**二十二 、封装MQuery**

**一、$函数**

**【思路】：**

1、参数是函数 function

$(function(){

//window.onload

alert(123);

})

$(function(){

//window.onload

alert(456);

})

2、参数字符串 string

#id

.class

div

[title=hello]

[name=xxx]

3、JS元素节点 this document window

//object

=======================================================

function MQuery(vArg){

//声明一个数组，存储选中的元素节点

this.elements = [];

switch(typeof vArg){

case "function":

addEvent(window, "load", vArg);

break;

case "string":

//判断首字母

switch(vArg[0]){

case "#": //id

var node = document.getElementById(vArg.substring(1));

this.elements.push(node);

break;

case ".": //class

var nodes = elementsByClassName(document, vArg.substring(1));

this.elements = nodes;

break;

case "[": //属性 [title=hello] [name=world];

var start = vArg.indexOf("=");

var key = vArg.substring(1, start);

var value = vArg.substring(start + 1, vArg.length - 1);

var nodes = elementsByKey(document, key, value);

this.elements = nodes;

break;

default: //tagName

var nodes = document.getElementsByTagName(vArg);

this.elements = nodes;

break;

}

break;

case "object":

this.elements.push(vArg);

break;

default:

alert("error");

break;

}

}

function $(vArg){

return new MQuery(vArg);

}

//通过属性=属性值获取元素节点

function elementsByKey(parentNode, key, value){

var result = [];

var nodes = parentNode.getElementsByTagName("\*");

for(var i = 0; i < nodes.length; i++){

if(nodes[i].getAttribute(key) == value){

result.push(nodes[i]);

}

}

return result;

}

/\*

通过className获取元素节点

\*/

function elementsByClassName(parentNode, className){

var result = []; //用于存储符合条件的元素节点

//1、获取当前节点下，所有的节点

var nodes = parentNode.getElementsByTagName("\*");

for(var i = 0; i < nodes.length; i++){

if(nodes[i].className == className){

//2、存储

result.push(nodes[i]);

}

}

return result;

}

//添加事件，跨浏览器的兼容写法

function addEvent(obj, type, func){

if(obj.addEventListener){

obj.addEventListener(type, func, false);

}else{

obj.attachEvent("on" + type, func);

}

}

function removeEvent(obj, type, func){

if(obj.removeEventListener){

obj.removeEventListener(type, func);

}else{

obj.detachEvent("on" + type, func);

}

}

**二、事件（click）**

**通过给MQuery对象添加方法来添加事件**

MQuery.prototype.click = function(func){

//遍历已经获取到元素节点,添加对应的事件

for(var i = 0; i < this.elements.length; i++){

addEvent(this.elements[i], "click", func);

}

}

**三、JQuery\_css**

MQuery.prototype.css = function(){

switch(arguments.length){

case 1:

if(typeof arguments[0] == "object"){

//json赋值

for(var i = 0; i < this.elements.length; i++){

//将json对象中所有的属性进行赋值

for(var attr in arguments[0]){

if(typeof arguments[0][attr] == "number"){

arguments[0][attr] += "px";

}

this.elements[i].style[attr] = arguments[0][attr];

}

}

}else{

//取值

return getStyle(this.elements[0], arguments[0]);

}

break;

case 2:

//赋值针对所有

for(var i = 0; i < this.elements.length; i++){

if(typeof arguments[1] == "number"){

arguments[1] += "px";

}

this.elements[i].style[arguments[0]] = arguments[1];

}

break;

default:

alert(error);

break;

}

}

function getStyle(element, attr){

return element.currentStyle ? element.currentStyle[attr] : getComputedStyle(element)[attr];

}

**四、JQuery\_attr**



**五、JQuery\_on**

MQuery.prototype.on = function(){

switch(arguments.length){

case 1: //json对象 on(json)

var json = arguments[0];

for(var i = 0; i < this.elements.length; i++){

for(var type in json){

addEvent(this.elements[i], type, json[type]);

}

}

break;

case 2: //绑定事件 第一个参数有可能是多个事件 on("click mouseover", func)

var types = arguments[0].split(" ");

for(var i = 0; i < this.elements.length; i++){

for(var j = 0; j < types.length; j++){

addEvent(this.elements[i], types[j], arguments[1]);

}

}

break;

case 3: //事件委托 on("click", "li", func)

var \_arugments = arguments;

for(var i = 0; i < this.elements.length; i++){

//给当前的节点添加事件，让第二个参数去相应

addEvent(this.elements[i], arguments[0], function(ev){

var e = ev || window.event;

var target = e.target || window.event.srcElement;

if(target.nodeName.toLowerCase() == \_arugments[1]){

\_arugments[2].call(target);

}

})

}

break;

default:

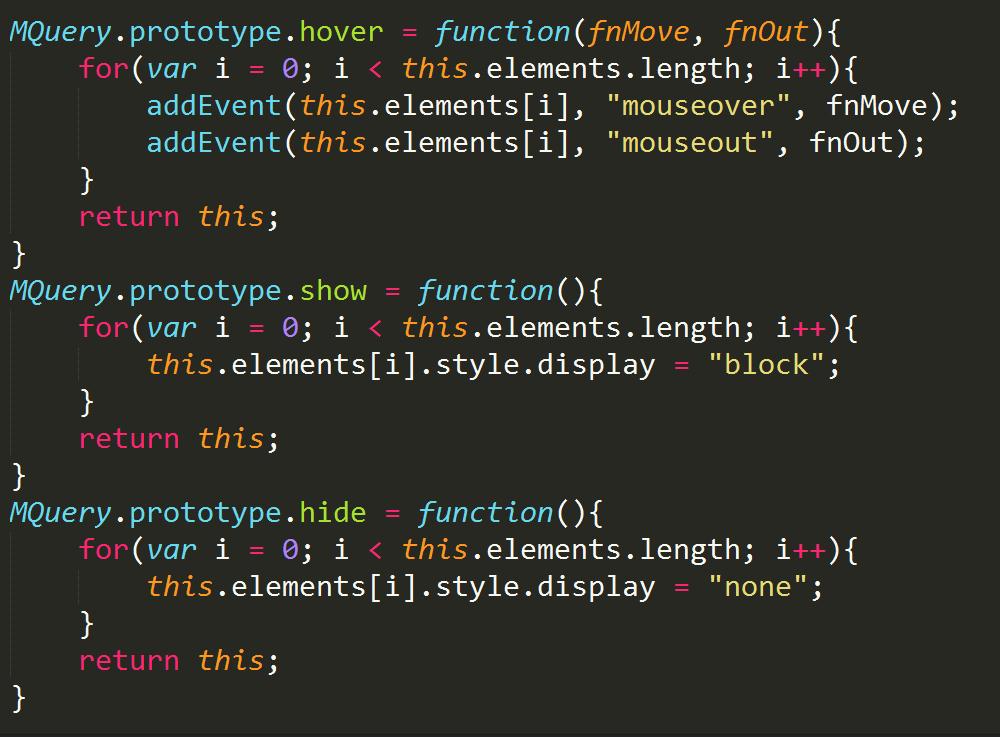
break;

}

return this;

}

**六、JQuery\_hover、show、hide**



**七、JQuery\_toggle**

**toggle是jquery-1.7.1中的方法**

**【注】参数个数是无限的，可以传任意的参数**

**参数都是函数**

**【函数执行的特点】：**

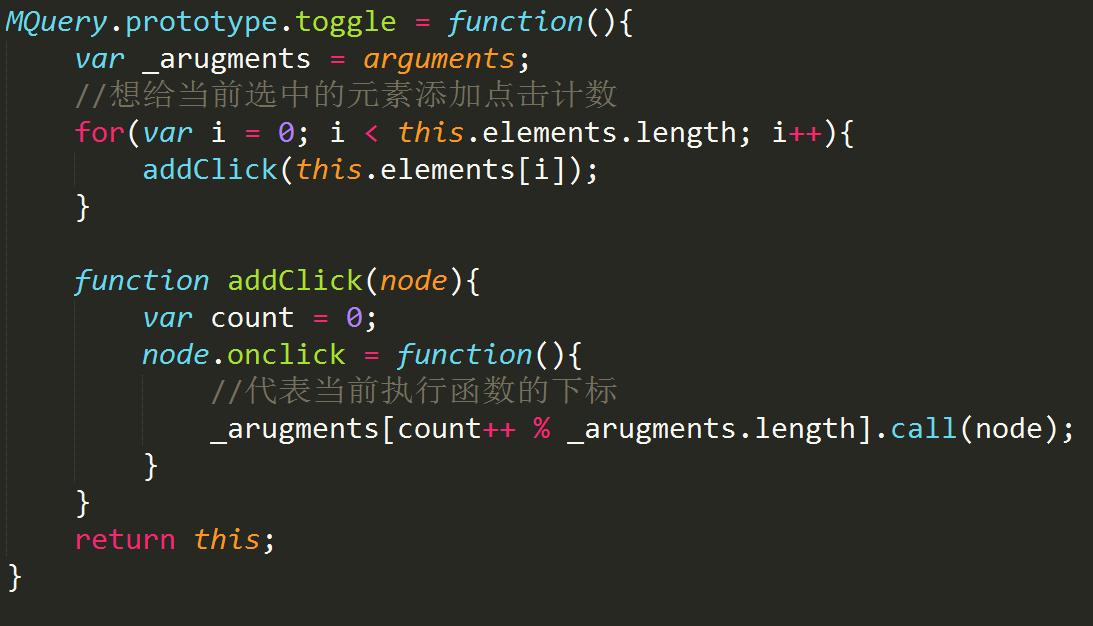
**<1>所有传参的函数，按照传参的顺序循环执行**

**<2>所有元素节点的函数执行顺序都是独立的。**

**【JS实例】点击计数（所有按钮都有一个独立的计数，%3之后可循环进行0,1,2的输出）**

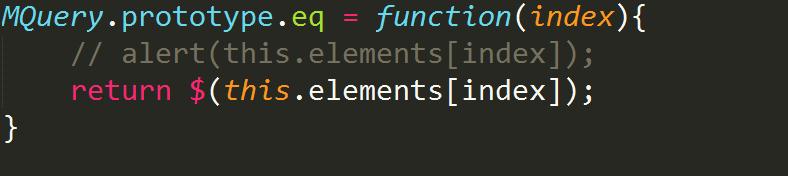


**【封装toggle方法】**



**八、JQuery\_eq**

**【1、通过下标获取元素；2、返回值是JQ对象】**



**九、JQuery\_find**

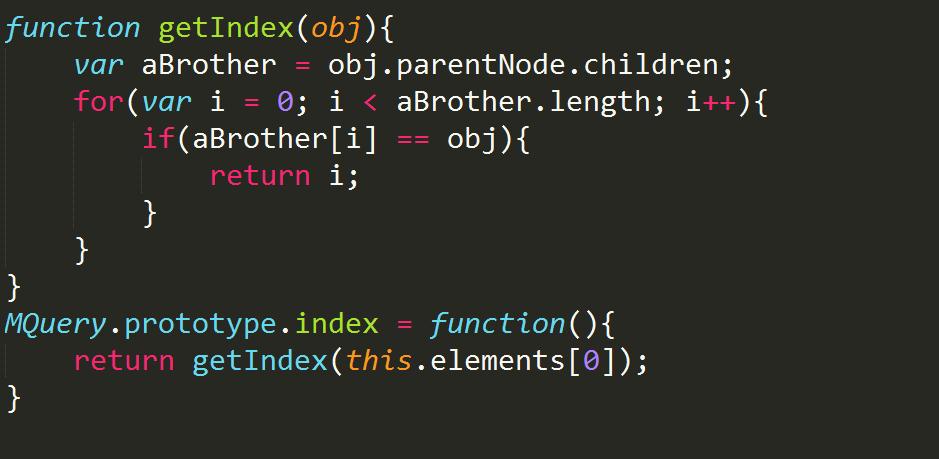
**1、find方法返回值是JQ对象**

**2、find找到子节点下，符合条件的元素。**

**参数: class tagName**



**十、JQuery\_index（获取元素下标）**



**十一、JQuery\_链式操作**

**【例1】**

$("#div1").css({width: 300, height: 300, background: "red"}).click(function(){

alert(123);

});

**【例2】函数的链式操作**

function show(str){

alert(str);

return show;

}

show("hello")("world")("xxx")("yyy");

【注】由于show的返回值为其本身这个函数，所以可不断的调用。

**结论：调用函数以后，返回值是一个JQuery**

**【封装MQuery解决】**

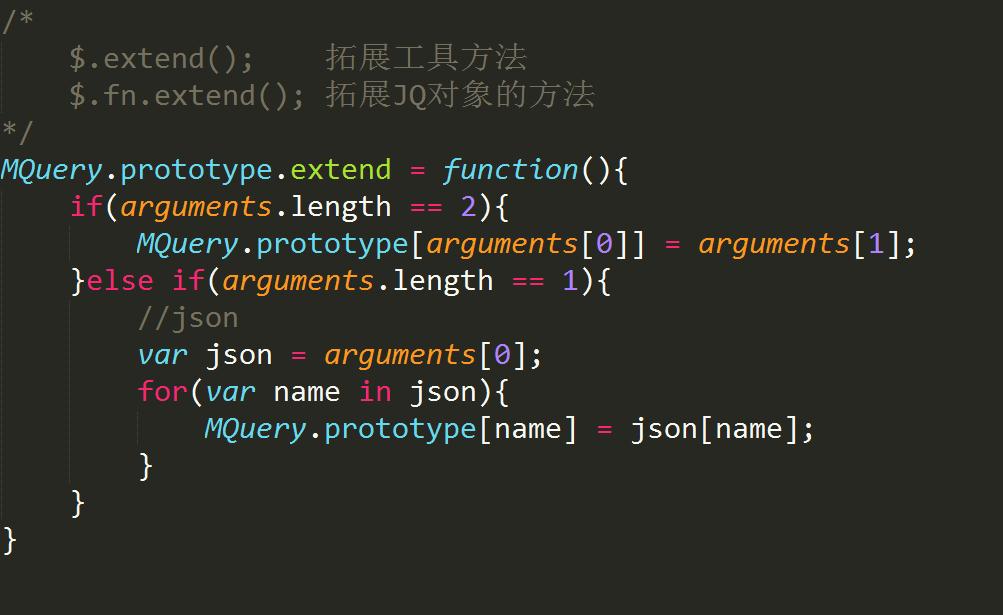
在封装的MQuery中的每一个MQuery.prototype函数最后都加**return this;**

由于this为当前函数的主人，是一个MQuery对象，所以**return this;**后其返回值为MQuery对象。

**十二、MQuery\_animate**

**可通过extend拓展animate函数**

**【封装extend函数】**



**【通过$().extend来调用这个函数】**

**【注】$().extend中的括号内可写内容，如果不写，会输出error,但是函数正常执行，如果删掉MQuery(vArg)中**

**default:**

**alert("error");**

**break;**

**的alert（"error"）可解决报错问题。**

