# 一 jQuery简介

## 1 jQuery的使用

原生JS的痛点：

1 window.onload事件只能出现一次，多次出现会覆盖之前的事件；

2 代码容错性差；

3 浏览器兼容问题；

4 简单功能实现起来复杂（各种循环，jQuery隐式迭代帮我们做了）。

jQuery的API都是方法，即要加小括号()，小传入的参数不同，功能不同。

版本一：1.x版本，兼容IE6/7/8

版本二：2.x版本，不兼容IE6/7/8

版本三：3.x版本，更精简，不再兼容低版IE

jQuery使用步骤：

<script src="./jQuery-1.11.3.min.js"></script> <!-- 引包 -->  
<script>  
 $(document).ready(function () { *//入口函数  
 //代码* });  
 *//也可以简写为：* $(function(){  
 *//代码* });  
</script>

## 2 jQuery入口函数与js入口函数

js入口函数指的是：window.onload = function() {};

区别一：书写个数不同

Js入口函数只能出现一次，出现多次会存在事件覆盖的问题。

jQuery的入口函数，可以出现任意多次，并不会存在事件覆盖问题。

区别二：执行时机不同

Js入口函数是在所有的 **文件资源** 加载完成后才执行。

jQuery的入口函数在 **文档** 加载完成后就执行。（即无需外部资源加载，DOM树加 载完成就执行）。

## 3 jQuery的$符号

jQuery使用$符号原因：书写简洁、相对于其他字符与众不同、容易被记住。

$实际上表示的**是一个函数**，即jQuery函数： jQuery ===$;

$函数根据传入参数的不同，进行不同的调用，实现不同的功能，返回的是jQuery对象。

## 4 jQuery的事件源 事件处理函数

$(function(){  
 $('#div').click(function () {  
 console.log(1)  
 });  
});

区别：jQuery的事件不带on

## 5 jQuery对象和DOM对象

DOM对象： 使用JS操作DOM返回的对象；

jQuery对象： 使用jQuery操作DOM获得的对象；

注意：jQuery对象其实是一个对DOM对象进行包装后的伪数组。

这样做的原因：DOM对象需要遍历，非常麻烦，且有兼容性问题，且会覆盖事件。

DOM对象转换成jQuery对象：$(DOM对象);

jQuery对象转换成DOM对象：

方式一：var btn = jQuery对象[0]; //推荐使用该方式

方式二：var btn = jQuery对象.get(0);

比如一个ul下所有的li获取：

$(‘li’).length length是jQuery属性

$(‘li’).get().length length是原生JS属性

$(‘li’)[1] 也可以直接获取第二个元素li

# 二 jQuery选择器

jQuery实现了从CSS1到CSS3所有的选择器以及其他常用的选择器。

## 1 基本选择器

id选择器： $('#btn');

类选择器： $('.btn');

标签选择器： $('div');

选择器的交集：$(#div,#green') 选择id为div，或id为green的元素

选择器的并集：$('#div#green') 选择id为div，且id为green的元素

## 2 层级选择器

后代选择器（空格）： $('#ul li') 选择id为ul的元素的所有后代li

子代选择器（>）： $('#ul > li') 选择id为ul的元素的直系后代li

## 3 过滤选择器

:eq(index) 选择匹配索引的元素 $('li:eq(2)') 选择索引号为2的li

:odd 选择匹配奇数索引元素 $('li:odd')

:even 选择匹配偶数索引元素 $('li:even')

## 4 筛选选择器

|  |  |  |
| --- | --- | --- |
| **符号(名称)** | **说明** | **用法** |
| **find(selector)** | 查找所有后代元素 | $(“#j\_wrap”).find(“li”).css(“color”, “red”);  选择id为j\_wrap的所有后代元素li |
| **children()** | 查找直接子代元素 | $(“#j\_wrap”).children(“ul”).css(“color”, “red”);选择id为j\_wrap的所有子代元素ul |
| **siblings()** | 查找所有兄弟元素  （不包括自己） | $(“#j\_liItem”).siblings().css(“color”, “red”);  选择id为j\_liItem的所有兄弟元素 |
| **parent()** | 查找父元素 | $(“#j\_liItem”).parent(“ul”).css(“color”, “red”);选择id为j\_liItem的父元素 |
| **eq(index)** | 查找指定元素的第index个元素 | $(“li”).eq(2).css(“color”, “red”);  选择所有li元素中的第二个 |

# 三 jQuery与DOM

## 1 jQuery操作样式

.css() 作用：设置或获取元素的样式属性值

$(selector).css(“color”, “red”); //设置单个样式

$(selector).css({“color”: “red”, “font-size”: “30px”}); //设置多个样式

$(selector).css(“font-size”); //获取样式

添加类样式：**(addClass)**

$(selector).addClass(“liItem”);

移除类样式：**removeClass(className)**

$(selector).removeClass(“liItem”);

$(selector).removeClass(); 不指定参数，表示移除被选中元素的所有类名

判断有没有类样式：hasClass(calssName) 返回true或false

$(selector).hasClass(“liItem”);

切换类样式：toggleClass(className) 该元素有类则移除，没有指定类则添加。

$(selector).toggleClass(“liItem”);

注意点：

操作类样式的时候，所有的类名都不带点

## 2 jQuery操作节点

### 2.1 动态创建元素

// $()函数的另外一个作用：动态创建元素

var $spanNode = $(“<span>我是一个span元素</span>”);

var node = $(“#box”).html（“<li>我是li</li>”）；

### 2.2 **append()**添加元素

**append()：**在元素的最后一个子元素后面追加元素（已存在或创建出来的元素都可以）。

$(selector).append($node); //追加传入jQuery对象

$(selector).append('<div></div>'); //直接传入html片段

其他操作：

appendTo(s) 添加到s元素最后面，原生没有

prepend() 添加到子元素最前面，类似原生的appendChild()

after() 添加到自己后面（作为兄弟）

before() 添加到自己前面（作为兄弟）

//获取到的元素剪切到某个位置（原生JS这里是选中后复制到某个位置）

insertBefore()

insertAfter() 原生JS没有insertAfter()

### 2.3 html()与text()

html(): 没有参数是获取包含标签的内容，有参数是插入内容，

设置内容时，如果是html标记，会动态创建元素，与原生 innerHTML相同

text(): 没有参数获取不包含标签的内容，有参数是插入内容，

设置内容时，类似原生innerText（火狐使用textContent获取），无视HTML标记 插入纯文本，但是text()不存在兼容问题

$(selector).html(‘<span>北京</span>’); // 动态创建元素，推荐该方式创建元素

$(selector).html(); // 获取html内容

### 2.4 清空/删除元素

$(selector).empty(); // 清空指定元素的所有子元素，推荐使用，因为会清除事件

$(selector).html(“”); // 作用同上，但是会出现内存泄露：元素事件不会被清空

$(selector).remove(); // 删除元素，包括自己，返回被删除的元素，事件也被删除

$(selector).detach(); // 作用同上，但是会保留事件

### 2.5 clone()复制元素

作用：复制匹配的元素，返回值为复制的新元素，参数的作用是复制操作行为

$(selector).clone();

$(selector).clone(true); //复制操作行为

### 2.6 其他节点操作

parents()

获取所有祖先节点生成集合，参数具备筛选功能，只要集合有对应的参数都会被选中；

offsetParent() : 获取有定位的父级

closesll()

获取最近的指定的祖先节点（包括元素自身），必须有参数，所以只能找到一个元素；

siblings()

查找所有的兄弟节点，参数具备筛选功能；

nextAll() prevAll()

nextAll查找下一个所有兄弟节点；preAll查找上一个所有的兄弟节点

nextUntil() prevUntil()

分别对应nextAll，preAll作用一致，但是可以写入参数，查找到指定位置；

parentsUntil()

wrap() wrapAll() wrapInner() unwrap()

包装方法：$(selector).wrap(‘<div></div>’); 给每个单独的选中标签外部包装一个div

wrapAll()：给选中的所有的标签整体包装一个标签；

wrapInner():将选中的每个标签的value值包装一个标签；

unwrap():删除父级，但是不包括body

add() slice()

add():组合操作，返回一个元素集；$(‘div’).add(‘span’);

slice(start,end):返回选择元素集中从第start-end位置的元素

## 4 jQuery操作属性与表单

### 4.1 attr()

使用style获取的都采用css()方法设置；

在标签里书写的样式，使用attr()获取

$(selector).attr(“title”, “jQeury简介”); //设置属性，设置多个传入对象

$(selector).attr(“title”); //获取属性

$(selector).removeAttr(“title”); //移除属性

### 4.2 prop()

注意：checked、selected、disabled要使用.prop()方法。(1.7版本之前仍然可以使用attr)。

prop方法通常用来影响DOM元素的动态状态，而不是改变的HTML属性。例如：input和button的disabled特性，以及checkbox的checked特性。

### 4.3 val()与text()

val()方法：获取元素的value，带参数就是设置value值

$(selector).val(); // 获取匹配元素的值，只匹配第一个元素

$(selector).val(“具体值”); // 设置所有匹配到的元素的值

text() 方法：没有参数获取所有内容，有参数是以文本形式插入

$(selector).text(); //会把所有匹配到的元素拼接为一个字符串！）

$(selector).text(“我是内容”); //参数表示要设置的文本内容

### 4.4 数据串连 serialize() serializeArray()

<form>

    <input type="text" name="a" value="1">

<input type="text" name="b" value="2">

<input type="text" name="c" value="3">

</form>

<script>

    $(function(){

        //console.log($('form').serialize()); //string : a=1&b=2&c=3

        console.log( $('form').serializeArray() );

        [

            { name : 'a' , value : '1' },

            { name : 'b' , value : '2' },

            { name : 'c' , value : '3' }

        ]

    });

## 5 jQuery操作尺寸位置

### 5.1 高度和宽度操作

height() ： 设置或获取匹配元素的高度值

width() ： 设置或获取匹配元素的宽度值

$(selector).height(200); //带参数表示设置高度

$(selector).height(); //不带参数获取高度

css()获取高度和height获取高度的区别？

$('div').height(); //返回值为Number类型，30，通常用来参与运算

$('div').css('height'); //返回值为String类型,30px

补充：

width() 获取width

innerWidth() 获取width+左右padding

outerWidth() 获取width+左右padding+左右边框宽度

outerWidth(true)获取width+左右padding+左右边框宽度+左右margin

原生的outerWidth无法获取隐藏元素的值，而jQquery可以。所以jQuery中可以获取到隐藏元素的一些属性值。

### 5.2 坐标值操作

offset() : 获取或设置元素相对于文档的位置

// 无参数表示获取，返回值为：{left:num, top:num}，值是相对于document的位置

$(selector).offset();

// 有参数表示设置，参数推荐使用数值类型

$(selector).offset({left:100, top: 150});

注意点：设置offset后，如果元素没有定位(默认值：static)，则被修改为relative

position(): 获取相对于其最近的具有定位的父元素的位置。只能获取，不能设置。

$(selector).position(); // 获取，返回值为对象：{left:num, top:num}

scrollTop() : 获取或者设置元素垂直方向滚动的位置

// 无参数表示获取偏移,有参数表示设置偏移，参数为数值类型

$(selector).scrollTop(100);

scrollLeft() : 获取或者设置元素水平方向滚动的位置

$(selector).scrollLeft(100);

总结：

$(“div”).offset(); // 获取或设置坐标值 设置值后变成相对定位

$(“div”).position(); // 获取坐标值 子绝父相 只能读取不能设置

$(“div”).scrollTop(); // 被卷曲的高度，即相对于滚动条顶部的偏移

$(“div”).scrolllLeft(); // 被卷曲的宽度，即相对于滚动条左部的偏移

**垂直滚动条位置 是可滚动区域 在 可视区域上方的 被隐藏区域的高度。**

**如果滚动条在最上方没有滚动 或者 当前元素没有出现滚动条，那么这个距离为0**

# 四 jQuery动画

## 1 显示/隐藏show/hide

show()/hide 显示/隐藏，用法一致

//不带参数，作用等同于 css(“display”, ”block”)，此时没有动画效果

$(selector).show();

//参数为数值类型，表示：执行动画时长，单位为：毫秒（ms），即多少毫秒动画结束

$(selector).show(2000);

//参数为字符串类型(jQuery预设的值:slow（600ms）、normal（400ms）、fast（200ms）

$(selector).show(“slow”);

//参数一可以是数值类型或者字符串类型，动画执行完后立即执行的回调函数

$(selector).show(2000, function() {});

### 2 滑入/滑出slideDown/slideUp

滑入动画效果：让元素以下拉动画效果展示出来

$(selector).slideDown(speed,callback);

滑出动画效果：让元素以上拉动画效果隐藏起来

$(selector).slideUp(speed,callback);

滑入滑出切换动画效果

$(selector).slideToggle(speed,callback);

注意：以上方法省略参数或者传入不合法的字符串，那么则使用默认值：400毫秒

### 3 淡入/淡出fadeIn/fadeOut

淡入动画效果：让元素以淡淡的进入视线的方式展示出来

$(selector).fadeIn(speed, callback);

淡出动画效果：让元素以渐渐消失的方式隐藏起来

$(selector).fadeOut(1000);

淡入淡出切换动画效果：通过改变透明度，切换匹配元素的显示或隐藏

$(selector).fadeToggle('fast',function(){});

### 4 改变透明度fadeTo

与淡入淡出的区别：淡入淡出只能控制元素的不透明度从 完全不透明 到完全透明；而fadeTo可以指定元素不透明度的具体值。时间参数是必需的！

作用：调节匹配元素的不透明度，用法有别于其他动画效果

// 第一个参数表示：时长

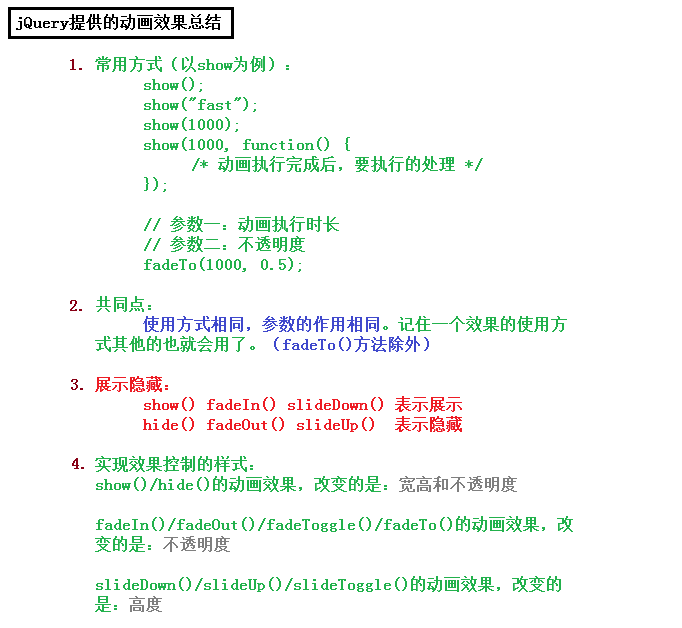
// 第二个参数表示：不透明度值，取值范围：0-1

$(selector).fadeTo(1000, .5); // 0全透，1全不透

// 第一个参数为0，此时作用相当于：.css(“opacity”, .5);

$(selector).fadeTo(0, .5);

### 5 jQuery动画总结



### 6 自定义动画animate

作用：执行一组CSS属性的自定义动画

//animate() :

//第一个参数 : {} 运动的值和属性

//第二个参数 : 时间(运动快慢的) 默认 : 400，可选

//第三个参数 : 运动形式 只有两种：swing(慢快慢，默认) linear(匀速) ，可选

//第四个参数 : 回调函数，可选

        $(this).animate({width : 300} , 4000 , 'linear',function(){

            alert(123);

        });

注意：所有能够执行动画的属性必须只有一个数字类型的值，比如：要改变字体大小，要使用：fontSize（font-size），不要使用：font

### 7 停止动画stop

为什么要停止动画？

如果一个以上的动画方法在同一个元素上调用，那么对这个元素来说，后面的动画将被放到效果队列中。这样就形成了动画队列， 动画队列里面的动画不会执行，直到第一个动画执行完成。

使用方式： $(selector).stop(clearQueue,jumpToEnd);

带参数方式：有两个参数，默认都为false

第一个参数表示后续动画是否要执行

true：后续动画不执行

false：后续动画会执行

第二个参数表示当前动画知否要执行

true：立即执行完成当前动画

false：立即停止当前动画

实战使用： $(selector).stop().slideDown(); //先停止别人的动画，再执行自己的动画。

当调用stop()方法后，队列里面的下一个动画将会立即开始。但如果参数clearQueue被设置为true，那么队列里剩余的动画就被删除了，并且永远也不会执行。

如果参数jumpToEnd被设置为true，那么当前动画会停止，但是参与动画的每一个CSS属性将被立即设置为它们的目标值。比如：slideUp()方法，那么元素会立即隐藏掉。如果存在回调函数，那么回调函数也会立即执行。

注意：如果元素动画还没有执行完，此时调用sotp()方法，那么动画将会停止。并且动画没有执行完成，那么回调函数也不会被执行。

    $('#div1').click(function(){

        $(this).animate({width : 300} , 2000).delay(1000).animate({height : 300} , 2000);

    });

    $('#div2').click(function(){

        //$('#div1').stop(); //默认 : 只会阻止当前运动

        //$('#div1').stop(true); //阻止后续的运动

        //$('#div1').stop(true,true); //可以立即停止到指定的目标点

        $('#div1').finish(); //立即停止到所有指定的目标点

    });