基于立即执行函数

(function (global, factory) {} (typeof window !==”undefined” : window ? This, function (window, noGlobal) {

}))

typeof window !==”undefined” : window ? This不用管，兼容以前的window可以被赋值为undefined

extend()为重，创建jQuery函数

window.jQuery = window.$ = jQuery，把接口留了出来，window上的对象jQuery，window.$()或$()

jQuery = function (selector, context) {

return new jQuery.fn.init(selector, context);

}

$()事会返回一个新的对象---->this

init = jQuery.fn.init = function (selector, context, root) {

}

init.prototype = jQuery.fn; return出来的对象继承了jQuery的方法

jQuery.fn = jQuery.prototype = {};

相当于init.prototype = $.prototype

基本框架：

(function () {

jQuery = function () {

return new jQuery.prototype.xxx();

}

jQuery.prototype = {xxx:function(){}}

window.$ = window.jQuery = jQuery;

var xxx = jQuery.prototype.xxx;

xxx.prototype = jQuery.prototype;

}())

init函数实现$()选择和创建的过程

get: function () {

Return num !== null ? (num < 0 ? this[num + this.length] : this[num]) : slice.call(this);

}

$()

$(‘#id’) $(‘.class’) $(‘.class’, ’上下文’) $(‘li’, ‘ul’) $(‘<div></div

>’) $(DOMElement) $(function () {}) $($(‘div’)) $(‘<div></

div>’, {html:’’(会加进div标签内), title:’’(将title变为div标签特性)})

$(‘div’)返回的对象是一个类数组

$.merge()拼接数组类数组，数组数组，类数组类数组

get()添负数从后倒着来，null,undefined这样等于null的返回的是包含所有元素的数组(从类数组截得)

pushStack( elem )推进栈，jQuery自用函数

把elem和一个空jQuery对象(this.constructor)merge成一个类数组，再return出来，再为了end()写一个ret.prevObject = this

end()回退操作，回退到上一个选择元素的地方，$(‘div’).eq(1)相当于栈操作，先进$(‘div’)再进eq(1)，回退操作栈的底层，代码就是return prevObject也就是说只要调用了pushStack的就可以进行回退操作

.extend()利用深度克隆

$.extend({abc:function(){}}) $.abc()这样添加为工具方法

$.prototype.extend = $.extend = function(json){

for(var key in json){

this[key] = json[key];}(this指代$或$.prototype)

}

$.prototype.extend({bcd:function(){}}) $().bcd() 这样添加为原型链上的方法，得$()执行后返回一个类数组对象才能用

$.extend(target, obj, obj1)将obj和obj1克隆到target，浅拷贝属性名相同不管什么情况都是后面的覆盖前面的，深拷贝时

var target = {name:{a:111}} var obj = {name:{b:222}}

$.extend(true, target, obj) target会变成{name:{a:111, b:222}}

没有的属性会加进去

当参数不固定时用arguments来表示

If( obj[name] === target ){ continue; }防止循环引用