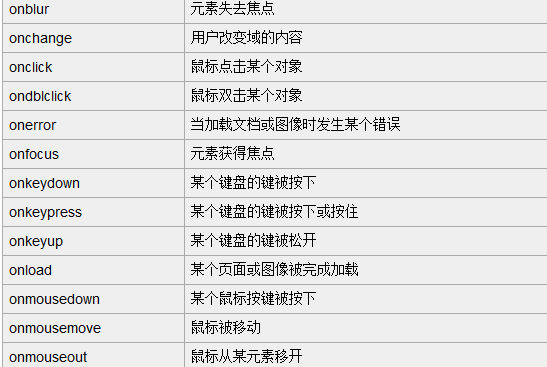
Javascript原生的onload事件只能注册一个函数。

* 思考：什么时候使用$(document).ready()，什么时候使用$(document).load事件？它们什么区别？



**简单事件**

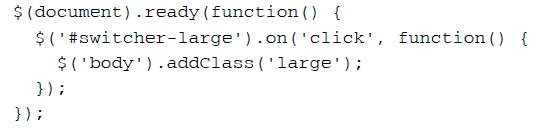
* 正如javascript提供了onload来截获页面加载事件一样，它对用户发起的其它事件也提供了相似的“挂钩”。



* 这些挂钩也会与onload一样具有类似的缺点，为此jQuery也为处理这些事件提供了一种的方式。
* 对匹配的元素进行特定的事件绑定: on()

**例3-1**

* 当单击Large Print按钮时，会运行函数中的代码。

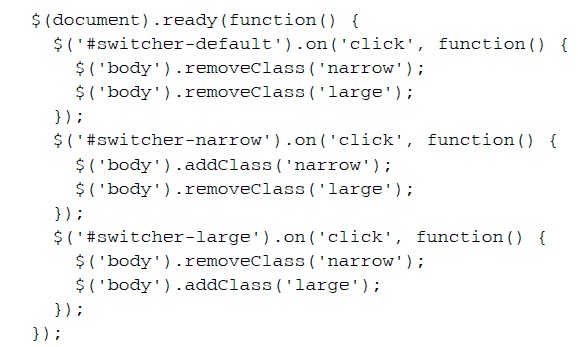


* $(*selector*).on(*event,childSelector,data,function,map*)
* on() 方法在被选元素及子元素上添加一个或多个事件处理程序。
* 自 jQuery 版本 1.7 起，on() 方法是 bind()、live() 和 delegate() 方法的新的替代品。该方法给 API 带来很多便利，我们推荐使用该方法，它简化了 jQuery 代码库。
* **注意：**使用 on() 方法添加的事件处理程序适用于当前及未来的元素（比如由脚本创建的新元素）。
* **提示：**如需移除事件处理程序，请使用 off() 方法。
* **提示：**如需添加只运行一次的事件然后移除，请使用 one() 方法。



**完善代码**

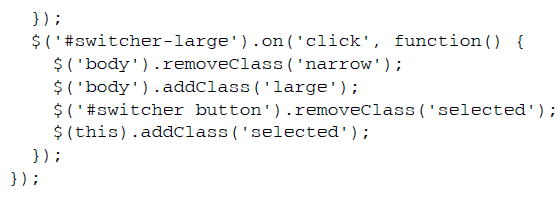
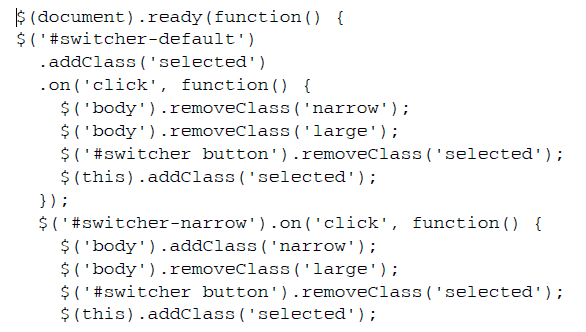
**例3-2完成其它2个按钮**



**this关键字的使用**

* 当触发事件处理程序时，关键字this引用的都是携带相应行为的DOM元素(处理事件的元素)。$()可以将DOM元素作为参数，所以事件处理程序中使用$(this)，可以为相应的元素创建jQuery对象，然后对它 进行操作。

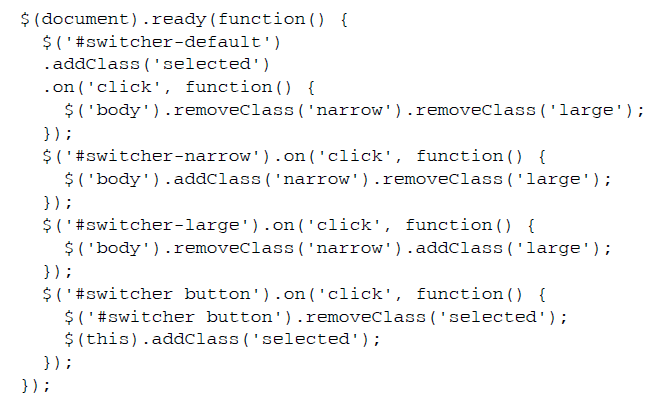
**例3-3添加单击按钮加粗样式**



**代码重构**

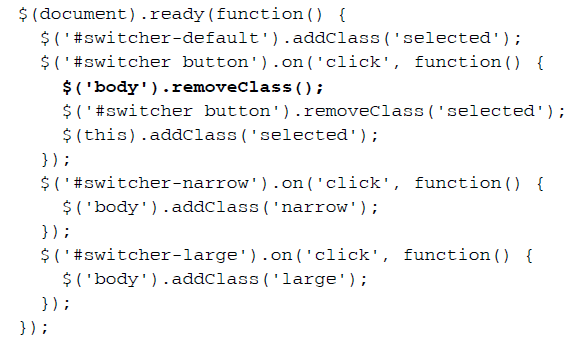
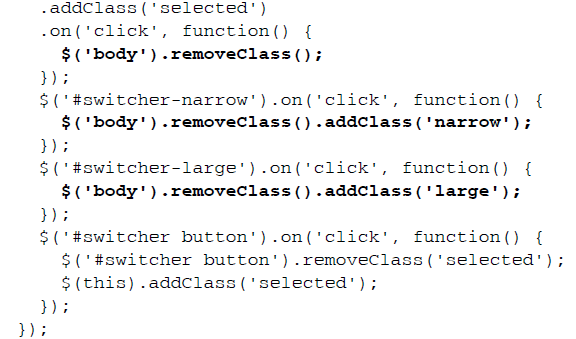
* 使用行为队列机制重构代码

**例3-4**



* 移除.removeClass()函数的参数，重构代码

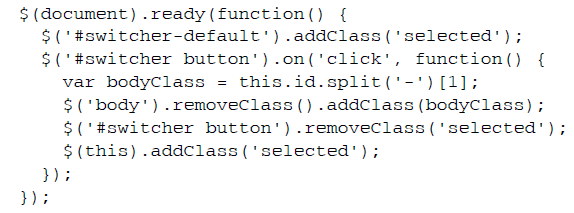
**例3-5**



**例3-6**，必须把通用的处理程序转移到特殊的处理程序上方，因为.removeClass()需要先于.addClass()执行。而之所以能够做到这一点，是因为jQuery总是按照我们注册的顺序来触发事件处理程序。

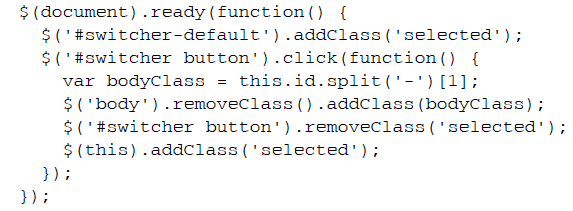
**进一步合并**

* 例3-7



**简写的事件**

* 鉴于绑定处理程序极为常用，jQuery提供了一种简化事件操作的方式使用.click()代替.on()
* 例3-8



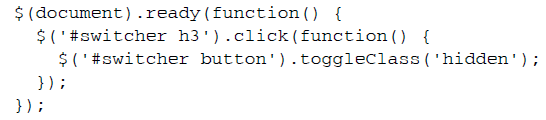
* jQuery为标准的DOM事件都提供了相应的简写事件方法

**复合事件**

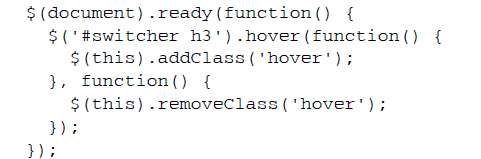
* 复合事件可以截取组合的用户操作，并且以多个函数作为响应，因此被称为复合事件处理程序。
* toggleClass(): ，能够根据相应的类是否存在而添加或删除类.
* hover(): 模拟光标悬停时间. 当光标移动到元素上时, 会触发指定的第一个函数, 当光标移出这个元素时, 会触发指定的第二个函数.

**例**

* 例3-9使用toggleClass应用和移除类



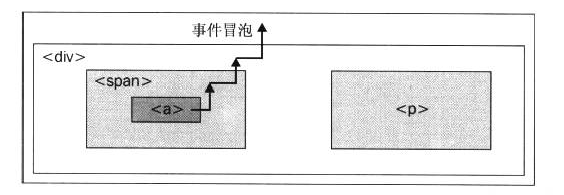
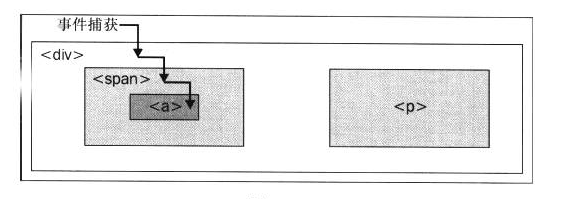
* 例3-10使用hover实现鼠标悬停效果



**事件的传播**

**事件捕获**

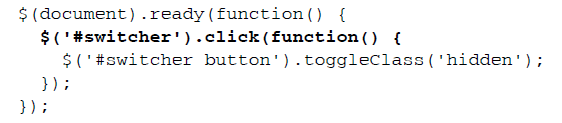
* 在事件捕获的过程中，事件首先会交给最外层的元素，接着再交给更具体的元素。
* 事件冒泡
* 当事件发生时，首先发送给最具体的元素， 在这个元素获得响应机会之后，事件会向上冒泡到更一般的元素。



* jQuery在事件**冒泡阶段**注册事件处理程序。

**事件冒泡的副作用**

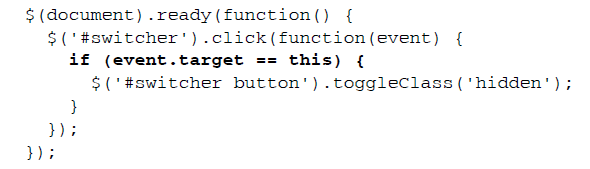
* 例3-11 增加折叠或扩展的可单击区域到div标签



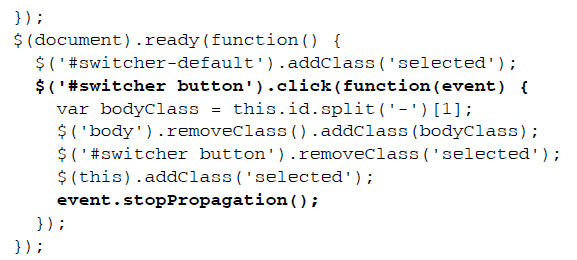
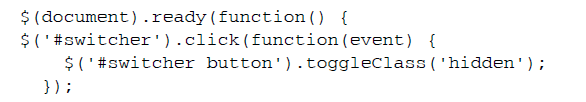
* 我们可以看到事件冒泡的副作用，点击按钮时也会折叠div标签

**防止事件冒泡的副作用**

* 要解决这个问题，必须访问事件对象。事件对象是一种DOM结构，它会在元素获得处理事件的机会时传递给被调用的事件处理程序。
* 事件目标
* target属性可以确定DOM中首先接收到事件的元素。



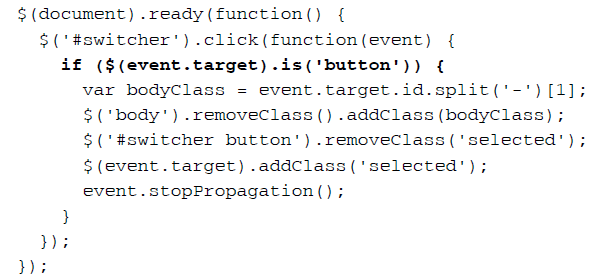
* 例3-13
* 停止事件传播 。 .stopPropagation()方法可以完全阻止事件冒泡。

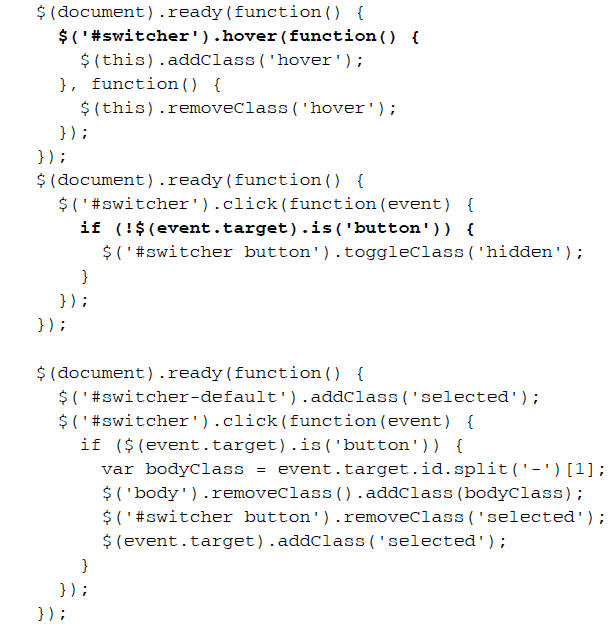


**默认操作**

* .preventDefault()可以在触发默认操作之前终止事件。
* 可以在事件处理程序中返回false同时停止事件传播和默认操作。

**事件委托**

* 事件冒泡并不是总带来问题，也可以利用它为我们带来好处
* 我们可以只在DOM中的一个祖先元素上指定一个单击处理程序。由于事件会冒泡，未拦截的单击事件最终会到达这个祖先元素，而我们可以在此时作出相应处理。
* 这里使用了一个新方法，即.is()。 is() 根据选择器、元素或 jQuery 对象来检测匹配元素集合，如果这些元素中至少有一个元素匹配给定的参数，则返回 true。
* 例3-14
* 进一步改进代码。消除点击按钮时，转换器折叠问题。
* 例3-15

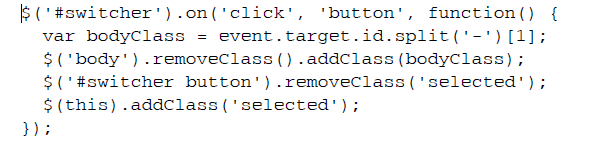


* 通过组合两个click事件处理程序并使用基于.is()测试

的if-else语句，可以减少重复的代码，例3-16

**使用内置的事件委托功能**

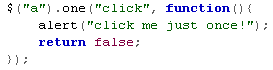
* 例3-17，.on()方法可以接受相应参数实现事件委托



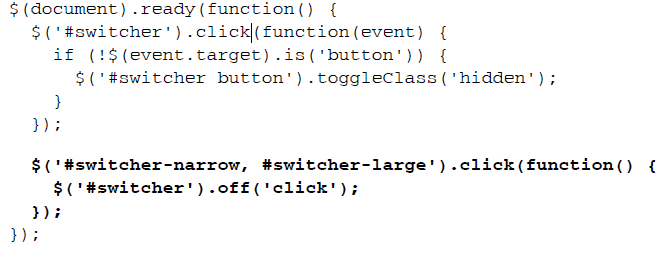
* 如果给.on()方法传入的第二个参数是一个选择符表达式，jQuery会把click事件处理程序绑定到#switcher对象，同时比较event.target和选择符表达式（这里的'button'）。如果匹配，jQuery会把this关键字映射到匹配的元素，否则不会执行事件处理程序。

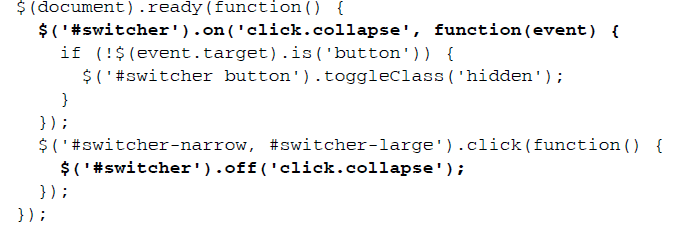
**移除事件处理程序**

* 移除某按钮上的所有 click 事件: $(“btn”). off(“click”)
* 移除某按钮上的所有事件: $(“btn”). off();
* one(): 该方法可以为元素绑定处理函数. 当处理函数触发一次后, 立即被删除. 即在每个对象上, 事件处理函数只会被执行一次.



**代码清单3-18**

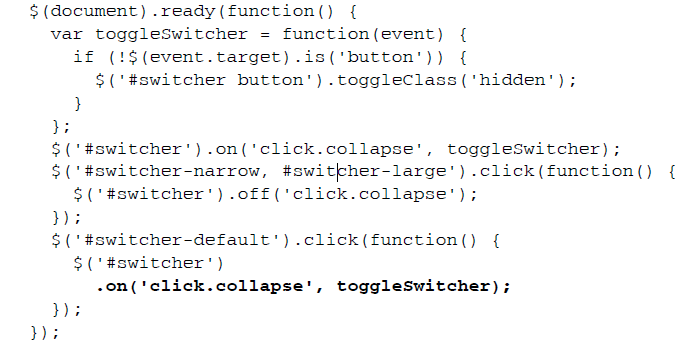
* 当NarrowColumn或 Large Print按钮被选中时，单击样式转换器的背景区域不应该引发任何操作。

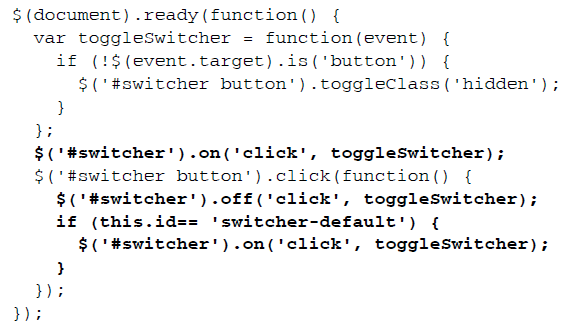
**为事件处理程序添加命名空间**

* 为事件处理程序添加命名空间
* 在调用$('#switcher').off('click')时，会导致按钮上绑定的两个事件处理程序都被移除。
* 通过附加的命名空间信息，则可以解除对这个特定处理程序的绑定，同时不影响为按钮注册的其他单击处理程序。

**重新绑定事件**

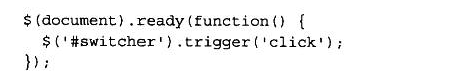
* 单击Default按钮时恢复转换器的折叠功能



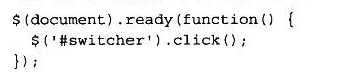
* 当在jQuery中把处理程序绑定到事件时，之前绑定的处理程序仍然有效。在这个例子中，在用户单击Narrow或LargePrint之前每多单击一次都会多调用一次这个函数。
* 为了解决这个问题，可以在用户单击任意按钮时解除绑定，并在确定单击按钮的ID是switcher-default的情况下再重新绑定
* 代码清单3-22 

**模仿用户操作**

* 有时候，即使某个事件没有真正发生，如果能执行绑定到该事件的代码将会很方便。
* 通过.trigger()方法就可以完成模拟事件的操作。

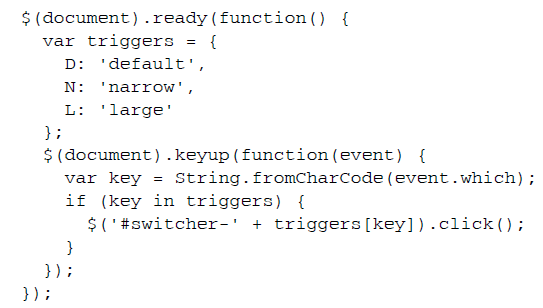


* .trigger()方法提供了一组与.on()方法相同的简写方法。当使用这些方法而不带参数时，结果将是触发操作而不是绑定行为。



**响应键盘事件**

* 直接对键盘按键给出响应的事件（keyup和keydown）和对文本输入给出响应的事件（keypress）， 如果想知道用户按了哪个键，应该侦听keyup或keydown事件；如果想知道用户输入的是什么字符，应该侦听keypress事件。
* ，需要在keyup处理程序被触发时知道用户按下了哪个键。此时可以检查相应的事件对象，事件对象的.which属性包含着被按下的那个键的标识符，对于字母键而言，这个标识符就是相应大写字母的ASCII值
* ，按下D,N,L这三个键中的任何一个，都会模拟鼠标对相应按钮的单击，代码清单3-25



* 代码的进一步整合。
* 代码清单3-26