1. 什么是JQuery
2. JQuery是第三方开发的执行DOM操作的极简化的函数库

(1).第三方：必须下载，才能使用

(2).指定DOM操作：JQuery还是在执行DOM操作（增删该查+事件绑定）

(3).极简化：JQuery将DOM的每一步操作，都进行了简化

比如DOM中，按id查找： var 元素=document.getElementById(“id名”);

Jquery: var 元素=$(“#id名”);

获得下一个兄弟元素：

DOM： 元素。nextElementSibling

JQuery: 元素.next();

1. .函数库：JQuery中，一切都是函数，没有属性
2. 为什么使用：2个优点：

(1).简单

(2).已经解决了大部分浏览器兼容性的问题。凡是JQuery让用的都没有兼容性的问题

3.何时：用的越来越少！

4.缺点：2个致命

(1).Jquery只支持PC端网页，不支持移动端和小程序

(2).只是对DOM的每一步进行了简化，没有彻底减少开发的步骤

二．如何使用JQuery:

1.下载：

(1).官网：jquery.com

(2).版本：

a. 1.x唯一兼容旧浏览器（IE8）的版本

1).压缩版：体积小，适合于生产环境快速下载使用

2).未压缩版：可读性好，适合于开发和学习之用

b. 2.x不再支持旧浏览器

c. 3.x也不再支持旧浏览器，才开始支持ES6新特性，比如：for of和promise等

2.

1. jQuery原理
2. 只要使用jQuery,必须先引入jQuery.js,在编写自定义的js脚本
3. 引入jQuery.js,其实是给js内存中添加了一种全新的类型（家庭）:2部分

(1).构造函数:专门用于创建该类型的子对象

(2).原型对象:专门保存所有子对象公用的方法

3.今后只要想使用jQuery中简化版函数代替DOM旧的函数操作DOM函数，都要：

(1).先创建jQuery家子对象，同时查找并保存要操作的DOM元素，到新创建的jQuery子对象

var jq对象=$(“选择器”);

强调：为了区分jQuery记得孩子，和DOM家的孩子，约定俗成：所有jQuery家孩子的变量名，都必须以"$"开头

1. .就可以用jq对象调用jQuery家提供的各种函数：
2. 所有简化版函数都会被自动翻译为DOM原生的对等的函数
3. 然后，对整个jq对象执行DOM操作，都等效于直接对jq对象内保存的DOM元素执行相同的DOM操作。执行jq对象，等同于执行DOM元素

4.btn.click(function(){...}) 中的this:

1. .btn.click(function(){...})会被翻译为btn.onclick=function(){...}

(2)this依然指.前的当前按钮btn对象---DOM元素对象，无法使用jQuery家函数

(3). 所以，必须: var $this=$(this)

等效于new jQuery(this) -> new jQuery(btn)

(4). 结果: $this才是jquery家函数，才能继续调用jQuery家函数

1. jQuery对象的本质
2. .jQuery是一个类数组对象
3. 因为交给$()的选择器，很可能在页面中找到的是多个符合要求的元素。只有类数组对象，才适合保存多个DOM对象
4. jQuery简化版函数的通用3大特点：
5. .几乎所有的简化版函数都内部自带for循环

所以：如果$()找到并保存了多个元素，则值对$()这一个jQuery对象整体调用了一次简化版函数，就等于自动遍历jq对象中保存的每个DOM元素，并对每个DOM元素执行等效的原生操作。

(2).所有和修改元素相关的函数（修改函数、属性、样式）都一个函数两用

1. 如果调用修改函数时，没有提供新值，则修改为执行获取现有值的操作  
    b.只有当调用函数时，如果没有给新值，才执行操作

（3）.

二．查找元素

1.jQuery支持所有css3选择器

2.但是jQuery还新增了个别独有的新选择器，(仅jq内可用,css不可用)

3.基本过滤选择器：

(1).回顾css中的子元素过滤选择器

a.什么是：根据子元素在其父元素内的相对位置匹配子元素

b.包括css中包含4个：

1). :first-child:选择作为其所在父元素内的第一个直接子元素的元素

2). ：last-child:选择作为其所在父元素内的最后一个直接子元素的元素

3). :nth-child(n)：选择作为其所在父元素内的第n个直接子元素的元素 css中从1开始

4). ：only-child: 选择作为其所在父元素内唯一一个直接子元素的元素

(2).什么是基本过滤选择器：先将所有符合条件的元素取出来，放在一个集合中，，从头到尾，统一编号，且编号从0开始

(3).结果：基本过滤选择器与元素在其父元素内部的相对位置无关

(4).包含：

a. :first 或 :last 选择所有符合条件的元素中的第一个或者最后一个元素

b. :gt(i) 或 :lt(i) 或 :eq(i) 选择所有符合条件的元素中 >i 位置或<i位置或者=位置的元素

c. :even 或 :odd 选择所有符合条件的元素中，偶数或者奇数下标位置的元素

强调：程序员眼中，下标是从0位置开始的，所以正常人眼中和程序员眼中的偶数和奇数位置是相反的，下标是0，则是偶数位置，是1，则是奇数位置

1. . 总结: 今后，如果一个效果，既可以用css做，又可以用js做，首选用css！因为css比js效率高！
2. 内容过滤选择器:
3. .根据元素的内容不同来选择元素
4. .包括：
5. :contains(文本) :选择元素内容包含”文本”的元素
6. :has(选择器) 选择包含符合选择器要求的子元素的元素
7. :parent 选择内容为非空的元素
8. 可见性过滤选择器:
9. .根据元素是显示还是隐藏来选择元素
10. 包括： ：visible 选择可见的元素 ：hidden 选择隐藏的元素

但是 :hidden只能匹配display:none 或者type=”hidden”的元素，无法匹配visibility:hidden和opacity:0隐藏的元素.

1. 表单元素过滤选择器:

(1). :input jQuery中独有的能够选择表单中所有的表单元素（input、textarea、button、select）

(2). jq针对每种type属性值分别提供了一个专门的选择器

a. :text 专门选择input type="text"的文本框

b. :password 专门选择input type="password"的密码框

c. :radio 专门选择 input type="radio"的单选按钮

d. :checkbox 专门选择input type="checkbox"的复选框

三．修改元素

1.内容3种：

(1).获取或修改元素开始标签和结束标签之间的原始的HTML内容：

.innerHTML

$元素.html（“新的HTML内容”） -----一个函数两用

(2).获取或修改元素开始标签和结束标签之间的纯文本内容：

.textContent

$元素.text（“新的文本内容”） -----一个函数两用

(1).获取或修改表单元素value内容：

.value

$表单元素.val（“新值”） -----一个函数两用

2. 属性: 3种:

(1). 字符串类型的HTML标准属性: 2种:

a. $元素.attr("属性名", "新属性值") —— 一个函数两用！

代替.getAttribute("属性名")和.setAttribute("属性名","新属性值")

b. $元素.prop("属性名", "新属性值") —— 一个函数两用

因为jQuery是一个函数库，不能用".属性名"，所以.prop()函数代替的是DOM中“.属性名”

(2). Bool类型的HTML标准属性: 1种:

a. DOM中: 不能用getAttribute()和setAttribute()获取或修改，只能用".属性名"获取或修改，且值必须是bool类型

b. jq中: 也不能用.attr()获取或修改，只能用.prop()获取或修改——jQuery即使DOM

(3). 自定义扩展属性: 1种:

a. DOM中(不考虑HTML5): 只能用getAttribute()和setAttribute()获取或修改。不能用".属性名"方式获取或修改

b. jq中: 也只能用.attr()来获取或修改，不能用.prop()获取或修改

总结: .attr() vs .prop()差别

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字符串类型  标准属性 | Bool类型  标准属性 | 自定义扩展属性  （值为字符串） |
| .attr() //get/setAttribute() | √ |  | √ |
| .prop() //.属性名 | √ | √ |  |

3. 样式:

(1). 获取或修改一个元素的一个css属性值:

a. $元素.css("css属性名", "新属性值") —— 一个函数两用

b. css()整合了DOM中的.style和getComputedStyle()两个操作

1). DOM中:

i. .style.css属性名 只能获取或修改内联样式

ii. 如果要完整样式，被迫改为用getComputedStyle()函数

2). Jq中:

i. 如果.css()时没有给新属性值，则自动在低层调用getComputedStyle()，执行获取css属性值的操作

ii. 如果.css()时，给了新属性值，则自动切换为在低层调用.style.css属性，执行修改css属性值的操作

c. 所以: jq中，无论获取还是修改css属性值，都用.css()一个函数就够了！

|  |
| --- |
| 总结: jQuery常用的三个修改相关的函数: $元素.attr()，$元素.prop()，$元素.css()  都可以一个函数同时修改多个属性，但是要用对象语法！  $元素.attr/prop/css({  属性名1: "值1",  属性名2:"值2",  ... : ...  }) |

1. 修改样式：通过修改class批量修改一个元素的多个css属性
2. 问题：使用.css()修改元素的属性时，一句话只能修改一个元素的css属性，即使使用{}简写，也必须把每个css属性都写出来。当批量修改一个元素的多个css属性时，代码就会很繁琐
3. 问题：DOM中虽然可以用className属性来修改元素的class，批量修改元素的多个css属性。但是一个元素被多个class修饰，那么，如果只修改一个class就会很麻烦

比如：<button id=”btn” class=”btn active btn-info”></button> 要移除active 要用 btn.replace(“active”,””);

1. jq中，为了方便对class做各种灵活的操作，提供了4中函数
2. . 添加一个class: $元素.addClass(“class名”);
3. .移除一个calss: $元素.removeClass(“class名”);
4. .判断一个元素是否包含class，$元素.hasClass(“class名”);
5. 因为经常需要在一个元素的某个class之间切换（有何没有某个calss）：
6. 只用一个函数： $元素.toggleClass(“class名”); 切换
7. 原理：toggleClass()内部自带了if else判断，所以不需要自己写if else了

相当于： if($元素.hasClass("元素名")){

      $元素.removeClass("元素名");

    }else{

      $元素.addClass("元素名");

    }

1. 按节点间关系查找：3大类关系
2. 父子关系：
3. .获得当前元素的父元素：$元素.parent() DOM: .parentNode()
4. 获得当前元素下符合要求的直接子元素：元素.children(“选择器”) 不写选择器意为所有的子元素 DOM: .children();

问题：只能在直接子元素中查找，无法在所有后代中查找

解决：jQuery新增了一个函数：$元素.find(“选择器”); 在所有后代子元素中查找

1. .获得当前元素的第一个直接子元素：没有专门的函数，可用children(“选择器”)来实现:$元素.children(“:first-child”);
2. .获得当前元素下的最后一个直接子元素：没有专门的函数，可用children(“选择器”)来实现:$元素.children(“:last-child”);
3. 兄弟关系：
4. .当前元素的前一个兄弟元素：$元素.prev(); DOM .previousElementSibling()

问题：prev()只能找前一个元素，有时候需要获得前所有的元素，

解决：jQuery新增了一个函数 $元素.prevAll(“选择器”);选择当前元素之前的符合要求的元素

1. .当前元素的后一个兄弟元素：$元素.next(); DOM .nextElementSibling()

问题：next()只能找后一个元素，有时候需要获得后面的所有元素，

解决：jQuery新增了一个函数 $元素nextAll(“选择器”);选择当前元素之后的符合要求的元素

1. 问题：有时需要选择除当前元素之外的所有前后的兄弟元素： $元素.siblings(“选择器”); 可以选择符合要求的兄弟元素
2. 添加/删除/替换/克隆
3. 添加：2步
4. . var $新元素=$(`html代码片段`);
5. 强调：$() 不是查找的意思，而是将html代码片段创建为DOM元素对象
6. 问题：仅创建出新的DOM元素，但是还不在DOM树上，用户看不到
7. .将$新元素添加到DOM树上指定位置---用户才可以看到
8. 在父元素下，所有子元素末尾追加一个新元素：$父元素.append($新元素); 返回父元素，后续可调用该父元素 DOM: .appendChild(新元素)

或 $新元素.appendTo(父元素);

1. 在父元素下，所有子元素开头添加一个新元素：$父元素.prepend($新元素);

或 $新元素.prependTo(父元素);

1. 在父元素下，指定子元素之前插入新元素：$现有子元素.before($新元素); DOM中: .insertbefore(新元素,子元素)

或 $新元素.insertBefore(现有子元素);

1. 在父元素下，指定子元素之后追加一个新元素：$现有子元素.after($新元素);

或 $新元素.insertAfter(现有子元素);

e.在父元素下，替换某个子元素为一个新元素：$父元素.replaceWith($新元素); DOM中: .replaceChild(新元素,自元素)

或 $(新元素).replaceAll(父元素);

1. 删除：$元素.remove();
2. 克隆: var $一摸一样的副本元素=$元素.clone();

四.事件绑定:

1.标准的事件绑定方式:

(1).$元素.on(“事件名”，事件处理函数) ---仅仅是DOM中addEventListener的简写

(2).$元素.off(“事件名”，事件处理函数) ---仅仅是DOM中removeEventListener的简写

(3). 对比:

相同点: 如果一个事件监听有可能需要被移除，则绑定时必须用有名称的函数来绑定。移除时，才能用函数名获得原函数对象

不同点: jq中同名函数，照样可以绑定多个事件监听，移除时，还可用一个函数名移除所有同名的事件监听。

(4). 示例: 使用on和off绑定事件和移除事件

3.事件委托：

(1).回顾：如果多个子元素都要绑定相同的事件处理函数时，就可用事件委托

(2).如何：3步：

1. 事件只绑定在父元素上一份即可

2. e.target代替this

3. 判断e.target是否是想要的！

(3).简写--了解，仅JQuery可用，今后框架不通用

a.相同点：事件必须绑定在父元素上一份即可

b.不同点：

(1).必须用on(),不能简写click()...

(2).给on()添加第二个参数：选择器，用来自动判断当前触发事件的元素是不是符合要求的---不用写if判断了

(3).this重新指向了最初触发的那个元素了，

4.页面加载后自动执行:

(1). 问题1: 多数页面在页面加载完成后，就要自动执行一些操作:

比如: ajax请求首屏数据，比如为元素绑定事件处理函数

这些代码写在哪儿？

(2). 错误的做法: 放在<body>之前的<script>中。

因为网页顺序执行！没有body中的HTML元素，ajax请求回的数据就不知道放哪儿，事件也不知道绑定给谁！

(3). 不好的解决方法: 放在body结尾的<script>中

因为不符合内容与行为分离的原则，不便于维护！

(4). 正确的好的解决方法: 将所有js代码都放在HTML文件之外的独立的js文件中单独保存，运行时，通过<script src="js/xxx.js">引入HTML中自动执行——符合内容与行为分离的原则，便于维护。

(5). 问题2: 别人引入你的js时，可不一定都放在结尾！我们希望无论js文件在HTML开头引入，还是在body结尾引入，都能正常执行

(6). 解决: 将所有要在页面加载后自动执行的代码，都放在window.onload事件的处理函数中。（window.onload会等待所有HTML+JS+CSS+图片都下载完，才自定执行）

(7). 结果: 将来无论被人将你的js代码放在HTML开头还是放在body结尾都只是一个事件绑定代码而已，不会立刻执行。只有等到所有页面内容加载完成后，才执行！

(8). 问题3: 因为图片下载非常慢，有些操作，用户等不及所有图片都下载完，才用。用户希望，只要看到按钮就可以点击！不关心CSS和图片！

因为window.onload必须等待css和图片都下载完才能执行，所以，有点晚！

(9). 解决: 其实在window.onload之前还有一次加载完成事件:

a. DOMContentLoaded : 仅DOM内容加载完成就可以提前触发事件，执行一些操作

b. DOMContentLoaded只等HTML+JS，无需等待CSS和图片，所以DOMContentLoaded事件会比window.onload事件提前很多就可开始执行！

c. 今后几乎所有和css和图片无关的页面初始化操作：比如ajax和事件绑定，都应该放在DOMContentLoaded事件中提前触发，让用户提前用上功能！

(10). 问题4: 因为DOMContentLoaded有兼容性问题，所以通常都用jquery提供的方法来绑定DOMContentLoaded事件

(11). 解决:

a. 完整写法: $(document).ready(function(){ ... 页面初始化相关操作 ... })

b. 简化:

1). $().ready(function(){ ... 页面初始化相关操作 ... })

2). $(function(){... 页面初始化相关操作 ... })

(12). 总结: 今后为了保证绝大多数页面初始化操作都能让用户提前用上，所有页面初始化代码都应该放在一个$(function(){ ... })中

(13). 其实: 今后几乎所有jquery的代码，都是放在$(function(){ ... })中的

总结：jQuery3大特点

1. 自带for循环
2. 一个函数两用
3. 多数没有返回值的函数，默认继续返回当前正在操作的.前的jQuery对象主语：

链式操作：如果前一个函数返回的结果，刚好是下一个函数需要的主语，则可以连续用.调用多个函数，像排火车一样

总结：$()有4中重载

1. $(“选择器”) 创建jQuery对象，同时查找符合选择器要求的DOM元素，并找到DOM元素对象保存进新创建的jQuery对象中
2. $(this)或$(“e.target”) 创建jQuery对象，无需查找，就直接将已经给的DOM元素保存进jQuery对象中---其实就是让一个普通的DOM元素变为jQuery对象
3. $(`html代码片段`) 创建jQuery对象，同时将html代码片段创建成真正的DOM元素对象，保存进jQuery对象中

|  |
| --- |
| 总结: 常用可简写事件名列表  blur 失去焦点  change 下拉列表选中项改变  click 单击  dblclick 双击  focus 获得焦点  keydown 键盘按键按下  keyup 键盘按键抬起  mousedown 鼠标按键按下  mouseenter 鼠标进入(jq)  mouseleave 鼠标移出(jq)  mousemove 鼠标移动  mouseout 鼠标移出(dom)  mouseover 鼠标进入(dom)  mouseup 鼠标按键抬起  resize 窗口大小改变  scroll 网页滚动 |

一. 事件:

1. 鼠标事件:

(1). DOM中:

a. mouseover 当鼠标进入元素范围内时自动触发一次

b. mouseout 当鼠标离开元素范围时自动触发一次

(2). 问题: 即使反复进出子元素，也会反复冒泡触发父元素上的鼠标进入和移出事件——容易造成误解。

(3). Jq中: 用mouseenter和mouseleave代替了mouseover和mouseout

(4). 优点: 即使反复进出子元素，也不会反复冒泡触发父元素上的鼠标进入和移出事件——和现实相符，不会造成误解。

(6). 问题：因为多数情况下鼠标进入和鼠标移出事件都是成对儿绑定的(鼠标进入变样，鼠标移出后恢复)。但是mouseenter和mouseleave太长！不愿意写！

(7). 简写: 如果同时绑定mouseenter和mouseleave，只需要绑定一个jq专属的事件函数hover()即可

a. $元素.hover( //=mouseenter+mouseleave

function(){ ... }, //给mouseenter

function(){ ... } //给mouseleave

)

b. 强调: 因为虽然只绑定一个函数hover，但是hover等效于同时绑定两个事件(mouseenter+mouseleave)，所以hover中通常要写两个事件处理函数。第一个给mouseenter，第二个给mouseleave

(8). 更简写：如果你有本事把hover中的两个函数，改造成两个相同的函数

a. $元素.hover(//=mouseenter+mouseleave

function(){ ... } //既给mouseenter，又给mouseleave

)

b. 强调: hover中如果只给一个函数，不是只绑定mouseenter的意思。依然是既绑定mouseenter又绑定mouseleave。且唯一的一个函数既给mouseenter，又给mouseleave

(6).模拟触发

(1). 什么是: 即使没有点击按钮，也能触发按钮上的单击事件

比如: 百度上搜索时，当在文本框中输入完关键词，不需要点查找按钮，只需要按回车，就可执行和点击查找按钮完全相同的操作！

(2). 如何:

a. 标准: $事件所在元素.trigger("事件名")

b. 意为: 触发事件所在的那个元素上的指定名称的事件的处理函数

c. 简写: 如果要触发的事件刚好是21种常用事件之一，则无需trigger，可直接使用事件名触发:

$事件所在元素.事件名() //()里不要写function！

1.简单动画：固定写死的三种动画效果

(1).显示隐藏 $元素.show() 和 $元素.hide()a. 在不加任何参数的情况下: 仅仅是display:block和display:none的简写。等效于.css("display","block")和.css("display":"none");——所有在jquery如果希望无过渡效果的显示隐藏一个元素，都可用show()/hide()/toggle()不带任何参数来实现。

b. 其实: 想让以上三个函数带过渡动画效果，只需要在()中添加一个动画持续时间毫秒数。

(2). 上滑下滑: $元素.slideUp() $元素.slideDown() $元素.slideToggle()

说明: 不需要加参数，也默认自带过渡效果

(3). 淡入淡出: $元素.fadeIn() $元素.fadeOut() $元素.fadeToggle();

说明: 不需要加参数，也默认自带过渡效果

|  |
| --- |
| 简单动画函数的致命缺陷: 2个  1. 丑，不便于维护！——所有变化过程，都是用js在jquery函数库源代码中的写死的！  2. 效率低——底层都是用js程序+js定时器实现的，效率远不如css动画高！  所以，今后做这类简单动画，首选css transition做！——自己写的css便于维护，且css比js程序效率高的多！ |

2.万能动画

(1). 什么是: 可以对任意css属性应用过渡效果——作用类似于css的transition，但是无论是效率，还是功能，都远不如css的transition

(2). 如何:

$元素.animate({

Css属性:目标值,

... : ...

}, 动画持续时间ms)

强调: {}里只支持单个数值的css属性，不支持非数值的属性（比如颜色，CSS3 transform...），远不如css transition。

(3). 原理:

a. animate会自动获得元素的当前状态

b. animate根据{}中给的目标状态和当前状态之间的差距，自动计算过渡的变化过程

c. 然后底层启动js定时器模拟过渡变化的效果，执行变化的过程。

(4). 优点: 比简单动画9个函数更灵活

(5). 缺点:

a. 效率低——底层是用js定时器模拟的

b. 支持的css属性，还不如css transition多！

(6). 结论:

a. 如果动画变化的过程，是固定的，不会有人工干预，首选css transition

b. 如果动画变化的过程由人为操作，只能用animate()!

3. 排队和并发:

(1). 并发:

a. 什么是: 多个css属性，同时变化

b. 如何: 放在一个animate({})内的多个css属性，默认并发变化

(2). 排队:

a. 什么是: 多个css属性，按顺序，先后依次变化

b. 如何: 对同一个元素，先后连续调用多个animate()函数，则先后调用的多个animate()函数中的css属性是排队变化的

4. 停止动画:

(1). $元素.stop()

(2). 坑: 当多个animate()排队执行时，.stop()只能停止当前正在执行的一个动画animate()，排队的后续animate()依然继续执行！

(3). 解决:$元素.stop(true) 停止当前animate()动画，同时清除之后所有animate()动画

5. jquery独有选择器: :animated 专门匹配正在执行animate()动画的元素

6. 动画结束后自动执行:

(1). 问题: animate()低层是js定时器，所以animate()是异步的函数。如果想在动画播放结束后，才执行一项任务，单纯将代码放在animate()之后，是无法实现的！

(2). 解决: 其实所有动画函数都有最后一个实参——一个回调函数。放在动画函数最后一个回调函数实参中的代码，注定只能在动画播放完成后，才自动调用执行.

三．类数组对象操作：jQuery仿照着数组家两个函数，为jQuery对象也定义了两个功能几乎完全相同的函数

1.仿照着数组家的indexOf()函数，为jQuery对象定义了一个index()函数

(1).回顾：indexOf()查找一个元素在数组中的下标位置 var i=arr.indexOf(元素)

(2).jq中：index()查找当前元素在所有jQuery查询结果集合中的位置的下标 var i=$结果集合.index(要找的元素);

2.仿照着数组家的forEach()函数，为jQuery对象定义了一个each()函数

(1).回顾：forEach(function(elem,i){

//elem:会自动获得当前正在遍历的数组元素值

//i会自动获得当前正在遍历的下标位置

})

(2).jq中： $查询结果.each(function(i, elem){

//i会自动获得当前正在遍历的下标位置

//elem 会自动获得当前正在遍历的查询结果中的一个DOM元素对象

})

1. 封装自定义插件
2. 为jQuery家添加自定义函数：
3. .就是自定义一个函数，添加到原型对象中

.jQuery,prototype.自定义函数名=function(){ ... }

1. .结果：将来只要是jQuery的查询结果导向，都可以调用元素原型对象中自定义的函数方法

$(“选择器”).自定义函数名()

1. .强调：jQuery原型对象方法中，也只能通过this来操作将来调用的这个自定义方法的.前的jQuery对象。又因为将来到用时，.前已经是$()jQuery对象了，所以不用再将this用$()包起来了
2. 封装自定义函数：
3. .什么是插件/组件：值拥有专属的HTML+CSS+JS的可重用的独立的页面功能区域
4. .为什么：可重用代码，提高开发效率
5. 何时：如果我们的页面中有一块功能区域在为项目中反复使用，则都应该先封装插件,再反复使用插件。
6. 参考：jQuery官方插件库：jQuery UI 如何使用：

a. 如何使用:

1). 下载jQuery的js和css以及图片: jqueryui.com

2). css和图片文件夹，放在项目的css文件夹中，jquery-ui.js放在项目的js文件夹里

3). 在HTML页面中: 依次引入jqueryui.css， jquery.js 和jqueryui.js

<link rel="stylesheet" href="css/jquery-ui.css">

<script src="js/jquery-1.11.3.js"> 先

<script src="js/jquery-ui.js"> 后

4). 按插件要求编写插件的HTML结构和内容 —— 暂时没有任何插件效果

说明: 使用jqueryui，HTML中不需要加任何class！

5). 在自定义的script中:

$("插件父元素").插件函数()

b. 优点: 简单！

原理: jQueryUI的插件函数，会根据自身的需要，自动为元素侵入class和自定义属性

c. 致命缺点: 2个:

1). 仅有PC端，没有移动端！

2). 可维护性差！所有class和事件绑定都是背着程序员自动侵入的。程序员想改时，不知道改哪里！

(5). 仿着jQueryUI做法，封装一个自己的jquery插件: 3步:

a. 前提: 已经在页面中用传统的HTML+CSS+jQuery方式实现了插件的效果。封装插件其实只是一个提取代码的过程。而不是从0开始开发！

b. 第一步: 先提取css代码保存到独立的css文件中

c. 第二步: 在独立的js文件中，向jquery的原型对象中添加一个自定义的插件函数

jQuery.fn.插件函数=function(){

//2件事：

//1). 给将来的插件元素们自动悄悄的添加所需的class——侵入

2). 将页面中已经实现的事件绑定代码，剪切到原型对象中插件方法结尾。

}

d. 第三步: 在页面中引用插件的css和js，并编写自定义脚本调用插件函数——过程和使用jqueryui的步骤完全一样！

(6). 将来如果希望重复使用这个插件，只要拷贝独立的.css和独立的.js文件到下一个项目即可！只需要一句话$(父元素).自定义插件函数()，就可立即应用插件！

一. ajax: jQuery将最常用的发送ajax请求的4步或5步，都封装为一句话！

$.ajax({

url:" 服务器端接口的地址 ",

type:"get"或"post", //请求类型

data:{ //要发送到服务器的参数

变量名: 变量值

},

dataType:"json", js可用的对象或数组<-JSON.parse(xhr.responseText)<-JSON字符串<-服务器端,

success:function(result){ //回调函数: 成功收到响应结果后自动执行

result 得到的是: 服务器端返回的结果被转为js中的对象或数组

//强调: 凡是想在ajax请求完成后，才执行的代码，必须放在success回调函数内!

}

})

调试: F12->network中->点击类型为xhr的请求，可看到请求发送的详情和结果

二．跨域

1. 什么是跨域: 一个网站中的网页，需要用到另一个网站的资源(文件,接口等)

2. 比如:

<link rel="stylesheet" href="其它网站的.css">

<script src="其它网站的.js">

<img src="其他网站的图片">

$.ajax({

url:"其他网站的接口",

... ...

})

3. 包括:

(1). 域名不同: http://www.a.com下的网页想用http://www.b.com下的资源

(2). 子域名不同: http://oa.tedu.cn下的网页想用http://hr.tedu.cn下的资源

(3). 域名相同端口号不同，也算跨域：

http://localhost:5500下的网页想用http://localhost:3000下的资源

(4). 协议不同，也算跨域:

http://12306.cn下的网页想用https://12306.cn下的资源

(5). 即使同一台及其同一个网站内，用域名与IP地址互访，也算跨域！

http://localhost:3000下的网页想用http://127.0.0.1:3000下的资源

4. 问题: 浏览器有一个"同源策略"——只对ajax起作用:

(1). 什么是同源策略: 要求一个网站(localhost:3000)下的网页中编写的ajax代码只能请求自己网站提供的接口(localhost:3000/index)。不能请求其他网站提供的接口。

(2). 示例: localhost:3000自己的index.html页面请求localhost:3000自己内部的接口——可以正常访问

(3). 但是一个网站（127.0.0.1:5500/index.html）下的网页，想请求两一个网站下的接口(localhost:3000/index)——结果: 报错！

Access to XMLHttpRequest at 'http://localhost:3000/index' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

用XMLHttpRequest(xhr)，从源头http://127.0.0.1:5500，向http://localhost:3000/index发送的请求，已经被CORS策略阻止了！因为请求来的资源上没有提供"Access-Control-Allow-Origin"头说明.

|  |
| --- |
| 补: 快速创建服务器端nodejs express项目脚手架代码的方法: 2步  1. 安装会下蛋的老母鸡：其实是一个工具，可用于反复自动创建脚手架代码——会下蛋  npm i express-generator -g  生成器  如果你本地node有问题，或网路有问题，暂时不要尝试安装  2. 让老母鸡反复下蛋：蛋->可直接使用的已经包含核心功能的服务器端nodejs express项目脚手架代码  选择你想创建项目的文件夹  express 项目名  //暂时没有node\_modules  cd 项目名 //进入项目目录  运行 npm i 安装node\_moduels  如果上一步npm i xxx 安装不了，第二步也不要试！  3. 运行脚手架示例代码: （可以用我发的压缩包！直接执行第三步！）  (1). Vscode打开刚创建的项目文件夹:  (2). 右键选择package.json，选在终端中打开  (3). 等弹出窗口中出现xxxx/xzserver> | 输入npm start按回车  看到: node ./bin/www，就算成功  (4). 打开浏览器，手动输入http://localhost:3000 回车  看到：  Welcome  xxx  说明服务器中的网页也运行成功! |

说明:用两个vscode分别打开public和xzserver文件夹

纯后端项目xzserver中，右键单击package.json文件，选择在终端中打开，等终端中出现文字后，输入npm start，看到./bin/www说明服务器端启动成功

可测试服务器端接口: 手动打开浏览器，地址栏输入: http://localhost:3000/index，就可看到服务器端返回的商品信息.

纯前端项目public，右键单击index.html网页，选择open with live server，自动打开浏览器，能看到页面内容，但是F12打开控制台，发现报跨域错误！说明127.0.0.1:5500/index.html不能请求localhost:3000的接口。

5.解决: 服务器端CORS方式解决跨域问题：

(1).原理：其实服务器端在返回数据之前是有机会篡改寄件人地址的。如果服务器端在返回结果之前，将返回数据的寄件人地址篡改为和浏览器网页地址一样，则浏览器就允许网页使用这个返回数据

(2).如何：

a. 笨办法：在服务器端每个几口==接口函数用以下代码代替res.send(result);

res.writeHead(200,{

    "Access-Control-Allow-Origin":"http://127.0.0.1:5500"

  });

  res.write(JSON.stringify(result));

  res.end();

b.问题：每个接口都需要跨域，但是如果每个接口都反复写这个代码，就会很繁琐

c. 好的解决: 为express服务器端提前安装并配置允许跨域的中间件cors

1). 安装: npm i -save cors

2). 在服务端程序的app.js中:

var cors=require("cors");

var app=express()

app.use(cors({

origin:["http://127.0.0.1:5500"]

}))

3). 所有接口函数内，可以继续使用res.send(result)，不用再考虑跨域问题

4). 结果: 凡是从这个服务器出去的接口数据，都被统一自动添加上了符合要求的寄件人地址。都可以通过客户端浏览器的检查。

d. 问题: 如果多个客户端的网站，都像请求这一个服务器中的资源:

1). 错误做法: app.use(cors({ origin:[不支持\*匹配所有地址]}))

2). 绝大多数公司的服务器端都是极其保密的！不允许外部的其他网站访问服务器接口，只允许自己公司的少量客户端网站访问自己的服务器:

app.use(cors({ origin:[ "客户端地址", "客户端地址",...]}))

只要写在origin数组中的客户端地址，都可以跨域访问

3). 少数大公司的服务器端可能同时支持大量客户端网站访问。比如: 中央气象台！

不可能把所有客户端地址都列在origin数组中。

被迫使用jsonp方式跨域！

6. 前后端分离：企业中普遍采用的项目开发方式

(1). 前端和后端是两个完全独立的团队

(2). 两个团队各自开发自己的一个项目，也就是公司中的一个项目，普遍由两个项目组成(前端项目和后端项目)

(3). 前端项目和后端项目唯一的纽带就是用ajax请求服务器端的接口，获得数据。

(4). 优点: 松耦合

a. 互不影响:

b. 两个团队可以同时并行开发！开发效率极高！