jQuery

一个非常优秀的js库

重在封装思想，使用方法会用就可以现查现用，一切的方法都添加在jQuery的原型对象上

zepto(适合移动端)从jQuery演化过来

只是辅助工具不能代替js

jQuery很庞杂，重点学习思想和用法

jQuery应用方法很多，应把常用的记熟，其它现查现用

原生dom对象无法调用jQuery的方法

选择元素：$() 1.$()--->css selector 2.$()--->原生dom

$(‘li’/oLi(原生)/.class).css({‘background’:’red’, ‘width’:’40px’;});

3.$()--->null不会报错返回undefined

4.$(selector, context)--->在context下匹配selector $(‘li’, ‘.oUl’)或

$(‘oUl li’)

1. $( $() )jQuery对象里可以调用jQuery对象
2. $(function () {}) DomContentLoaded，dom解析完执行

jQuery特有的选择规则

$(‘ul>li:first’); 类数组中第一个，还有last

$(‘ul>li:eq(2)’); 类数组中第几个或$(‘li’).eq(1)

$(‘li’).get(1)转换成原生的js对象可以调用原生的方法

$(‘ul>li:odd/even’); odd选出索引是奇数的元素 event选偶数

filter(‘.box/[title=hello]’); $(‘li’).filter(‘.class’)过滤留下class的li

$(‘li’).filter(function (index) {})，假如有20个li就循环20次，index

从0-20变，函数return一个值，为true就过滤出来

$(‘li’).has(‘div’); 有div子元素的li被选出来 前后都可以写原生

is(); 判断前后选出的是否是一个元素，返回值为Boolean

not(); $(‘li’).not(‘.class’)除了.class的元素都被选出来

.not(function (index) {}) 函数返回true的去掉，index为索引

$(‘ul’).find(‘li’); 在前一个基础上向下查找

.html() 取值取一个，赋值赋一组 .html(function (index) { })

例如选中十个li，就会循环10次，添加到元素中的值的值为函数返回的值，index值从0-9

$(‘body’).html(‘<div style=”width:100px; height:100px;”></div>’);

$(‘li’).text() 文本

css() 取值相当于getComputedStyle，赋值相当于ele.style.xxx

css(‘background’)它就会返回rgb()

attr()主要依赖于Element对象的get/setAttribute()两个方法

不管是属性还是特性在ele上都会有显示

prop()主要依赖于js中原生对象的属性获取和设置方式

设置:$(‘ul’).prop(‘rich’, ‘hgf’)属性时无映射，特性有

获取:$(‘ul’).prop(‘rich’)能取到，但是ele不会有显示

注意:attr的checked,select,disabled就是表示该属性初始状态的值，返回checked,select,disabled

prop的checked,select,disabled才表示该属性的实时状态值，返回true或false

next()下一个兄弟节点 prev()上一个兄弟节点

index()当前节点中的索引 $(‘li’).index(原生dom)返回这个dom的索引

before/after xxx在\*\*\*前面 insertBefore 把XXX放在\*\*\*前面

append 在xxx里添加\*\*\* appendTo 把xxx添加到XXX里

prepend prependTo(添加到第一个元素前)

把元素位置插好后还想链式调用其他方法，想用哪个元素就可以用

remove() detach()把元素删除后返回，detach会保留函数的事件

注册事件 on(‘click’, ‘li:odd’, data, function () {]) 解绑事件off()

还可以传个参数data，在事件对象e上找到它

$(‘li’).on(‘mouseup’, up).on(‘mousedown’, down1).on(‘mousedo

wn’, down2) 执行顺序为up,down1,down2

$(‘li’).on({mousedown:down1, mouseup:up, mousedown:down2}),

顺序为up,down2，同时注册事件同类型事件会覆盖

off()不填参数解绑所有事件 off(‘mousedown’, down1)只解绑down1一个函数

怎么绑定的函数就要怎么解绑如$(‘ul’).on(‘mouseup’, ‘li’, up)，解绑为$(‘ul’).off(‘mouseup’, ‘li’, up)

innerWidth() 除了border的可视区域包括padding

outerWidth() 包含border的可视区域 传个true为参数连margin也算上

注册框：$(‘<div></div>’)可以创造一个div标签或$(‘<div class=”

wrapper” />’)

关于事件

1. e->e.pageX/Y e.lientX/Y 距离顶部的距离，page算上了滚动条距离

e.which e.button

1. return false 既能阻止事件冒泡又能阻止默认事件
2. one()绑定事件，只绑定一次

offset().left/top相对于浏览器边框 position().left/top 注意没有定位的情况，且是相对于有定位的父级

parent()找到父级标签 offsetParent()找到最近有定位的父级

size() $(‘li’).length都是返回有几个li

each()遍历操作元素组，$(‘li’).each(function (index, ele) {})index为从头开始的索引，ele为相应的元素

hover(function () {}, function () {})鼠标移入触发第一个函数，移出触发第二个函数

(hide(1000)隐藏元素 show(1000, function() {})展示元素，参数为多长时间运动到原先元素大小，还可以传一个回调函数，运动完毕后执行

fadeIn/Out()淡入淡出，透明度改变

sildeDown/Up()卷入卷出，上下卷)这些运动函数工作一般不用

end()回退操作 $(‘li’).eq(3).css().end().eq(4) 从eq(3)回退到li再eq(4)，方便链式调用，退回到上个元素

parents()查找祖先元素 不填参数选出所选元素的所有祖先元素，

包括html标签，parents(‘div’/标签选择器)祖先元素是div才被选中

closest()查找最近的祖先元素，参数可以写自己，会选到自己

clone()普通克隆 clone(true)深度克隆，包括事件

wrap(原生dom/jq对象/’<div></div>’) 在操作元素外面加个父级元素，是复制一个要加的父级元素，不是剪切，有几个操作元素就加几个父级元素，wrap(function (index) {return ‘<div></div>’})

wrapAll() 给选出的整组操作元素添加父级元素，会破坏dom结构强制让操作元素变为父级元素的子元素

wrapInner() 把填的元素放到操作元素里变为操作元素里的子元素的父元素

unwrap()直到body就不能再解除父级元素了

add() $(‘li’).add(‘p’).css() 将它们合在一起操作

slice() $(‘li’).slice(0,4) 选出索引为0,1,2,3的元素

$(‘form’).serilize() 将数据拼接为字符串

$(‘form’).serilizeArray 将数据转换为数组

animate({位置}，1500时间，’swing’或默认的线性运动或运动插件里的，function () {}回调函数动到指定位置后运行该函数)

stop()阻止当前运动继续后续运动 stop(true)阻止后续所有运动

stop(true,true)停止当前所有运动，停在当前目标点

finish()直接运动到要到的位置

delay(1000)过1000毫秒再运动

bind()废弃 on()注册事件

trigger(翻译引起其他事件的事件)() $(‘div’).on(‘cilck’, function(e

, dom(从trigger传过来的想传几个都行))主要用来触发自定义事件，冒泡机制还是会有

$(‘li’).on(‘click’, function(){$(‘div’).trigger(‘click’)})在函数内写条件，执行trigger时触发div的事件

$(‘div’).on(‘eat’, function(e, data){}) $(‘button’).on(‘click’, function

(){var data = {}; data.count = this.num; data.dom = this; $(‘div’).trig

ger(‘eat’, data))

工具方法，不需要获取dom就可用的方法

原生js判断是否为数组的方法1.Object.prototype.toString.call(arr)

== “[object,Array]”

1. arr instanceof Array 3.arr.constructor == “Array”

子页面的数组在父页面无法判断，结果是false，但依旧可以用Object.prototype.toString.call(arr)来判断

$.type()，null类型是null，array类型是array

$.trim() 消除空格

$.proxy() 改变指针方向，参数一要改的function 参数二：指向哪个对象 原生bind(this)改变this指向当前对象

var show1 = $.proxy(show, document) show1的上下文this变为

document，show的上下文this还是原先的

$.noConflict() 防止冲突 很多JavaScript库用$作变量或者函数名，就像jQuery，$只是一个别名，所以不用依旧有效，如果需要同时使用jQuery和其他库，我们可以用$.noConflict()把$控制权交给其他库

var Cst = $.noconflict(); Cst(‘’).css()

$.parseJSON(str) 必须严格按照JSON格式写才能转

var str = ‘{“name”:”hgf”, “age”:18 }’

$.makeArray()把类数组转换为数组

插件扩展

$.extend({fn: function () {}})提供新的jQuery工具方法

$.fn.extend({fn: function () {}})提供新的实例方法，是实例不是工具方法

var cb = $.Callbacks() 回调函数---->返回回调函数对象

cb.add(function aaa() {})给回调对象添加函数

add的添加方式add(aaa,bbb)或add([aaa, bbb])

cb.fire()---->触发添加的函数，多次fire多次触发，fire可以传参，

为函数实参

在局部作用域里用cb.add()函数也会触发

$.Callbacks() 四个参数 ’once’不管有几个fire只一次 ‘memory’ 记忆功能，不管在哪fire在哪add，只要add函数了就会执行 ‘unique’ 相同的元素添加多次只执行一次 ‘stopOnFalse’不再运行返回了false的函数的后面的函数

$.Deferred()延迟对象

添加对象done fail progress

触发函数 resolve(解决) reject(拒绝) notify(通知)

相当于有状态的callback，三种方式一一对应，前两种相当于自带once memory，第三种为memory，progress添加的函数可以执行多次直到done(成功)或fail(失败)为止

$.Deferred().promise() 不能触发函数无状态

$.when() 返回的延迟对象，参数也是延迟对象，当参数延迟对象全部触发done的时候最后会触发when返回延迟对象的done函数，如果有一个reject就触发fail否则触发progress

1. 延迟对象，作用控制关键点，做链式运动效果
2. $.ajax这种有状态的异步操作核心是deferred
3. when当发送多个ajax请求时，因为都是异步，可以使用多个

Deferred，每当ajax成功后再触发一个，最后全部成功，可以在

when后一并处理

$.when(df1, df2, df3).done().fail().progress()

$.ajax({}) type：’GET’/’POST’ url：’’ data success：function(){}

error：function(){} context：改变this指向 (dataType：’jsonp’一般请求不用添加，请求jsonp时须填写jsonp) jsonp：’callback’ cache：是否缓存，默认为true，可以改为false

也能发送JSONP请求

sizzle()