### 什么是jQuery?

jQuery(js+Query)是一款优秀的JavaScript库,帮助开发人员用最少的代码做更多的事情,官网网站http://jquery.com/

### 为什么学习jQuery？

\* jQuery让元素选取,事件处理,动画,DOM操作,属性操作,Ajax变得简单,通过链式调用和隐形迭代让代码更加简洁,更重要的是他可以几乎兼容所有的浏览器

\* 几乎所有的巨头都在使用jQuery

\* 丰富的插件支持和强大的生态体系

\* 解读优秀框架,提高编程的视野

### jQuery学什么？

\* 熟悉jQuery的方法

\* 理解jQuery的原理

### 下载jQuery和版本选择

\* 下载地址:https://code.jquery.com/

\* 版本选择

> \* 1.x: 兼容ie678,但相对其它版本文件较大

> \* 2.x: 不兼容ie678,比1.x的文件小

> \* 3.x: 不兼容ie678,比1.x的文件小

> \* 由于兼容性的问题,国内大部分网站依旧选择1.x的版本,所以建议使用1.x

\* 压缩版和源码版

> \* 压缩版,将源代码进行了压缩,文件体积更小,项目上线时为了提高页面的加载速度使用

> \* 源码版,没有压缩,文件体积相对大,开发学习时为了理解jQuery的源码使用

### jQuery的引入

\* 方法一:将jQuery下载到本地,通过script标签引入到要使用的页面

```javascript

<script src="js/jquery.js"></script>

```

\* 方法二:通过script标签引入CDN的资源

```javascript

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

```

\* 由于CDN资源需要从网络上获取,所以在开发阶段使用本地文件,线上产品可以使用CDN资源

\* 在线上的项目为了确保jquery被引入,可以使用判断的方法引入

```javascript

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<script>

window.jQuery || document.write('<script src="js/jquery.js"><\/script>');

</script>

```

### jQuery是使用

\* 方法一

```javascript

//当页面DOM结构加载完毕后执行回调函数

jQuery(document).ready(function(){

//jq代码1...

});

jQuery(document).ready(function(){

//jq代码2...

});

```

> 注意:

> > \* 上述jq代码1和jq代码2都会被执行,不会覆盖

> > \* 不会覆盖的原因是jQuery函数是一个闭包,传递进去的函数都有自己的作用域

\* jQuery(document).ready() 和 window.onload的区别

window.onload必须等到页面全部加载完成然后执行,并且只能执行一次,jQuery(document).ready()只需页面中的DOM结构加载完毕就执行,可以执行多次

\* 方法二

```javascript

//当页面DOM结构加载完毕后执行回调函数

$(document).ready(function(){

//jq代码1...

});

$(document).ready(function(){

//jq代码2...

});

```

> 注意:

>> \* $和jQuery是jQuery框架通过window对象暴露出来的全局变量,并且$=jQuery

>> \* jQuery内部实现:window.jQuery = window.$ = jQuery;

>> \* 用$代表jQuery是为了提高效率

\* 方法三

```javascript

//当页面DOM结构加载完毕后执行回调函数

$(function(){

//jq代码1...

});

$(function(){

//jq代码2...

});

```

> 注意:

>> \* 方法三是方法二的简写形式

>> \* 方法三中的$可以用jQuery代替

### $冲突解决

当其他框架也使用了$时就会和jQuery的$发生冲突,解决方法有

\* 方法一:在使用jQuery前释放jQuery的$使用权,用jQuery来代替$

```javascript

jQuery.noConflict();

jQuery(function(){

});

```

\* 方法二:在使用jQuery前释放jQuery的$使用权的同时自定义自己的符号,用自定义符号来代替$

```javascript

var kz = jQuery.noConflict();

kz(function(){

});

```

### jQuery的核心函数

\* 接收一个函数,返回一个包含了document的jQuery对象

\* 接收一个字符串

> \* 字符串选择器,返回一个包含了DOM元素的jQuery对象

> \* 字符串的HTML代码片段,返回一个包含了DOM元素的jQuery对象

\* 接收一个DOM元素,返回一个包含了DOM元素的jQuery对象

### jQuery对象

jQuery对象是一个伪数组

### jQuery对象和DOM对象转换

\* $(DOM对象)会得到包含了DOM元素的jQuery对象

> \* 原生的DOM元素不能使用jQuery的方法,如果需要使用jQuery对象,需要转换

\* jQuery对象.get(index) 或者 jQuery对象[index] 获取对应下标的DOM元素

### jQuery对象的静态方法

\* $.each(数组,回调) 遍历数组或者伪数组,返回被遍历的数组

```

$.each(arr,function(index,value){

})

```

\* $.map(数组,回调) 映射数组或者伪数组,返回映射后的数组

```

$.map(arr,function(value,index){

})

```

\* $.trim(str) 去除字符串两边的空格并返回

\* $.isWindow(arg) 判断参数对象是不是window对象

\* $.isArray(arg) 判断参数对象是不是真数组

\* $.isFunction(arg) 判断参数对象是不是函数

\* $.holdReady(布尔值) 参考为true时暂停jQuery入口函数的执行,参数为false时恢复jQuery入口函数的执行

### 选择器规则参考CSS中的选择器

\* $('CSS中的选择器');

### 过滤器

\* 基本过滤器

> $(':eq(index)') 选择索引对应的DOM元素

>> 可以用jQuery对象的index()方法获取当前jQuery对象的索引

\* 内容过滤器

> $('选择器:empty') 选择的元素中没有文本内容和子元素的

> $('选择器:parent') 选择的元素中有文本内容或者子元素的

> $('选择器:contains(文本内容)') 选择的元素中包含有指定内容的

> $('选择器:has(选择器)') 选择的元素中有指定的选择器的

\* 过滤方法

> jQuery对象.eq(index) 选择索引对应的DOM元素

> jQuery对象.siblings() jQuery对象中包含元素之外的兄弟元素

\* 注意:以上方法都返回jQuery对象

### 属性和属性节点

\* 属性就是添加在对象(自定义对象或者DOM对象)上的变量

\* 属性节点是HTML标签中指定的属性

> 属性节点可以watch

### 原生JS操作操作属性节点

\* 获取属性节点的值

> DOM对象.getAttribute("属性节点名称")

> DOM对象.属性节点名称,注意class用className

> DOM对象.["属性节点名称"]

> 注意:后面两种方法只适合DOM对象原有的属性

\* 设置属性节点的值

> DOM对象.setAttribute("属性节点名称", "值");

> DOM对象.属性节点名称 = 值

> DOM对象.["属性节点名称"] = 值

> 注意:后面两种方法只适合DOM对象原有的属性

\* 删除属性

> DOM对象.removeAttribute("属性节点名称")

### jQuery对象方法操作属性

\* 获取

> jQuery对象.attr("属性节点名称") 或者 jQuery对象.prop("属性节点名称")

> 注意:只能获取第一个元素的值

\* 设置

> jQuery对象.attr("属性节点名称","值1 值2...") 或者 jQuery对象.prop("属性节点名称","值1 值2")

\* 删除

> jQuery对象.removeAttr("属性节点名称") 或者 jQuery对象.removeProp("属性节点名称")

> 注意:删除所有元素的值

\* attr和prop的区别

> \* 属性节点的值具有true和false时用prop,如checked,selected,disabled,其它的属性节点用attr

> \* 属性节点的值具有true和false时,没有添加该属性时attr会返回undefined,prop返回true

### 操作class

\* 添加

> jQuery对象.addClass("class1 class2...")

\* 删除

> jQuery对象.removeClass("class1 class2...")

\* 切换(存在就删除不存在就添加)

> jQuery对象.toggleClass("class1 class2...")

### 操作代码/文本/值

\* jQuery对象.html([val])

> 设置或者获取元素中的HTML,相当于innerHTML

\* jQuery对象.text([val])

> 设置或者获取元素中的HTML,相当于innerText

\* innerHTML和innerText的区别

> \* 获取HTML内容包括HTML标签和文本，而获取文本内容只有标签里面的文本

> \* 设置带有标签的HTML内容,标签会起作用，而设置带有标签的文本内容，标签会被当做文本来显示

\* jQuery对象.val([val])

> 设置或获取元素value属性的值

### 操作css

\* 获取 jQuery对象.css("样式名称"）

\* 设置

> \* jQuery对象.css("样式名称", "值");

> \* jQuery对象.css("样式名称", "值").css("样式名称", "值");

> \* jQuery对象.css({"样式名称1":"值1","样式名称2":"值2"});

### 操作尺寸

\* jQuery对象.width([val]) jQuery对象.height([val])

> 设置或获取元素的宽高

\* jQuery对象.innerWidth([val]) jQuery对象.innerHeight([val])

> 设置或获取元素的宽高,包含内边距

\* jQuery对象.outerWidth([val]) jQuery对象.outerHeight([val])

> 设置或获取元素的宽高,包含边框和内边距

### 操作位置

\* jQuery对象.offset([对象])

> \* 设置或获取元素相对于body的偏移位

> \* offset返回不传参时返回一个包含偏移值对象

> \* offset传值需要传一个包含偏移值的对象

\* jQuery对象.position()

> \* 获取元素相对于父级定位元素的偏移位,返回一个包含偏移值对象

\* jQuery对象.scrollTop([val])

> 设置或获取元素垂直滚动距离

\* jQuery对象.scrollLeft([val])

> 设置或获取元素水平滚动距离

### 事件的绑定

\* 方法一: jQuery对象.事件名(function(){})

$('.btn').click(function(){})

\* 方法二: jQuery对象.on('事件名',function(){})

$('.btn').on('click',function(){})

\* 注意点:

> \* 方法一部分事件jQuery没有实现,方法二中可以写js中所有的事件

> \* 无论是方法一还是方法二,绑定的事件都会不会覆盖

### 事件的移除

\* jQuery对象.off() 移除所有的事件

\* jQuery对象.off('事件名') 移除所有事件名对应的事件

\* jQuery对象.off('事件名',函数名) 移除事件名对应的函数名的相关事件

### 阻止事件冒泡

\* 方法一 return false

\* 方法二 event.stopPropagation()

### 阻止默认行为

\* 方法一 return false

\* 方法二 event.preventDefault()

### 事件自动触发

\* jQuery对象.trigger('事件名') 不阻止事件冒泡和默认行为的事件触发

> 注意a标签用trigger不阻止事件冒泡,但是会阻止默认行为,如果需要发生默认行为需要触发a标签的子元素

\* jQuery对象.triggerHandler('事件名') 阻止事件冒泡和默认行为的事件触发

### 自定义事件

\* 自定义事件的方法

> \* 用jQuery对象.on('自定义事件名',function(){})绑定一个自定义事件

> \* 用jQuery对象.trigger('自定义事件名')来触发自定义的事件

### 事件命名空间

\* 事件命名空间的使用

> \* 用jQuery对象.on('事件名.名称空间',function(){})绑定一个自定义事件

> \* 用jQuery对象.trigger('事件名.名称空间')来触发自定义的事件

> \* 注意点:

>> \* 用trigger触发子元素有命名空间的事件时,父元素相同相同命名空间的相同事件会触发,没有命名 空间的事件不会触发

>> \* 用trigger触发子元素没有命名空间的事件时,子元素所有相同类型的事件和父元素所有相同类型的事件都会被触发

### 事件的委托

\* 事件委托的方式:$('父元素选择器').delegate('子元素选择器','事件名', 回调函数)

> 注意点：

>> \* 回调函数中的this是触发事件的元素,及具体的某个子元素

>> \* 页面中动态增加的元素需要通过事件委托的方式来添加事件

### 移入移出事件

\* mouseover/mouseout 移入移出子元素会触发

\* mouseenter/mouseleave 移入移出子元素不会触发

\* 用jQuery对象.hover(回调1,回调2)

> \* 移入执行回调1,移出执行回调2

> \* 如果只传一个回调函数,移入移出都执行该回调

> \* 移入移出子元素不会触发

### 显示和隐藏

\* jQuery对象.show(时长,回调)

\* jQuery对象.hide(时长,回调)

\* jQuery对象.toggle(时长,回调)

### 卷入卷出

\* jQuery对象.slideDown(时长,回调)

\* jQuery对象.slideUp(时长,回调)

\* jQuery对象.slideToggle(时长,回调)

### 淡入淡出

\* jQuery对象.fadeIn(时长,回调)

\* jQuery对象.fadeOut(时长,回调)

\* jQuery对象.fadeToggle(时长,回调)

\* jQuery对象.fadeTo(时长,淡入到值,回调)

### 自定义动画

\* jQuery对象.animate(需要动画的属性对象,时长,回调)

> \* 需要动画的属性对象可以使用累加累减动画

> \* 多个属性写在一起可以执行同步动画

### 列队动画

\* 多个动画链式调用,后面的动画会等待前面的动画执行完毕

\* 在动画方法的回调中再执行动画,回调中的动画会等待前面的动画执行完毕

\* 多个动画按调用的顺序执行

### 动画方法

\* 停止动画 jQuery对象.stop(是否清空未执行完的列队动画,是否跳到正在执行动画的最后状态);

> \* 如果不传参数,停止当前正在执行的动画,然后把后续的动画执行完毕

> \* 第一个参数默认是false,表示不清空,如果是true,后续动画不执行

> \* 第二个参数默认是false,表示停止在当前状态,如果是ture,则跳到正在执行动画的最后状态

\* 延迟动画 jQuery对象.delay(时长);

> 延迟动画只对后面的动画起作用

### 动画的速度

以上动画在回调前面可以指定速度,默认是'swing',可用参数"linear"

### 添加节点

\* 内部插入

> \* 将子节点插入到父节点内部后面

>> \* 父节点jQuery对象.append(子节点jQuery对象)

>> \* 子节点jQuery对象.appendTo(父节点jQuery对象)

> \* 将子节点插入到父节点内部前面

>> \* 父节点jQuery对象.prepend(子节点jQuery对象)

>> \* 子节点jQuery对象.prependTo(父节点jQuery对象)

\* 外部插入

> \* jQuery对象1.after(jQuery对象2) jQuery对象1中的节点后面插入jQuery对象2中的节点

> \* jQuery对象1.before(jQuery对象2) jQuery对象1中的节点前面插入jQuery对象2中的节点

### 删除节点

\* jQuery对象.remove() 删除jQuery对象中的节点同时删除事件

\* jQuery对象.empty() 清空jQuery对象总节点的内容

\* jQuery对象.detach() 删除jQuery对象中的节点但保留事件

### 替换节点

\* jQuery对象1.replaceWith(jQuery对象2|HTML代码) 用jQuery对象2中的节点替换jQuery对象1中的节点

\* jQuery对象1.replaceAll('选择器') 用jQuery对象1中节点替换掉所有选中的节点

### 复制节点

jQuery对象.clone(false|true) 复制一个节点,默认是false不会复制事件,true会复制事件

### ajax接口

```javascript

$.ajax({

url:'./data.json', //请求的地址

type:'GET',//指定请求的方法,默认是GET

data:{name:'Tom',age:18},//发送到服务器的数据，字符串键值对或者对象

dataType:'json',//返回数据的类型,可以是xml json script text html,默认是text

success:function(返回数据,状态信息,jqXHR对象){},//请求成功后的回调函数

error:function(jqXHR对象,状态信息){},//请求失败后的回调函数

complete:function(jqXHR对象,状态信息){},//请求完成后的回调函数

timeout:3000, //设置请求超时的时间,单位为毫秒

statusCode:{304:function(){},503:function(){}.....}//设置不同状态码的回调函数

headers:{test:'test'}//设置请求头

})

```

### 接口封装

\* $.get(url[,参数][,回调函数][,返回数据类型])//发送GET请求

\* $.post(url[,参数][,回调函数][,返回数据类型]) //发送POST请求

\* $.getScript(url[,回调函数]) //获取JS代码

\* $.getJSON(url[,回调函数]) //获取JSON数据

\* jQuery对象.load(url[,参数][,回调函数]) //将服务器返回的数据加载到jQuery对象的DOM元素中

> 所有的回调函数接收的参数依次为 返回数据,返回状态信息(成功为success 失败为error),jqXHR对象

### 全局配置

\* $.ajaxSetup(对象参数) //给所有的ajax请求设置默认的初始化参数

### 全局回调

\* $(document).ajaxStart(回调函数),请求开始,未完成其它请求时执行回调函数

\* $(document).ajaxStop(回调函数),请求结束,所有请求结束时执行回调函数

\* $(document).ajaxSuccess(回调函数),当有请求成功时执行回调函数

\* $(document).ajaxComplete(回调函数),当有请求完成时执行回调函数

\* $(document).ajaxError(回调函数),当有请求失败时执行回调函数

\* $(document).ajaxSend(回调函数),当有请求发送请求时执行回调函数

> 全局回调一般写在ajax请求前面,如果某个ajax不想触发全局事件,可以设置 global:false来取消

### jqXHR对象

\* jqXHR对象是浏览器原生XHR对象的一个超集

\* 可以通过调用jqXHR对象上的方法来实现连缀调用,常见的方法有:

> \* .done(回调函数) 请求成功时执行,回调函数的参数依次为 返回数据,状态信息,jqXHR对象

> \* .fail(回调函数) 请求失败时执行,回调函数的参数依次为 jqXHR对象,状态信息

> \* .always(回调函数) 无论成功还是失败都执行,如果成功,回调函数的参数和done的一样,如果失败，回调函数的参数和fail的一样

###扩展自己的插件

\* 类级别的

```javascript

$.myFunc = function(){

}

```

```javascript

$.extend({

myFunc:function(){

}

})

```

```javascript

var obj = {

myFunc:function(){

}

};

$.tom = obj;

\* 对象级别的,本质就是在jQuery对象的原型对象上增加方法

```javascript

$.fn.extend({

colorToRed:function(){

this.css({

color:'red'

});

return this;

}

})

````

```javascript

;(function($){

$.fn.tab = function(options){

//默认的配置属性

var defaults = {

}

//将默认属性和传入的属性合并

var options = $.extend(defaults,options);

this.each(function(){

//具体的逻辑实现

})

}

})(jQuery)

````

###使用已有的插件

\* 电子书插件 turnjs(http://www.turnjs.com/)

\* 表单验证插件 jQuery validation(https://jqueryvalidation.org/)

\* 全屏加载插件 FullPage (https://alvarotrigo.com/fullPage/)