# 选择器（selector）

|  |  |  |  |
| --- | --- | --- | --- |
| 选择符 | CSS | jQuery | 说明 |
| A | A | $(‘A’) | DOM中的标签 |
| ID | #some-id | $(‘#some-id’) | DOM中ID为some-id的一个元素 |
| 类 | .some-class | $(‘.some-class’) | DOM中类为some-class的所有元素 |

# DOM遍历方法

|  |  |  |
| --- | --- | --- |
| 遍历方法 | 返回的对象包含 | 备注 |
| .filter(selector) | 与给定的选择符selector匹配的元素 | $('td').filter('[id^=td2]')  Td标签中，id以’td2’字符开头的 |
| .filter(callback) | 回调函数callback返回treu的元素 | $('td').filter(testfun)  **function** testfun(){  **if**($(**this**).attr('id').indexOf('td2')==0){  **return** **true**;  }**else**{  **return** **false**  }  } |
| .not(selector) | 与.filter功能相反 | $('td').not('[id=td1]')  Td标签中id不为td1的Td元素 |
| .find(selector) | 与selector匹配的子元素 | $('tr').find('[id=td1]')  Tr标签中id为td1的Td元素 |

# DOM操作方法

|  |  |  |
| --- | --- | --- |
| 选择符 | 含义 | 备注 |
| .attr(key) | 获取属性key的值 | $('#text1').attr('value') |
| .attr(key,value) | 设置属性key的值 | $('#text1').attr('value','hello world!'); |
| .attr(key,fn) | 设置属性key的值为fn的返回结果 | $('tr').attr('value',**function**(){  **return** $(**this**).attr('id');  }); |
| .addClass(class) | 为每个匹配的元素增加类 |  |
| .removeClass(class) | 为每个匹配的元素移除类 |  |
| .html() | 获得第一个匹配元素的html |  |
| .html(value) | 设置第一个匹配元素的html为value |  |
| .text() | 获取每个匹配元素的文本内容，返回一个字符串 |  |
| .text(value) | 设置每个匹配元素的文本内容 |  |
| .val() | 获得第一个匹配元素的value属性的值 |  |
| .val(value) | 设置第一个匹配元素的value属性的值为value |  |
| .css(key) | 获取css属性 |  |
| .css(key,value) | 设置css属性 |  |
| .height() | 获得第一个匹配元素的高度 |  |
| .height(value) | 设置每个匹配元素的高度 |  |
| .width() | 获得第一个匹配元素的宽度 |  |
| .width(value) | 设置每个匹配元素的宽度 |  |

# 事件方法

## 给元素增加、删除事件（bind、ubind）

|  |
| --- |
| <input type=*"button"* id=*"btn1"* value=*'btn1'*/>  <script type=*"text/javascript"*>  $().ready(**function** (){  $('#btn1').**bind**('click',**function**(){  alert('aaaa');  });  });  </script>  <script type=*"text/javascript"*>  $().ready(**function** (){  $('#btn1').**unbind**('click');  });  </script> |

## 增加事件的简写方式

|  |
| --- |
| <input type=*"button"* id=*"btn1"* value=*'btn1'*/>  <script type=*"text/javascript"*>  $().ready(**function** (){  $('#btn1').**click**(**function**(){  alert('aaaa2');  });  });  </script> |

## 模拟事件触发

$('#btn1').trigger('click');

$('#btn1').click();

## 可以使用简写方式的事件

|  |  |  |  |
| --- | --- | --- | --- |
| blue | change | click | dblclick |
| error | focus | keydown | keypress |
| keyup | load | mousedown | mousemove |
| mouseout | mouseover | mouseup | resize |
| scroll | select | submit | unload |



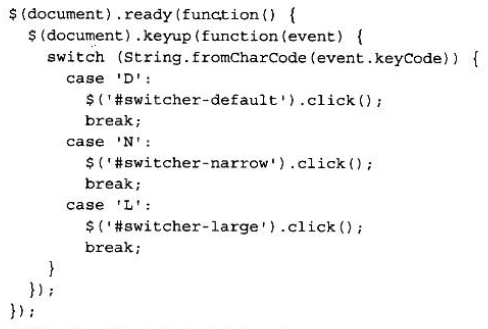
# 常用代码块

交替调用两个方法toggle

## 设置元素的样式

$('#test').css('display','none');

## 键盘（keyup,keydown,keypress）



## 取消$的控制权

jQuery.noConflict();

用jQuery代替$

jQuery(document).ready(function($){

//在这里可以正常使用$

});

jQuery(function($){

//在这里可以正常使用$

});

## 修改或获取元素的属性

* 获取：$('#text1').attr('value')
* 修改：$('#text1').attr({'value':'sssss'})

## 元素.each

$('input').**each**(**function**(**index**){

alert($(**this**).val()+'-'+**index**);

});

$('input').**each**(**function**(){

alert($(**this**).val()+'-'+index);

});

## $.each

* 核心代码

$.getJSON('test2.js',**function**(data){

**$.each(data,function(entryIndex,entry){**

textValue = textValue + entryIndex + entry ;

});

$('#text1').attr('value',textValue);

});

* data数据结构，数据或map等；
* function(entryIndex,entry)回调函数

当data为数据时，entryIndex为0,1,2,3…… 索引

当data为map时，entryIndex为map的key

* div

<div id=*'chapter'*>

</div>

* test1.js

{

"key1" : "value1",

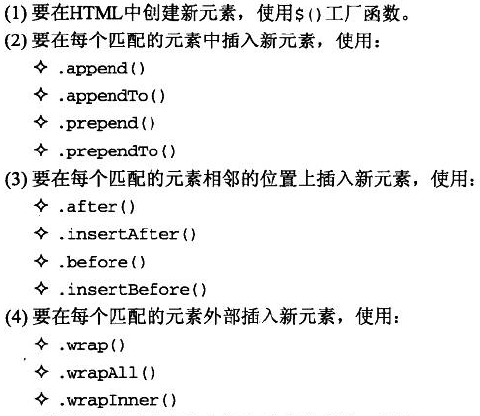
"key2" : "value2"

}

* test2.js

["value1","value2","value3"]

## 操作元素





### 增加元素

|  |  |
| --- | --- |
| $('<a href="#top" >back to top</a>').**insertAfter**('div'); | 插入到div后 |
| $('div').**after**('<a href="#top" >back to top</a>'); | 插入到div后 |
| $('<a href="#top" >back to top</a>'). insertBefore('div'); | 插入到div前 |
| $('div'). before('<a href="#top" >back to top</a>'); | 插入到div前 |
| $('div'). prependTo('<a href="#top" >back to top</a>'); | 插入到div中的第一个元素 |
| $('div'). appendTo('<a href="#top" >back to top</a>'); | 插入到div中的最后一个元素 |

### 包装元素

把test元素封装到li标签内。

$('#test').wrap('<li></li>');

### 移动元素

把test元素插入到div中的第一个元素

$('#test').prependTo('div');

### 复制元素

复制test元素

$('#test').clone(); //没有复制test元素中的事件

$('#test').clone(true); //复制了test元素中的事件

## 把数组转成字符串

**var** s = ['1','2','3'];

s.join('w'); 结果为：1w2w3

s.join(''); 结果为：123

s.join(); 结果为：1,2,3

## 加载其它页面（元素.load）

$('#chapter').load('test1.html');//把test1.html中的内容加载到id=chapter的元素中

<div id=*'chapter'*>

</div>

## 操作JS对象（$.getJSON）

* 核心代码

$.getJSON('test.js',**function**(data){

$('#chapter').text(data.key1);//key1不需要加引号

});

//把test.js中的key1键对应的值加载到id=chapter的元素中

* div

<div id=*'chapter'*>

</div>

* test.js

{

"key1" : "value1",

"key2" : "value2"

}

## 操作xml（$.get）

* 核心代码

**var** textValue = '';

$.get('test1.xml',**function**(data){

$(data).find('entry').filter('[id^=dog]').each(**function**(index){

textValue = textValue + $(**this**).text()+',';

});

$('#text1').attr('value',textValue);

});

//获取test1.xml中id以dog开头的entry节点的text

* test1.xml

<?xml version="1.0" encoding="UTF-8"?>

<entries>

<entry id="dog1">dog1-name</entry>

<entry id="dog2">dog2-name</entry>

<entry id="cat1">cat1-name</entry>

</entries>

## 执行其它js文件（getScript）

* $.getScript('test3.js');
* test3.js

alert(111);

## 插件

### 自定义全局函数（添加、调用）

|  |  |  |
| --- | --- | --- |
| 功能 | 代码 | 说明 |
| 添加 | $.myfun = **function**(){  alert('dfsdf');  }; | 使用$ |
| 添加 | jQuery.myfun = **function**(){  alert('dfsdf');  }; | 使用JQuery |
| 添加 | $.extend({  myfun1:**function**(){  alert('myfun1');  },  myfun2:**function**(){  alert('myfun2');  }  }); | 使用extend |
| 添加 | $.myplugin = {  myfun1: **function** (){alert('myfun1')},  myfun2: **function** (){alert('myfun2')}  }; | 使用自定义空间 |
| 调用 | $.myfun1();  或  jQuery.myfun1(); |  |

### 自定义对象方法（添加、调用）

* 添加

$.fn.print = **function** (){

alert($(**this**).val();

)};

* 调用

$('#text1').print();

### 自定义选择符表达式

* 定义：

Dom中所有type属性为text，并且id= ?的元素

$.extend($.expr[':'],{

'mytext' : **function**(element,index,matches,set){

**if**($(element).attr('type')=='text'){

**if**(matches[3]==$(element).attr('id')){

**return** **true**;

}**else**{

**return** **false**;

}

}**else**{

**return** **false**;

}

}

});

Elemet：当前元素；

Index：当前元素在结果集中的索引；

Matches[3] ：mytext(b)中的b；

Set：到目前为止匹配的整个DOM元素集合；

1. 调用：

Dom中所有type属性为text，并且id= text1的元素

$(':mytext(text1)').each(**function** (){

alert($(**this**).val());

});

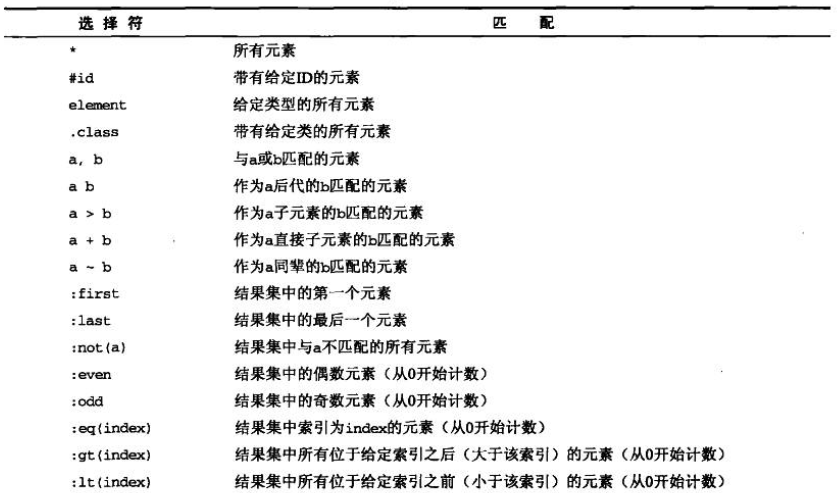
### 默认参数

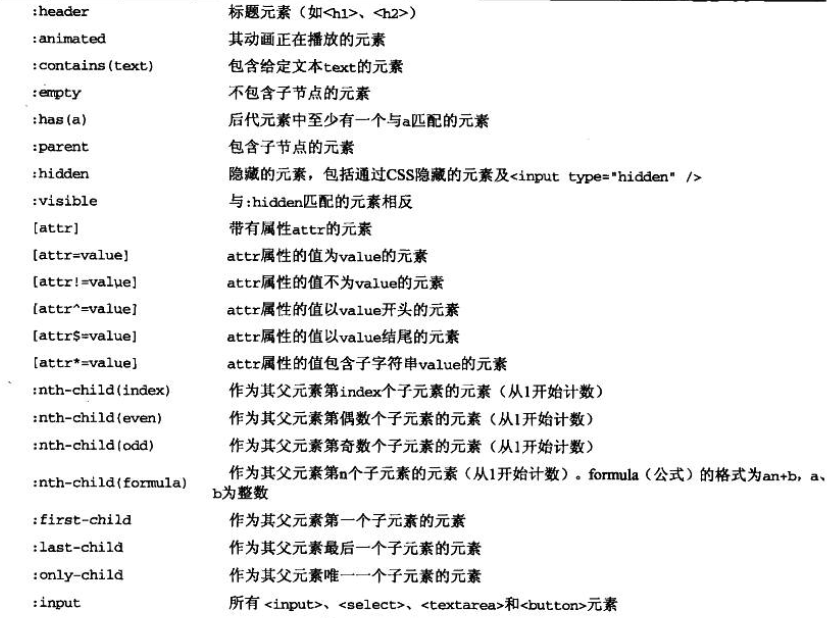
|  |  |  |
| --- | --- | --- |
| 功能 | 代码 | 说明 |
| 定义 | $.print = **function** (map){  **var** defMap = {  width : 100,  height : 200  };  $.extend(defMap,map);  alert('宽：'+defMap.width+','+'高：'+defMap.height);  }; |  |
| 调用 | $.print(); | 无参数传入时使用默认参数 |
| 调用 | $.print({  width:150,  height:33  }); | 有参数传入时覆盖默认参数 |
| 调用 | $.print({  width:150  }); | 只有部分参数传入时，传入什么值，覆盖什么默认值 |

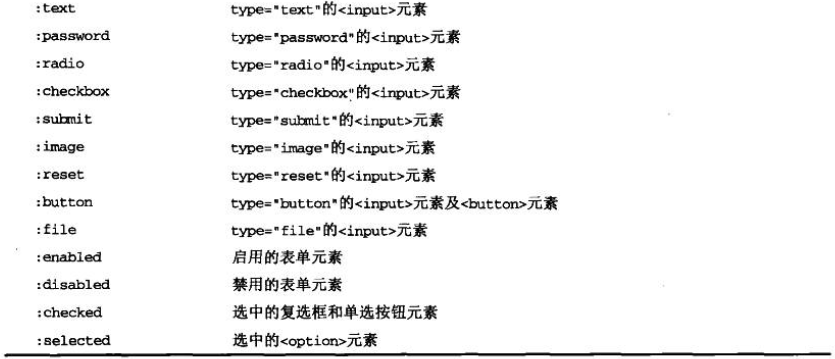
# 快速参考

## 选择符（selector）

|  |  |  |
| --- | --- | --- |
| 选择符 | 匹配 | 说明 |
| \* | 所有元素 | 如：${‘\*’} |
| #id | 带有给定id的元素 | 如：${‘#id’} |
| element | 综合类型的所有元素 | 如：${‘a’} |
| .class | 带有给定类的所有元素 | 如：${‘.testclass’} |
| a,b | 与a或b匹配的元素 | $('div,tr')  Dom中所有的div和tr元素 |
| a b | 作为a后代的b匹配元素 | $('#table1 tr')  Id=table1中所有的tr元素 |
| a > b | 作为a子元素的b匹配元素 |  |
| a + b | 作为a直接子元素的b匹配元素 |  |
| a ~ b | 作为a同辈的b匹配元素 |  |
| :first | 结果集中的第一个元素 |  |
| :last | 结果集中的最后一个元素 |  |
| :not(a) | 结果集中与a不匹配的所有元素 | $('#table1 :not(tr)')  Id=table1中不为tr标签的元素 |
| :even | 结果集中的偶数元素（从0开始） |  |
| :odd | 结果集中的奇数元素（从0开始） |  |
| :eq(index) | 结果集中索引为index的元素（从0开始） |  |
| :gt(index) | 结果集中所有位于index之后的元素（从0开始） |  |
| :lt(index) | 结果集中所有位于index之前的元素（从0开始） |  |
| :header | 标题元素如<h1><h2> |  |
| :animated | 动画正在播放的元素 |  |
| :contains(text) | 包含给定文本text的元素 |  |
| :empty | 不包含子节点的元素 |  |
| :has(a) | 后代元素中至少有一个与a匹配的元素 |  |
| :parent | 包含子节点的元素 |  |
| :hidden | 隐藏的元素，包括通过css隐藏的和  <input type=’hidden’>的元素 |  |
| :visible | 与:hidden相反 |  |
| [attr] | 带有属性attr的元素 | $('[id]')  $('#table1 [id]') |
| [attr=value] | attr属性的值为value的元素 | $('[id=test]')  Id=test的元素 |
| [attr!=value] | attr属性的值不为value的元素 | $('[id^=td]')  Id以td开始的元素 |
| [attr^=value] | attr属性的值以value开头的元素 |  |
| [attr$=value] | attr属性的值以value结尾的元素 |  |
| [attr\*=value] | attr属性的值包含字符串value的元素 |  |
| :nth-child(index) | 作为其父元素第index个子元素的元素（从1开始计数） | $('#tr1 :nth-child(2)') |
| :nth-child(even) | 作为其父元素第偶数个子元素的元素（从1开始计数） | $('#tr1 :nth-child(even)') |
| :nth-child(odd) | 作为其父元素第奇数个子元素的元素（从1开始计数） | $('#tr1 :nth-child(odd)') |
| :nth-child(formula) | 作为其父元素第n个子元素的元素（从1开始计数）。formula的格式为an+b，a,b为整数 |  |
| :first-child | 作为其父元素第1个子元素 | $('#tr1 :first-child) |
| :last-child | 作为其父元素最后一个子元素 | $('#tr1 :last-child') |
| :only-child | 作为其父元素唯一一个子元素 |  |
| :input | 所有<input><select><textarea><button>元素 | $(':input') |
| :text | type=text的<input>元素 |  |
| :password | type= password的<input>元素 |  |
| :radio | type= radio的<input>元素 |  |
| :checkbox | type= checkbox的<input>元素 |  |
| :submit | type= submit的<input>元素 |  |
| :image | type= image的<input>元素 |  |
| :reset | type= reset的<input>元素 |  |
| :button | type= button的<input>元素 |  |
| :file | type= file的<input>元素 |  |
| :enabled | 启用的表单元素 |  |
| :disabled | 禁用的表单元素 |  |
| :checked | 选中的单元和复选框按钮 | $('input[name="items"]:checked')  Input标签name=’items’，并且选中的复选框 |
| :selected | 选中的<option>元素 |  |

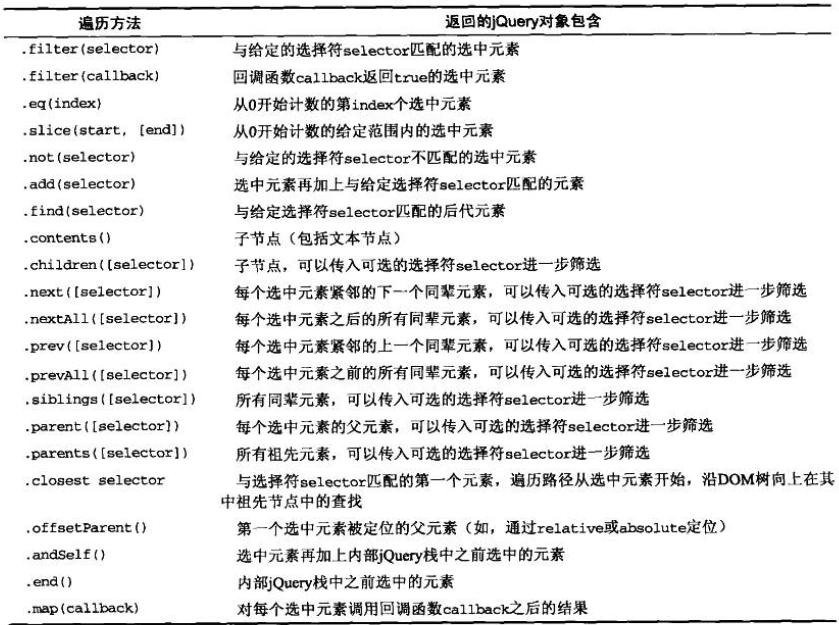






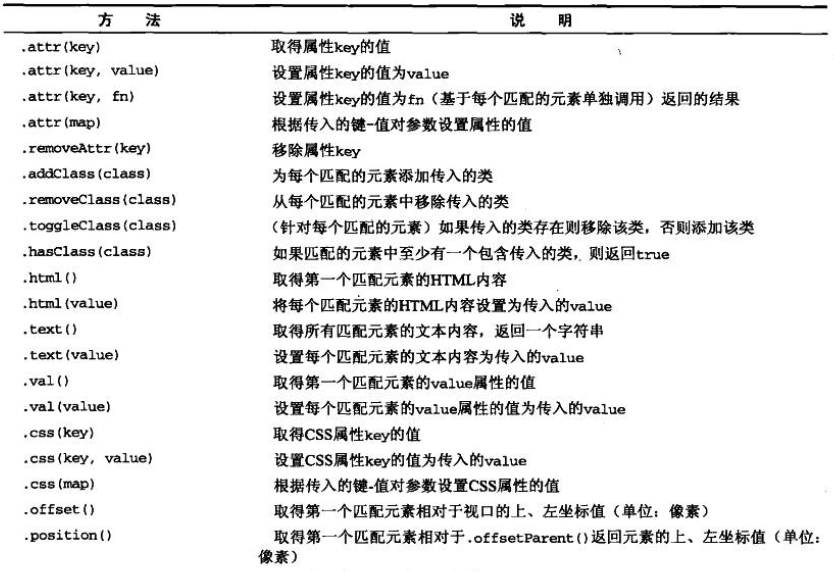
## DOM遍历方法

|  |  |  |
| --- | --- | --- |
| 遍历方法 | 返回的对象包含 | 备注 |
| .filter(selector) | 与给定的选择符selector匹配的元素 | $('td').filter('[id^=td2]')  Td标签中，id以’td2’字符开头的 |
| .filter(callback) | 回调函数callback返回treu的元素 | $('td').filter(testfun)  **function** testfun(){  **if**($(**this**).attr('id').indexOf('td2')==0){  **return** **true**;  }**else**{  **return** **false**  }  } |
| .eq(index) | 从0开始的第index个元素 |  |
| .slice(start[,end]) | 从0开始的给定范围的元素 |  |
| .not(selector) | 与.filter功能相反 | $('td').not('[id=td1]')  Td标签中id不为td1的Td元素 |
| .add(selector) | 增加选定条件 | $('td').add('tr')  增加tr标签 |
| .find(selector) | 与selector匹配的子元素 |  |
| .contents() | 子节点 |  |
| .children([selector]) | 子节点，可以根据selector进一步筛选（与find类似） |  |
| .next([selector]) | 下一个同辈元素，可以根据selector进一步筛选 |  |
| .nextAll([selector]) | 之后所有同辈元素，可以根据selector进一步筛选 |  |
| .prev([selector]) | 上一个同辈元素，可以根据selector进一步筛选 |  |
| .prevAll([selector]) | 之前所有同辈元素，可以根据selector进一步筛选 |  |
| .siblings([selector]) | 所有同辈元素，可以根据selector进一步筛选 |  |
| .parent([selector]) | 父幸元素，可以根据selector进一步筛选 |  |
| .parents([selector]) | 所有祖先元素，可以根据selector进一步筛选 |  |
| .closest(selector) | 从选中元素开始，沿DOM树向上在其祖先节点中查找 |  |
| .offsetParent() | 第一个选中元素被定位的父元素 |  |
| .andSelf() | 选中元素再加上内部jquery栈中之前选中的元素 |  |
| .end() | 内部jquery栈中之前选中的元素 |  |
| .map(callback) | 对每个选中元素调用callback之后的结果 |  |

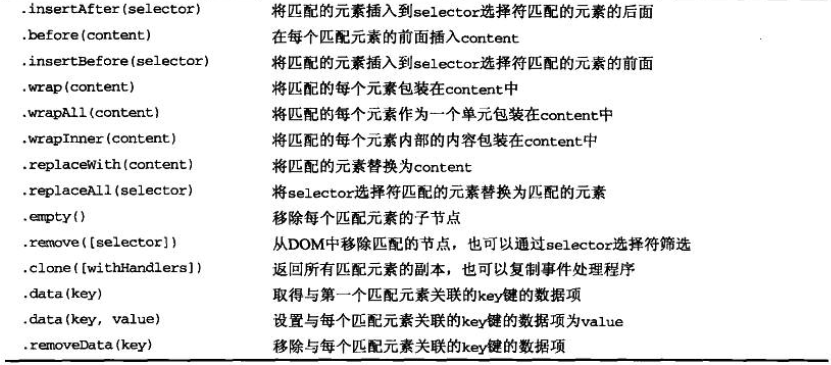


## DOM操作方法

|  |  |  |
| --- | --- | --- |
| 选择符 | 含义 | 备注 |
| .attr(key) | 获取属性key的值 | $('#text1').attr('value') |
| .attr(key,value) | 设置属性key的值 | $('#text1').attr('value','hello world!'); |
| .attr(key,fn) | 设置属性key的值为fn的返回结果 | $('tr').attr('value',**function**(){  **return** $(**this**).attr('id');  }); |
| .attr(map) | 根据传入的键值对参数设置属性的值 |  |
| .removeAttr(key) | 移除属性key |  |
| .addClass(class) | 为每个匹配的元素增加类 |  |
| .removeClass(class) | 为每个匹配的元素移除类 |  |
| .toggleClass(class) | 每个匹配的元素，如果类存在则删除，不存在则增加 |  |
| .hasClass(class) | 匹配的元素只要有一个类包含class，则返回true |  |
| .html() | 获得第一个匹配元素的html |  |
| .html(value) | 设置第一个匹配元素的html为value |  |
| .text() | 获取每个匹配元素的文本内容，返回一个字符串 |  |
| .text(value) | 设置每个匹配元素的文本内容 |  |
| .val() | 获得第一个匹配元素的value属性的值 |  |
| .val(value) | 设置第一个匹配元素的value属性的值为value |  |
| .css(key) | 获取css属性 |  |
| .css(key,value) | 设置css属性 |  |
| .offset() | 获得第一个匹配元素相对于视口的坐标（像素） |  |
| .position() | 获得第一个匹配元素相对于.offsetParent()返回元素的坐标（像素） |  |
| .scrollTop() | 获得第一个匹配元素的垂直滚动位置 |  |
| .scrollTop(value) | 设置每个匹配元素的垂直滚动位置 |  |
| .scrollLeft() | 获得第一个匹配元素的水平滚动位置 |  |
| .scrollLeft(value) | 设置每个匹配元素的水平滚动位置 |  |
| .height() | 获得第一个匹配元素的高度 |  |
| .height(value) | 设置每个匹配元素的高度 |  |
| .width() | 获得第一个匹配元素的宽度 |  |
| .width(value) | 设置每个匹配元素的宽度 |  |
| .innerHeight() | 获得第一个匹配元素包含内边距但不包含边框的高度 |  |
| .innerWidth() | 获得第一个匹配元素包含内边距但不包含边框的宽度 |  |
| .outerHeight(includeMargin) | 获得第一个匹配元素包含内边、边框及可选的外边框的高度 |  |
| .outerWidth(includeMargin) | 获得第一个匹配元素包含内边、边框及可选的外边框的宽度 |  |
| .append(content) | 在每个匹配元素内部的末尾插入content |  |
| .appendTo(selector) | 将匹配的元素插入到selector的元素内部的末尾 |  |
| .prepend(content) | 在每个匹配元素内部的开头插入content |  |
| .prependTo(selector) | 将匹配的元素插入到selector的元素内部的开头 |  |
| .after(content) | 将匹配的元素的后面插入content |  |
| .insertAfter(selector) | 将匹配的元素插入到selector的元素的后面 |  |
| .wrap(content) | 将匹配的元素封装到content内 |  |
| .wrapAll(content) |  |  |
| .wrapInner(content) | 将匹配的元素内部的内容封装到content中 |  |
| .replaceWith(content) | 将匹配的元素替换为content |  |
| .replaceAll(selector) | 将selector的元素替换为当前元素 |  |
| .empty() | 移除匹配元素的子节点 |  |
| .remove([selector]) | 从DOM中移除节点，也可根据selector |  |
| .clone([withHandles]) | 返回匹配元素的副本，也可复制事件处理 |  |
| .data(key) | 取得与第一个匹配元素关联的key键的数据项 |  |
| .data(key,value) | 设置与每个匹配元素关联的key键的数据项 |  |
| .removeData(key) | 移除与每个匹配元素关联的key键的数据项 |  |





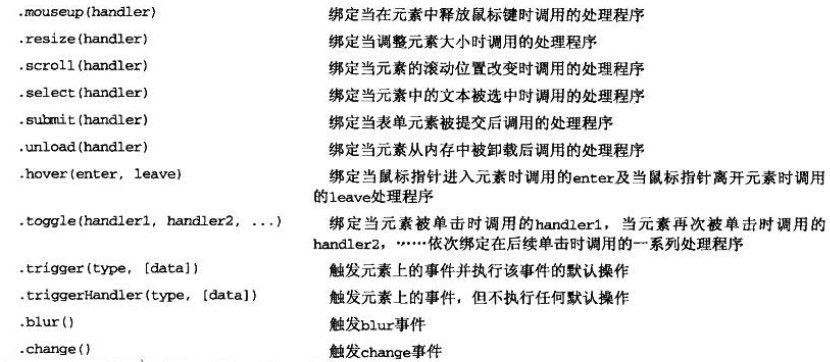


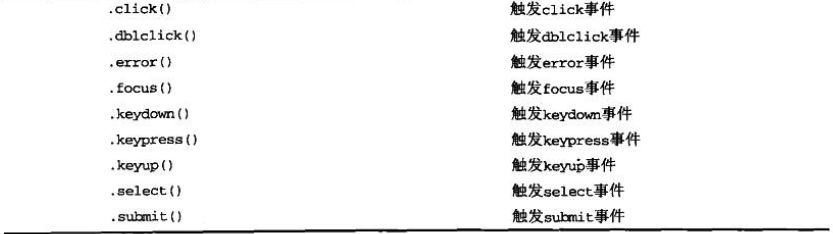
## 事件方法

|  |  |  |
| --- | --- | --- |
| 事件方法 | 含义 | 备注 |
| .ready(handler) | 绑定在dom和css完全加载后调用 |  |
| .bind(type, [data], handler) | 绑定给定的类型type |  |
| .one(type, [data], handler) | 绑定给定的类型type，且只执行一次 |  |
| .unbind([type] , [handler]) | 解除绑定在元素上的处理程序 |  |
| .live(type, handler) |  |  |
| .die(type, [handler]) |  |  |
| .toggle(handler1,handler2……) | 依次调用被绑定的程序 |  |
| .trigger(type,[data]) | 触发元素上的事件，并执行默认操作 |  |
| .triggerHandler(type,[data]) | 触发元素上的事件，但不执行默认操作 |  |
| Bind、trigger代替方法 | | |
| .blue(handler) | 失去焦点 | .blue() |
| .change(handler) | 值改变 | .change() |
| .click(handler) | 点击 | .click() |
| .dblclick(handler) | 双击 | .dblclick() |
| .error(handler) | 接收到错误事件 | .error() |
| .focus(handler) | 获得键盘焦点 | .focus() |
| .keydown(handler) | 拥有键盘焦点且有按键被按下时 | .keydown() |
| .keypress(handler) | 拥有键盘焦点且有按键事件发生时 | .keypress() |
| .keyup(handler) | 拥有键盘焦点且有按键被释放时 | .keyup() |
| .load(handler) | 加载完成时 |  |
| .mousedown(handler) | 按下鼠标时 |  |
| .mouseenter(handler) | 鼠标进入元素时，不受事件冒泡影响 |  |
| .mouseleave(handler) | 鼠标离开元素时，不受事件冒泡影响 |  |
| .mousemove(handler) | 鼠标移动时 |  |
| .mouseout(handler) | 鼠标进入元素时 |  |
| .mouseover(handler) | 鼠标离开元素时 |  |
| .mouseup(handler) | 释放鼠标键时 |  |
| .resize(handler) | 调整元素大小时 |  |
| .scorll(handler) | 滚动位置改变时 |  |
| .select(handler) | 文本被选中时 | .select() |
| .submit(handler) | 表单元素被提交时 | .submit() |
| .unload(handler) | 元素从内存被卸载后 |  |
| .hover(handler) |  |  |



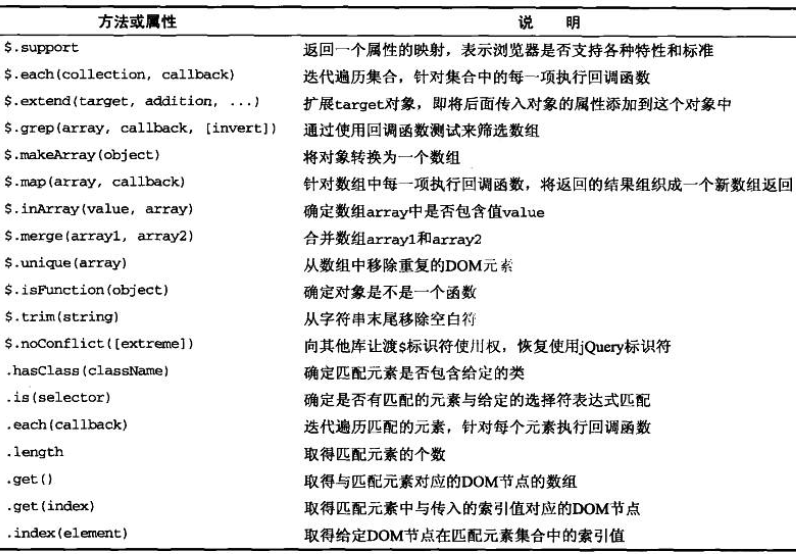






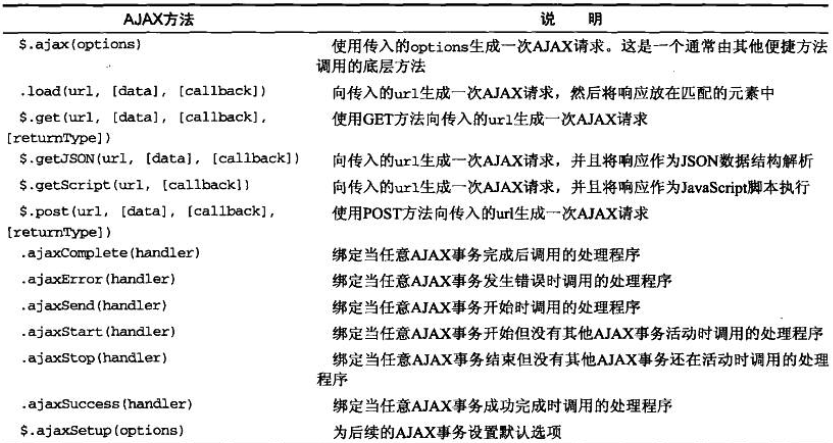
## 其它方法

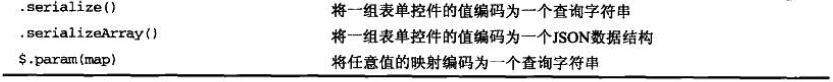
|  |  |  |
| --- | --- | --- |
| 方法或属性 | 说明 | 备注 |
| $.support | 返回一个属性的映射，表示浏览器是否支持各种特性和标准 |  |
| $.each(collection, callback) | 迭代遍历集合，针对集合中的每一项执行回调函数 |  |
| $.extend(target,addition, ……) | 扩展target对象 | Target,addition都是映射（map） |
| $.grep(array, callback, [invert]) | 通过使用回调函数测试来筛选数组 |  |
| $.makeArray(object) | 将对象转换为一个数组 |  |
| $.map(array, callback) | 针对数组中每一项执行回调函数，将返回的结果组织成一个新数据返回 |  |
| $.inArray(value, array) | 数组array中是否包含value |  |
| $.merge(array1,array2) | 合并数组array1和array2 |  |
| $.unique(array) | 从数组中移除重复的DOM元素 |  |
| $.isFunction(object) | 判断object是否为函数 |  |
| $.trim() | 删除字符串尾部空白符 |  |
| $.noConflict([extreme]) | 放弃$的使用权，改用jQuery |  |
| .hasClass(className) | 匹配元素是否包含给定类 |  |
| .is(selector) | 匹配给定元素是否与selector匹配 |  |
| .each(callback) | 遍历给定元素，每个元素都执行回调函数callback |  |
| .length | 匹配元素的个数 |  |
| .get() | 匹配元素对应dom节点的数组 |  |
| .get(index) | 取得给定元素对应索引的dom节点 |  |
| .index(element) | 确定element在给定元素中的索引位置 |  |



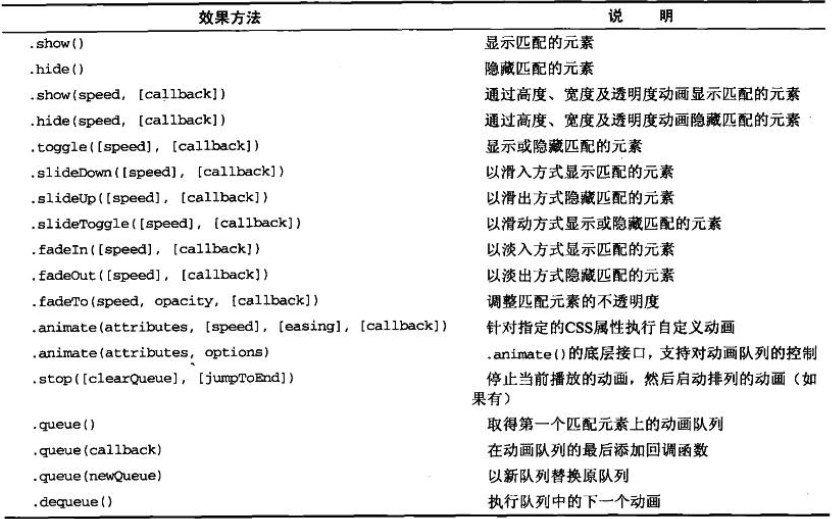
## Ajax方法

|  |  |  |
| --- | --- | --- |
| 方法 | 说明 | 备注 |
| $.ajax(options) | 使用传入的options生成一次ajax请求，这是一个通常由其他便捷方法调用的底层方法。如$.getJSON、$.getScript等。 | $.ajax({  url: ‘a.html’,  type: ’get’,  dataType: ’html’,  success: function(data){  $(‘div’).html(data);  }  }); |
| .load(url, [data], [callback]) | 向传入的url生成一次ajax请求，然后将响应放在当前的元素中 | $('#chapter').load('test1.html'); |
| $.get(url, [data], [callback], [returntype]) | 使用get方法向传入的url生成一次ajax请求 | $.get('test1.xml',**function**(data){  }); |
| $.getJSON(url, [data], [callback]) | 向传入的url生成一次ajax请求,并将响应作为JSON数据结构解析 | $.getJSON('test.js',**function**(data){  $('#chapter').text(data.key1);//key1不需要加引号  }); |
| $.getScript (url, [data], [callback]) | 向传入的url生成一次ajax请求,并将响应作为javascript脚本执行 | $.getScript('test3.js'); |
| $.post(url, [data], [callback], [returntype] ) | 使用post方法向传入的url生成一次ajax请求 | $.post('test4.jsp',**function**(data){  $('#text1').attr('value',data);  }); |
| .ajaxComplete(handler) | 绑定当任意ajax事务完成后调用的处理程序 |  |
| .ajaxError (handler) | 绑定当任意ajax事务发生错误时调用的处理程序 |  |
| .ajaxSend(handler) | 绑定当任意ajax事务开始时调用的处理程序 |  |
| .ajaxStart(handler) | 绑定当任意ajax事务开始但没有其他ajax事务活动时调用的处理程序 |  |
| .ajaxStop (handler) | 绑定当任意ajax事务结束但没有其他ajax事务活动时调用的处理程序 |  |
| .ajaxSuccess (handler) | 绑定当任意ajax事务成功完成时调用的处理程序 |  |
| $.ajaxSetup (options) | 为后续ajax事务设置默认选项 |  |
| .serialize() | 将一组表单控制的值编码为一个查询字符串 | $('#form1').serialize()  结果：  name1=value1&name2=value2 |
| .serializeArray() | 将一组表单控制的值编码为一个JSON数据结构 |  |
| $.param(map) | 将任意的映射编码为一个查询字符串 |  |





## DOM效果方法



# Jquery-ui

## Js,css,jquery



## 手风琴

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div id=*"accordion"*>  <h3>First</h3>  <div>Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.</div>  <h3>Second</h3>  <div>Phasellus mattis tincidunt nibh.</div>  <h3>Third</h3>  <div>Nam dui erat, auctor a, dignissim quis.</div>  </div>  <script>  $( "#accordion" ).accordion();  </script>  </body>  </html> |

## 自动填充

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div>  <input id=*"autocomplete"* title=*"type &quot;a&quot;"*>  </div>  <script>  **var** availableTags = [  "ActionScript",  "AppleScript",  "Asp",  "BASIC",  "C",  "C++",  "Clojure",  "COBOL",  "ColdFusion",  "Erlang",  "Fortran",  "Groovy",  "Haskell",  "Java",  "JavaScript",  "Lisp",  "Perl",  "PHP",  "Python",  "Ruby",  "Scala",  "Scheme"  ];  $( "#autocomplete" ).autocomplete({  source: availableTags  });  </script>  </body>  </html> |

## 按钮和单选按钮

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <button id=*"button"*>A button element</button>  <form style="margin-top: *1em*;">  <div id=*"radioset"*>  <input type=*"radio"* id=*"radio1"* name=*"radio"*><label for=*"radio1"*>Choice 1</label>  <input type=*"radio"* id=*"radio2"* name=*"radio"* checked=*"checked"*><label for=*"radio2"*>Choice 2</label>  <input type=*"radio"* id=*"radio3"* name=*"radio"*><label for=*"radio3"*>Choice 3</label>  </div>  </form>  <script>  $( "#button" ).button();  $( "#radioset" ).buttonset();  </script>  </body>  </html> |

## Tabs

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div id=*"tabs"*>  <ul>  <li><a href=*"#tabs-1"*>First</a></li>  <li><a href=*"#tabs-2"*>Second</a></li>  <li><a href=*"#tabs-3"*>Third</a></li>  </ul>  <div id=*"tabs-1"*>1111</div>  <div id=*"tabs-2"*>2222</div>  <div id=*"tabs-3"*>3333</div>  </div>  <script>  $( "#tabs" ).tabs();  </script>  </body>  </html> |

## 对话框

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <a href=*"#"* id=*"dialog-link"* class=*"ui-state-default ui-corner-all"*> Open Dialog</a>  <div id=*"dialog"* title=*"Dialog Title"*>  <p>你还好吗？</p>  </div>  <script>  $( "#dialog" ).dialog({  autoOpen: **false**,  width: 400,  buttons: [  {  text: "Ok",  click: **function**() {  $( **this** ).dialog( "close" );  }  },  {  text: "Cancel",  click: **function**() {  $( **this** ).dialog( "close" );  }  }  ]  });  $( "#dialog-link" ).click(**function**( event ) {  $( "#dialog" ).dialog( "open" );  event.preventDefault();  });  </script>  </body>  </html> |

## 滑块（slider）

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div id=*"slider"*></div>  <script>  $( "#slider" ).slider({  range: **true**,  values: [ 17, 67 ]  });  </script>  </body>  </html> |

## 进度条（progressbar）

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div id=*"progressbar"*></div>  <script>  $( "#progressbar" ).progressbar({  value: 20  });  </script>  </body>  </html> |

## 日期选择（Datepicker）

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <div id=*"datepicker"*></div>  <script>  $( "#datepicker" ).datepicker({  inline: **true**  });  </script>  </body>  </html> |

## 下拉框

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <select id=*"selectmenu"*>  <option>Slower</option>  <option>Slow</option>  <option selected=*"selected"*>Medium</option>  <option>Fast</option>  <option>Faster</option>  </select>  <script>  $( "#selectmenu" ).selectmenu();  </script>  </body>  </html> |

## 菜单

|  |
| --- |
| <html>  <head>  <link href=*"jquery-ui.css"* rel=*"stylesheet"*>  <script src=*"external/jquery/jquery.js"*></script>  <script src=*"jquery-ui.js"*></script>  <style>  **body**{  font: *62.5% "Trebuchet MS", sans-serif*;  margin: *50px*;  }  </style>  </head>  <body>  <ul style="width:*100px*;" id=*"menu"*>  <li>Item 1</li>  <li>Item 2</li>  <li>Item 3  <ul>  <li>Item 3-1</li>  <li>Item 3-2</li>  <li>Item 3-3</li>  <li>Item 3-4</li>  <li>Item 3-5</li>  </ul>  </li>  <li>Item 4</li>  <li>Item 5</li>  </ul>  <script>  $( "#menu" ).menu();  </script>  </body>  </html> |

# Jquery - dataTables

## Js,css引入

|  |
| --- |
| <script src=*'/study/js/jquery/dataTable/js/jquery.js'*></script>  <script src=*'/study/js/jquery/dataTable/js/jquery.dataTables.min.js'*></script>  <link rel=*"stylesheet"* type=*"text/css"* href=*'/study/js/jquery/dataTable/css/jquery.dataTables.min.css'*> |

## 0配置

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  <script src=*'/study/js/jquery/dataTable/js/jquery.js'*></script>  <script src=*'/study/js/jquery/dataTable/js/jquery.dataTables.min.js'*></script>  <link rel=*"stylesheet"* type=*"text/css"*  href=*'/study/js/jquery/dataTable/css/jquery.dataTables.min.css'*>  </head>  <body>  <table id=*"example"* class=*"display"* cellspacing=*"0"* width=*"100%"*>  <thead>  <tr>  <th>Name</th>  <th>Position</th>  <th>Office</th>  <th>Age</th>  <th>Start date</th>  <th>Salary</th>  </tr>  </thead>  <tfoot>  <tr>  <th>Name</th>  <th>Position</th>  <th>Office</th>  <th>Age</th>  <th>Start date</th>  <th>Salary</th>  </tr>  </tfoot>  <tbody>  <tr>  <td>Tiger Nixon</td>  <td>System Architect</td>  <td>Edinburgh</td>  <td>61</td>  <td>2011/04/25</td>  <td>$320,800</td>  </tr>  <tr>  <td>Garrett Winters</td>  <td>Accountant</td>  <td>Tokyo</td>  <td>63</td>  <td>2011/07/25</td>  <td>$170,750</td>  </tr>  <tr>  <td>Ashton Cox</td>  <td>Junior Technical Author</td>  <td>San Francisco</td>  <td>66</td>  <td>2009/01/12</td>  <td>$86,000</td>  </tr>  </tbody>  </table>  </body>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $('#example').DataTable();  });  </script>  </html> |

## 属性配置

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 含义 | 说明 | 例子 |
| bLengthChange | 开关，是否显示一个每页长度的选择条（需要分页器支持） | 如果20,50,100 |  |
| aLengthMenu | 选择每页的条目数 |  | default [10, 25, 50, 100]，可以为一维数组 |
| bPaginate | 是否显示分页功能 |  |  |
| bProcessing | 载入数据时，是否显示‘进度’提示 |  |  |
| bInfo | 是否显示有关table中现有数据的信息，执行过滤数据操作(filtered data)时显示对应信息 | 如：  Showing 1 to 10 of 57 entries |  |
| bAutoWidth | 是否启动自动计算各个字段(td单元格)所在宽度 |  |  |
| bScrollInfinite | 是否启动初始化滚动条 | 通常和sScrollY结合使用。不能和pagination分页功能并存，分页了数据就不会过多了(垂直方向） | $(document).ready( function() {   $('#example').dataTable( {     "bScrollInfinite": true,     "bScrollCollapse": true,     "sScrollY": "200px"   } ); } ); |
| bServerSide | 是否启动服务器端数据导入 | 要和sAjaxSource结合使用 | $(document).ready( function () {  $('#example').dataTable( {  "bServerSide": true,  "sAjaxSource": "xhr.php"  } );  } ); |
| bSort | 是否启动各个字段的排序功能 | bSortable可对应启动单个字段的排序功能 |  |
| aaSorting | 某列的排序方式，按index确定 | [[0, "asc"]] |  |
| bStateSave | 是否启动状态保留 | 开启cookie记住table信息如分页信息，数据长度，filtering过滤和sorting过滤等信息。 |  |
| sScrollX | 字段过多，启动横向滚动条 | '100%' 类似的字符串 | $(document).ready( function() {  $('#example').dataTable( {  "sScrollX": "100%",  "bScrollCollapse": true  } );  } ); |
| sScrollY | 数据过多，使用垂直滚动条 | '200px' 类似的字符串 | $(document).ready( function() {  $('#example').dataTable( {  "sScrollY": "200px",  "bPaginate": false  } );  } ); |
| bFilter | 是否启用客户端过滤功能 |  |  |
| iDisplayLength | 一个分页显示多少行数据 |  |  |
| iDisplayStart | 从第几行开始显示数据 |  |  |
| sPaginationType | 分页方式 | 内建方法有two\_button和full\_numbers | 这类方法是可以自定义的，如：自己写个listbox方法，提供一个下拉框来选择显示当前一个table中显示多少数据，  $(document).ready(function() {  $('#example').dataTable( {  "sPaginationType": "listbox"  } );  } ); |
| aoColumnDefs | 隐藏第几列 |  | [{ "bVisible": false, "aTargets": [0]}] |
| oLanguage | 多语言设置 |  | "oLanguage": {  "sProcessing": "正在加载中......",  "sLengthMenu": "每页显示 \_MENU\_ 条记录",  "sZeroRecords": "对不起，查询不到相关数据！",  "sEmptyTable": "表中无数据存在！",  "sInfo": "当前显示 \_START\_ 到 \_END\_ 条，共 \_TOTAL\_ 条记录",  "sInfoFiltered": "数据表中共为 \_MAX\_ 条记录",  "sSearch": "搜索",  "oPaginate": {  "sFirst": "首页",  "sPrevious": "上一页",  "sNext": "下一页",  "sLast": "末页"  }  } |
| aoSearchCols | 给每个列单独定义其初始化搜索列表特性 |  |  |
| oSearch | 默认{ "sSearch": "", "bRegex": false, "bSmart": true } | 初始时指定搜索参数相关 |  |
| asStripClasses | 指定要被应用到各行的class风格，会自动循环 | default ['odd', 'even'], 比如['strip1', 'strip2', 'strip3'] |  |
| bDestroy |  | 用于当要在同一个元素（如：div）上执行新的dataTable绑定时，将之前的那个数据对象清除掉，换以新的对象设置 |  |
| bScrollCollapse | 指定适当的时候缩起滚动视图 |  |  |
| iDeferLoading |  | 延迟加载，它的参数为要加载条目的数目，通常与bServerSide，sAjaxSource等配合使用 |  |
| iDisplayLength |  | 用于指定一屏显示的条数，需开启分页器 |  |
| iDisplayStart |  | 用于指定从哪一条数据开始显示到表格中去 |  |
| iScrollLoadGap |  | 用于指定当DataTable设置为滚动时，最多可以一屏显示多少条数据 |  |
| sAjaxDataProp | 字符串，default 'aaData' | 指定当从服务端获取表格数据时，数据项使用的名字 |  |
| sAjaxSource | URL字符串，default null | 指定要从哪个URL获取数据 |  |
| sCookiePrefix | 字符串，default 'SpryMedia\_DataTables' | 当打开状态存储特性后，用于指定存储在cookies中的字符串的前缀名字 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## dataTables官方demo索引



**注：由于demo中用到了ajax，所有要想完整展示，需要把此内容部署到web项目下。**

### 基本配置

|  |  |  |
| --- | --- | --- |
| Basic initialisation | 基本初始化 | 说明 |
| Zero configuration | 零配置 | 入门介绍 |
| Feature enable / disable | 基本属性的使用 | 以下三个属性的配置  paging  ordering  info |
| Default ordering (sorting) | 排序（只对已经加载的数据） | 页面打开时默认使用排序列 |
| Multi-column ordering | 多列排序（只对已经加载的数据） |  |
| Multiple tables | 按class初始数据 |  |
| Hidden columns | 隐藏列 |  |
| Complex headers (rowspan and colspan) | 多级表头 |  |
| DOM positioning | 改变table各元素中的位置 | Page,filter |
| Flexible table width | 表的宽度 |  |
| State saving | 保存页面各元素的状态 |  |
| Alternative pagination | 设置翻页的属性（只对已经加载的数据） |  |
| Scroll - vertical | 垂直方向的滚动条 |  |
| Scroll - horizontal | 水平方向的滚动条 |  |
| Scroll - horizontal and vertical | 垂直、水平方向的滚动条 |  |
| ~~Scroll - vertical with jQuery UI ThemeRoller~~ | ~~使用jquery ui 主题的垂直滚动~~ | ~~基本不用~~ |
| Language - Comma decimal place | 千分位，小数位分隔符 |  |
| Language options | 显示信息国际化 |  |

### 高级设置

|  |  |  |
| --- | --- | --- |
| Advanced initialisation | 高级初始化 | 说明 |
| DOM / jQuery events | 给dom上增加jquery事件 | 非常有用 |
| DataTables events | Table分页、查询、排序的事件 | 非常有用 |
| Column rendering | 自定义列的显示内容 | 非常有用 |
| Page length options | 每页显示条数的配置 |  |
| Multiple table control elements | 增加多个控制元素 | 如头部、脚部显示分页功能 |
| Complex headers (rowspan / colspan) |  |  |
| Read HTML to data objects |  |  |
| HTML5 data-\* attributes - cell data |  |  |
| HTML5 data-\* attributes - table options |  |  |
| Language file |  |  |
| Setting defaults |  |  |
| Row created callback | 行回调，可以根据行内不同的数据，显示行样式 |  |
| Row grouping | 按某列进行分组显示 |  |
| Footer callback | 页角回调函数 |  |
| Custom toolbar elements | 自定义工具条元素 |  |
| Order direction sequence control | Sequence排序 |  |

### API

|  |  |  |
| --- | --- | --- |
| API | 解释 | 说明 |
| Add rows | 增加行 |  |
| Individual column searching (text inputs) | 每列独立查询输入框 |  |
| Individual column searching (select inputs) | 每列独立查询下拉框 |  |
| Highlighting rows and columns | 高亮鼠标所在行、列 |  |
| Child rows (show extra / detailed information) | 行的父子关系 |  |
| Row selection (multiple rows) | 多行选择 |  |
| Row selection and deletion (single row) | 单行选择 |  |
| Form inputs | 信息输入 |  |
| Index column | 行号 |  |
| Show / hide columns dynamically | 动态显示、隐藏列 |  |
| Using API in callbacks | 回调函数 |  |
| Scrolling and jQuery UI tabs |  |  |
| Search API (regular expressions) | 高级查询 |  |

### 样式设置

|  |  |  |
| --- | --- | --- |
| Styling | 样式 | 说明 |
| Base style | 基本样式 |  |
| Base style - no styling classes | 无样式 |  |
| Base style - cell borders | 单元格边框 |  |
| Base style - compact | 行之间紧凑 |  |
| Base style - hover | 鼠标所在行高亮 |  |
| Base style - order-column |  |  |
| Base style - row borders | 行之间边框 |  |
| Base style - stripe |  |  |

### ajax

|  |  |  |
| --- | --- | --- |
| Ajax |  |  |
| Ajax data source (arrays) | Json数组 |  |
| Ajax data source (objects) | Json对象 |  |
| Nested object data (objects) | 嵌套对象 |  |
| Nested object data (arrays) | 嵌套数据 |  |
| Orthogonal data |  |  |
| **Generated content for a column** | **生成某列内容** | **如：生成一个按钮** |
| Custom data source property | 指定数据源是json中的哪个字段 |  |
| Flat array data source |  |  |
| Deferred rendering for speed |  |  |

### 扩展

|  |  |  |
| --- | --- | --- |
|  |  |  |
| AutoFill | 像excel中自动填充 |  |
| ColReorder | 拖拽列的显示顺序 |  |
| ColVis | 控制列是否显示 |  |
| FixedColumns | 固定列 | 当列数比较多时，首列一直显示 |
| FixedHeader | 表头、表尾固定显示 |  |
| KeyTable | 使用键盘控制单元格的焦点位置 |  |
| Scroller | XY轴的滚动条 |  |
|  |  |  |
|  |  |  |

## dataTable中的回调函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 回调函数名称 | 参数 | 返回值 | 默认 | 功能 |
| fnCookieCallback | 1.string: Name of the cookie defined by DataTables 2.object: Data to be stored in the cookie 3.string: Cookie expires string 4.string: Path of the cookie to set | string: cookie formatted string (which should be encoded by using encodeURIComponent()) | null | 当每次cookies改变时，会触发这个函数调用 |
| fnDrawCallback | 无 | 无 | 无 | 在每次table被draw完后调用，至于做什么就看着办吧 |
| fnFooterCallback | 1.node : "TR" element for the footer 2.array array strings : Full table data (as derived from the original HTML) 3.int : Index for the current display starting point in the display array< 4.int : Index for the current display ending point in the display array 5.array int : Index array to translate the visual position to the full data array | 无 | 无 | 用于在每次重画的时候修改表格的脚部 |
| fnFormatNumber | 1.int : number to be formatted | String : formatted string for DataTables to show the number | 有默认的 | 用于在大数字上，自动加入一些逗号，分隔开 |
| fnHeaderCallback | 1.node : "TR" element for the header 2.array array strings : Full table data (as derived from the original HTML) 3.int : Index for the current display starting point in the display array 4.int : Index for the current display ending point in the display array 5.array int : Index array to translate the visual position to the full data array | 无 | 无 | 用于在每次draw发生时，修改table的header |
| fnInfoCallback | 1.object: DataTables settings object 2.int: Starting position in data for the draw 3.int: End position in data for the draw 4.int: Total number of rows in the table (regardless of filtering) 5.int: Total number of rows in the data set, after filtering 6.string: The string that DataTables has formatted using it's own rules | string: The string to be displayed in the information element. | 无 | 用于传达table信息 |
| fnInitComplete | 1.object:oSettings - DataTables settings object | 无 | 无 | 表格初始化完成后调用 |
| fnPreDrawCallback | 1.object:oSettings - DataTables settings object | Boolean | 无 | 用于在开始绘制之前调用，返回false的话，会阻止draw事件发生；返回其它值，draw可以顺利执行 |
| fnRowCallback | 1.node : "TR" element for the current row 2.array strings : Raw data array for this row (as derived from the original HTML) 3.int : The display index for the current table draw 4.int : The index of the data in the full list of rows (after filtering) | node : "TR" element for the current row | 无 | 当创建了行，但还未绘制到屏幕上的时候调用，通常用于改变行的class风格 |
| fnServerData | 1.string: HTTP source to obtain the data from (i.e. sAjaxSource) 2.array objects: A key/value pair object containing the data to send to the server 3.function: Function to be called on completion of the data get process that will draw the data on the page. | void | $.getJSON | 用于替换默认发到服务端的请求操作 |
| fnStateLoadCallback | 1.object:oSettings - DataTables settings object 2.object:oData - Object containing information retrieved from the state saving cookie which should be restored. For the exact properties please refer to the DataTables code. | Boolean - false if the state should not be loaded, true otherwise | 无 | 在cookies中的数据被加载前执行，可以方便地修改这些数据 |
| fnStateSaveCallback | 1.object:oSettings - DataTables settings object 2.String:sValue - a JSON string (without the final closing brace) which should be stored in the state saving cookie. | String - the full string that should be used to save the state | 无 | 在状态数据被存储到cookies前执行，可以方便地做一些预操作 |