jQuery版本

1.x 兼容IE678，文件较大，大型网站都用的这个

2.x 不兼容IE678

3.x 不兼容IE678 只支持最新版本浏览器

head标签引入jQuery

<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

入口函数

<script>

//js原生写法

window.onload=function(ev){

}

//jQuery写法

//jQuery写在<script>标签里

$(document).ready(function(){

alert("fffff");

});

</script>

$.holdReady(true);暂停入口函数ready

$.holdReady(false);恢复ready函数

原生JS和jQuery加载模式不同

1.原生JS等到DOM元素加载完毕，图片也加载完毕后执行

2.jQuery等到DOM元素加载完毕，但不会等到图片加载完毕时就已经执行

$(function(){

alert("hello world!");

});

jQuery解决冲突问题

1.释放$使用权

jQuery.noConflict();

然后就不要使用$符号，全部变为jQuery

2.自定义访问符号

var s1=jQuery.noConflict();

1. **var** s1=jQuery.noConflict();
2. **s1**(**function**(){
3. alert("hello world!");
4. });

$();代表jQuery**核心函数**

* 1.接受一个函数

作为入口函数，执行的jQuery代码都写在里边

$(function(){

alert("hello world!");

});

* 2.接收一个字符串
* 2.1.接受一个字符串选择器

返回一个jQuery对象，对象包装了找到的DOM元素

var div=$("div")

var $div=$(".box")

var $div2=$("#box2")

* 2.2.接受一个代码片段

var $box=$("<p>This is a paragraph<p>");

$div.append($box);

根据代码内容创建DOM元素

返回一个jQuery对象，对象中包装了创建的DOM元素

* 3.接受一个DOM元素

var $span=document.getElementsByTagName("span")[0];

var span=$($span);

原生的DOM元素被包装成一个jQuery对象，返回给我们

var div=$("div");

返回一个类数组 这个教程叫伪数组有length属性，属性名都是索引

$("div").**eq**(0)

选中第一个

$("span").find("img")

找出所有span下所有的img，等效于$("span img")

**Javascript实例方法**

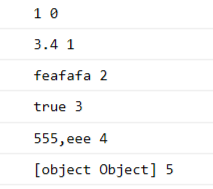
1. **function** Class1(){
2. }
3. Class1.prototype.f2=**function**(){
4. alert("fefafafa");
5. }  //prototype是一个对象！！！
6. //Class1.f2();不能这样用，报错Object doesn't support property or method 'f2'
7. var x=new Class1();
8. x.f2();

**静态方法**

1. **function** Class1(){
2. }
3. Class1.f1=**function**(){
4. alert("fewfwef");
5. }
6. Class1.f1(); //静态方法直接在类上创建，通过类调用
7. //x.f2();不能这样用，报错Object doesn't support property or method 'f1'

**JavaScipt forEach()**

1. **var** array=[1, 3.4, 'feafafa', **true**, [555,'eee'], {}];
2. array.forEach(**function**(value, index){
3. console.log(value, index);
4. });

控制台：

只能数组使用

参数是一个函数，函数有两个参数或3个参数。两个参数时，第一个参数是值，第二个参数是索引；3个参数时，第一个参数是值，第二个参数是索引，第三个参数是数组对象

书上明确说：forEach方法除了数组对象外，还可以用于具有length属性且具有已按数字编制索引的属性名的任何对象，如关联数组对象，Arguments等

教程明确说：forEach只能用于数组，不能用于类数组

亲测好像不行Object doesn't support property or method 'forEach'

。。。

**jQuery each**方法

var obj={

0: 'feafafe',

1: 333,

2: true,

length: 3

}

$.each(obj, function(index , value){

console.log(index, value);

});

可以用于类数组

第一个参数：要遍历的数组或类数组

注意顺序不一样，JS forEach第一个参数value第二个参数index，jQuery each第一个参数index第二个参数value

$.map 返回一个数组

var str=' ffsaa ';

var str1=$.trim(str);

返回一个新字符串，去除字符串两端空格

一些**静态函数**

isArray() 判断是否是真数组，类数组返回False

isFunction()

isWindow() 判断是否是window对象

console.log($.isFunction(jQuery));返回true

jQuery本质上是一个匿名函数

jQuery**内容选择器**

:empty

没有子元素没有文本内容的元素

:parent 有文本元素或子元素的元素

:contains('hello world');

找到包含这段文本内容的元素

:has('span')

包含子元素span的元素

**属性节点**

编写HTML代码时，在HTML标签中添加的属性就是属性节点

<span class="box" name="span1">hello</span>

在浏览器找到span这个DOM元素后，看到的都是属性？调试工具的Source下的Watch中？？

在attributes属性里保存的都是属性节点？？？

attributes就是一个属性

var span=document.getElementsByTagName("span")[0];

span.setAttribute("name", "xiaowang");

给span的属性节点赋值

.attr()

1.$("span").attr("class")  
传一个参数，获取属性节点的值

无论找到多少个元素（上面这么写可能找到多个），只返回第一个元素属性节点的值

2.$("span").attr("class", "box")

传两个参数，修改属性节点

找到多少个元素（上面这么写可能找到多个）,设置多少个，全部设置

如果设置的属性节点不存在，自动新增

.removeAttr("class name") //删两个属性节点可以这么写

删除所有找到的元素的属性节点

操作属性的方法

.prop()

和attr特点一样

能操作属性也能操作属性节点

官方推荐：操作属性节点时，具有True和false两个属性的属性节点，如checked，selected或者disabled使用prop(),其他使用attr()

小练习：

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. <title>点一下就换图片</title>
6. <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
7. </head>
8. <body>
9. <input type="input" name="input1">
10. <button>点击换图</button><br>
11. <img src="https://www.baidu.com/img/superlogo\_c4d7df0a003d3db9b65e9ef0fe6da1ec.png?where=super" style="background-color: black">
12. <script type="text/javascript">
13. var b1=document.getElementsByTagName("button")[0];
14. var input1=document.getElementsByTagName('input')[0];
15. b1.addEventListener('click', f1, **false**);
16. function f1(){
17. var img1=document.getElementsByTagName('img')[0];
18. $(img1).attr("src", input1.value);
19. }
20. </script>
21. </body>
22. </html>

Notice：

1.必须包装成一个jQuery对象才能用jQuery方法，比如$(img).attr(),img.attr()就不行

2.input1.value获取输入框内容

3.type="input" name="input1"中的type和name就叫属性节点

4.教程上的都写在了一个

$(function(){

});

里边

或者在body最后加script标签里面写

$(document).ready(function(){

});

5. $(" img").css("width")获取的值为500px，转换为500时去掉px：parseInt()

操作类的方法

1.addClass()

如果添加多个，多个类名之间用空格隔开即可

2.removeClass()

如果删除多个，多个类名之间用空格隔开即可

3.toggleClass()

有就删除，没有就添加

<div class="c1">qewr</div>

$("div").addClass("c2");

现在相当于<div class="c1 c2">qewr</div>

4. hasClass("box")

$("div").click(function(){

if ( $(this).hasClass("protected") ){

}

});

检查当前的元素是否含有某个特定的类，如果有，则返回true。

相当于is("." + class)。

操作文本

1. .html()

和原生JS的innerHTML一模一样

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

var html1=$("div").html();//获取

2. .text()

和原生JS的innerText一模一样

3. .val()

$("input").val("请输入内容");

var x=$("input").val(); //获取

只有input有val()

操作CSS

逐个设置

$("div").css("样式名", "样式值");

$("div").css("width”, "100px");像这样

链式设置

$("div").css("width”, "100px").css("height", "100px")

不推荐

批量设置

$("div").css({

width: "100px",

height: "100px",

background-color: "black"

});传一个对象进去,推荐使用

注意，由于是对象，值必须用字符串形式

获取

$("div").css("width");

位置和尺寸

.width()

.height()

包括边框和内边距，不包括外边距

.offset().left

.offset().right

.offset().bottom

.offset().top

获取距离窗口的偏移位

设置方法：

.offset({

left: 10

});

奇了怪了这个不能写成"10px",必须数字才行？？

.position().left

.position().right

.position().bottom

.position().top

距离最近的定位元素的偏移位

position只能读不能写

$(".scroll").scrollTop()

获取滚动条位置

$(".scroll").scrollTop(300)

这样设置滚动条位置

$("html").scrollTop()+$("body").scrollTop();

获取页面滚动条偏移位，为了兼容IE

$("html, body").scrollTop(300); //设置方法

**事件**

绑定事件的两种方法

1. .eventName(fn)

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

alert("hello world")

});

2. .on(eventName, fn)

$("button").on("click", function(){

alert();

});

部分事件1方法jQuery没有实现，只能用第二种方法

注册多个事件甚至是相同种类的事件，不会覆盖

blur() 触发、或将函数绑定到指定元素的 blur 事件

change() 触发、或将函数绑定到指定元素的 change 事件

click() 触发、或将函数绑定到指定元素的 click 事件

dblclick() 触发、或将函数绑定到指定元素的 double click 事件

error() 触发、或将函数绑定到指定元素的 error 事件

focus() 触发、或将函数绑定到指定元素的 focus 事件

keydown() 触发、或将函数绑定到指定元素的 key down 事件

keypress() 触发、或将函数绑定到指定元素的 key press 事件

keyup() 触发、或将函数绑定到指定元素的 key up 事件

load() 触发、或将函数绑定到指定元素的 load 事件

mousedown() 触发、或将函数绑定到指定元素的 mouse down 事件

mouseenter() 触发、或将函数绑定到指定元素的 mouse enter 事件

mouseleave() 触发、或将函数绑定到指定元素的 mouse leave 事件

mousemove() 触发、或将函数绑定到指定元素的 mouse move 事件

mouseout() 触发、或将函数绑定到指定元素的 mouse out 事件 鼠标移出

mouseover() 触发、或将函数绑定到指定元素的 mouse over 事件 鼠标移入

mouseup() 触发、或将函数绑定到指定元素的 mouse up 事件

resize() 触发、或将函数绑定到指定元素的 resize 事件

scroll() 触发、或将函数绑定到指定元素的 scroll 事件

select() 触发、或将函数绑定到指定元素的 select 事件

submit() 触发、或将函数绑定到指定元素的 submit 事件

unload() 触发、或将函数绑定到指定元素的 unload 事件

移除事件

$("button").off();

不传参数移除所有事件

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

传一个参数移除所有指定类型事件

$("button").click(test1);

function test1(){  
 alert("666");

}

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

传两个参数，移除指定类型的指定事件

阻止事件冒泡

1.子元素的回调函数中，return false

2.$("button").click(funtion(event){  
 alert("666");

event.stopPropagation();

}

都是子元素里边写的

阻止默认事件

比如a标签的跳转事件

1.$("a").click(funtion(){

alert("555");

return false;

}

2. $("a").click(funtion(event){  
 alert("666");

event.preventDefault();

}

事件自动触发

$(".son").trigger("click");

自动触发元素的click事件

$(".son").triggerHandler("click");

区别

1.trigger()方法会触发事件冒泡

triggerHandler()不会触发事件冒泡

2.trigger()触发事件同时触发默认事件

triggerHandler()自动取消了默认行为，但是a标签的默认事件不触发，有点特殊

想要触发a标签的默认事件，还想自动触发，这样写

HTML:

<a><span>点击弹框并跳转</span></a>

jQuery:

$("span").click(function(){

alert("666");

}

$("span").trigger("click");

把a中的内容封装在另一个元素里，事件绑定和触发都用另一个元素

自定义事件

1.事件通过on绑定

2.事件通过trigger或triggerHandler触发

$(".son").on("myClick", funtion(){

alert("rrr");

}

$(".son").trigger("myClick"):

也就是说自定义事件只能自动触发

jQuery事件命名空间

1.必须通过on绑定

$(".son").on("click.linran", function(){

alert();

});

$(".son").on("click.lizhigang", function(){

alert();

});

2.必须trigger触发

$(".son").trigger("click.linran");//只触发我写的click事件

利用trigger触发子元素带命名空间的事件，父元素带相同命名空间的事件被触发，而父元素没有带相同命名空间的事件不触发

利用trigger触发子元素不带命名空间的事件吗，子元素所有相同类型的事件和父元素所有相同类型的事件，不管有没有命名空间，都触发

在jQuery中，如果核心函数找到的元素不止一个，那么添加事件的时候会给所有找到的元素添加事件

$("ul>li")找到所有的li

事件委托

1. **<ul>**
2. **<li>**aaaaaaaa**</li>**
3. **<li>**bbbbbbbb**</li>**
4. **<li>**cccccccc**</li>**
5. **<li><button>**新增一个li**</button></li>**
6. **</ul>**
7. $("ul li button").click(**function**(){
8. $("ul").append("<li>新增的li</li>");
9. });
10. $("ul").**delegate("li", "click",** **function**(){
11. console.log($(**this**).html());
12. });

把li的click事件委托给ul执行

因为jQuery代码在HTML加载完毕后执行，执行的时候新增的li还没有出现，所以不能用传统方法

委托给ul，ul在DOM元素加载完毕时已经产生，所以可以一直监听

function里的this指li

在入口函数之前就有的元素监听动态产生的元素的事件

$("body").delegate(".login>span", "click", function(){

$(".mask").remove();

});

body监听动态框的关闭按钮

.one()添加的事件只执行一次

mouseover/mouseout鼠标移入移出事件，子元素被移入移出也会触发父元素的移入移出

推荐使用mouseenter,mouseleave

$(".father").hover(function(){

}, function(){

});

两个参数，一个移入，一个移出

也可以一个参数，既监听移入又监听移出

$(this).index()

找到自己在兄弟之中排第几

$(this).index("li")

在所有li中排第几

$("p").eq(1)

获取匹配的第二个元素

$("p").eq(-2)

获取匹配的倒数第二个元素（参数为负数时表示倒数，倒数第一个是-1）

eq返回一个jQuery对象

get(0)

返回DOM对象本身

$('li').first()

$('li').last()

获取匹配的第一个/最后一个元素

$(".button").siblings()

返回同级别的元素(包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合)

我也看不懂自己记了啥。返回兄弟就对了

$("div").siblings(".selected")

找到每个div的所有同辈元素中带有类名为selected的元素

$("div").children()

返回所有子元素

$("div").children(".selected")可以加条件

$("input[type='checkbox']").parent().is("form")

返回true

检测匹配元素集合，如果其中至少有一个元素符合这个给定的表达式就返回true

$("div").show(1000, function(){

alert();

});

显示动画执行完毕后执行的函数

针对hide()方法隐藏，或CSS里设置了display:none的元素

* 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)。有一篇文章上说最好使用字符串
* easing默认'swing’先慢后快再慢，可用"linear"
* fn:在动画完成时执行的函数，每个元素执行一次。

以下所有动画效果的方法，参数都有这几个

$("div").hide(1000)

$("div").toggle(1000)

切换，状态是show就hide，状态是hide就show

默认是从左上角延伸到右下角

$("div").slideUp(1000, function(){  
 alert("展开完毕执行的动作");

});

$("div").slideDown(1000, function(){

alert("收起完毕执行的动作");

});

$("div").slideToggle(1000, function(){});

$("div").**stop**([clearQueue],[jumpToEnd])

* $("div").stop() stop(false, false) stop(false)效果一样

调用动画之前最好调用stop()

停止当前正在进行的动画，如果接下来还有动画等待继续进行，则以当前状态开始接下来的动画

* $("div").stop(true)

$("div").stop(true, false)

停止当前和后续的所有动画（把当前元素接下来尚未执行完的动画队列清空）

$("#panel").hover(function() {

$(this).stop(true).animate({height: "150"}, 5000).animate({borderWidth: "20px"},3000);//这就是一个动画队列

},function() {

$(this).stop(true)

.animate({width: "300"}, 2000)

.animate({fontSize: "24px"},3000);//也可以这样写

});

如果两个都是stop()，则

鼠标移入——height改变——鼠标移出——stop()触发——borderWidth改变——width改变——fontsize改变

两个都是stop(true)时

鼠标移入——height改变——鼠标移出——stop()触发——width改变——fontsize改变

* $("div").stop(false, true)完成当前的，执行后续的
* $("div").stop(true, true)

停止当前动画并直接到达当前动画的末状态（完成当前），并清空动画队列。

$("div").fadeIn(1000, function(){

alert("淡入结束后执行的函数);

}

$("div").fadeIn(1000, 0.2, function(){

alert("淡入结束后执行的函数);

}

淡入到20%不透明度

$("div").**animate**(params,[speed],[easing],[fn])

[ ]内为可选参数

$("#block").animate({

width: "90%",//可以使用百分数

height: "+=100", //累加

fontSize: "10em",

border: "hide" //关键字,还有toggle没有变成有，有变成没有

}, 1000 );

$("div").delay(2000)

延迟

监听网页滚动

$(window).scroll(function(){

}

滚动条一滚动function就触发

* Content Delivery Network CDN
* jQuery类型的变量最好加个$前缀。
* javascript和css实际应用中使用外部文件可以提高页面速度，因为JavaScript和CSS文件都能在浏览器中产生缓存。内置在HTML文档中的JavaScript 和CSS则会在每次请求中随HTML文档重新下载
* .多级查找中，右边尽量指定得详细点而左边则尽量简单点

// 丑陋 $("div.data .gonzalez");

// 优化后 $(".data td.gonzalez");

* 操作任何元素前先将其从文档卸载，然后再贴回去

var $myList = $("#list-container > ul").detach();

//...a lot of complicated things on $myList

$myList.appendTo("#list-container");

* 一个页面只写一个文档ready事件的处理程序

.不要用匿名函数来做事件的回调。匿名函数不易调试维护测试和复用

$("#myLink").on("click", function(){...}); // 不要这样

function myLinkClickHandler(){...}// 这样

$("#myLink").on("click", myLinkClickHandler);

* 处理文档ready事件的回调也不要用匿名函数，匿名函数不易调试维护测试和复用

$(function(){ ... }); // 糟糕的做法：无法利用此函数也无法为其写测试用例

// 好的做法

$(initPage); // 抑或 $(document).ready(initPage);

function initPage(){

// 这里你可以进行程序的初始化了

}

**节点操作**

1.创建节点

var $li=$("<li>新增的<li>");

2.添加节点

* 内部插入到最后/最前

$("ul").append($li); $li.appendTo($ul);

$("p").append("<p>Hello world!</p>");

$("ul").prepend($li); $li.prependTo($ul);

* 外部插入

li添加到ul后面

$("ul").after($li)

li添加到ul前面

$("ul").before($li)

$li.insertAfter("ul")  
$li.insertBefore("ul")

3.删除节点

$("div").remove()

删除自身及子节点

$("div").empty()

删除子节点，保留自身

$("div").empty(".content")

删除content类的div

detach()和remove()一模一样

4.替换节点

$li.replaceWith($h1)

把li换成h1

$h1.replaceAll(selector)

5.复制节点

var $li1=$("li:first").clone(false);

浅复制

var $li1=$("li:first").clone(true);

深复制

深复制连事件也一起复制了

图片按比例缩放

img{

width:auto;

height:auto;

max-width:100%;

max-height:100%;

}

Ajax

发送get请求

$.ajax({

url: "p1.php",

type: "get",//不分大小写

data: "userName=linran&userPwd=123",

//data还可以写成这种形式

//data:{"name": "myName"};

success: function(msg){

alert(msg.responseText);

},//success 当请求之后调用。传入返回后的数据，以及包含成功代码的字符串

error: function(xhr){

alert(xhr.status);//error 在请求出错时调用。传入XMLHttpRequest对象，描述错误类型的字符串以及一个异常对象（如果有的话）

}

})

p1.php

<?php

echo $\_GET["userName"];

echo $\_GET["userPwd"];

?>

$.ajax({

url: "p1.php",

type: "get",

data: "userName=linran&userPwd=123",

success: function(msg){

var str=msg.responseText;

var obj=JSON.parse(str);

alert(obj.name);

},//success 当请求之后调用。传入返回后的数据，以及包含成功代码的字符串

error: function(xhr){

alert(xhr.status);//error 在请求出错时调用。传入XMLHttpRequest对象，描述错误类型的字符串以及一个异常对象（如果有的话）

}

})

获取单选框的value

$("input[name='age']:checked").val()

获取输入框的value

$("input[name='roomid']").val();

轮播图

$(".banner").hover(function () {

clearInterval(t);//鼠标悬停时清除定时器

}, function () {

t = setInterval(function () {

i++;

move();

}, 2000); //鼠标移出时开始定时器

});

console.log( jQuery.inArray( "Karl", arr ) );

console.log( jQuery.inArray( "Pete", arr, 2 ) );

<div>"Pete" 在数组中，但是不在索引值大于2的位置，所以返回-1

* $.inArray() 函数用于在数组中查找指定值，并返回它的索引值（如果没有找到，则返回-1）

$.inArray( value, array [, fromIndex ] )

参数 描述

value 任意类型 用于查找的值。

array Array类型 指定被查找的数组。

fromIndex 可选。Number类型 指定从数组的指定索引位置开始查找，默认为 0

# jQuery UI

1.代码不够健壮：缺乏全面的测试用例，部分组件 Bugs 较多，不能达到企业级产品开发要求。

2.构架规划不足：组件间 API 缺乏协调，缺乏配合使用帮助。

3.控件较少：相对于 Dojo、YUI、Ext JS 等成熟产品，可用控件较少，无法满足复杂界面功能要求。