# 1-开始

## 优势：

1. 轻量级
2. 强大的选择器
3. 出色的DOM操作封装
4. 可靠的事件处理机制
5. 完善的Ajax
6. 不污染顶级变量，所有函数，变量都封装在名为jQuery的对象中，与其他库不冲突
7. 出色的浏览器兼容
8. 链式操作，对发生在同一个jQuery对象上的一组动作，无需重复获取对象就可以直接连写
9. 隐式迭代，jQuery里的方法被设计成自动操作返回的对象集合，而不是单独的对象，因此无需循环对每个对象操作
10. 行为层与结构层分离
11. 丰富的插件
12. 完善的文档
13. 开源

## $与jQuery

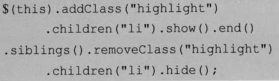
在jQuery中，$就代表jQuery这个对象

## $(document).ready()与window.onload

|  |  |  |
| --- | --- | --- |
|  | Onload | Ready |
| 执行时间 | 网页所有内容加载完毕（包括图片） | 所有Dom结构绘制完成 |
| 编写个数 | 不能有多个onload，如果有，只执行最后一个 | 可以有多个ready函数，都会被执行 |
| 简化写法 | 无 | 可以简写为$（function（）{……}） |

## 链式写法

1. 对一个对象不超过3个操作的，可以写成一行
2. 对于同一个对象操作较多的，建议每行写一个操作，或操作相关的写成一行
3. 对多个对象的少量操作，可以每个对象写一行，如果涉及子元素，适当缩进



## DOM对象与jQuery对象

JQuery对象无法使用DOM对象的任何方法，DOM也不能使用jQuery对象的方法

### 相互转换

jQuery->DOM

提供了【index】

var $jq=$(“#id”) //jQuery对象

var dom=$jq[0] //DOM对象

get（index）方法

var $jq=$(“#id”) //jQuery对象

var dom=$jq.get(0) //DOM对象

DOM->jQuery

$(DOM对象)

var dom=document.getElementById(“id”)

var $jq=$(dom);

## $冲突

最好在其他库导入之前导入JQuery，那么就可以直接使用jQuery来做一些工作，而$()方法作为其他库的快捷方式，不会出现冲突。

在任何时候都可以使用jQuery.noConflict()将$使用权交给别的js库

想自定义快捷方式

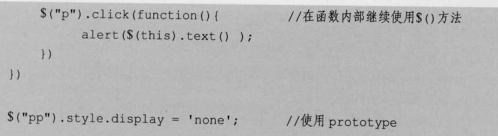
var $j= jQuery.noConflict();

$j(“#id”).click()

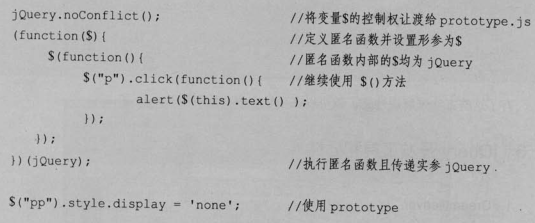
其他解决冲突方法：

方法1：将$传入function中即可





方法2：



# 2-jQuery选择器

css应用到网页的3中方式（按优先级高低）：

* 行间样式（优先级最高，不推荐使用，不符合表现与结构分离的规范）
* 内部样式
* 外部样式

需要注意的是：$(“#id”)获取的永远是对象，即使网页上没有此元素，因此当要判断某元素在网页上是否存在时，

If($(“#id”)){

//doing

}

需要这样：

If($(“#id”).length>0){

//doing

}

## 基本选择器

|  |  |  |
| --- | --- | --- |
|  |  | 说明 |
| $(“#id”) | Id选择器 |  |
| $(“.class”) | 类选择器 |  |
| $(“element”) | 标签选择器 |  |
| $(“selector1,selector2,……”) | 群组选择器 |  |
| $(“\*”) | 全部选择器 |  |

图 基本选择器-1

## 层次选择器

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 说明 |  |
| $(“ancestor descentdant”) | 后代选择器 | 选择ancestor元素内的所有子元素 |  |
| $(“parent>children”) | 子选择器 | 选择parent元素下的子元素，不包括更内层的元素 |  |
| $(“prev+next”) | 紧邻选择器 | 选择prev元素下一个的元素， | 等价于$(“prev”).nextl(next) |
| $(“prev~sibling”) | 后兄弟选择器 | 选择prev元素后的兄弟节元素， | 等价于$(“prev”).nextAll(sibling) |
| $(“prev”).siblings(“div”) | 全部兄弟选择 |  |  |

图 层次选择器-2

$(“prev”).siblings(“div”)是全部的兄弟节点，包括前后的

## 过滤选择器

### 基本过滤器

|  |  |  |
| --- | --- | --- |
| $(“selector：first”) | 选择第1个元素（div） |  |
| $(“selector：last”) | 选择最后一个元素（div） |  |
| $(“selector：not（selector）”) | 反选择 | $(“input:not(.myClass)”)选取class不是myClass的对象 |
| $(“selector：even”) | 选择偶数元素 |  |
| $(“selector：odd”) | 选择奇数元素 |  |
| $(“selector：eq（index）”) | 选择等于索引index的元素（从0开始） | 索引是根据DOM构建时排序的，递归获取的 |
| $(“selector：gt（index）”) | 选择大于索引index的元素 | 索引是根据DOM构建时排序的，递归获取的 |
| $(“selector：lt（index）”) | 选择小于索引index的元素 | 索引是根据DOM构建时排序的，递归获取的 |
| $(“：header”) | 选择所有的标题元素 |  |
| $(“selector：animated”) | 选择当前正在执行动画的所有元素 | $(“div：animated”)正在执行动画的div |
| $(“selector：focus”) | 选择当前获取焦点的元素 |  |

图 过滤选择器-3

### 内容过滤器

|  |  |  |
| --- | --- | --- |
| $(“selector：contain（’text’）”) | 选择包含text的元素 |  |
| $(“selector：empty”) | 选择不包含子元素或文本的元素 |  |
| $(“selector：has（selector2）”) | 选择含有选择器2所匹配的元素 |  |
| $(“selector：parent”) | 选择含有子元素的元素 |  |

图 过滤选择器-4

### 可见性过滤器

|  |  |  |
| --- | --- | --- |
| $(“selector：hidden”) | 选择不可见的元素 |  |
| $(“selector：visible”) | 选择可见的元素 |  |

图 过滤选择器-5

### 属性过滤器

|  |  |  |
| --- | --- | --- |
| [attribute] | 拥有该属性的元素 | $(“div[id]”)选择拥有属性id的元素 |
| [attribute=value] | 属性值等于value的元素 | $(“div[title=test]”)选择id属性为test的元素 |
| [attribute!=value] | 属性值不等于value的元素 |  |
| [attribute^=value] | 属性值以value开头的元素 |  |
| [attribute$=value] | 属性值以value结束的元素 |  |
| [attribute\*=value] | 属性值含有value的元素 |  |
| [attribute|=value] | 属性值等于value或以value-为前缀的元素 |  |
| [attribute~=value] | 属性值以空格分隔后包含value的元素 |  |
| [attribute=value][attribute]…… | 多个属性过滤器 |  |

### 子元素过滤器

|  |  |  |
| --- | --- | --- |
| :nth-child(index,even,odd,equation) | ：eq是从0开始，且只返回一个元素。而这个是从1开始的，可以返回多个元素 | :nth-child(3n+1)选择索引值是3n+1的元素 |
| :first-child | first只返回第一个元素，first-child将为每个父元素匹配第一个子元素 | $(“ul li:first-child”) 将选择每个ul下第一个li元素 |
| :last-child | last只返回最后一个元素，last-child将为每个父元素匹配最后子元素 | $(“ul li:last-child”) 将选择每个ul下最后一个li元素 |
| :only-child |  | $(“ul li:only-child”) 将选择每个ul下是唯一子元素的li元素 |

### 表单过滤器

|  |  |  |
| --- | --- | --- |
| ：input | 选择所有<input>、<textarea>、<select>、<button>元素（所有表单元素） | $(“#form：input“)与$(“#form input”)是不同的 |
| ：text | 单行文本 |  |
| ：password | 密码框 |  |
| ：radio | 单选框 |  |
| ：checkbox | 多选框 |  |
| ：submit | 提交按钮 |  |
| ：image | 图像 |  |
| ：reset | 重置 |  |
| ：button | 按钮 |  |
| ：file | 上传域 |  |
| ：hidden | 不可见元素 |  |

### 表单属性过滤器

|  |  |  |
| --- | --- | --- |
| ：enable | 选择所有可用元素 |  |
| ：disable | 选择不可用元素 |  |
| ：checked | 选择所有被选中的元素（单选框，复选框） | $(“input：checked”) |
| ：selected | 选择被选中的选项元素（下拉框） | $(“select option:selected”) |

### filter与 find

* Filter操作的是集合元素，参数传入多个选择器，filter（“：contain（‘text’）,：hidden”），是对自身集合元素进行筛选，如：

var $lis=$(ul li);

$lis.filter(“：contain（‘text’）,：hidden”)

* Find操作的是单个元素，是对这个元素中的子元素进行筛选

var $id=$(“#id”);

$lis.find(expr);

# 3-DOM操作

## 节点操作

### 创建节点

$(“html节点”)

例如：

var $li=$(“<li title=‘嘻嘻嘻‘>嘻嘻嘻</li>”); //新建节点

$(“ul”).append(“$li”) //添加到ul中

### 插入节点

|  |  |  |
| --- | --- | --- |
| A.append(B) | 将B插入到A内部的后面 | A:<P>你好</p>  B:<b>我</b>  结果：<P>你好<b>我</b></p> |
| B.appendTo(A) | 同上 |  |
| A.prepend(B) | 将B插入到A内部的前面 | A:<P>你好</p>  B:<b>我</b>  结果：<P><b>我</b>你好</p> |
| B.prependTo(A) | 同上 |  |
| A.after(B) | 将B插到A后面 | A:<P>你好</p>  B:<b>我</b>  结果：<P>你好</p><b>我</b> |
| B.insertAfter(A) | 同上 |  |
| A.before(B) | 将B插在A之前 | A:<P>你好</p>  B:<b>我</b>  结果：<b>我</b><P>你好</p> |
| B.InsertBefore(A) | 同上 |  |

### 删除（剪切）节点

**remove**

var d=$(“selector”).remove（“selector”）

$().append(d); //粘贴到别的地方

删除该元素内的元素，并返回删除的元素，绑定的事件全部消失

**detach（保留事件）**

与remove类似，但删除的元素绑定的事件会保留下来

**empty**

并非删除该节点，而是清空该节点内的东西

### 复制节点

**Clone**

$(“this”).clone().appendTo(“selector”) //不复制事件

$(“this”).clone(**true**).appendTo(“selector”) //将事件一起复制

### 替换节点

|  |  |
| --- | --- |
| A.replaceWith(B) | 用B替换A |
| B.replaceAll(A) | 用B替换A |

### 包裹节点

|  |  |  |
| --- | --- | --- |
| A.wrap(B) | 用B包裹每个A中的元素 | A:<li>收到</li>  <li>收到</li>  B:<div></div>  结果：  <div><li>收到</li></div>  <div><li>收到</li></div> |
| A.WrapAll(B) | 用B包裹A（在A外层包裹），  如果被包裹的多个元素之间有其他元素，则其他元素会被放到包裹元素之后 | A:<li>收到</li>  <li>收到</li>  B:<div></div>  结果：  <div>  <li>收到</li>  <li>收到</li>  </div> |
| A.WrapInner(B) | 用B包裹A内部的内容 | A:<li>收到</li>  B:<div></div>  结果：  <li><div>收到</div></li> |

### 属性操作

* 获取属性：attr(“属性”)
* 设置单个属性：attr（“属性”，“value”）
* 设置多个属性：attr({“属性1”：“value1”，“属性2”：“value2”})
* 删除属性：removeAttr(“属性”)

### 样式操作

* **获取样式**：attr(“class”)
* **设置样式：**attr(“class”,”新class”) //会替换掉原来的class
* **添加样式**：addClass（“class”） //添加后马上生效
  + 一个元素添加多个class，就相当于合并了他们的样式
  + 多个不同的class设定了同一个样式属性，后者覆盖前者的设置
* **删除样式**：removeClass（“类名”）
  + 删除多个类，可以使用空格隔开，removeClass（“类名1 类名2 类名3”）
  + 删除全部类，不传参数即可，removeClass（）
* **切换样式：**toggleClass（“类名”）
  + 如果类名存在则删除，如果不存在，则添加该类名
* 判断是否含有某个类名：hasClass（“类名”）

### 文本操作