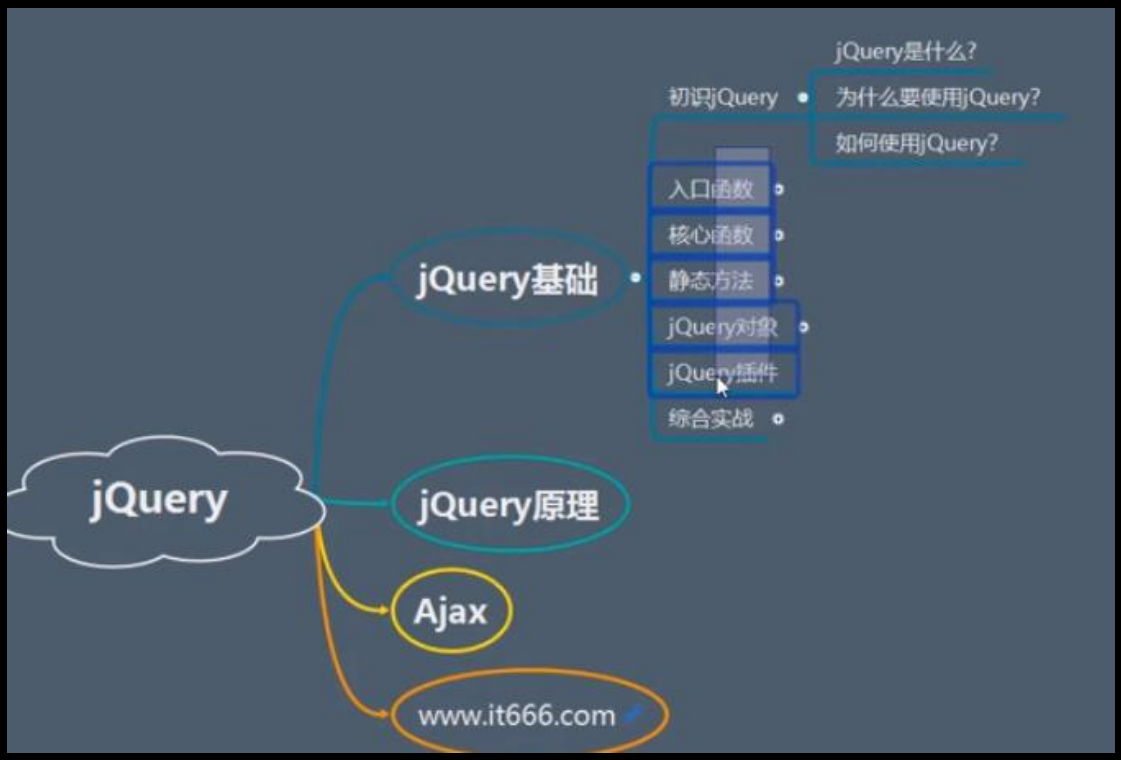
JQuery学习笔记

网易云课堂—李江南

概述



JQuery：优秀的 JS 库，js+Query，所以最重要的用途是查询，其他还能做很多简化工作、浏览器兼容。

版本：

- 1.x：兼容ie678，但相对其它版本文件较大，官方只做BUG维护，功能不再新增，最终版本：1.12.4 (2016年5月20日).
- 2.x：不兼容ie678，相对1.x文件较小，官方只做BUG维护，功能不再新增，最终版本：2.2.4 (2016年5月20日)
- 3.x：不兼容ie678，只支持最新的浏览器，很多老的jQuery插件不支持这个版本，相对1.x文件较小，提供不包含Ajax/动画API版本。

大公司都用 1.x

每个版本有压缩版和原生版，压缩版格式压缩，没有换行，单词也不完整，不易阅读。

开发的时候使用未压缩的，上线的时候使用压缩的。

JQuery 使用

1.引入 js 包：

在 head 中引入：

```
<script src="js/jquery-1.12.4.js"></script>
```

2.onload 写法

```
<head>
    <meta charset="UTF-8">
    <title>02-jQuery-HelloWorld</title>
    <script src="js/jquery-1.12.4.js"></script>
    <script>
        // 1.原生JS的固定写法
        window.onload = function (ev) { }

        // 2.jQuery的固定写法
        $(document).ready(function () {
            alert();
        });
    </script>
</head>
```

如果；两种同时使用，JQ 优先级高一点，先显示。

入口区别

加载模式不同

- 原生 js 会等到 DOM 元素加载完毕，并且图片也加载完毕才会执行
- JQuery 会等到 DOM 元素加载完毕，但是不会等图片加载完毕就会执行。

优先级

原生的：后面写的入口函数会覆盖之前的

JQ：后写的不会覆盖前面的

入口函数 4 种写法

```
// 1.第一种写法
$(document).ready(function () {
    // alert("hello lnj");
});

// 2.第二种写法
jQuery(document).ready(function () {
    alert("hello lnj");
});
```

```
// 3.第三种写法
$(function () {
    alert("hello lnj");
});

// 4.第四种写法
jQuery(function () {
    alert("hello lnj");
});
```

推荐写法

冲突解决

```
// 1.释放$的使用权
// 注意点：释放操作必须在编写其它jQuery代码之前编写
//        释放之后就不能再使用$,改为使用jQuery
jQuery.noConflict();
jQuery(function () {
    alert("hello lnj");
});
```

不过不想使用 JQuery 这个单词——自定义：

```
// 2.自定义一个访问符号
var nj = jQuery.noConflict();
nj(function () {
    alert("hello lnj");
});
```

核心函数

$(): 就表示调用 JQ 的核心函数

- 接受一个函数

```
$(function () {
    alert("hello lnj");
```

- 接受一个字符串
  - 接受一个字符串选择器：返回 JQ 对象，该对象保存了 DOM 元素

```
var $box1 = $(".box1");
var $box2 = $("#box2");
```

  - 接受一个字符串代码片段：返回 JQ 对象

```
var $p = $("<p>我是段落</p>");
console.log($p);
$box1.append($p);
```

- 接受一个 DOM 元素

```
// 3.接收一个DOM元素
// 会被包装成一个jQuery对象返回给我们
var span = document.getElementsByTagName("span")[0];
console.log(span);
var $span = $(span);
console.log($span);
```

JQuery 对象

JQ 对象：是一个伪数组，有 length 属性。其中还有许多的属性。

伪数组：有 0----length-1 的

比如：

```
var obj = {0:1, 1:3, 2:5, 3:7, 4:9, length:5};
```

注意：

- 普通数组使用 []
- 伪数组使用：{key:value… , length:len}

匿名函数 var a =function(){}和 function a(){}的区别

```
//代码一：

a(1);          //执行这个会报错
var a  =  function(index){
        alert(index);
}
a(2);          //执行这个不会报错



//代码二：
a(1);          //执行这个不会报错
function a(index){
        alert(index);
}
a(2);          //执行这个不会报错
```

JS:函数和变量声明的提前行为，匿名函数只有在被调用时才初始化。

而非匿名函数在加载时就被初始化了。

静态方法 V 实例方法

静态方法：通过类名调用

实例方法：通过对象实例调用

```
function func(){}
func.staticMethod = function () {
    alert("我是定义的静态方法！");
};
func.prototype.instanceMethod = function () {
    alert("我是定义的实例方法！")
};
func.staticMethod();
new func().instanceMethod();
```

静态 foreach 方法

原生

```
var arr = [1,2,3,4,5,6,7,8,9];
arr.forEach(function (value, index) {
    document.writeln(index,value);
})
// 注意：原生的foreach只能遍历数组，不能遍历伪数组。
```

注意：forEach 方法三个参数：value, index, array

但是也可以省略个别。

JQuery

语法：

$.each(数组/伪数组，函数{ 执行的方法 });

```
$.each(arr,function (index,value) {
    document.writeln(index,value);
})
```

注意：

- each 的返回值：遍历谁就返回谁。
- 不支持在回调函数中对数组进行操作!!!

静态 Map 方法

原生：不能遍历伪数组

```
var a =[1,2,3,4,5];
var res = a.map(function (value, index, array)
    return array[index]*10;
});
document.write(res);
```

JQuery

$.map(para1，函数 func(value ，index){})

Para1：**数组/伪数组(和 foreach 一样)**

Func：执行每一个函数后的回调函数（callbackfn）

返回值：默认空数组，可以在 func 中设置生效

静态 trim

Var res = $.trim(str)

去除首位空格，返回字符串

静态 isWindow

判断一个对象是否是一个 window 对象

$.iswindow(obj)

返回 true、false

静态 isArray

$.isArray(obj)

判断一个数组是一个真数组

返回 bool

静态 isFuncation

$.isFunction(obj)

判断是否是函数

返回 bool

注意：$.isFuncation(JQuery) 返回 true 说明 JQuery 本质是一个函数

静态 holdReady

$.holdReady(true) 暂停 ready 方法的执行，即暂停入口函数的执行。

$.holdReady(false) 恢复 ready 方法的执行，即恢复入口函数执行

用处：

用在有时需要先执行一些其他的代码再执行入口函数中的一些函数，即**调整执行先后顺序**。

WebStorm 初始化模板

File—setting—editor—Live Templates—HTML--+-- 填写模板

—abbreviation 添加快捷键—define—all -apply

内容选择器

: empty

:empty 表示一个标签中是空的，里面没有内容，没有子元素

```
var $emptyDiv = $("div:empty");
```

: parent

找到有文本或子元素的指定元素（做父亲的）

```
var $parentDic = $("div:parent");
```

:contains(text)

找到包含特定文本元素的元素

```
var $contentDiv = $("div:contains('XX')");
```

注意：是包含，text 包含特定字符串的即可

: has(son)

找出包含特定子元素的元素

```
var $Div_has_son = $("div:has('span')");
```

层次选择器

派生选择器

在 S1 内部获得全部的 S2 节点（不考虑层次）

```
$("div  span")// 派生选择器
<div>
<span></span>//找到
<p>
<span></span>//找到
</p>
</div>
<span></span>//找不到
```

直接子元素选择器

$("S1 > S2")

获取子元素

```
$("div > span")
<div>
<span></span>//找到
<p>
<span></span>
</p>
<span></span>//找到
</div>
<span></span>
```

直接兄弟选择器

$(S1+S2)

S1 后边获得紧紧挨着的第一个*兄弟关系*的 S2

<S1></S1>

<S2></S2> //get it

全部兄弟选择器

$(S1~S2)

获得 **S1 后面的全部兄弟 S2**

S1 S2 S1>S2 S1+S2 S1~S2

并且选择器

:first
用法: $("tr:first") ; 单个元素的组成的集合
匹配找到的第一个元素
:last
匹配找到的最后一个元素
:not(selector)
去除所有与给定选择器匹配的元素
:even
匹配所有索引值为偶数的元素，从 0 开始计数
:odd
匹配所有索引值为奇数的元素，从 0 开始计数
:eq(index)
匹配一个给定索引值的元素，从 0 开始计数
:gt(index)
匹配所有大于给定索引值的元素，从 0 开始计数
:lt(index)
匹配所有小于给定索引值的元素，从 0 开始计数
:header
匹配如 h1, h2, h3 之类的标题元素

（这些应该是同 JavaScript 的）

序选择器

修改同一标签中的某个标签

同级别

first-child:对同级别中的第一个标签进行修改;

nth-child(n):对同级别中第n个标签进行修改;

last-child:对同级别中最后一个标签进行修改;

only-child:对同级别中的唯一标签进行修改。

```
<style>
    p:first-child {
        color: red;
    }

    p:nth-child(2) {
        font-family: "华文楷体";
    }

    p:last-child {
        font-size: 30px;
    }
</style>
```

同类型

first-of-type:修改同类型标签中的第一个标签；

last-of-type:修改同类型标签中的最后一个标签；

nth-of-type(n):修改同类型标签中的第n个标签；

nth-last-of-type(n):修改同类型标签中倒数第n个标签；

only-of-type:修改同类型标签中的唯一标签。

```
<style>
    p:first-of-type {
        color: red;
    }

    p:last-of-type {
        font-family: 华文楷体;
    }

    p:nth-of-type(2) {
        font-size: 20px;
    }
</style>
```

nth-child(odd):修改同级别奇数标签；

nth-child(even):修改同级别偶数标签；

nth-child(xn+y):输入任意的x与y，修改任意行的标签(n从0到标签最大值)；

nth-of-type(odd):修改同类型奇数标签；

nth-of-type(even):修改同类型偶数标签；

nth-of-type(xn+y):输入任意的x与y，修改任意行的标签(n从0到标签最大值)；

选择器大全

选择器

基本
- #id
- element
- .class
- *
- selector1,selector2,selectorN

层级
- ancestor descendant
- parent > child
- prev + next
- prev ~ siblings

基本
- :first
- :not(selector)
- :even
- :odd
- :eq(index)
- :gt(index)
- :lang[1.9+]
- :last
- :lt(index)
- :header
- :animated
- :focus
- :root[1.9+]
- :target[1.9+]

内容
- :contains(text)
- :empty
- :has(selector)
- :parent

可见性
- :hidden
- :visible

属性
- [attribute]
- [attribute=value]
- [attribute!=value]
- [attribute^=value]
- [attribute$=value]
- [attribute*=value]
- [attrSel1][attrSel2][attrSelN]

子元素
- :first-child
- :first-of-type[1.9+]
- :last-child
- :last-of-type[1.9+]
- :nth-child
- :nth-last-child()[1.9+]
- :nth-last-of-type()[1.9+]
- :nth-of-type()[1.9+]
- :only-child
- :only-of-type[1.9+]

表单
- :input
- :text
- :password
- :radio
- :checkbox
- :submit
- :image
- :reset
- :button
- :file
- :hidden

表单对象属性
- :enabled
- :disabled
- :checked
- :selected

```
ul>li:nth-child(-n+3) span{
    background: deeppink;
}
```

牛逼，前三

属性　属性节点

属性：对象保存的变量，代表其性质。

获取属性：

1.a.b

2.a[b]

属性节点：在 HTML 标签中添加的属性，保存在 Attributes 中。

注意：任何对象都有属性，但是只有 DOM 元素才有属性节点。

属性节点设置：

```
Dom.setAttributes(key,value);
Dom.getAttributes(key);
```

Attr()

作用：获取/设置属性节点的值

参数：

- 一个参数：代表获取

```
($("span").attr("class")
```

注意：无论获得了多少元素，都**只会返回第一个元素**的属性节点值。

- 二个参数：设置

```
$("span").attr("class", "box");
```

Obj.attr(attribute, value)

注意：

- 找到多少元素，就会设置多少元素。

- 如果设置的属性不存在，会**自动新增**一个属性。

RemoveAttr()

移除元素的属性节点。

```
$("span").removeAttr("class");
```

注意：

- 会删除找到的所有元素的属性节点
- 如果需要同时删除指定的元素的指定节点，使用空格隔开多个元素

```
$("span").removeAttr("class name");
```

Prop() and removeProp()

.eq(n)方法：表示获取一个元素序列的第几个，一般用于找到的元素集合中获取某个。

```
$("span").eq(0)
$("span").eq(1)
```

Prop 方法和 attr 方法大同小异！

Prop(attr, value)：设置属性节点

Prop(attr):获取属性节点，只会使用第一个

RemoveProp（attr)：移除所有属性节点

注意：

- prop 不仅能操作属性，还能够操作属性节点。
- 操作属性节点时，具有 *true、false 两个属性的属性节点*如

checked, select, disable 使用 prop()，其他的使用 attr().

类操作方法

AddClass(calss | fn)

添加一个类，如果添加多个，使用空格隔开

```
// $("div").addClass("class1");
$("div").addClass("class1 class2");
```

RemoveClass()

删除一个类，多个使用空格隔开

```
// $("div").removeClass("class2");
$("div").removeClass("class2 class1");
```

ToggleClass（）

切换类：*有就删除，没有就添加*

文本值操作方法

Html(var | fn)

和原生 JS 中的 innerHTML 一样，添加一个元素的内部 HTML:

```
$("div").html("<p>我是段落<span>我是span</span></p>");
```

Text()

和原生 JS 中的 innerText 一样，添加一个元素的内部文本

```
$("div").text("<p>我是段落<span>我是span</span></p>");
```

注意：使用 text()，只是用 text 是不会起作用的。

Val()

- Obj.val(context): 设置 value
- Obj.val(): 获取 value

CSS 样式操作方法

.css(k,v)

- 逐个:

  Obj.css("attr", "value")

```
$("div").css("width", "100px");
$("div").css("height", "100px");
$("div").css("background", "red");
```

- 链式:

```
$("div").css("width", "100px").css("height", "100px").css("background", "blue");
```

  为了不影响可读性，一般不超过 3 个连续链式

- 批量设置: 推荐使用

  Obj.css({ k:v, k1:v1, k2:v2……})

```
$("div").css({
    width: "100px",
    height: "100px",
    background: "red"
});
```

.css(attr)

获取 CSS 样式

```
console.log($("div").css("background"));
```

CSS 尺寸 位置操作

Width() height()

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

无参数：获取

```
console.log($(".father").width());
```

有参数：设置

```
$(".father").width("500px")
```

innerHeight() innerWidth()

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

Offset()

- 获取元素宽口（浏览器窗口）的偏移位

```
$(".a").offset()
```

返回的是一个对象

若要获取四周的值：

```
$(".a").offset().left;
```

或者直接使用：

```
offsetX (jQuery)
offsetY (jQuery)
```

添加参数，即可设置。

Position()

获取元素距离**定位元素**的距离

```
$(".son").position({
    left: 10
});
```

（有多个值的都可以使用（{}）方式设置）

注意：position（）只能获取不能设置（设置使用.css）

scrollTop()

- 使用 overflow：auto：当元素内部的文本装不了时，自动使用滚动条
- scrollTop()：获取滚动条滚动的位置

  scrollTop(n)：设置滚动条偏移位，参数为整数
- 获取整个网页的偏移位：

  元素 body 或 html(浏览器兼容性不同)

  ```
  console.log($("body").scrollTop()+$("html").scrollTop());
  ```

  设置整个网页的偏移位：

  或：使用逗号来分隔多个元素标签。

事件绑定

JQuery 中有 2 中绑定事件的方式：

- EventName(fn)：

```
$("button").click(function () {
```

- On(name, fn)

```
$("button").on("click", function () {
    alert("hello lnj2");
});
```

区别：

- 前者编码效率高，但是部分事件 JQ 没有实现，不完整
- 后者可以使用任何事件，自定义的也行。

注意：可以添加多个相同的事件，***不会覆盖***；

事件解绑

Obj.off()

- 无参数：Obj.off() 全部移除
- 一个参数：Obj.off(eventName)：移除 eventName 的所有事件

```
$("button").off("click");
```

- 二个参数：Obj.off(eventName, funcName)

  移除指定事件的指定方法

```
$("button").off("click", test1);
```

事件冒泡和默认行为

事件冒泡：给父元素和子元素都添加事件，当子元素事件触发时，父元素也会触发事件(事件向上传递)。

注意：就算子元素没有绑定这个事件，也会发生事件冒泡，去看父元素是否有响应！

阻止事件冒泡：

- 方式一：在子元素事件处理加上返回值：**return false.**

```
$(".son").click(function () {
    alert("son");
    return false;
});
$(".father").click(function () {
    alert("father");
});
```

- 方式二：事件响应的函数中传递 event，然后在添加语句：

event.stopPropagation()

```
$(".son").click(function (event) {
    alert("son");
    // return false;
    // event.stopPropagation();
});
```

默认事件:如 a 标签,fom 的 submit,就算你给他们添加绑定了事件，当这些事件执行后，都会自动执行他们默认的事件，比如 a 跳转。

阻止默认事件：有时不希望执行默认的事件。

- 方式一：在事件函数中添加 return false.
- 方式二：添加参数 event，调用 event.preventDefault()

```
$("a").click(function (event) {
    alert("弹出注册框");
    // return false;
    event.preventDefault();
});
```

事件自动触发

- 方式一：

  先需要给元素绑定事件 eventName，如 click

  然后在给事件添加一个特殊事件：trigger(eventName)，用以自动触发上面绑定的事件

- 方式二：使用 triggerHandler（eventName）

```
$(".father").click(function () {
    alert("father");
});
// $(".father").trigger("click");
$(".father").triggerHandler("click")
});
```

区别：

1. Trigger 给子元素添加自动触发时会触发事件冒泡，使父元素事件也被触发，而 triggerHandler 则不会触发事件冒泡。
2. Trigger 会触发默认行为

   TriggerHandler 不会触发默认行为

```
$("input[type='submit']").trigger("click");
$("input[type='submit']").triggerHandler("click");
```

即：triggerHandler 不会触发事件冒泡和默认事件，二 trigger 都会触发。

自定义事件

自定义的事件类型：如 myClick

注意：自定义事件不能以 eventName 形式定义

```
$(".son").myClick(function (event) {
    alert("son");
});
```

必须要使用 on 形式定义：

```
$(".son").on("myClick", function () {
    alert("son");
});
$(".son").trigger("myClick");
```

注意：自定义事件不能被系统方式触发，必须使用 trigger 或 triggerHandler 触发

事件命名空间

企业开发中，多人协作的时候，多个人可能会给同一个元素添加多个事件，一次触发即全部响应，不好。添加命名空间可以区分不同的人的事件。

添加命名空间的方法：

1. 事件必须通过 on 绑定，不能通过 eventName 形式绑定。
2. On 中的参数 1 即 name 中使用“.usespace”来添加命名空间
3. 通过自动触发 trigger 来只触发个别命名空间的响应（实际上也可以把自动触发代理给另一个非自动触发实现手动触发）。

注意：

1. 若父元素和子元素含命名空间和事件类型一致，则会有事件冒泡，父元素同命名空间的事件被触发，无命名空间的则不会被触发
2. 利用 trigger 触发子元素不带命名空间的事件，那么子元素所有

同类型的事件和父元素所有同类型的事件都会被触发。

```
$(".son").on("click.zs", function () {
    alert("click1");
});
$(".son").on("click.ls", function () {
    alert("click2");
});
// $(".son").trigger("click.zs");
$(".son").trigger("click.ls");
```

事件委托

【`table>tr>td{我是第$个td}*10` 使用$自动生成序号】

在入口函数中绑定的事件或其他，在 HTML 加载时就会注册，但是对于后来新增的元素，原来绑定的事件却不会生效。（比如原来给 li 都绑定了事件，现新增一个 li，它却不会绑定到原来的事件。）

解决：事件委托

事件委托格式：

```
Obj.delegate(selector , type , data , fn)
```

```
    <script>
        $(document).ready(function () {
            $("ul>li").click(function () {
                console.log($(this).html());
            });
            $("button").click(function () {
                $("<li>I'm new</li>").appendTo($("ul"));
            });
            //把li的click事件委托给ul去监听，
            //响应事件的是ul，ul在文件加载的时候就有了
            //对于新增的li，它本身没有click事件，但是有事件冒泡，会使ul响应
            $("ul").delegate("li","click",function () {
                console.log($(this).html());
            });
            $.d
        })
    </script>
</head>
<body>
<ul>
    <li>I'm 1</li>
    <li>I'm 2</li>
    <li>I'm 3</li>
</ul>
<button>AddOne</button>
```

使用的地方：动态出现的元素。比如点击按钮，弹出一个输入框，可以通过点击然后 append 添加进去，然后将其代理给原来存在的元素

鼠标移入移除事件

- .mouseover

  .mouseout

  注意：子元素移出移入也会触发父元素

- .mouseenter

  .mouseleave

  注意：子元素移出移入不会触发父元素，推荐使用

- .hover(fn_in, fn_out)

  两个函数分别监听移入移出（不触发父元素）

Obj.siblings()

.siblings()方法选取除开当前元素的所有兄弟。

$(this).siblings()

显示隐藏哪个动画

- show([speed,[easing],[fn]])
  - speed:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如: 1000)
  - easing:(Optional) 用来指定切换效果，默认是"swing"，可用参数"linear"
  - fn:在动画完成时执行的回调函数，每个元素执行一次。

```
$("button").eq(0).click(function () {
    $("div").show(2000,"linear",function ()
    });
});
```

- hide() 同理

```
$("button").eq(1).click(function () {
    $("div").hide(2000,"linear",function ()
        alert("sd");
    });
});
```

- toggle() 切换：有就隐藏，没有就显示

```
$("button").eq(2).click(function () {
    $("div").toggle(1000);
});
```

（实战：对联广告）

.scroll(fn)

监听是否滚动了

```
$(window).scroll(function () {
    // 1.1获取网页滚动的偏移位
    var offset = $("html,body").scrollTop();
```

展开 收起动画

- slideDown()
- slideUp()
- slideToggle()

同 show hide()

适用：所有元素按**动画**平铺展开

实战：二级菜单在一级菜单下显示

动画——stop()

对于在事件中添加的动画效果，如果快速反复触发事件，事件将会排成队列，直到全部执行完，造成一种延迟效果，所以在使用动画前，调用，stop()方法能够停止当前进行的动画效果，就不会造成延迟。

.children()

```
// 1.3拿到所有非当前的二级菜单
var otherSub = $(this).siblings().children(".sub");
```

addClass removeClass

很常用，通过赋予类标签的方式来给一个元素添加性质。

Eq（）

获取现有伪数组的第几个对象，0-n-1

```
$("button").eq(0).show(2000);
```

$(this)

$(this)也是极其常用的

表示当前操作的对象。

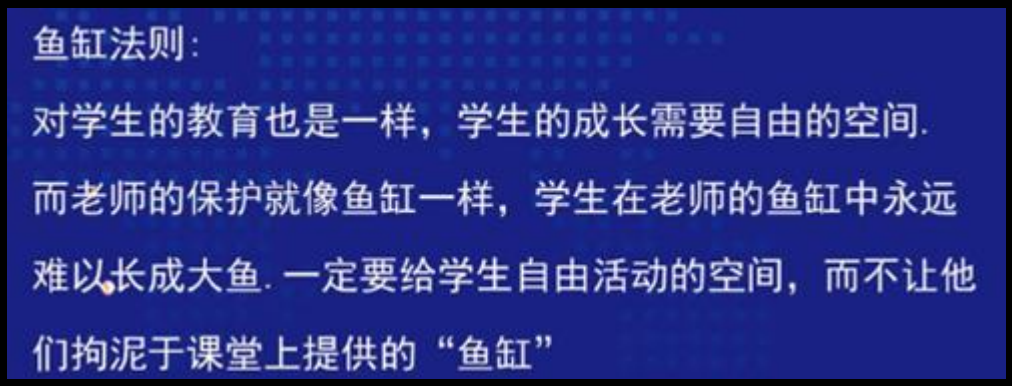
```
var index = $(this).index();
// 1.3根据索引找到对应的图片
var $li = $(".content>li").eq(index);
// 1.4显示找到的图片
```

index

eq

parseInt parseFloat

将字符串转化为整数或者是小数。



按钮一次点击触发多次

先解绑在绑定：

```
$("#b1").unbind().click(function () {
```

为什么？大家都知道怎么做，但是很少有人去打破砂锅问到底：原因是什么？看透事物本质才是关键！

提示信息显示后自动消失

方法一：

复制代码 代码如下：

```
$("#errormsg").html("您的信息输入错误，请重试!").show(300).delay(3000).hide(300);
```

方法二：

复制代码 代码如下：

```
$("#errormsg").html("ok").hide(3000);// 这个是渐渐消失
$("#errormsg").html("ok").fadeTo(3000).hide(); //这个是让他显示 5 秒（实际就是没有改变透明度），然后隐藏
```

Position:fixed 固定

几种选择器：

后代选择器：father children{} 可以跨代

子选择器：father > son{}

与后代选择器相比，子元素选择器（Child selectors）只能选择作为某元素子元素的元素

相邻兄弟选择器：bro + bro2 {}

选择紧接在另一个元素后的元素

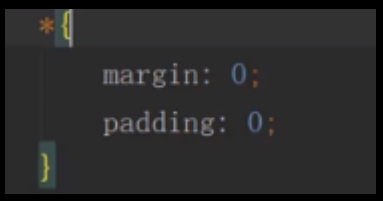
属性选择器：

选择器[属性]{}

Change()

Change（function(){}）：会在失去、获得焦点的时候触发、执行

*号——所有



animate()

效果

animate() 方法执行 CSS 属性集的自定义动画。

该方法通过 CSS 样式将元素从一个状态改变为另一个状态。CSS 属性值是逐渐改变的，这样就可以创建动画效果。

只有数字值可创建动画（比如 "margin:30px"）。字符串值无法创建动画（比如 "background-color:red"）。

注释：使用 "+=" 或 "-=" 来创建相对动画（relative animations）

1. 语法 1

   $(selector).animate(styles,speed,easing,callback)

2. 语法 2

   $(selector).animate(styles,options)

   options ：

   可选。规定动画的额外选项。

   可能的值:

   speed - 设置动画的速度

   easing - 规定要使用的 easing 函数

   callback - 规定动画完成之后要执行的函数

   step - 规定动画的每一步完成之后要执行的函数

   queue - 布尔值。指示是否在效果队列中放置动画。如果为

false，则动画将立即开始

specialEasing - 来自 styles 参数的一个或多个 CSS 属性的映射，以及它们的对应 easing 函数

插入兄弟节点

1、在每个匹配的元素之后插入内容。

```
$("p").after("<b>Hello</b>");
```

2、在每个匹配的元素之前插入内容。

```
$("p").before("<b>Hello</b>");
```

3、把所有匹配的元素插入到另一个、指定的元素元素集合的后面

```
$("p").insertAfter("#foo");
```

4、把所有匹配的元素插入到另一个、指定的元素元素集合的前面。

```
$("p").insertBefore("#foo");
```

jQ 追加的节点无法添加事件无效

需要 jQuery 中的 on 方法绑定事件。不能直接添加相应的事件

以 click 事件为例

```
$('#resultWrap li').click(function(){})
```

我们要改写为

```
$('#resultWrap').on('click','li',function(){})
```

```
$(document).on( types: 'click', selector: ".addRow", data: function () {
```

```
$(document).on("click",'#lyysb a',function(){
 if(!$(this).hasClass('cur')){
  $(this).addClass('cur');
 } else {
  $(this).removeClass('cur');
 }
});
```

官方API例子

```
<div id="zkdiv">
 <input type="button" value="展开" id="zk" class="zk"/> <br>
</div>

//展开按钮点击触发事件

$("#zkdiv").on("click",".zk",function(){
 console.log("on 点击一次");
});
var html2 = "<input type='button' class='zk' value='新生成的展开'
$("#zkdiv").append(html2);
```

点击删除自身

Remove()

如： $(“p”).remove()

Remove()也可以添加选择器

```
$(document).on( types: 'click', selector: ".delRow", data: function () {
    var tr = $(this).parent().parent().parent();
    if (tr.siblings().length!=0){
        tr.remove();
    }
});
```

$(sector).click()和$(document).on('click',sector,function(){})

都可以对一个元素进行点击事件的绑定，但是有一个小小的区别。

$(sector).click()只针对本页面已经存在的元素，对于后面通过 js 动态添加的元素，是没有完成事件绑定的。

but

$(document).on('click',sector,function(){})则可以对所有元素进行事件绑定，不管是原本存在，还是动态添加。

可以说前者是后者的子集。

后者个功能包含前者个功能。

$().on()的知识点

1.从 jQuery 1.7 开始，on()函数提供了绑定事件处理程序所需的所有功能，用于统一取代以前的 bind()、delegate()、live()等事件函数。

$().bind()直接绑定在元素上，和 click，blur，mouseon一样的点击事件。

$().live()是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到 document DOM 节点上。

$().delegate()是更精确的小范围的使用事件代理。

$().on()结合了这三个方法的优势摒弃了劣势。

2.该函数可以为同一元素、同一事件类型绑定多个事件处理函数。触发事件时，jQuery 会按照绑定的先后顺序依次执行绑定的事件处理函数。

3. 阻止事件冒泡和事件委托的方法：

A: return false。

在事件的处理中，可以阻止默认事件和冒泡事件。

B: event.preventDefault()

在事件的处理中，可以阻止默认事件但是允许冒泡事件的发生

C: event.stopPropagation()..

在事件的处理中，可以阻止冒泡但是允许默认事件的发生。

错误日志

Checked 一直是 undefined

在 jquery1.6 版本+:

【checked 属性在页面初始化的时候已经初始化好了，不会随着状态的改变而改变。 也就是说如果 checkbox 在页面加载完毕是选中的，那么返回的永远都是 checked，如果一开始没被选中，则返回的永远是 undefined !】

解决：`.prop("checked",this.checked);`

故：checked，select，disable 使用 prop()，其他的使用 attr()

Eg:实现反选

```
b.prop("checked",!b.prop("checked"));
```

走得太快，容易扯着蛋，慢一点，理解，运用在下一步！

其实，前面 HTML、CSS、JS 还有很多没掌握。