jQuery

1 jQuery的优势：轻量级、强大的选择器、出色的DOM操作及其封装、可靠地事件处理机制、完善的Ajax、不污染顶级变量、出色的浏览器兼容性、链式操作方式、隐式迭代、行为层和结构层的分离、丰富的插件支持

2 释放$变量：jQuery.noConflict();

3 相关方法

匹配指定索引值的元素 eq(n)

设置css样式 css()

设置html代码 html()

设置文本内容 text()

把元素添加到已存在的元素组合中 add()

遍历数据对象 $.each(data, function(index, obj) {})

4 标签选择器

常见的有基本选择器、多项选择器、层级选择器、属性选择器、过滤器等

id $('#')

element $(element)

class $('.class')

通配符 $('\*')

多项 $('a, b, c')

祖先后代 $('ancestor descendant')

直接后代 $('parent > child')

next $('prev + next')

兄弟 $('prev ~ siblings')

属性名 $('[attribute]')

属性值 $('[attribute = value]')

非属性值 $('[attribute != value]')

属性值前缀 $('[attribute ^= value]')

属性值后缀 $('[attribute $= value]')

包含属性值 $('[attribute \*= value]')

多个属性值 [selector1][selector2][selectorN]

5 过滤器（伪类选择器）

$(选择器:过滤器')

child系列（先统计个数n，再匹配标签类型）

:first-child

:last-child

:nth-child(n | even | odd | formula)

:nth-last-child(n | even | odd | formula)

:only-child

type系列（先筛选符合类型的标签，再统计个数n）

:first-of-type

:last-of-type

:nth-of-type(n | even | odd | formula)

:nth-last-of-type(n | even | odd | formula)

:only-of-type

n从1开始

formula为计算公式，如an + b

6 表单选择器

可输入项目 :input 包括<input>、<textarea>、<select>、<button>

单行文本 :text 等同input[type='text']

其它input类型 :password/:radio/:checkbox/:image/:reset/:button/:file

可用状态 :enabled

不可用状态 :disabled

已选中 :checked

已选中（下拉框）:selected

7 查找和过滤

所有后代中元素 find(expr | object | element)

查找子代中元素 children([expr])

查找父代中元素 parent([expr])、parents([expr])

查找紧邻的元素 next([expr])、prev([expr])

查找指定索引值的元素 eq(n|-n)

查找所有同辈的元素 siblings([expr])

过滤所有元素 filter(expr | object | element | fn) fn为筛选条件函数

排除所有元素 not(expr | object | element | fn)

8 事件

method([[data], fn])

fn参数event

鼠标点击 click、dblclick

鼠标按下放开 mousedown、mouseup

鼠标进入移出 mouseenter、mouseleave

鼠标经过 hover([overFn,] outFn)

鼠标经过及其子元素 mouseover、mouseout

鼠标移动 mousemove

鼠标滚动 scroll

键盘按下放开 keydown、keyup、keypress event.key、event.keyCode

keydown和keyup中的keyCode返回键盘编码，不区分大小写；捕获控制键和功能键

keypress中的keyCode返回ascii码，区分大小写；不捕获控制键和功能键

dom载入就绪 ready(fn)

调整浏览器窗口 window.resize

获得失去焦点 focus、blur

元素值发生改变 change

输入框选择文本时 select

提交表单 form.submit

阻止表单提交 form.submit(function() { return false; })

事件绑定 on(events, [selector], [data], fn)

事件取消 off(events, [selector], fn)

一次性绑定 one(type, [data], fn)

9 动画

渐变元素css animate(cssObject, time)

结束动画（防止连续播放） stop()

延迟执行 delay(time)

显示/隐藏 show(time)、hide(time)、toggle(time)

淡入淡出 fadeln(time)、fadeout(time)、fadeToggle(time)

垂直滑入滑出 slideDown(time)、slideUp(time)、slideToggle(time)

time可设置为关键字 'fast'、'normal'、'slow'

div.stop()

.hide(2000)

.delay(1000)

.show(2000)

.animate({ opacity: 0.25, width: '256px', height: '256px' }, 3000);

10 js原生计时器

延迟一段时间后执行一次 setTimeout(fn, time)

延迟一段时间后循环执行 setInterval(fn, time)

取消计时器 clearTimeout(object)、clearInterval(object)

11 原生ajax

优点：

通过异步模式，提升用户体验

优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用

ajax引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载

缺点：

不支持浏览器back按钮

安全问题，暴露了与服务器交互的细节

对搜索引擎的支持比较弱

ajax核心：XMLHttpRequest对象

可以向服务器提出请求并处理响应，而不阻塞用户

可以在页面加载以后进行页面的局部更新

使用步骤：

// 1 创建XMLHttpRequest对象

var xhr = new XMLHttpRequest();

// 2 创建一个新的HTTP请

xhr.open(method, url, async);

// 3 设置响应请求状态变化的函数

xhr.onreadystatechange = function() {

if (xhr.readyState === 4 ) {

if ((xhr.status >= 200 && xhr.status < 300 || xhr.status === 304)) {

console.log(xhr.responseText);

}

}

}

// 4 发送请求

xhr.send(null | data);

post场合设置请求头信息

xhr.setRequestHeader(header, value)

服务器返回数据对象

字符串形式 responseText

xml形式 responseXML

请求方法：get与post

必须使用post场合

无法使用缓存文件（更新服务器上的文件或数据库）

向服务器发送大量数据（浏览器对post没有数据量限制）

发送包含未知字符的用户输入时，更稳定可靠

12 jquery ajax

$.ajax({

url: "http://xxx",

type: "post",

async: true,

datatype: "json",

success: function(result) {},

error: function(result) {}

})

$.get()

$.post()

$.getJson()

13 json

可以表示简单值：字符串、数值、布尔和null

必须使用双引号，数值必须是十进制，不能使用NaN和Infinity

不支持undefined

字符串转换为json对象 JSON.parse(responseText)、eval("(" + responseText + ")")

json对象转换为字符串 JSON.stringify("str")

14 跨域

网页请求资源时需要遵循同源策略：域名，协议，端口均相同

解决方案：

跨域资源共享（CORS）、使用JSONP（常用）、修改document.domain、使用window.name

JSONP是一种非正式传输协议，服务端返回数据时会将用户传递的callback 参数作为函数名来包裹住 JSON  数据，这样客户端就可以定制函数来处理返回数据了。

JSONP原理：通过script标签引入js文件，js文件载入成功后，执行在url参数中指定的函数

原生js实现

// 得到查询结果后的回调函数

var getJSONPHandler = function(data){

alert(data.result);

};

// 提供jsonp服务的url地址（不管是什么类型的地址，最终生成的返回值都是一段javascript代码）

var url = "http://localhost/getJSONP?code=CA1998&callback=getJSONPHandler";

// 创建script标签，设置其属性

var script = document.createElement('script');

script.setAttribute('src', url);

// 把script标签加入head，此时调用开始

document.getElementsByTagName('head')[0].appendChild(script);

jquery在处理jsonp类型的ajax时，会自动生成回调函数并把数据取出来供success属性方法来调用

<script type="text/javascript">

jQuery(document).ready(function(){

$.ajax({

type: "get",

async: false,

url: "http://localhost/getJSONP?code=CA1998",

dataType: "jsonp",

jsonp: "callback",

jsonpCallback:" getJSONPHandler",

success: function(json){

…

}

});

});

</script>

补充

ajax和jsonp这两种技术在调用方式上”看起来”很像，目的也一样，都是请求一个url，然后把服务器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装。

但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jsonp的核心则是动态添加

15 自定义封装

var $ = {

ajax: function(options) {

…

}

}

$.ajax({

…

})