█执行时机

Jquery有两种方法

●网页中所有DOM结构绘制完毕后就执行，可能DOM元素关联的东西并没有加载完。

// 正常写法

$(document).ready(function(){

})

// 简写

$(function(){

})

●必须等待网页中所有的内容加载完毕后（包括图片）才能执行。类似window.onload，但window.onload不能同时编写多个。而这个jquery方法可以编写多个。

// 正确写法

$(window).on("load",function(){

})

// 以下会报错,只能用on绑定load才不会报错

// $(window).load(function(){

// alert("aa");

// })

●window.onload方法。必须等待网页中所有的内容加载完毕后（包括图片）才能执行。不能同时编写多个。

// 如出现多个,则只执行最后一个.有的环境还会报错

window.onload = function(){

};

█转换

●将jquery对象转换成DOM对象有两种方法[index]和get(index)

// 使用[index]方法将jquery转换成DOM对象

var $d1=$("#d1");

var d1=$d1[0];

// 使用get(index)方法将jquery转换成DOM对象

var $d2=$("#d2");

var d2=$d2.get(0);

●将DOM对象转换为jquery对象的方法是将DOM对象用$()包裹取来。

// 将DOM对象转换成jquery对象

var d3=document.getElementByid("d3");

var $d3=$(d3);

●如上面例子，在用var声明变量的时候，如果创建的是jquery对象，在变量名前面加上$这样就能方便区分变量是DOM对象还是jquery对象：

例如：

DOM对象： d5

Jquery对象： $d5

█选择器

●Jquery选择器分类

基本选择器

层次选择器

过滤选择器

基本过滤选择器

内容过滤选择器

可见性过滤选择器

属性过滤选择器

子元素过滤选择器

表单对象属性过滤选择器

表单选择器

●基本选择器

$("#test") //选择id为test的元素 （返回单个）

$(".test") //选择所有class为test的元素 （返回集合）

$("p") //选择所有p元素（返回集合）

$("\*") //选择所有元素（返回集合）

$("div,span,p,.test,#test") //选择所有div,span,p和class为test，id为test的元素（返回集合）

●层次选择器

$("div span") //选择div里的所有span，空格代表div往下所有层级 （返回集合）

$("div > span") //选择div下元素名是span的子元素，> 代表选择子元素，只往下一级 （返回集合）

$(".one + div") //选择class 为one的下一个div同辈元素，推荐用next()方法代替 （返回集合）

$(".one").next("div") //等同于$(".one + div")

$("#two~div") //选择id为two的元素后面所有div同辈元素，推荐用nextAll()方法代替 （返回集合）

$("#two").nextAll("div") //等同于$("#two~div")

●过滤选择器 基本过滤选择器

$("div:first") //选择所有div元素中第1个div元素（返回单个）

$("div:last") //选择所有div元素中的最后一个div元素（返回单个）

$("input:not(.myClass)") //选择class不是myClass的input元素（返回集合）

$("input:even") //选择索引是偶数的input元素，索引从0开始（返回集合）

$("input:odd") //选择索引是奇数的input元素，索引从0开始（返回集合）

$("input:eq(1)") //选择索引等于1的input元素，索引从0开始（返回单个）

$("input:gt(1)") //选择索引大于1的input元素，索引从0开始（返回集合）

$("input:lt(1)") //选择索引小于1的input元素，索引从0开始（返回集合）

$(":header") //选择所有标题元素，如h1,h2,h3（返回集合）

$("div:animated") //选择正在执行动画的div元素（返回集合）

$(":focus") //选择当前获取焦点的元素（返回集合）

●过滤选择器 内容过滤选择器

$("div:contains('我')") //选择含有文本“我”的div元素（返回集合）

$("div:empty") //选择不包含子元素（可包含文本元素）的div空元素（返回集合）

$("div:has(p)") //选择含有p元素的div元素（返回集合）

$("div:parent") //选择拥有子元素的（包含文本元素）的div元素（返回集合）

●过滤选择器 可见性过滤选择器

$(":hidden") //选择所有不可见的元素（返回集合）

$("div:visible") //选择所有可见的div元素（返回集合）

●过滤选择器 属性过滤选择器

$("div[id]") //选择拥有属性id的元素（返回集合）

$("div[title=test]") //选择属性title为"test"的div元素（返回结合）

$("div[title!=test]") //选择属性title不等于“test”的div元素（没有属性title的也会被选取）（返回集合）

$("div[title^=test]") //选择属性title以“test”开始的div元素（返回集合）

$("div[title$=test]") //选择属性title以“test”结束的div元素（返回集合）

$("div[title\*=test]") //选择属性title含有“test”的div元素（返回集合）

$("div[title|='test']") //选择属性title等于“test”或以“test”为前缀（该字符串后跟一个连接字符“-”）的div元素（返回集合）

$("div[title~='test']") //选择属性title用空格分隔的值中包含字符“test”的div元素（返回集合）

$("div[id][title$='test']") //选择拥有属性id，并且属性title以”test“结束的div元素（返回集合）

●过滤选择器 子元素过滤选择器

$("ul li:nth-child(2)") //选择每个ul的索引第2个子li（返回集合）

//eq(index)只匹配一个元素，索引从0开始。而nth-child将为每一个父元素匹配子元素，并且nth-child(index)的index索引从1开始。

$("ul li:first-child") //选择每个ul中第1个li元素（返回集合）

$("ul li:last-child") //选择每个ul中最后一个li元素（返回集合）

$("ul li:only-child") //选择ul中是唯一子元素的li元素（返回集合）

●过滤选择器 表单对象属性过滤选择器

$("#form1:enabled") //选择id为“form1”的表单内的所有可用元素（返回集合）

$("#form2:disabled") //选择id为“form2”的表单内的所有不可用元素（返回集合）

$("input:checked") //选择所有被选中的input元素（返回集合）

$("select option:selected") //选择所有被选中的选项元素（返回集合）

●表单选择器

$(":input") //选择所有input, textarea, select, button元素 （返回集合）

$(":text") //选择所有单行文本框 （返回集合）

$(":password") //选择所有密码框（返回集合）

$(":radio") //选择素有单选框（返回集合）

$(":checkbox") //选择所有复选框（返回集合）

$(":submit") //选择所有提交按钮（返回集合）

$(":image") //选择所有图像按钮（返回集合）

$(":reset") //选择所有重置按钮（返回集合）

$(":button") //选择所有按钮（返回集合）

$(":file") //选择所有上传域（返回集合）

$(":hidden") //选择所有不可见元素（返回集合）

■DOM操作

●DOM操作分类

DOM Core（核心）

HTML-DOM

CSS-DOM

●查找节点

查找节点包括查找元素节点和查找属性节点。

◎查找元素节点就是选择器的使用。

◎查找属性节点使用attr()方法：

$("p").attr("title") //获取P元素节点的title属性

●创建节点

◎创建元素节点。使用$(html)方法

$("<li></li>") //创建一个li元素节点

◎创建文本节点

$("<li>香蕉</li>") //创建一个包含文本节点的li元素节点，"香蕉"就是文本节点

◎创建属性节点

$("<li title='水果名'>香蕉</li>") //创建属性节点

●插入节点

插入节点的方法有：

append()

appendTo()

prepend()

prependTo()

after()

insertAfter()

before()

insertBefore()

$("p").append("<b>你好</b>") //结果是<p>我想说<b>你好</b></p>

$("<b>你好</b>").appendTo("p") //结果是<p>我想说<b>你好</b></p>

$("p").prepend("<b>你好</b>") //结果是<p><b>你好</b>我想说</p>

$("<b>你好</b>").prependTo("p") //结果是<p><b>你好</b>我想说</p>

$("p").after("<b>你好</b>") //结果是<p>我想说</p><b>你好</b>

$("<b>你好</b>").insertAfter("p") //结果是<p>我想说</p><b>你好</b>

$("p").before("<b>你好</b>") //结果是<b>你好</b><p>我想说</p>

$("<b>你好</b>").insertBefore("p") //结果是<b>你好</b><p>我想说</p>

●删除节点

◎remove()方法：

remove()方法删除节点以及该节点的所有后代节点。返回一个指向已被删除节点的引用，将来可以再次使用它们（但是所有绑定的事件，附加的数据等就没有了）。

var $li=$("ul li:eq(1)").remove(); //删除节点后，remove()方法返回一个指向已被删除节点的引用。可以在以后再使用这些元素。

$li.appendTo("ul"); //把刚才删除的节点又重新添加到ul元素里。

◎detach()方法

detach()方法删除节点以及该节点的所有后代节点。返回一个指向已被删除节点的引用，将来可以再次使用它们（与remove()不同，所有绑定的事件，附加的数据等都会保留下来）。

◎empty()方法

empty()方法是清空节点，它能清空元素中的所有后代节点。

●复制节点clone()

$("ul li").click(function(){

$(this).clone().appendTo("ul"); //复制当前单击的节点并将它追加到ul元素中

})

●替换节点

replaceWith()

replaceAll()

$("p").replaceWith("<strong>你喜欢吃水果马？</strong>"); //把p元素替换成<strong>你喜欢吃水果马？</strong>

$("<strong>你喜欢吃水果马？</strong>").replaceAll("p"); //效果同上，把p元素替换成<strong>你喜欢吃水果马？</strong>

●包裹节点

$("strong").wrap("<b></b>") //得到<b><strong>文本内容</strong></b>

$("strong").wrapInner("<b></b>")

//得到<strong><b>文本内容</b></strong>

$("li").wrapAll("<b></b>")

//得到<b><li>1</li><li>2</li><li>3</li></b>

●属性操作

◎属性的获取和设置

$("#d1").attr("title") //获取属性

$("#d1").attr("title","苹果") //设置单个属性

$("#d1").attr({"title":"苹果", "name":"水果1"}) //设置多个属性

◎删除属性

$("p").removeAttr("title");

●样式操作

$("p").attr("class") //获取p的class

$("p").attr("class","dd1") //设置p的class为dd1

$("p").addClass("dd2") //为p增加dd2的class

$("p").removeClass("dd1 dd2") //移除p的dd1和dd2的class

$("p").removeClass() //移除p的所有class

$("p").toggleClass("dd3") //为p提供切换样式。如果有dd3就移除，如果没有dd3就添加

$("p").hasClass("dd5") //判断p是否有dd5的class

●获取HTML, 文本和值

$("p").html() //获取p的html

$("p").text() //获取p的文本

$("p").val() //获取p的值

●CSS-DOM操作

$("p").css("color") //获取p的颜色样式

$("p").css("color", "red") //设置单个CSS样式

$("p").css({"fontSize":"30px", "backgroundColor":"#888888"}) //设置多个CSS样式

$("#d1").height() //获取d1的高度，等同于$("#d1").css("height")

$("#d1").width() //获取d1的高度，等同于$("#d1").css("width")

◎offset()方法

◎position()方法

◎scrollTop()方法和scrollLeft()方法

●遍历节点

$("#d1").children() //匹配d1的所有子元素

$("p").next() //取得紧邻p元素后的同辈元素

$("p").prev() //取得紧邻p元素前的同辈元素

$("p").siblings() //取得p元素前后的所有同辈元素

$("p").parent() //取得p元素的父级元素

$("p").parents() //取得p元素的祖先元素（爸爸，爷爷，祖宗）

■事件

●事件绑定

bind(type[, data], fn);

bind()方法有三个参数：

第一个参数type是事件类型，包括：blur, focus, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error等

第二个参数是可选参数，作为event.data属性值传递给事件对象的额外数据对象。

第三个参数是用来绑定的处理函数

//正常写法

$("#d1").bind("click", function(){

alert(11);

})

//简写形式

$("#d1").click(function(){

alert(11);

})

●合成事件

◎hover()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<h1>标题1</h1>

<p>标题1的内容</p>

<script type="text/javascript">

$("p").hide();

$("h1").hover(function(){

$(this).next().show();

}, function(){

$(this).next().hide();

});

</script>

</body>

</html>

◎toggle()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<h1>标题1</h1>

<p>标题1的内容</p>

<script type="text/javascript">

$("p").hide();

$("h1").bind("click", function(){

$(this).next().toggle(); //切换元素可见和不可见的状态

});

</script>

</body>

</html>

●阻止事件冒泡

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<p>内容<span>里面的内容</span></p>

<script type="text/javascript">

$("p").bind("click", function(){

alert(11);

});

// 以下的结果,如点击span则会先弹出22然后冒泡弹出11

// $("span").bind("click", function(){

// alert(22);

// });

// 以下使用阻止冒泡,点击span则只会弹出22

$("span").bind("click", function(event){

alert(22);

event.stopPropagation();

});

</script>

</body>

</html>

●阻止默认行为

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<form>

<input type="text" name="xingming" id="d2"/>

<input type="submit" id="d1"/>

</form>

<script type="text/javascript">

$("#d1").bind("click", function(event){

var $key=$("#d2").val();

if($key=="123"){

event.preventDefault(); //如果输入123则取消submit的默认行为，不提交表单，可用于表单验证

}

});

</script>

</body>

</html>

●移除事件

$("#d1").unbind("click"); //移除绑定在d1上的所有click事件

如果有很多个同样类型的事件绑定在一个元素，可以用以下的方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<form>

<ul>

<li>点击我</li>

</ul>

</form>

<script type="text/javascript">

$("li").bind("click", myFn1 = function(){

$("ul").append("<li>插入第一条</li>");

}).bind("click", myFn2 = function(){

$("ul").append("<li>插入第二条</li>");

}).bind("click", myFn3 = function(){

$("ul").append("<li>插入第三条</li>");

})

// 移除myFn2事件

$("li").unbind("click", myFn2);

</script>

</body>

</html>

●trigger()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<form>

<p>点击我</p>

</form>

<script type="text/javascript">

$("p").bind("click", function(){

alert(11)

});

$("p").trigger("click"); //可自动出发p的点击事件

$("p").click(); //简写形式

</script>

</body>

</html>

●one()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<form>

<p>点击我</p>

</form>

<script type="text/javascript">

$("p").one("click", function(){

alert(11); //由于用one绑定的click事件，只会出发一次

});

</script>

</body>

</html>

■动画

●show()和hide()

●fadeIn()和fadeOut()

●slideUp()和slideDown()

●自定义动画animate()

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

/\*必须把position设置成relative或absolute,动画才能生效\*/

position: relative;

}

</style>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<div id="d1"></div>

<script type="text/javascript">

$("#d1").bind("click", function(){

$(this).animate({left:"+=100px", top:"+=50px"},1000);

});

</script>

</body>

</html>

●停止动画

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

/\*必须把position设置成relative或absolute,动画才能生效\*/

position: relative;

}

</style>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<p>停止动画</p>

<div id="d1"></div>

<script type="text/javascript">

$("#d1").bind("click", function(){

$(this).animate({left:"+=100px", top:"+=50px"},5000);

});

$("p").bind("click", function(){

$("#d1").stop(); //停止动画

});

</script>

</body>

</html>

●判断是否处于动画状态

if(!$(element).is(":animated")){

}

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

/\*必须把position设置成relative或absolute,动画才能生效\*/

position: relative;

}

</style>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<p>停止动画</p>

<div id="d1"></div>

<script type="text/javascript">

$("#d1").bind("click", function(){

$(this).animate({left:"+=100px", top:"+=50px"},5000);

});

$("p").bind("click", function(){

if(!$("#d1").is(":animated")){

alert("目前没有动画"); //如果d1没有处于动画中，则弹出alert

};

});

</script>

</body>

</html>

●延迟动画delay()

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

/\*必须把position设置成relative或absolute,动画才能生效\*/

position: relative;

}

</style>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<div id="d1"></div>

<script type="text/javascript">

$("#d1").bind("click", function(){

//下面的动画是，向右下方移动后，停顿2秒，再向左上方移动

$(this).animate({left:"+=100px", top:"+=50px"},1000).delay(2000).animate({left:"-=50px", top:"-=25px"},1000);

});

</script>

</body>

</html>

■编写jQuery插件

●jQuery的闭包

// 为了更好的兼容性,在开始前加个分号

;(function($){ //此处的$作为匿名函数的形参

//此处编写jQuery插件的代码，可以使用$作为jQuery的缩写别名

})(jQuery) //通过最后这个（）运算符来执行函数。括号里将jQuery作为实参传递给了匿名函数

●三种jQuery插件方法简介

◎封装对象方法的插件

此方法用来对通过选择器获取的jquery对象进行操作，例如$(“p”).parent()

使用jQuery.fn.extend()方法来实现：

// 为了更好的兼容性,在开始前加个分号

;(function($){ //此处的$作为匿名函数的形参

//此处编写jQuery插件的代码，可以使用$作为jQuery的缩写别名

$.fn.extend({

"color":function(value){ //编写一个名为color的插件

//这里写具体代码

}

});

})(jQuery) //通过最后这个（）运算符来执行函数。括号里将jQuery作为实参传递给了匿名函数

// 以上代码可以简写成以下形式

;(function($){ //此处的$作为匿名函数的形参

//此处编写jQuery插件的代码，可以使用$作为jQuery的缩写别名

$.fn.color = function(){ //编写一个名为color的插件

//这里写具体代码

};

})(jQuery) //通过最后这个（）运算符来执行函数。括号里将jQuery作为实参传递给了匿名函数

◎封装全局函数的插件

将对立的函数加到jquery的命名空间下，例如jQuery.noConflict()

使用jQuery.extend()方法来实现：

◎选择器插件

使用jQuery.extend()方法来实现：

■Ajax

●load()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<h1>题目1</h1>

<div id="d1"></div>

<script type="text/javascript">

$(function(){

$("h1").bind("click", function(){

$("#d1").load("ceajax2.html");

});

});

</script>

</body>

</html>

●序列化

$(选择器).serialize()方法将jQuery对象转化成字符串

$(选择器).serializeArray()方法将jQuery对象转化成JSON对象

$.param()方法将数组或对象按照key/value进行序列化

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<form id="form1">

<span>姓名：</span><input type="text" name="name" id="na"/><br/>

<span>密码：</span><input type="password" name="password" id="pass"/><br/>

<span>多余：</span><input type="text" id="duo"/><br/>

<input type="button" id="btn" value="提交"/><br/>

</form>

<p></p>

<script type="text/javascript">

$("#btn").bind("click", function(){

var dd=$("#form1").serialize(); //转化成字符串

var dd2=$("#form1").serializeArray(); //转化成json对象

//[{…}, {…}]

//0:{name: "name", value: "aa"}

//1:{name: "password", value: "bb"}

//length:2

alert(dd1);

alert(typeof dd1);

console.log(dd1);

alert(dd2);

alert(typeof dd2);

console.log(dd2);

$.each(dd2, function(index, element) {

$("p").append("<span>" + index + element.name + element.value + "</span><br/>")

});

});

//param方法

var b={a:1, b:2, c:3};

var p=$.param(b);

alert(p);

</script>

</body>

</html>

●jQuery种的ajax全局事件

ajaxStop(callback)

ajaxStart(callback)

ajaxComplete(callback)

ajaxError(callback)

ajaxSend(callback)

ajaxSuccess(callback)

■.each()方法

●jQuery对象的.each()方法

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<ul>

<li>内容1</li>

<li>内容22</li>

<li>内容333</li>

</ul>

<p></p>

<script type="text/javascript">

// index为索引位置,element为当前元素也可以用this

// $(selector).each(function(index,element){})

$("li").each(function(index, element){

var tt=$(element).text(); //也可以用$(this).text()

$("p").append("<span>" + index + tt + "</span><br/>");

});

</script>

</body>

</html>

●全局的$.each()方法

先获取某个集合对象，例如： var d=$(“li”);

然后再使用$.each()方法：

$.each(d, function(index, element){})

// d是要遍历的对象;index是索引号;element为当前元素也可以用this

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>

</head>

<body>

<ul>

<li>内容1</li>

<li>内容22</li>

<li>内容333</li>

</ul>

<p></p>

<script type="text/javascript">

var d=$("li");

$.each(d, function(index,element){

var nn=$(element).text();

$("p").append("<span>" + index + nn + "</span><br/>");

});

</script>

</body>

</html>

z