# 1第一章jQuery简介

框架

Js

库

Jquery-------pc端

Zepto-------移动端

## jQuery的优势：

轻量级、强大的选择器、出色的DOM封装、可靠的事件处理机制、完善的Ajax、不污染顶级变量、出色浏览器兼容性、键式操作方法、隐式迭代、丰富的插件、完善的文档、开源

$(document).ready(function(){ $(“div”).addClass(“aa”) })

$(function(){ }) 代表文档建在完成之后，所有代码写在里面

## 引入jQuery文件：

jQuery最库右两种类型，完整版和压缩版，前者用与测试开发，后者主要用于项目应用，目前最常用的版本为1.8.3，最新版本3.4.1，从jQuery2.0开始jQuery不再支持ie6/7/8

jQuery库的引入方式和我们引入js文件是一样的：

<script src=”jquery-1.8.3.min.js” type=”text/javascript”></script>

jQuery编写，和js编写方式一样，在html页面中<script></script>中书写，或采用独立js文件的方式。

注意：自己编写的jQuery代码必须要放到引入jQuery库文件的后面。

## Js和jQuery之间的转化(区别)：

Js:document.getElementsByTagName(“div”)

Jquery:&(“div”)

# 第二章选择器

用来获取BOM结构的元素的方式。使用jQuery选择器目的是为了生成jQuery对象，然后对其进行操作。

jQuery选择器继承了css选择器的风格，支持css1到css3几乎所有选择器，并且不需要考虑兼容性

## 基本选择器

--基本选择器式jQuery中最常用的选择器，也是最简单的选择器

--它通过元素id、class和标签等来查找DOM元素。

--在网页中，每个id名称只能使用一次，class允许重复使用

|  |  |  |  |
| --- | --- | --- | --- |
| 选择器 | 描述 | 返回 | 示例 |
| #id | 根据给定的id匹配一个元素 | 单个元素 | $(“#test”)选取id为test的元素 |
| .class | 根据给定的类别名匹配元素 | 集合元素 | $(“.test”)选区class为test的元素 |
| Element | 根据给定的元素名匹配元素 | 集合元素 | $(“p”)选取所有<p>元素 |
| \* | 匹配所有元素 | 集合元素 | $(“\*”)选取所有元素 |
| Selector1,selector2,… | 将每一个选择器匹配到的元素合并后一起返回 | 集合元素 | $(“#div.span.p.myClass”)选取所有的<div>,<span>和拥有class为myclass的<p>标签的一组元素 |

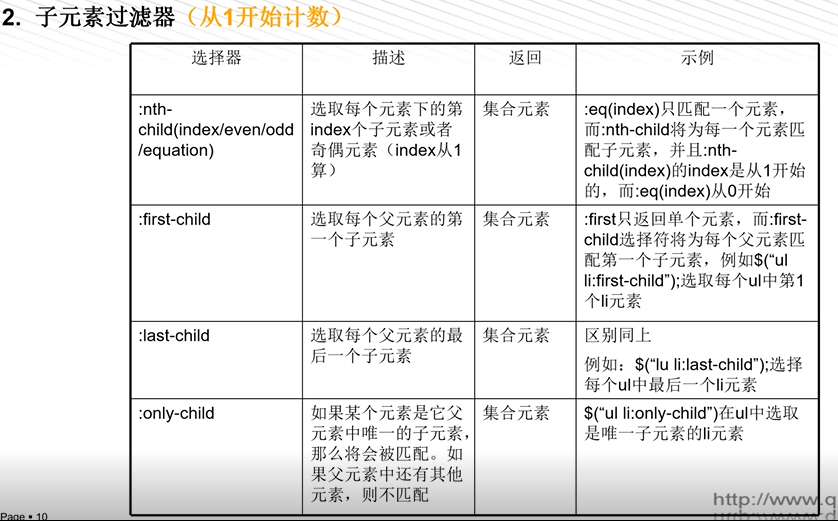
## 基本过滤选择器：

--过滤器主要是通过特定的过滤规则来筛选出所需要的DOM元素，过滤规则与CSS的伪类选择器语法相同，即选择器都以一个冒号：开头。

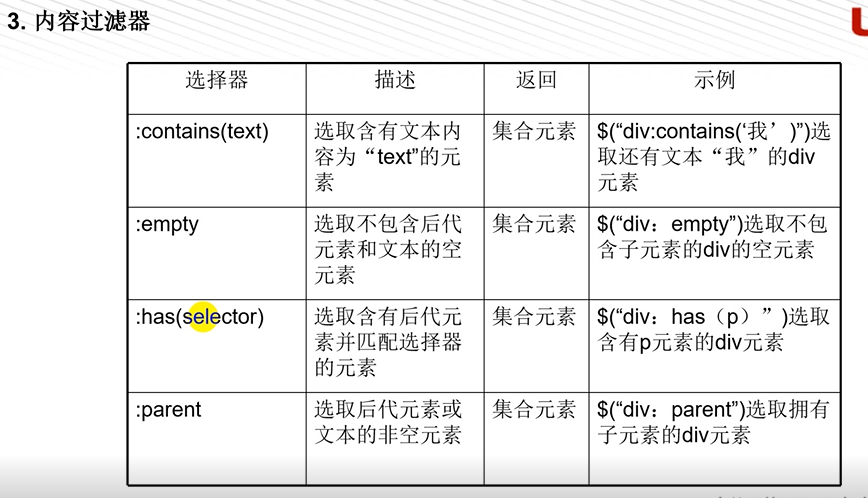
--过滤器可以分为基本过滤器、内容过滤器、可见性过滤器、属性过滤器、子元素过滤器



## 子元素过滤器：



## 内容过滤器：



基本过滤器：从0开始

子元素过滤器：从1开始

## 可见性过滤器：

|  |  |  |  |
| --- | --- | --- | --- |
| 选择器 | 描述 | 返回 | 实例 |
| ：hidden | 选取所有不可见元素 | 集合元素 | $(“:hidden”)选区所有不可见的元素。包括<input type=hidden>,<div style=”display:none”>如果只想选取<input>元素可以使用$(“input:hidden”) |
| ：visible | 选取所有可见元素 | 集合元素 | $(“div:visible”)选取所有可见元素 |

## 属性过滤器：

|  |  |  |  |
| --- | --- | --- | --- |
| [attribute] | 选取拥有此属性的元素 | 集合元素 | $(“div[id]”)选择拥有属性id的元素 |
| [attribute=value] | 选取属性的值为value | 集合元素 | $(“div[title=test]”)选取属性title为“test”的<div>元素 |
| [attribute!=value] | 选取属性的值不等于value的元素 | 集合元素 | $(“div[title!=test]”)选取属性title不等于“test”的div元素（没有属性title的div也会选取） |
| [attribute^=value] | 选取属性的值为value开始的元素 | 集合元素 | $(“div[title^=test]”)选取属性title以“test”开始的div元素 |
| [attribute$=value] | 选取属性的值以value结束的元素 | 集合元素 | $(“div[title$=test]”)选取属性title含有“test”结束的div元素 |
| [attribute\*=value] | 选取属性的值含有value的元素 | 集合元素 | $(“div[title\*=test]”)选取属性title含有“test”的div元素 |
| [attribute~=value] | 选取多个属性的值中包含value值的元素 | 集合元素 | $(“div[class~=test]”)选取属性class含有“test“的div元素 |

## 层次选择器：

可以通过dom元素之间的层次关系来获得特定元素

|  |  |  |  |
| --- | --- | --- | --- |
| $(“选择器1 选择器2”) | 从匹配选择器1的后代元素中选取匹配选择器2的元素 | 集合元素 | $(“div span”)选取div里所有span元素 |
| $(“选择器1>选择器2”) | 从匹配选择器1的子元素中选取匹配选择器2的元素 | 集合元素 | $(“div>span”)选取div元素下元素名称为span的子元素 |
| $(“选择器1+选择器2”) | 从匹配选择器1后面的第一个兄弟元素中选取匹配选择器2的元素 | 集合元素 | $(“.one+div”)选取class为one的下一个div兄弟元素 |
| $(“选择器1~选择器2”) | 从匹配选择器1后面的所有兄弟元素中选取匹配选择器2的元素 | 集合元素 | $(“#two~div”)选取id为two的元素后面的所有div元素 |

## 表单对象元素过滤器：

|  |  |  |  |
| --- | --- | --- | --- |
| :enabled | 选取所有可用元素 | 集合元素 | $(“#form1:enabled”);选取id为“form1”的表单内的所有可用元素 |
| :disabled | 选取所有不可用元素 | 集合元素 | $(“#form2:disabled”)选取id为“form2”的表单内的所有不可用元素 |
| :checked | 选取所有被选中的元素（单选框、复选框） | 集合元素 | $(“input:checked”);选取所有被选中的input元素 |
| :selected | 选取所有被选取中的选项元素（下拉列表） | 集合元素 | $(“select:selected”)选取所有被选中的选项元素 |

一般情况下操作checked的元素没有效果，我们可以操作后面的紧跟着的兄弟

表单选择器：

|  |  |  |  |
| --- | --- | --- | --- |
| :input | 选取所有的input、textarea、select和button | 集合元素 | $(“:input”)选取所有input textarea、select和button元素 |
| :text | 选取所有的单行文本框 | 集合元素 | $(“:text”)选取所有的单行文本框 |
| :password | 选取所有的密码框 | 集合元素 | $(“:password”)选取所有的密码框 |
| :radio | 选取所有的单选框 | 集合元素 | $(“radio”)选取所有的单选框 |
| :checkbox | 选取所有的复选框 | 集合元素 | $(“checkbox”)选取所有的复选框 |
| :submit | 选取所有的提交按钮 | 集合元素 | $(“submit”)选取所有的提交按钮 |
| :image | 选取所有的图像按钮 | 集合元素 | $(“image”)选取所有的图像按钮 |
| :reset | 选取所有的重置按钮 | 集合元素 | $(“reset”)选取所有的重置按钮 |
| :button | 选取所有的按钮 | 集合元素 | $(“button”)选取所有的按钮 |
| :file | 选取所有的上传文件 | 集合元素 | $(“file”)选取所有的上传文件 |
| :hidden | 选取所有不可见元素 | 集合元素 | $(“:hidden”)选取所有不可见元素（已经在不可见性过滤选择器中讲过） |

# 第三章更改jQuery对象

## 更改操作概念

（1）更改对象操作

-jQuery所做的事情就是生成jQuery对象，操作jQuery对象：像JavaScript一样如果对象之间存在某种关系我们就可以通过这种关联来改变对象，例如通过父元素找到子元素

（2）链式操作设定

-生成一个jQuery对象后，既要对它进行一次或多次的普通操作，同时还要对他进行更改操作，于是有必要把生成的jQuery对象存储在一个变量里，满足多次调用的需要。

-如果按照这种方式，每进行一次对象生成或更改操作，都需要引入一个变量存储新的jQuery对象，使高效的程序变得不够简洁，于是jQuery引入了链式操作：执行普通操作后，会返回对象本身，于是可以持续地对该对象执行下一次普通操作，指导改变jQuery对象时，执行一个更改操作。

## 给改为后代元素集合

1. Children()方法 选择子元素

children(“选择器”)选择子元素中，所有匹配选择器地子元素

children()/children(“\*”)选择父元素地所有直接子元素

1. Find()方法 选择后代元素中，所有匹配选择器地后代元素

Find(“选择器”)选择后代元素中，所有匹配选择器地后代元素

Find()/find(“not(\*)”)不选择原先元素人后后代元素

1. Contents()方法 选择文本块和子元素

## 更改为祖先元素：

1. parent(“选择器”)选择父元素中，所有匹配选择器的元素

parent()选择所有父元素

1. parents()方法 选择祖先元素

parents(“选择器”)选择祖先元素中，所有匹配选择器的元素

parents()选择所有祖先元素

1. parentsUntil()方法：

parentsUntil(“选择器”)选择祖先元素，直到找到匹配选择器的元素，但不包括匹配元素

parentsUntil()选择所有祖先元素，

1. offsetParent()方法：

选择最近的祖先定位元素，且该元素的css属性display不能为none，所谓定位元素是指其css属性position取值为relative、absolute或fixed。如果祖先元素中没有合适的元素，则会选择<body>元素

1. closest()方法
2. 选择本身或其祖先元素所匹配选择器地元素，也就是所说先看该元素本身是否匹配，如果不匹配会一层一层向上找到最先匹配地祖先元素

## 更改为兄弟元素集合：

1. Next()和pre()方法

Next(“选择器”)在原先元素后面的第一个兄弟元素中，选择匹配选择器的元素（仅匹配后面的第一个兄弟）；

Next()选择原先元素后面的第一个兄弟元素

Prev(“选择器”)在原地元素前面的第一个兄弟元素中，选择匹配选择器的元素（仅匹配前面的第一个兄弟）；

Prev()选择原先元素前面的第一个兄弟元素

1. nextAll()、prevAll()和siblings()方法

nextAll(“选择器”)在原先元素后面的所有兄弟元素中，匹配选择器的元素

nextAll()选取原先元素后面的所有兄弟

prevAll(“选择器”)在原先元素前面的所有兄弟元素中，匹配选择器的元素

prevAll()选取原先元素前面的所有兄弟

siblings(“选择器”) 选择你原先元素所用兄弟元素中，匹配选择器的元素

siblings()选择所有兄弟元素

## 更改为更多元素集合

* 1. add()方法

add(“选择器”)在原先元素的基础上添加选取匹配选择器的元素（括号里是加上它都有效果）

举例：$(“选择器1”)add(“选择器2”)等同于$(“选择器1”，“选择器2”)

2> andself()方法

在更改对象后，将原对象添加到选择的元素集合（找到的加先前的）

## 更改为部分元素集合

1. eq()、first()和last()方法  
   eq(index)选择元素中索引值等于index的元素，正向从0开始，逆向从-1开始-2-3等

first()选择匹配元素中第一个，等于eq（0）

last()选择匹配元素中最后一个，等于eq(-1);

1. slice()方法

slice(start,[end])方法：筛选索引从start到（end-1）的元素（不包含结束位置的元素）。

例如：slice(2,5);指得是从索引值为2到5之前，也就是2.3.4元素

1. filter()和not()方法

filter(“选择器”)在原先元素中选择匹配选择器的元素（包含那个选择器的都有效果）

例如：<div class=”clear top”></div> <div class=”last top”></div>

Filter(“top”),上面两个div都会有效果

$("p").filter(function(index){return index%2==0}).addClass("hilight)

所有为偶数的p有效果，里面可以是一个方法。Index就是索引值，里面的把索引值返回给function后面的index

not(“选择器”)在原先元素中选择不匹配选择器的元素

has(“选择器”)在原先元素中筛选出拥有匹配选择器后代元素的元素

例子：$("div").has("p").addClass("hilight")

有p的div全都有效果

## 还原jQuery对象的方法

执行更改jQuery对象操作后，选取的元素会发生改变，如果希望选取的元素还原到改变之前，可以使用end()方法，如果希望还原多个更改操作，可以连续使用多次end()方法，直到最后会返回一个空集

链式操作：用一行代码完成操作

# 第四章jQuery对象的文档处理

## 移动元素：

1.

--Append()：向每个匹配的元素内部尾随追加内容。例如：$(“选择器1”).append(“选择器2”)；会将匹配选择器2的元素，移动到匹配选择器1的内部尾部。

例子：$(“div”).append($(“p”)).addClass(“highlight”);

--appendTo():将所有匹配的元素追加到指定的元素中

--appendTo()和append方法颠倒过来

AppendTo是把想追加的东西放到前面，append是把想追加到的地方放到前面

--例子：$(“p”).appendTo($(“div”)).addClass(“highlight”);//功能同上

2.

--prepend():向每个匹配的元素内部前置内容

--prependTo():将所有匹配的元素前置到指定的元素中。

3.

--after():在每个匹配的元素之后插入内容

--insertAfter():将所有匹配的元素插入到指定元素的后面

4.

--before():在每个匹配的元素之前插入内容

--insertBefore():将所有匹配的元素插入到指定的元素的前面

## 添加元素：

概述：4.1节方法不仅能够把页面元素移动到指定位置，还能够把新添加的元素移动到指定位置。添加的新元素需要额外生成

1. 创建生成jQuery对象

例如：$(“<p>p1</p>”).appendTo($(“div”))

1. 复制生成jQuery对象：

Clone()方法生成被选元素的副本，包含子节点、文本和属性

参数：true可选。规定是否复制元素的所有事件处理。默认地，副本中不包含事件处理器。

## 替换元素：

ReplaceWitch()方法

移动页面上原有的元素（创建新的jQuery对象）来替换当前选定地元素

举例：$(“p”).replaceWith($(“.last”))

说明：删除所有地p元素，同时把类名为last的元素移动到最后一个匹配p元素的位置

ReplaceAll()方法

$(selector1).raplaceAll(selector2)

删除所有匹配selector2的元素，并将匹配selector1的元素移动到每个匹配selector2的位置

// replaceAll\replaceWith的用法

// replaceAll:用前边的替换后边的每一个，每一个都会被替换掉

// replaceWith：用后边的替换前边所有的，并且放到最后一个替换元素的位置

## 包裹元素：

包裹元素，就是个给某个节点添加父元素

1. Wrap()方法
   1. 复制页面上原有的元素来包裹当前选定的页面元素
   2. $(“p”).wrap($(“.last”)), 复制类别名为last的元素来包裹p元素
   3. $(“p”).wrap(“<div></div>”), 创建新的jQuery对象，生成div标记来包裹p标记。
2. Unwrap()方法
   1. $(selector1).unwrap()
   2. Unwrap()方法与wrap()方法的作用正好相反，用来去除当前元素的父元素
3. wrapAll()方法
   1. wrapAll()方法不同于wrap()方法，会把所有元素包裹到一起
   2. $(selector1).wrapAll(selectoe2)
   3. 将匹配selector2的第一个元素进行复制，用来包裹被移动到一起的匹配selector1的全部元素
4. wrapInner()方法
   1. wrapInner()方法不同于wrap()方法，会分别包裹每个元素内部的文本和后代元素

## 删除和清空元素

Remove()方法可以删除当前元素，该元素所包含的文本内容和后代元素同时被删除

举例：$(“div>p:first”).remove();删除div的子元素第一个p元素

Detach()方法

与remove()一样可以删除当前元素，但是使用链式操作可以将删除的元素移动到其他位置，原先绑定在remove该元素上的事件会保留

Empty()方法

清空当前元素，该元素的文本内容以及所有的后代元素都会被删除。

举例：$(“div”).empty()清空div中所有的内容。

# 第五章jQuery对象的属性处理

## 元素的属性处理：

Attr()方法：获取元素的某个属性值，或者传递一个参数来设置该属性的值

用法:$(seletor).attr(“width”,”400”);

removeAttr()方法：删除某个属性

用法：$(“img”).removeAttr(“title”);

元素class属性处理

Class属性的一般处理：class也是元素的一个属性，因此完全可以使用attr()和removeattr（）方法对元素的class属性的值进行获取、设置、删除等处理

例1：$(“img”).attr(“class”);//获取第一个<img>元素的class值

例2：$(“img”).attr(“class”,”inner”);//设置所有<img>元素的class值为inner

例3：$(“img”).removeAttr(“class”);//删 所有<img>元素的class属性值

1. addClass()方法：

attr()设置属性值，先清空其他属性值，而addClass()方法可以再现有class追加另外的值。

例如：$(“img:eq(1)”).addClass(“test”);//给第2个<img>元素，追加一个属性值test.

1. removeClass()方法
   1. 元素溢出一个或多个属性值。例如$(“选择器”).removeClass(value);会从匹配元素的class属性中溢出指定得值。
      1. 例如：$(“img:eq(1)”).removeClass(“test”);//移除 第2个<img>元素上的test值。
2. toggleClass();方法：切换class属性中的一个或多个属性值的切换。例如:$(“选择器”).toggleClass(value)会切换所有匹配选择器的元素的class属性值，如果属性值存在则删除它，如果属性值不存在，则添加它。
   1. 例如：$(“img”).toggleClass(“inner”);
3. hasClass();方法：判断匹配元素class属性中是否含有某个值，有返回true，没有返回false;
   1. 例如：$(“img”).hasClass(“inner”);

## 元素内部的HTML、文本处理

Html()方法：使用html()方法获取或设置内部的html代码，包括文本内容以及html标记

Text()方法：可以用来读取或设置某个元素中的文本内容

Val()方法：获取或设置匹配选择器的表单元素的value属性值

例如：$(“input").val();//获得第一个<input>元素的value属性的值

例如：$(“input”).val(“设定值1”);//设置所有input元素的value值为选定值1

表单选项的checked、selected属性

使用val()方法来设置checked和selected的默认值。

$(“input:checkbox”).val([“html”]);

# 第六章 jQuery对象的css处理

## Css基本属性的处理

1. css():方法：获取css属性值。$(“选择器”).css(name);获取匹配选择器的元素指定css属性的值

例1：$(“p”).css(“color”);//获取第一个<p>元素的css属性的color值。

说明：css()方法允许获得，合并书写或简写的属性的值，例如background、margin、border等，如果要获得margin必须分别写margin-top、margin-right等。

例2：$(“p”).css(“background-color”);

$(“p”).css(“backgroundColor”);

1. css():方法：设置css属性值。$(“选择器”).css(name,value);设置匹配选择器的元素指定css属性的值。

例1：$(“p”).css(“color”,”green”);//设置所有<p>元素的css属性的color值。

说明：css()方法允许获得，合并书写或简写的属性的值，但是支持设置合并书写或简写的属性值。例如background、margin、border等，

例2：$(“p”).css(“border”,”1px solid white”);

说明：也可以一次设置多属性

例3：$(“p”).css({“background-color”:”green”,”color”:”white”})

## Css尺寸属性处理

1. height()和width()方法：如果要获取或设置css属性height、width的值，可以使用更为简洁的height()、width()方法。单位是像素。

说明：需要注意的是，height()、width()方法获取的都是元素在页面中的实际高度值和宽度值，与设置的不一定相同。

例1：$(“div”).width();//获取第一个div的width值

例2：$(“div”).width(30);//设置所有div的width为30px

例3：$(“div”).width(“70%”);//设置所有div的width为父元素的70%

1. innerHeight()和innerWidth()方法，用户获得或设置元素的实际高度和宽度值(获取的是padding+width的值,只能获取，不能设置)

例1：$(“div”).innerWidth();//获得实际宽度，包括padding+width不包括border

1. outerHeight()和outerWidth()方法。用户获得或设置元素的实际高度和宽度指。想包含margin必须给后面加true（不能设置，只能获取）

例1：$(“div”).outerWidth(true);//获得实际宽度，包括padding+border+width+margin

## Css位置属性处理

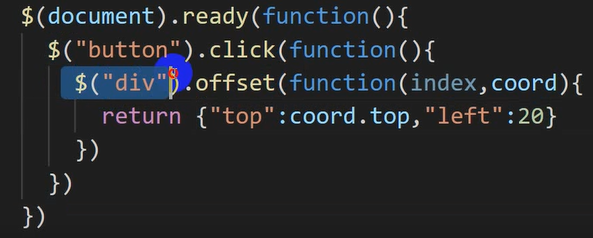
1. offset()方法:获取设置当前元素的相对窗口偏移量。

例如:$(“p”).offset().top;//获得第一个p距窗口上边的便宜距离。

$(“p”).offset().left;//获得第一个p距窗口左边的便宜距离。

例如:$(“p”).offset({“top”:40,”left”:15});

只设置left值，top不动。Index不用也必须加上，index代表索引值。



1. position()方法:获取当前元素相对最近的定位元素的偏移量。(父元素定位了获取的是距父元素的，父元素没定位到body的距离)

例如:$(“p”).position().top;//获得第一个p距父元素上边的偏移距离。

$(“p”).position().left//获得第一个p距父元素左边的偏移距离。

1. scrollTop和scrollLeft()方法：获取或设置滚动条的位置。

例如：（“div:eq(1)“）.scrollTop();//获得滚动条的垂直距离

例如：(“div:eq(1)”).scrollTop(37);//

# 第7章jQuery动画

## 显示与隐藏动画效果：

1. hide():方法：动态改变当前匹配元素的高度、宽度和不透明度，最终隐藏当前元素，此时元素的属性display修改为none.（宽、高、不透明度同时改变）

语法：hide(动画运行时间)；slow是0.6秒、normal是0.4秒和first是0.2秒

2. show():方法：动态改变当前匹配元素的高度、宽度和不透明度，最终显示当前元素，此时元素的属性display恢复为none以外的值（宽、高、不透明度同时改变）

3. toggle()：方法：会动态的改变当前元素的高度、宽度和不透明度，最终切换当前元素的可见状态，也就是说，如果元素是可见的，则被切换成隐藏状态：如果元素是隐藏的，则被切换为可见状态。（宽、高、不透明度同时改变）

## 淡入与淡出动画效果：

1. fadeOut():方法：动态的改变当前元素的不透明度（其他的不变），最终隐藏当前元素。此时元素的css属性修改为none,它的语法结构hide()和show()相同。（只改变不透明度）
2. fadeIn():方法：动态改变当前元素的不透明度(其他的不变)，最终显示当前元素。此时元素的css属性修改为none以外的值。
3. fadeToggle():方法：会动态的改变当前元素不透明度（其他不变），最终切换当前元素的可见状态。也就是说，如果元素是可见的，则通过淡出鲜果切换成隐藏状态；如果元素是隐藏的，则通过淡入此效果则切换为可见状态。
4. fadeTo():方法：会把当前元素动态的调整到指定不透明度（其他不变）。

语法：fadeTo(动画时间，不透明度)，不透明度去0-1之间的值，0表示透明，1表示不透明。

## 滑入与滑出效果：

1. slideUp():方法：动态的改变当前元素的高度（其他的不变），最终隐藏当前元素。此时元素的css属性修改为none.它的语法结构同hide()和show相同。
2. slideDown():方法：动态的改变当前元素的高度（其他的不变），最终显示当前元素。此时元素的css属性修改为none.
3. SlideToggle():方法：会动态的改变当前元素高度（其他不变），最终切换当前元素的可见状态。也就是说，如果元素是可见的，则通过滑出效果切换成隐藏状态；如果元素是隐藏的，则通过滑入此效果切换为可见状态。

## 自定义动画效果：

1. Animate():方法：可以动态改变当前元素的各种css属性。

语法：

Animate({“样式名”:”值”},时间)；

说明:animate():方法：只能修改可以取数字值的css属性。例如width、height、border-width、padding等等。

动态改变元素的css属性left、top、bottom、right，可以使元素运动到不同的位置，不过必须确保元素的css属性position已经被设置“relative“、”absolute“或者”fixed”。

增量

$(“p”).animate({“left”,”+=300px”},400)://p换元素在0.4面内向右移动，使css属性的left值在原来的基础上增加300px.

## 动画队列：

上面例子里面的多个动画是同时发生的，如果想要按照顺序执行每个动画效果，例如：让p元素先向右滑动，然后再放大他的高度，只需要使用多次animate（）方法，每个animate（）方法执行一个动画效果就可以了。

例1：$(“p”).animate({“left”:”300px”}).animate({“height”:”100px”});

总之，在css属性left改变之前，css属性height将不会改变。向这样，动画效果执行具有先后顺序，称之为动画序列。

## 延迟动画队列：

延迟动画对垒概述：如果需要延迟执行当前元素的某个动画效果。可以使用delay（）方法。

语法：delay（动画时间）；动画时间可以使用slow、normal、fast，也可以是毫秒。

## 中止动画队列：

终止动画队列概述：如果需要在某时刻停止当前元素正在进行的动画效果，可以使用stop（）;效果。

语法：stop(是否要清空未执行完的动画队列，是否直接将正在执行的动画跳转到末状态)；两个参数均是布尔值true和false；参数也可以省略，将立即停止动画。

例如：原始动画。

$(“p”).animate({“left”:”+=400px”},2000)

.animate({“top”:”+=200px”},2000)

.animate({“left”:”-=400px”},2000)

.animate({“top”:”-=200px”},2000);

例如1：在p元素从状态1右移变到状态22的过程中，单机ID为btn1的元素。使用stop（）方法终止动画1，从终止位置开始。P元素会继续执行动画2、动画3、动画4、最终会到状态1的左边。

$(“btn1”).click(function(){

$(“p”).stop();

})

终止动画队列（续）

Stop()方法来中止正在运行的动画队列，stop()方法有两个可选参数，不同参数的设置代表了的中止形式

Stop(stopAll,gotoEnd);

stopAll()规定是否停止被选元素的所有加入队列的动画

gotoEnd规定是否允许完成当前的动画

1. stop()无参数，从中止位置开始继续执行剩下的动画。
2. stop(true)只有一个参数true，立即停止所有动画，并跳转到当前动画的结束状态。
3. stop(true,true)立即停止所有动画，并跳转到当前动画的结束状态。
4. stop(false,true)立即停止当前正在进行的动画，并跳转到该动画的结束状态，接着执行后面的动画。

## 循环动画：

如果希望循环执行一系列动画，可以建立一个包含动画队列的函数，然后在最后一个动画方法回调这个函数，

例如：function loop(){

$(“p”).animate({“left”,”+=400px”},2000)

.animate({“top”,”+=200px”},2000)

.animate({“left”,”-=400px”},2000)

.animate({“top”,”-=200px”},2000,function(){

Loop()

})

}

$(“btn”).click(function(){

Loop()

})

## 动画回调函数：

动画回调函数概述：假设在元素动画结束后，设置它的css属性。如果这是按照常规思路，在动画方法后使用css()方法，并不能得到预期的效果，实际情况是，动画开始之前，css()方法就被执行了。

例如：$(“p”).animate({“width”:”300px”},1000).css(“border-width”,0);

出现这个问题的原因是css()并不会加入到动画队列中，而是立刻执行。可以使用回调函数对非动画方法实现排队。

解决方法：把css()方法写在动画方法的回调函数里即可。

例如：$(“p”).animate({“width”:”300px”},1000,function(){

$(“this”).css(“border-width”,0);

})

如果多组元素上都有动画效果，默认情况下动画是同时发生的，可以使用回调函数改变执行顺序。

例如：$(“p”).animate(“left”:”300px”);

$(“div”).animate(“left”:0);

解决：

$(“p”).animate({“left”:”300px”},function(){

$(“div”).animate({“left”:0});

})

# 第8章jQuery对象的事件对象

## Window.onload跟$(document).ready(function(){})区别：

1. 前者不能简写，后者可以简写成$(function(){})
2. 前者只能执行一次，后者有多少就执行多少次
3. 前者等所有执行完成再加载，后者是相应的doom加载完成再执行

## 页面加载

页面加载的顺序：先加载<head></head>之间的内容，然后加载<body></body>之间的内容

直接在<head></head>之间书写jQuery代码，浏览器会立即执行脚本。但是由于页面元素在<body>部分才出现，此时尚未加载，所以jQuery操作无法应用到页面元素上。

1. example01.html

解决方法：

1. 将代码放到$(function(){

代码；

代码；

})

说明：$(function(){

})

函数等到页面加载完毕之后执行。相当于onload事件

## 绑定事件—bind()方法

1. bind()方法

语法：bind(“事件名称”，处理函数());

常用事件有：click、dblclick、mouseup、mouseup、mousedown、mouseenter、mouseleave、moouseover、mouseout、focus、blur、focusin、keyup、keydown、keypress、change、select、submit、load、unload、resize、scroll、error等。

例1：$(“p”).bind(“click”,function(){

$(this).append(“<span>发生了鼠标单击事件</span>”);

})

例2：为一个元素一次绑定多个事件，多个事件都调用同一个函数

例如：$(“p”).bind(“mouseover mouseout”,function(){

$(this).append(“<span>发生了鼠标划入滑出事件</span>”)

})//当鼠标划入、滑出这个p的时候，会在其内部尾端添加一个span元素。

例3：bind()方法可以对元素多次绑定同一个事件类型，并且每次绑定的事件处理函数不i会互相覆盖，而是会按照属性的顺序依次执行。

例如：$(“p”).bind(“click”,function(){

$(this).append(“<span>1.发生了鼠标单击事件</span>”)；

})

$(“p:first”).bind(“click”,function(){

$(this).append(“<span>2.发生了鼠标单击事件</span>”)；

})//点击第一个p元素时，会在其内部尾端添加两个span，而点击其他p元素时，只会在其内部尾端添加一个span元素。

## 绑定事件—unbind()方法

Unbind()方法，如果需要移除绑定在元素上的所有事件，那么可以使用不带参数的unbind()方法。Unbind(type),指定需要移除的类型。

## 绑定事件—one()方法

One()方法，对于只需要触发一次，随后就要立即移除绑定事件的情况，可以使用更为方便的one()方法。它的语法结构和使用方法与bind()方法一致。

## 绑定事件—on()方法

On()方法，绑定事件，对于使用on进行绑定的事件使用off进行解绑

--绑定一个事件

$(“p”).on(“click”,function(){

$(this).css(“background-color”,”pink”);

});

$(“button”).click(function(){

$(“p”).off(“click”);

})

On()方法，绑定事件，对于使用on进行绑定的事件使用off进行解绑

--绑定多个事件

$(“p”).on(“mouseover mouseout”,function(){

$(“p”).toggleClass(“intro”)；

})

On()方法，绑定事件，对于使用on进行绑定的事件使用off进行解绑

--多个事件绑定不同函数

$(“p”).on({

Mouseover:function(){$(“body”).css(“background-color”,”lightgray”);},

Mouseout:function(){$(“body”).css(“background-color”,”lightblue”);},

Click:function(){$(“body”).css(“background-color”,”yellow”);}

})

On()方法，绑定事件，对于使用on进行绑定的事件使用off进行解绑

--绑定自定义事件

$(“p”).on(“myOwnEvent”,function(event,showName){

$(“this”).text(showName+”!What a beautiful name!”).shou();

});

$(“button”).click(function(){

$(“p”).trigger(“myOwnEvent”,[”Anja”]);

})

On()方法，绑定事件，对于使用on进行绑定的事件使用off进行解绑

--传递数据到函数

$(document).ready(function(){

$(“div”).on(“click”,{msg:”You just clicked me!”},function(event){

Alert(event.data.msg);

})

})

//如果需要绑定事件的时候传数据的话，直接使用的是对象的方式来表示的，{属性：值，属性：值} 这个值：是字符串直接加“”，是数值直接写数值就行

//获取的时候直接e.data.属性名获取即可

//E:代表的是事件对象

//data:代表的是事件对象上边存数据的一个东西

## 委派事件—live()方法

问题1：为所有p元素绑定click事件。当单机某个p元素绑定click事件。当单击某个p元素是，会在内部的后面添加一个span元素。然后再body元素的内部尾端添加一个p元素。

实现：$(“p”).bind(“click”,function(){

$(“this”).append(“<span>发生例点击事件</span>”)；

})；

$(“body”).append(“<p></p>”);

但是，虽然在body后面添加了p元素，但是添加的p元素没有被绑定click事件，所以可以看出，当调用bind()方法时，匹配的元素将被事件处理函数，但是以后添加的匹配元素并不会被附加上该事件处理程序。

## 委派事件—die()方法

Die()方法概述：如果需要移除所有委派在元素上的事件，那么可以使用不带参数的die()方法。

例如：$(“p”).live(“mouseover mouseout”,function(){

$(this).append(“<span>发生了鼠标划入滑出事件</span>”)；

})

$(“body”).append(“<p></p>”);

$(“#btn”).click(function(){

$(“p”).die();

})//移除委派在p元素上的所有事件

## Bind live one on的区别：

Bind:正常绑定，绑定几个执行几个，不能给新添加的元素绑定

Live:会给新添加的元素也绑定上指定的事件

One:只执行一次，执行完成之后自动解除绑定

On:绑定可以同时绑定多个不同事件处理函数的事件（事件名：function(){}）,事件名：function(){}

Bind---unbind解绑

On—off解绑

Live—die解绑

One 不需要解绑

## 触发事件—触发内置事件

1. 触发内置事件，如果需要通过模拟用户操作来触发内置事件，那么可以使用trigger()方法，此时需要传入一个参数即模拟触发的事件类型。

例如：$(“p”).bind(“click”,function(){

$(this).append(“<span>发生了鼠标单击事件</span>”)；

})

模拟触发：

$(“p”).triggle(“click”);//模拟用户触发绑定到p元素上面的单击事件

## 触发事件—触发命名空间上的事件

可以为元素绑定相同的事件类型，然后根据命名空间的不同，使用trigger()方法按需要触发。其中一个特殊的方法是在trigger()方法中的事件类型参数后添加一个感叹号，用来匹配不包含在任意命名空间中的该事件类型。

例如：$(“p”).bind(“click”,function(){

$(this).append(“<span>1.发生了鼠标单击事件</span>”)；

})；

$(“p”).bind(“click.plugin”,function(){

$(this).append(“<span>2.发生了鼠标单击事件</span>”)；

})

触发事件：

$(“#btn1”).click(function(){$(“p”).trigger(“click”);});//触发命名空间plugin命名空间里面的click事件

$(“#btn2”).click(function(){$(“p”).trigger(“click.plugin”);});//只触发命名空间plugin里面的click

$(“#btn3”).click(function(){$(“p”).trigger(“click！”);});//只触发不在命名空间plugin里面的click

## 触发事件—触发事件时传递数组数据

概述：触发元素上的事件时，trigger()方法还可以使用一个数组形式的参数，给事件处理函数传递附加的数据

例如:$(“p”).bind(“myEvent”,function(event.param1,param2){

$(this).append(“<Span>发生了“+param1+param2+”</span>”);

});

$(“p”).trigger(“myEvent”,[“自定义事件”,“1”])；

$(“p”).trigger(“myEvent”,[“自定义事件”,“2”])；

## 事件类型

### 一．鼠标点击事件：

1. click事件—单机某个元素或其后代元素事件
2. dbclick事件—双击某个元素或后代元素事件
3. mousedown事件—鼠标左键按下某个元素或其后代元素事件
4. mouseup事件—鼠标左键抬起某个元素或其后代元素事件

### 二．鼠标移动事件

1. mouseenter事件—鼠标指针进入某个元素事件

2. mouseleave事件—鼠标指针离开某个元素事件

3. mouseover事件—鼠标指针进入某个元素或其后代元素事件

4. mouseout事件—鼠标指针离开某个元素或其后代元素事件

5. mousemove事件—当鼠标在某个元素或其后代元素移动事件

## 事件类型—合成事件

### 三．合成事件

1. hover事件—如果要为元素同时绑定mouseenter和mouseleave事件，那么可以使用hover()方法，

语法：hover(处理函数1(),处理函数2());鼠标移动到元素上时，调用处理函数1(),移出时调用处理函数2()

例如：$(“p:eq(0)”).hover(

Function(){

$(this).append(“<span>发生了mouseenter鼠标移动事件</span>”)；

}，

Function(){

$(this).append(“<span>发生了mouseleave鼠标移动事件</span>”)

})

1. toggle事件—如果要多次触发元素的click事件，且需要循环执行不同的函数，那么可以使用toggle()方法。

语法：toggle(处理函数1(),处理函数2(),处理函数3()……);第一次单机，执行处理函数1()，第二次单机执行处理函数2()……,最后一次以后，执行处理函数1()，循环执行。

例如：$(“p:eq(1)”).toggle(

Function(){

$(this).append(“<span>发生了单击事件1</span>”)；

}

Function(){

$(this).append(“<span>发生了单击事件2</span>”)；

}

Function(){

$(this).append(“<span>发生了单击事件3</span>”)；

})

### 四． 得到焦点和失去焦点

焦点概述：焦点是用于确定哪个元素正在接受键盘的输入，例如，获得焦点的文本框元素可以接受输入字符等。

1. focus事件—通过鼠标单机或Tab键定位元素触发。
2. blur事件—元素失去焦点事件
3. focusin事件—通过鼠标单机或Tab键定位元素或元素的后代元素获得焦点
4. focusout事件—通过鼠标单机或Tab键定位元素或元素的后代元素失去焦点

### 五．键盘事件

1. keydown事件—按下键盘上的任意键，会触发获得焦点元素的keydown事件

2. keyup事件—按下键盘上的任意键，会触发获得焦点元素的keyup事件

3. keypress事件—当想获得焦点的元素输入一个字符是，会触发该元素记其祖先元素发送的keypress

### 六．表单事件

1. change事件—当表单元素的value属性值或选中项发生变化时，会触发该元素记其祖先元素发送的change事件

2. select事件—挡在<input type=”text”>或<textarea>元素内进行文本选中时，会向该元素及其祖先元素发送select事件，需要注意，在IE浏览器中并不会向触发元素的祖先元素发送select事件

3.submit事件—通过<input type=”submit”>、<input type=”image”>或<button type=”submit”>元素或当其获得焦点时按下回车，可以提交表单。

### 七．加载和卸载事件

1. load事件—当某个元素及其后代元素已经完全加载时

2. unload事件—当离开当前页面时

## 事件对象

事件对象是一个事件发生时，保存事件各个相关信息的对象。在程序中使用事件对象非常简单，只需要为事件处理函数添加一个event参数即可。

例如：$(“p”).bind(“click”,function(event){

})//点击p元素的时候，就创建了event事件对象，这个事件对象只在事件处理函数里面才能访问到，事件处理函数执行完毕后，事件对象被销毁

## 事件对象—阻止事件冒泡的方法

前面学过的事件中，除了mouseenter、mouseleave、focus、blur外，绑定在祖先元素上的事件一般都可以通过后代元素来触发，而如果后代元素本身也绑定了统一类型的事件，那么在触发后代元素的事件时，祖先元素上的事件也会被触发，这种现象交做冒泡现象。

Jquery阻止事件冒泡 event.stopPropagation();

### 事件对象—阻止事件冒泡和默认行为的方法

阻止事件冒泡和默认行为的方法：return false;

(1).返回当前执行的所有对象都为false

(2).return false后面的所有代码都不会执行（阻止事件处理函数执行）

补充：triggerHandler()

使用trigger()方法触发事件时，不光会执行相应的事件处理函数，还会执行浏览器的默认操作，例如单机超链接后会自动跳转页面，单机提交按钮后表单会自动提交

使用triggerHandler()方法就可以阻止默认事件，只执行绑定的事件函数

### jQuery阻止事件冒泡的三种方法：

1. return false
2. stopPropagation
3. triggerHandler

## 事件对象传递参数

事件对象传递参数：当调用bind()或one()方法为元素绑定事件时，还可以选用参数data来传递额外的信息，并用事件对象的data属性在事件处理函数中进行接收，通常这个参数可以解决由于事件处理函数使用外部变量造成的问题。

例如:var message=”data1”,

$(“p:eq(0)”).bind(“click”,function(){

$(this).text(“变量为message的值为”+mesage)；

})；

Var message=”data2”;

$(“p:eq(1)”).bind(“click”,function(){

$(this).text(“变量message的值为”+mesage)

})//为两个p元素分别绑定click事件，当鼠标单机某个元素时，会写该其文本的内容，在这两个click事件处理函数里面，外部变量message的值均为data2.

由此可见，事件处理函数只能使用外部函数只能使用外部变量的最终值。而无法获得外部变量的当前值。

## 事件对象传递参数—问题解决

问题解决：

例如：var message=”data1”;

$(“p:eq(0)”).bind(“click”,{msg:message},function(event){

$(this).text(“变量message的值为”+event.data.msg);

});

Var message=”data2”;

$(“p:eq(1)”).bind(“click”,{msg:message},function(event){

$(this).text(“变量message的值为”+event.data.msg);

})//为两个p元素分别绑定click事件，当鼠标单机其中某个元素时，会修改其文本内容。在这两个click事件的处理函数里面。事件对象的data属性会接受到外部变量message的当前值，分别是字符串“data1”和“data2”

## 事件对象的触发键

如果需要知道触发事件的按键。那么可以使用事件对象的which属性，在鼠标点击事件中，which属性：

Which的鼠标按键值：

1. 表示左键
2. 表示中键
3. 表示右键

Which的键盘按键值：查看ASC||表

## 事件对象的发生位置和发生时间

如果用跟踪事件触发时的鼠标位置，那么可以使用pageX和pageY属性，这两个属性分别提供了鼠标指针位置的相对于页面左上角的X坐标和Y坐标。

例如：$(“p”).mousemove(function(event){

$(“p:eq(0)”).text(“移动鼠标相对于页面的坐标:”+event.pageX+”,”+event.pageY)

})

如果需要跟踪事件发生的时间，那么可以使用事件对象的timeStamp属性，这个属性能后返回时间触发时距离1970年1月1日的毫秒数。

# 第9章jQuery事件对象的其他处理

## 对元素进行遍历操作

如果要遍历一个jQuery对象，对其中每个匹配元素进行相应的处理，那么可以用each()方法。

语法：$(“选择器”).each(function(){

});

说明：对所有匹配选择器的元素逐个调用后面的函数。

说明：由于jQuery命令都具有隐式迭代的特点，所有完全无需要使用each()方法显示的循环遍历每一个元素进行操作，就能达到相同的效果。

对元素进行遍历操作—可以根据索引值index进行判断。

例如：$(“div”).each(function(){

If(findex!=2&&index<5)

$(this).css(“border”:”6px solid black”)

})

## 对元素进行数据存取

1. 在元素上存取数据

将数据存储在元素data方法的参数中

语法：data(“变量名”，变量值)；

1. Data()方法

Data()方法允许在元素上附加任意类型的数据。

语法：

设置元素中存储的变量值：data（“变量名”，变量值）；

获得第一个匹配元素变量值：data(“变量”)；

例如：$(“p”).data(“test”,”哈哈”)创建变量test且赋值({first:1,last:”two”});

Alert($(“p”).data(“test”))//输出1

Alert($(“p”).data(“test”).last);//输出two

1. removeData()方法：清除附加在元素上的变量

语法：removeData(name)

删除指定参量名的数据。

如果要清楚元素上的全部变量使用removeData()不带参数。

## 元素的个数和索引

1. size()方法：计算jQuery对象中元素的个数

例如：$(“p”).size();//4

说明：通过jQuery对象的length属性，也能得到jQuery对象中元素的个数

例如:$(“p”).length;//4

扩展应用：可以判断jQuery对象是否为空：

例如：if($(“span”).length==0)

Alert(“不存span元素！”)；

1. index()方法：返回某个元素相对于其他元素的索引位置。

例如：如果index（）方法，不带参数，他会计算第一个匹配选择器的元素的相对于其他兄弟的索引的位置

$(“#core”).index()://索引值3

例如：如果index(“选择器”)，带参数，他会计算出某一类元素中相对于其他同类元素的索引的位置

$(“#core”).index(“p”);//索引值为2，因为他是第3个p，索引从0开始。

例如：如果想知道，某一类元素中某个元素的位置

$(“p”).index(“#core”);//索引值为2。

# 第10章jQuery的基本方法

获得需要的jQuery对象和操作获得的jQuery对象,这是jQuery的核心内容。除此之外，jQuery还提供了一些非常好用的工具函数，用来处理一般的数据，例如，字符串、对象、数组、函数等。这些工具函数的用法均为：$(函数名)（）

## 处理字符串的方法：

去除字符串中两端空格$.trim(str)

方法说明：str为需要删除左右两边空白字符的字符串，字符串中间的空白字符不会受到影响，依旧被保留下来。

$.trim()+toLowserCase()把取出来的数据存到数据库中

## 处理对象的方法

要遍历对象的数据并进行操作$.each(obj,fn(name,value))

解释说明：obj表示为需要进行遍历的对象（json对象和array对象），fn(name,value)为需要在对象数据上执行的函数，name和value用于接收数据对象的属性名和值；

## 处理数组的方法

（1）对数组中数据项进行筛选，返回一个新数组$.grep(arr,fn(n,l))

语法说明：arr表示要进行筛选的数组，fn表示要执行的删选数组，n,I分别表示数组值和索参数。使用$.grep()方法只有其中的函数返回true是，该数据才会进入到新的数组当中去

（2）要对数组中的数据项进行修改、删除、添加、来获得一个新的数组$.map(arr,fn(n,i))

语法说明：arr为需要变更的数组，参数fn（n,i）为需要在数组数据上执行的处理函数，该函数本身可以接收数组值和数组索引为参数。

（3）把两个数组中的数据项合并在一起，来获得一个新数组，$.marge(arr1,arr2),参数arr1和arr2为需要被合并的数组。使用$.merge()方法时，不会光返回数组合并的结果，而且会向数组arr1添加数组arr2的数据项，从而改变了数组arr1.

例1.

Var arrOld1=[1,2,3];

Var arrOld2=[3,4,5];

Var arrNew=$.merge(arrOld1,arrOld2);

Alert(arrNew);// 输出1,2,3,4,5

Alert(arrOld1);//输出1,2,3,4,5

Alert(arrOld2)://输出3,4,5

(4)要在数组中搜某个数据值,$.inArray(value,arr)

说明：value为需要搜索出的数据值，参数arr为被搜索的数组，如果找到了指定数据值，就返回该数据值在数组中的索引，否则返回-1

Var arr=[1,2,3];

Alert($.inArray(3,arr));//输出2

Var arr=[1,2,3];

Alert($.inArray(4,arr));//输出-1

# 第十二章jQuery插件

Each:对象.Each(function(){}).$each(对象,function(){})

$.each()🡪扩展的就是jQuery的方法

$(obj).each()🡪扩展的就是jQuery对象的方法

## 创建插件的方法分为两种:

第一种对jQuery方法方法进行扩展的，$.\*\*\*方法名（参数，参数）

第二种创建插件的方法：扩展的是jQuery对象的方法 ,对象.方法名(function(){})

jQuery插件主要分成两类，一类插件是用来开发扩展jQuery的方法，另一类插件使用扩展jQuery对象的方法。

第一类插件，扩展jQuery的方法

语法结构：

$.funName=function([参数1，参数2，参数3…]){代码块}

调用方法：

$.funName([参数1，参数2，参数3…]);

## 第一类插件另一种构造方法，通过$.extend()方法

语法结构：

$.extend({

funName.function(参数1，参数2…){语句块}

funName.function(参数1，参数2…){语句块}

})

调用方法还是$.funName(参数1，参数2，…);

## 构造第二类jquery插件：

再jquery里面，extend这个方法专门是用来创建插件的，在extend中写的时候需要遵循对象的写法{插件名：function(){},插件名：function(){}}

调用的时候，用的是哪种方法创建插件的，就是用哪种方法进行调用

但是一般情况下我们在使用的时候使用扩展对象的方法是最多的。$.fn.funName=function(){}

因为在这个方法中有操作对象，想要当前的操作对象进行其他操作的话可以直接找到这个都对象，就是在调用完了插件之后，继续进行其他的操作。

语法结构：

$.fn.funName=function([参数1，参数2…]){语句块}

调用方法：

$(selector).funName([参数1，参数2…])

$.fn.插件名=function(){}🡪创建方法

$(obj).插件名()🡪调用方法

在构造第二类jquery插件时，还可以使用$.fun.extend()方法

语法结构：

$.fn.extend({

funName:function(参数1，参数2…){语句块}，

funName:function(参数1，参数2…){语句块}

})

## 优化jquery插件

对象.Each()🡪对数组进行遍历的，他操作的是每一个需要遍历的对象，所以如果想要返回去操作这个数组对象的话，那么需要加上return

返回操作的jquery对象

使用this.each()来遍历jquery对象中的每一个元素并会返回jquery对象，所以将this.each()的返回值作为插件的返回值即可

使用方法：

$.fn.funName=function(参数1，参数2){

Return $(this).each(function(){语句块})

}

设置传入参数的默认值

设置插件的参数，如果用户不传入参数的情况下，我们可以设置默认值，还进行执行，当用户设置了值，传入参数是，传入的参数就会代替默认值，设置方式很简单，只需要在插件中做一个判断

隐藏私有的函数和变量

Jquery插件方法可以使用外部的函数和变量，但是这些函数和变量需要进行额外的隐藏保护，否则就有可能被外界的一些顶级变量改写，就会破环jquery插件的功能，解决方法是使用一个闭包匿名函数将jquery插件的方法和私有变量全部放到这个匿名函数中

语法结构：

（function($){jquery插件方法}）（jquery） jquery插件调用

Js中创建变量使用的var进行创建的，创建出来之后就是一个全局变量，使用var创建出来的变量可以进行变量提升，在提升的时候会把当前变量提升为全局变量，之后如果需要使用的话就会污染当前这个全局变量，为了避免污染所以需要把这些方法放在匿名函数中进行调用

Es5中只有一种创建变量的方式就是var

Es6中新增了两种

所以我们在创建插件的时候，把创建插件的方法全都放在匿名函数中，作为私有函数和变量进行使用

# Firebug

\*Firebug插件概述

\*控制台及命令

\*查看和编辑html

\*查看和编辑css

\*Css段落文字

Firebug是firefox下的一款开发类插件，现属于firefox的五星级强力推荐插件之一。它集HTML查看和编辑、javascript控制台、网络状况监视器于一体，是开发JavaScript、css、HTML和Ajax的得力助手。

## 控制台—显示信息的命令：

Firebug内置一个console对象，提供5种方法，用来显示信息。

1. 最简单的方法是console.log（），可以用来取代alert()或document.write().比如，在网页脚本中使用console.log(“Hello world”)，加载时控制台就会自动显示。
2. 另外，根据信息的不同性质，console对象还有4种显示信息的方法，分别是一般信息console.info()、出错信息console.debug()、警告提示console.warn()、错误提示console.error()



## 控制台—占位符

Console对象的上面5种方法，都可以使用printf风格的占位符。不过，占位符的种类比较少，只支持字符（%s）、整数（%d或%i）、浮点数（%f）和对象（%o）四种。

例如：console.log(“%d年%d月%d日”,2011，3，26)；

console.log（“圆周率是%f”,3.1415926）；

## 控制台—分组显示

如果信息太多，可以分组显示，用到的方法是console.group()和console.groupEnd().

例如：

Console.group(“第一条信息”)；

Console.log(“第一组第一条”)；

Console.log(“第一组第二条”)；

Console.groupEnd();

Console.group(“第二组信息”)；

Console.log(“第二组第一条”)

Console.log(“第二组第二条”)

Console.groupEnd();

## 控制台—console.dir()

Console.dir()可以显示一个对象所有的属性和方法

例如：

比如，现在为第二节的dog对象，添加一个bark()方法

Dog.bark=function(){alert(“汪汪汪;”)};

然后，显示该对象的内容

Console.dig(dog);

## 控制台—console.assert()

Console.assert()用来判断一个表达式或变量是否为真。如果结果为否，则在控制台输出一条相应的信息，并且抛出一个异常。

## 控制台—console.trace()

Console.trace()用来追踪函数的调用轨迹。

## 控制台—计时功能

Console.time()和console.timeEnd(),用来显示代码的运行时间

例如：

Console.time(“计时器一”)

For(var i=0;i<1000,i++){

For(var j=0;j<1000;j++){}

}

Console.timeEnd(“计时器一”)

## 控制台—性能分析

性能分析（profiler）就是分析程序各个部分的运行时间，找出瓶颈所在，使用的方法是console.profile(),

例如：console.profile(“性能分析器一”)；

Foo() //方法调用

Console.profileEnd();

# fullPage.js

fullPage.js是一个基于jQuery的插件，它能够很方便、很轻松的制作出全屏网站，主要功能有：

支持鼠标滚动

支持前进后退和键盘控制

多个回调函数

支持手机、平板触摸事件

支持css3动画

支持窗口缩放

窗口缩放时自动调整

可设置滚动宽度、背景颜色、滚动速度、循环选项、回调、文本对齐方式等等

## 兼容性：

Jquery兼容

兼容jQuery1.7以上版本

浏览器兼容

IE8+

Chrome

Firefox

Opera

## 使用方法：

引入相关文件

<link rel=”stylesheet” type=”text/css” href=”js/jquery.fullpage.min.css”/>

<script type=”text/javascript” src=”js/jquery-1.8.3.min.js”></script>

<script type=”text/javascript” src=”js/jquery.fullpage.min.js”></script>

<!—jquery.easings.min.js用于easing参数，也可以使用完整的jQuery UI 代替。easing是指定动画在不同点上的行进速度-->

<script type=”text/javascript” src=”ja/jquery.easings.min.js”></script>

<!—如果scrollOverflow设置为true使用插件内置滚动条，则需要引入jquery.Slimscroll.min.js,一般情况下不需要-->

<script type=”text/javascript” src=”js/slimscroll”.min.js”></script>

补充：

fullPage的生产环境（环境搭建）

引入fullpage.css文件

然后是jquery

之后是jquery-ui

最后是fullpage.js

## 基本结构：

<div id=”fullpage”>

<div class=”section”>

<h3>第一屏</h3>

</div>

<div class=”section”>

<h3>第二屏</h3>

</div>

<div class=”section”>

<h3>第三屏</h3>

</div>

<div class=”section”>

<h3>第四屏</h3>

</div>

</div>

每个.section代表一屏，默认显示“第一屏”，中间可以插入页面内容，section中内容默认水平居左垂直居中。可以使用css设置背景颜色以及背景图片

调用fullpage.js相关方法

基础方法：

<script type=”text/javascript”>

$(document).ready(function(){

$(“#fullpage”).fullpage();//该方法实现整屏滚动效果

})

</script>

## 带有左右切换效果的整屏切换

<div id=”fullpage”>

<div class=”section”>第一屏</div>

<div class=”section”>第二屏</div>

<div class=”section”>

<div class=”slide”> 第三屏的第一屏</div>

<div class=”slide”> 第三屏的第二屏</div>

<div class=”slide”> 第三屏的第三屏</div>

<div class=”slide”> 第三屏的第四屏</div>

</div>

<div class=”section”>这是最后一屏</div>

</div>

可以在.section元素内加入.slide元素，实现横屏切换效果

## Fullpage初始化参数

配置选项采用json格式，用来对fullpage()方法进行设置，通过参数的设置，可以指定全屏滚动的不同效果

语法格式：

$(“选择器”).fullpage({

Name1:value1,

Name2.value2,

…

})

Anchors:[“值1”,”值2”，…]定义锚链接，数组元素为对听锚链接的名称

ScrollingSpeed:700 滚动速度，单位为毫秒，默认值700

Easing:easeInQuart 设置滚动方式，默认值easeInQuart，设置该参数需要引如jquery。Easings.min.js文件，具体参数参照jQuery UI API-Easings(设置不生效)

## Fullpage初始化参数

loopBottom：false 设置滚动最底部时是否滚回顶部，默认值false

looptop:false 设置滚到最顶部时是否滚回底部，默认值false

anchors:定义锚链接

controlArrowColor:#fff 左右滑块的箭头的背景颜色

menu:false 绑定菜单，设定的相关属性与anchors的值对应后，菜单可以控制滚动

scrollingSpead:700滚动速度，单位为毫秒

paddingTop:0 设置每屏上内边距，默认值0

paddingBottom:0 设置每屏下内边距，默认值0

keyboardScrolling:true 设置方向键导航，默认值true

verticalCentered:true 内容是否垂直居中

resize:false 字体是否随着窗口缩放而缩放

slidesColor:设置背景颜色

autoScrolling:true 是否使用插件滚动方式，如果选择false,则会出现浏览器自带的滚动条

scrollOverflow：false 内容超过满屏后是否显示滚动条

css3:false 是否使用css3transforms滚动

continuousVertical:false 是否循环滚动，与loopTop及loopBottom不兼容

easing:easeInQuart 滚动动画方式

loopHorizontal:true 设置左右滑动是否循环滚动，默认值true。

## 绑定菜单导航

Html格式

<ul id=”menu”>

<li data-menuanchor=”page1”><a href=”#page1”>第一屏</a></li>

<li data-menuanchor=”page2”><a href=”#page2”>第二屏</a></li>

<li data-menuanchor=”page3”><a href=”#page3”>第三屏</a></li>

<li data-menuanchor=”page4”><a href=”#page4”>第四屏</a></li>

</ul>

Css样式

具体样式根据需求设置即可，必须设置#id .active a{}激活状态时的样式

Fullpage绑定

$(“#menu”).fullpage({

Anchors:[“page1”,”page2”,”page3”,”page4”],

Menu:”#menu”

})

## 项目导航

项目导航

fullpage()中的参数进行设置

navigation:false 是否显示项目导航，默认值false不显示

navigationPosition:”right”项目导航的位置，可选left和right

navigationTooltips[“page1”,”page2”,”page3”…]设置项目导航鼠标划入提示信息

navigationColor:#000 项目导航的颜色

实例：项目导航.html

自定义项目导航.html

左右滑块项目导航

slidesNavigation:false 设置是否显示左右滑块项目导航

slidesNavPosition:”bottom” 设置左右滑块项目导航的位置，可选top和bottom

controlArrowColor:”#fff” 设置左右滑块箭头的颜色

## fullpage方法函数

前面介绍了fullpage的配置参数，接下来为大家介绍一些fullpage的方法函数，这些函数是在插件初始化外调用，不同于回调函数，且不受参数的影响。

**moveSectionUp()**

设置section向上滚动

$.fn.fullpage.moveSectionUp();

**moveSectionDown()**

设置section向下滚动

$.fn.fullpage.moveSectionDown();

**moveSlideRight()**

设置slide向右滑动

$.fn.fullpage.moveSlideRight()

**moveSlideLeft()**

设置slide向左滑动。

$.fn.fullpage.moveSlideLeft()

**setAutoScrolling(boolean)**

可以实时控制页面滚动的方式，可选的参数false/true

如果参数被设置为true，所有的section将自动滚动

如果参数被设置为false，suoyoudesection将不会自动滚动，需要用户 手动或者使用浏览器的滑条滚动网页

注意，scrollOverflow参数如果设置为true，他可能很难滚动鼠标滚动轮或触摸板当部分滚动

$.fn.fullpage.setAutoScrolling(false)

**setAllowScrolling(Boolean,[directions])**

添加或者禁止fullpage自动绑定的鼠标滑轮和移动触摸事件，不过用户仍然可以通过和点击快速导航的方式切换section/slide.

directions,可选参数，可以设置的值：all，up,down,left,right或者设置组合的参数，例如down，right，它设置的两个方向上将禁止或者激活滚动。

**setKeyboardScrolling(Boolean,[directions])**

添加或者进制键盘对section/slide的控制，这个事件是默认绑定的。

directions，可选参数，可以设置的值：all,up,down,left,right或者设置组合的参数，例如down、right，它设置的两个方向上将禁止或这激活键盘的滚动。

**setScrollingSpeed(milliseconds)**

定义每个section/slide滚动的时间，默认的时间单位为毫秒(ms).

$.fn.fullpage.setScrollingSpeed(700;)

## Fullpage回调函数

上一节中介绍了fullpage的方法函数，那些函数只适合单独使用，如果想更加详细的控制fullpage,就需要使用回调函数，接下来得文档将为您详细介绍fullpage中的会点函数使用方法和参数。

**afterLoad（anchorLink，index）**

滚动到某一屏后的回调函数，接收anchorLink和index两个参数。

anchorLink是锚链接的名称

index是section的索引，从1开始计算。

**onLeave(index,nextIndex,direction)**

滚动前的会点函数（离开当前页面时），接收index、nextIndex和direction3个参数

index是离开的“页面“的序号，从1开始计算；

nextIndex是滚动到的“页面“的序号，从1开始计算；

direction判断往上滚动还是往下滚动，值是up或down.

**afterRender()**

这个问题函数只是在生成页面结构的时候调用，这是要用来初始化其他插件或删除任何需要的文件准备好代码的回调。

**AfterSlideLoad(anchorLink,index,slideAnchor,slideIndex)**

滚动到某一水平滑块后的回调函数，与afterLoad类似，接收anchorLink、index、slideIndex、direction4个参数，slideIndex从0开始计算

**onSlideLeave(anchorLink,index,slideIndex,direction,nextSlideIndex)**

某一水平滑块滚动前的回调函数，与onLeave类似，接二手anchorLink、index、slideIndex、direction、nextSlideIndex5个参数。slideIndex和nextSlideIndex都是从0开始计算

# 响应式开发

## 什么是响应式：

响应式网页设计最初是由Ethan Marcotte提出的一个概念：为什么一定要为每个用户群各自打造一套设计和开发方案？web设计应该做到根据不同设备环境自动响应及调整。当然响应式web设计不仅仅是关于屏幕分辨率自适应以及自动缩放的图片等等，它更像是一种对于设计的全新思维模式：我们应当向下兼容、移动优先。

优雅降级:向下兼容

渐近增强:向上兼容

响应式布局应该是一种弹性布局，不同尺寸下弹性适应，如以下页面中各模块在不同尺寸下的位置

## 响应式布局老方法

Viewport分为三种:

Visual

Layout

Ideal

Meta标签定义

使用viewport meta标签在手机浏览器上控制布局

<meta name=”viewport” content=”width=device-width, initial-scale=1.0maximum-scale=1.0, user-scalable=0”/>”

通过快捷打开时全屏显示

<meta name=”apple-mobile-web-app-capable” content=”yes”/>

隐藏状态栏

<meta name=”apple-mobile-web-app-status-bar-style” content=”blank”/>

iPhone会将看起来像电话号码的数字添加电话链接，应当关闭

<meta name=”format-detection” content=”telephone=no”/>

## 利用css3@media only screen and (min-width:\*px)和rem设置

rem是设置通过根元素（html）进行适配的

rem得兼容性

以屏幕640为标准rem算法

@media (min-device-width:1024px) and (max-width:980px), screen and (max-deice-width:480px), (max-device-width:480px),(max-device-width:480px) and (orientation:landscape),(min-device-width:480px) and (max-device-width:1024px) and (orientation:portrait) (srules)