小程序: WEB问题速查

1. 问题清单: 小程序->首页->JQUERY->day01

2. 高清无码小视频: 小程序->在线->JQUERY->day01

今日扩展视频:

无

不用今天都看完！！！！！！\*\*\*两个月后\*\*\*鄙视面试前\*\*\*看完就行\*\*\*！

3. 单词列表: 小程序->单词->JQUERY->day01

第三阶段上课约定:

1. 上课千万不要跟着记笔记和写代码！

2. 拼写错误！自己找！让我帮你找，你需要手抄20遍，给我拍照！

3. 现在正处在照猫画虎的阶段！每个DOM效果至少写3遍:

3.1 注释+代码抄一遍！

3.2 保留注释，删除代码，把代码再填回来！

3.3 删除注释和代码，试着用自己的话把注释写回来，根据自己的注释，把代码填回来

一个DOM效果，只有自己能写出人话注释，才算自己会！——标准！

如何提问:

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻微信私信问我！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻微信私信问我！

3. 个人项目中哪个功能不会做！立刻微信私信问我！

4. 刷笔试题时，遇到不会的笔试题！立刻微信私信问我！

下载源代码: 右键->链接另存为！

正课:

1. 什么是jQuery

2. 如何使用jQuery

3. jQuery原理

4. 查找元素

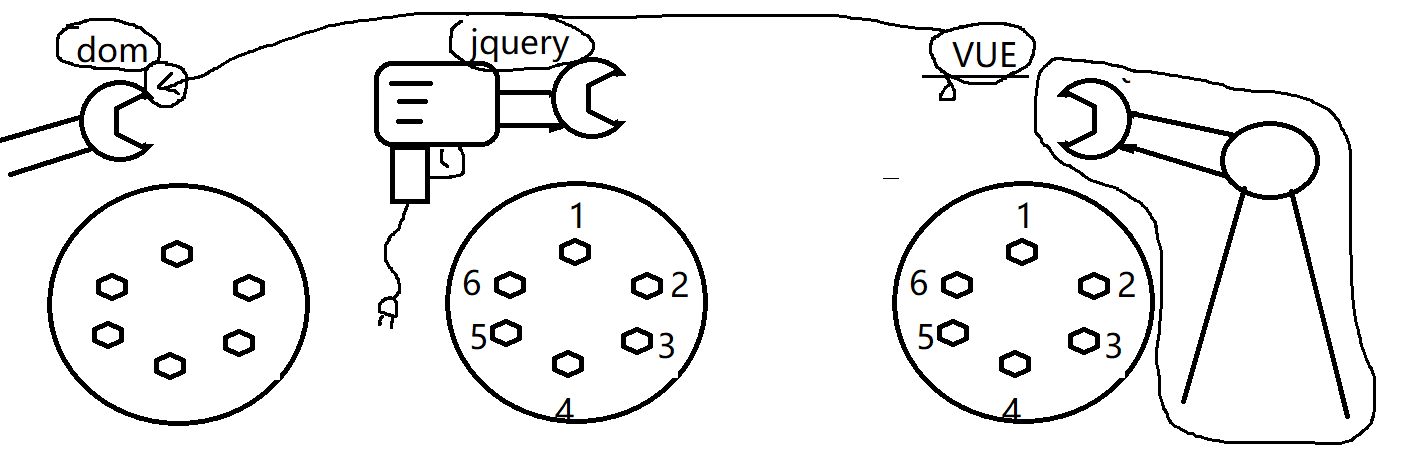
5. 修改元素

对应小程序视频: 小程序->在线->JQUERY->day01 1. 什么是jQuery, 如何使用jQuery

一. 什么是jQuery：

1. 什么是:

(1). 第三方开发的: 先下载才能使用。

(2). 执行DOM操作的: jQuery还是执行DOM5件事: 增删改查+事件绑定。学习jQuery还是在学DOM。  
 (3). 极简化的: jQuery是对DOM操作的终极简化。  
  
 (4). 函数库: jQuery使用函数来解决一切问题。

2. 为什么:

(1). 简化

(2). 解决了大部分浏览器兼容性问题

3. 何时: 几乎所有旧的项目和框架都是用jQuery做出来的。

4. 问题: jQuery将来会用的越来越少！

(1). 只简化了每一步操作，但是没有简化步骤！依然存在大量重复的编码！

(2). 不支持移动端！

二. 如何使用jQuery

1. 下载:   
 (1). 官网: jquery.com

(2). 版本:

a. 1.x: 兼容旧浏览器

b. 2.x: 不再兼容旧浏览器, 暂不支持es6

c. 3.x: 不再兼容旧浏览器，支持ES6

(3). 压缩 vs 未压缩:

a. 未压缩版： jquery-1.11.3.js

1). 有完备的注释、代码格式、见名知意的变量名。

2). 好处: 可读性好，适合于开发和学习之用

3). 缺点: 体积大，不便于快速下载使用，不适合于生产环境

b. 压缩版：jquery-1.11.3.min.js

1). 去掉了所有注释和代码格式，极简化了变量名。

2). 好处: 体积小，便于快速下载使用，适合于生产环境

3). 缺点: 可读性查，不适合于学习之用

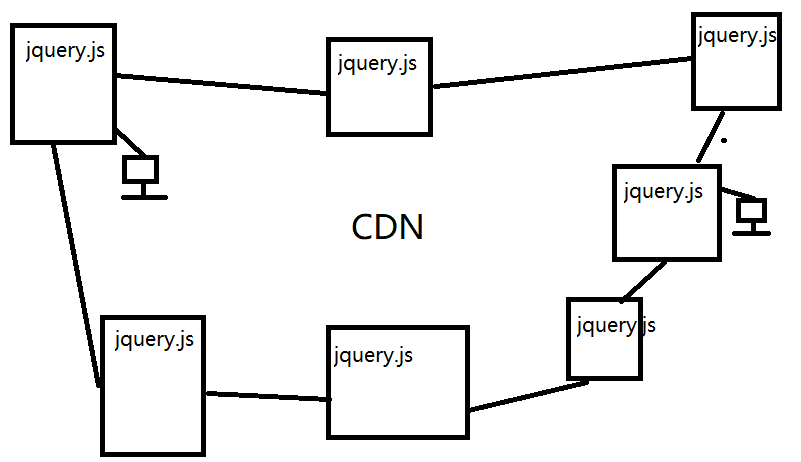
2. 引入网页: 2种:

(1). 使用项目本地的js文件: 先将jquery.js文件下载到项目本地的文件夹中，在网页中使用相对路径引入:

<script src="js/jquery-1.11.3.js">

(2). 使用CDN网络中共享的js文件:

a. CDN网络: 在全球范围多点共享资源，并能够根据用户所在的位置，自动选择离客户最近的服务器下载资源 的智能网络



b. 如何使用cdn网络中共享的jquery.js: 只要找到网址即可:

<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.3.js">

3. 我的第一个jquery程序:

对应小程序视频: 小程序->在线->JQUERY->day01 2. 第一个jQuery程序 jQuery原理

(1). 示例: 使用DOM记录按钮点击次数：

1\_DOM.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>  </head>  <body>    <h1>jQuery/$工厂函数</h1>    <button id="btn1">click me(0)</button>    <!--              0123456789-1                     .innerHTML                     .slice(   9,-1)-->    <script>      //DOM 4步      //1. 查找触发事件的元素      var btn=document.getElementById("btn1")      //2. 绑定事件处理函数      btn.onclick=function(){        //3. 查找要修改的元素        //本例中: 修改自己this，不用找        //4. 修改元素        //4.1 取出现在的点击次数转为整数        //                      复习第一阶段slice        var n=parseInt(this.innerHTML.slice(9,-1));        //4.2 次数+1        n++;        //4.3 放回按钮内容中，但是拼的格式要和最初一模一样        this.innerHTML=`click me(${n})`;      }    </script>  </body>  </html>  运行结果: |

(2). 示例: 使用jQuery记录按钮点击次数

1\_jQuery.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>  </head>  <body>    <h1>jQuery/$工厂函数</h1>    <button id="btn1">click me(0)</button>    <!--              0123456789-1                     .innerHTML                     .slice(   9,-1)-->    <script src="js/jquery-1.11.3.js"></script>    <script>      //DOM 4步      //1. 查找触发事件的元素      // var btn=document.getElementById("btn1");      //创建一个jQuery类型的子对象，并查找id为btn1的DOM按钮对象，保存到新创建的jQuery子对象中，等待操作。      var $btn=$("#btn1")      //2. 绑定事件处理函数      // btn.onclick=function(){      //为btn绑定单击事件处理函数      $btn.click(function(){        //3. 查找要修改的元素        //本例中: 修改自己this，不用找        //因为this默认指原生DOM元素对象，所以不能直接调用jQuery家简化版函数。所以必须用$()将this包装为一个新的jQuery家子对象，才能继续使用jQuery家简化版函数。        var $this=$(this);        //4. 修改元素        //4.1 取出现在的点击次数转为整数        // var n=this.innerHTML.slice(9,-1);        var n=parseInt($this.html().slice(9,-1));        //4.2 次数+1        n++;        //4.3 放回按钮内容中，但是拼的格式要和最初一模一样        // this.innerHTML=`click me(${n})`;        $this.html(`click me(${n})`)      //}      })    </script>  </body>  </html>  运行结果: |

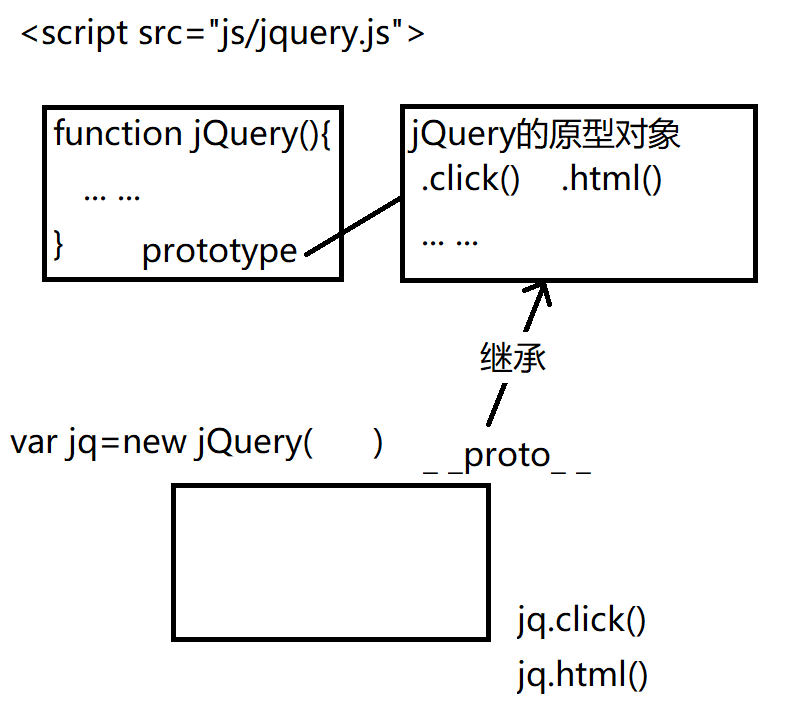
三. jQuery原理

1. 引入jquery.js后: 在全局创建了一种新的类型jQuery

(1). 构造函数: 负责创建当前类型的一个新的子对象

(2). 原型对象: 负责保存该类型所有子对象共有的方法

2. 要想使用jQuery家所有简化版函数，必须先创建jQuery家的子对象。然后，才能用子对象调用jQuery原型对象中的简化版函数



3. 问题: jQuery的简化版函数，都是为了简化DOM操作，来操作DOM元素的。比如: .click()代替DOM的onclick为元素绑定事件。.html()就是代替innerHTML获取或修改元素的内容。但是，如果只创建jQuery家子对象，是不知道该用这些函数操作哪个DOM元素的！

4. 解决: 今后创建jQuery子对象时，都会顺便传入一个DOM元素，供jQuery简化版函数去操作！2种方式:

(1). 在创建jQuery子对象时，提供一个选择器，去查找DOM元素，并将找到的DOM元素放入新创建的jQuery子对象中。

var jq子对象=new jQuery("选择器")

(2). 也可以直接在new jQuery()中放入一个已经获得的DOM元素参数。表示将一个原生DOM元素封装为jQuery类型的子对象

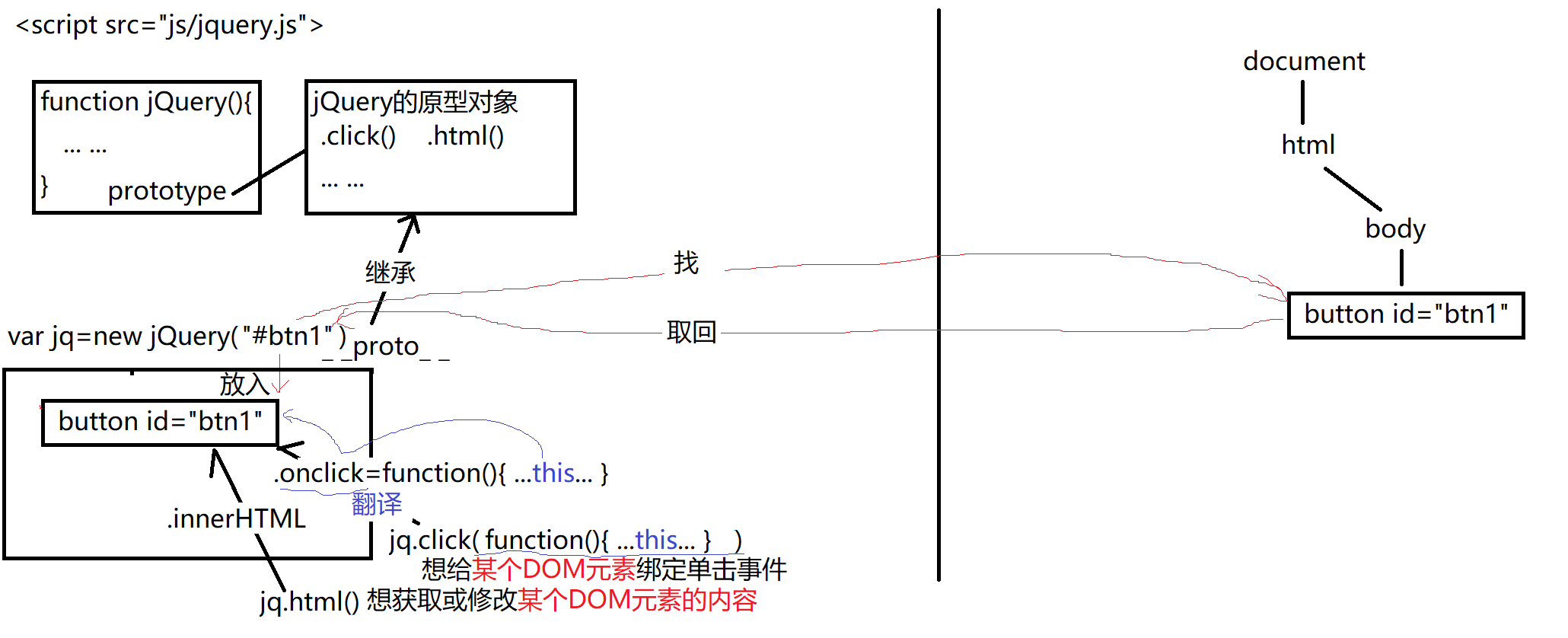
var jq子对象=new jQuery(DOM元素对象)

比如: var $this=$(this)

5. 结果: 将来对jQuery子对象调用那些简化版的函数，其实操作的就是找到的这个DOM元素。

6. 问题: 简化版函数如何变成原生DOM的属性和方法呢？

7. 解决: 其实，每当我们调用一个简化版函数时，jq子对象都会自动将简化版函数翻译为对应的原生的属性和方法



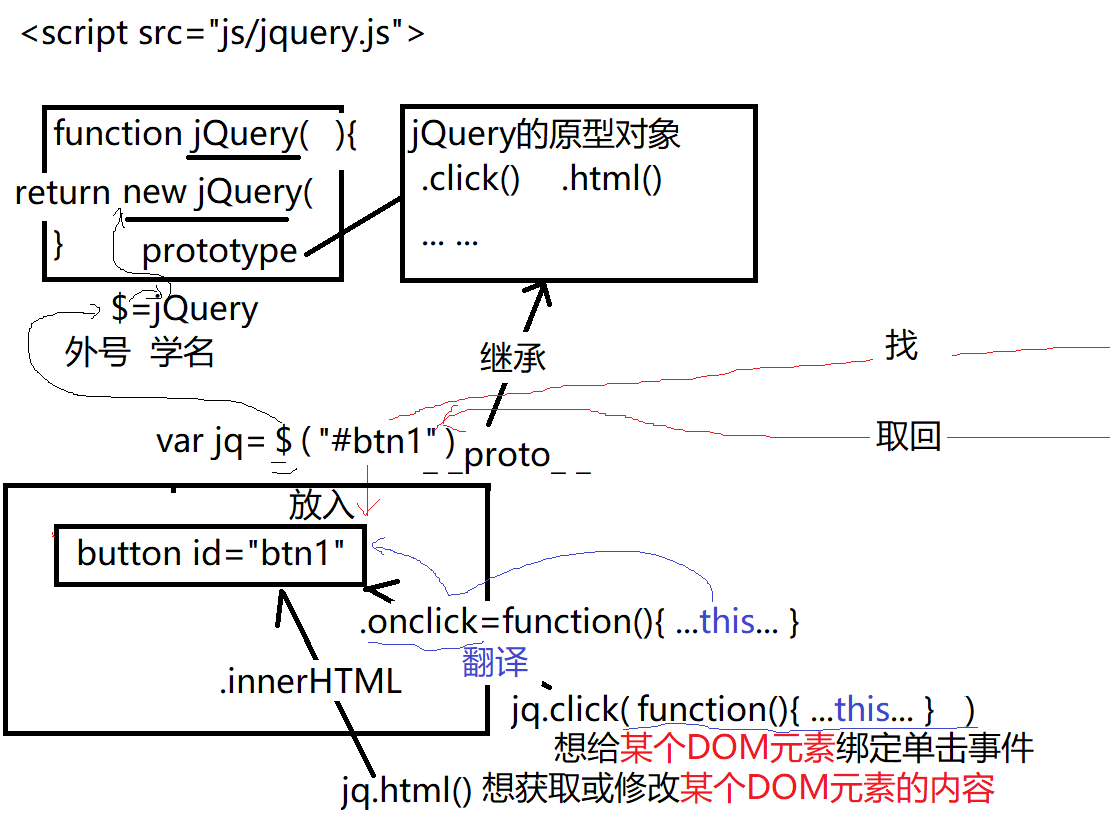
8. 问题: jquery的作者认为，每次使用jquery都要new jQuery()太繁琐！

9. 解决: 简化:

(1). 将new jQuery()放入jquery构造函数内。结果: 今后创建jQuery子对象时，不用每次都写new，只写jQuery()即可

(2). 给jQuery起了一个别名叫$。从此要想创建jquery类型的子对象，之用$()即可。

10. $的本质: 创建一个jQuery类型的子对象，等效于new jQuery()



11. 问题: 使用原生的DOM查找函数找到的元素对象和使用jquery的$()找到的结果对象是不一样的！

(1). 原生的DOM查找函数，找到的直接就是原生的DOM元素

(2). $()最后获得的是一个jQuery类型的子对象，其中包含着找到原生DOM元素对象

如何区分？

12. 解决: 今后，凡是用$()找到的结果对象，都要以$开头命名！

13. 问题: jQuery事件处理函数中的this

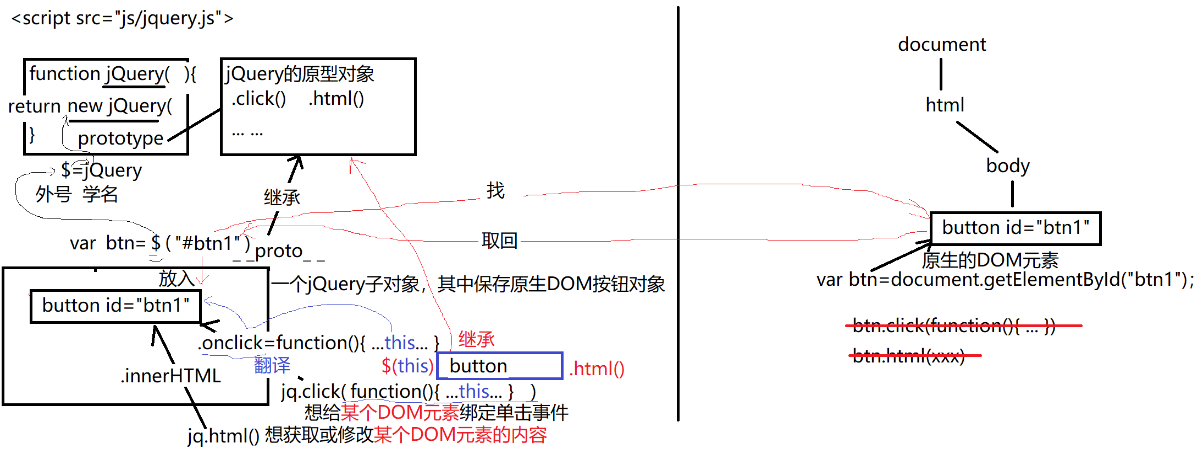
(1). jQuery事件处理函数的原理和DOM事件处理函数的原理是一模一样的。$btn.click(function(){ ... })就等效于btn.onclick=function(){ ... }

(2). 所以, jQuery事件处理函数中的this->当前触发事件的DOM元素对象

(3). 但是, this默认指的是当前正在触发事件的原生DOM元素，无法使用jQuery家所有简化版函数

14. 解决: 只要将this，用$()包装一下，包装成一个jQuery类型的子对象。就可以使用jQuery类型的子对象调用简化版函数，来操作this指向的DOM元素对象。

var $btn=$(this)



15. 问题: 案例中, 获取按钮的内容用.html()，修改按钮的内容用的还是.html()。一个函数，怎么既能获取，又能修改呢？

16. 总结: 其实大部分jquery简化版函数都有三大特征:

(1). 一个函数两用: 多数和修改相关的函数，都是一个函数两用。即能获取旧值，又能修改新值。如何判断本次是该修改，还是该获取？重载。

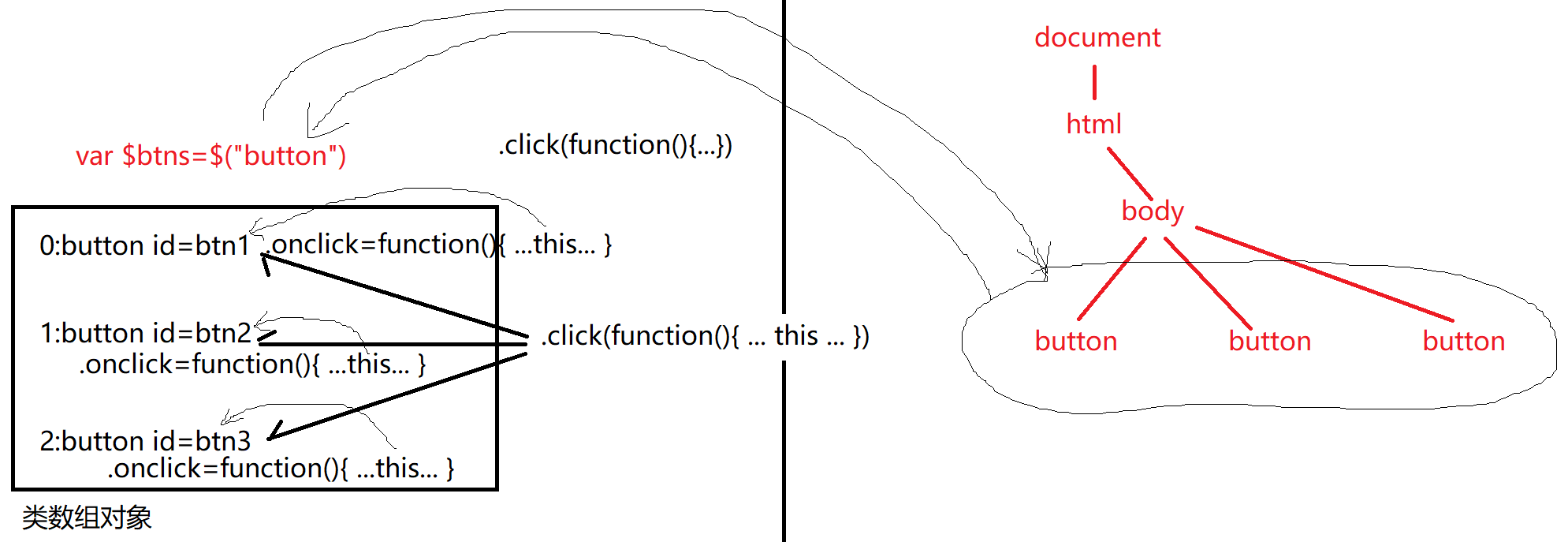
a. 如果本次调用函数时，没有传新值，就默认执行获取旧值的操作

b. 如果本次调用函数时，传了新值，就自动切换为修改操作。

(2). 自带遍历:

a. 其实, jquery查找结果对象是一个类数组对象。无论选择器找到多少元素，都可以包裹进jquery对象中

b. 只要对整个jQuery子对象调用一次简化版函数，就等效于对jquery对象内每个DOM元素都调用一次对应的原生DOM属性或方法。无需自己写for循环。



c. 示例: 使用jQuery记录三个按钮各自的单击次数

2\_jQueryAPI.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>  </head>  <body>    <h1>jQueryAPI特点</h1>    <button id="btn1">click me(0)</button>    <button id="btn2">click me(0)</button>    <button id="btn3">click me(0)</button>    <script src="js/jquery-1.11.3.js"></script>    <script>      //DOM 4步      //1. 查找触发事件的元素      //本例中三个按钮都可点击      var $btns=$("button");      console.log($btns);      //2. 绑定事件处理函数      //为每个按钮都绑定单击事件      $btns.click(function(){    //相当于: for(...)    //$btns[0].onclick=function(){ ...this... };    //$btns[1].onclick=function(){ ...this... };    //$btns[2].onclick=function(){ ...this... };        //3. 查找要修改的元素        var $this=$(this);        //4. 修改元素        //4.1 取出现在的点击次数转为整数        // var n=this.innerHTML.slice(9,-1);        var n=parseInt($this.html().slice(9,-1));        //4.2 次数+1        n++;        //4.3 放回按钮内容中，但是拼的格式要和最初一模一样        // this.innerHTML=`click me(${n})`;        $this.html(`click me(${n})`)      })      </script>  </body>  </html>  运行结果: |

(3). （待续...）

四. 查找:

1. jQuery中支持使用CSS选择器查找元素。并且扩展了一些css没有，但是jquery中独有的新选择器。

2. 基本过滤选择器:

对应小程序视频列表: 小程序->在线->JQUERY->day01 3. 子元素过滤 child filter 基本过滤

(1). 回顾子元素过滤选择器: (css中的)

a. 什么是: 根据元素在其父元素内的相对下标位置来选择子元素。

b. 包括：

1). :first-child : 选择作为其所在的父元素下的第一个孩子的子元素。

2). :last-child : 选择作为其所在的父元素下的最后一个孩子的子元素。

3). :nth-child(i) : 选择作为其所在的父元素下的第i个孩子的子元素。i从1开始！

4). :only-child : 选择作为其所在的父元素下的唯一一个子元素的元素。

c. 示例:使用子元素过滤选择并修改指定元素:

5\_child filter.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>   <meta charset="utf-8" />   <title></title>  </head>  <body>    <h3>子元素过滤选择器.</h3>    <ul>      <li>child1-basic0</li>      <li>child2-basic1</li>      <li>child3-basic2</li>    </ul>    <ul>      <li>child1-basic3</li>      <li>child2-basic4</li>      <li>child3-basic5</li>    </ul>    <ul>      <li>child1-basic6</li>    </ul>    <script src="js/jquery-1.11.3.js"></script>    <script>      //复习第二阶段子元素过滤选择器。      //查找每个ul中第一个li      $("ul>li:first-child")      .css("border","2px solid red");      //相当于.style.border="2px solid red";      //查找每个ul中最后一个li      $("ul>li:last-child")      .css("color","green");      //查找每个ul中处于偶数位置的      $("ul>li:nth-child(2n)")      .css("box-shadow","0 0 5px blue")      //查找每个ul中第2个li      $("ul>li:nth-child(2)")      .css("background-color","yellow")      //查找作为ul下唯一子元素的li      $("ul>li:only-child")      .css("font-size","32px")    </script>  </body>  </html>  运行结果: |

(2). 基本过滤选择器: （jq新增，css中没有）

a. 什么是: 先将所有符合条件的元素集中保存在一个大的集合中，统一编号！然后，再按照元素在集合中的大排名，选择元素！

b. 包括：

1). :first 选择所有符合要求的元素中第一个元素

2). :last 选择所有符合要求的元素中最后一个元素

3). :even 选择所有符合要求的元素中下标位置为偶数的元素

4). :odd 选择所有符合要求的元素中下标位置为奇数的元素

5). :eq(i) eq是equal(等于)的简写。选择所有符合要求的元素中下标等于i位置的元素

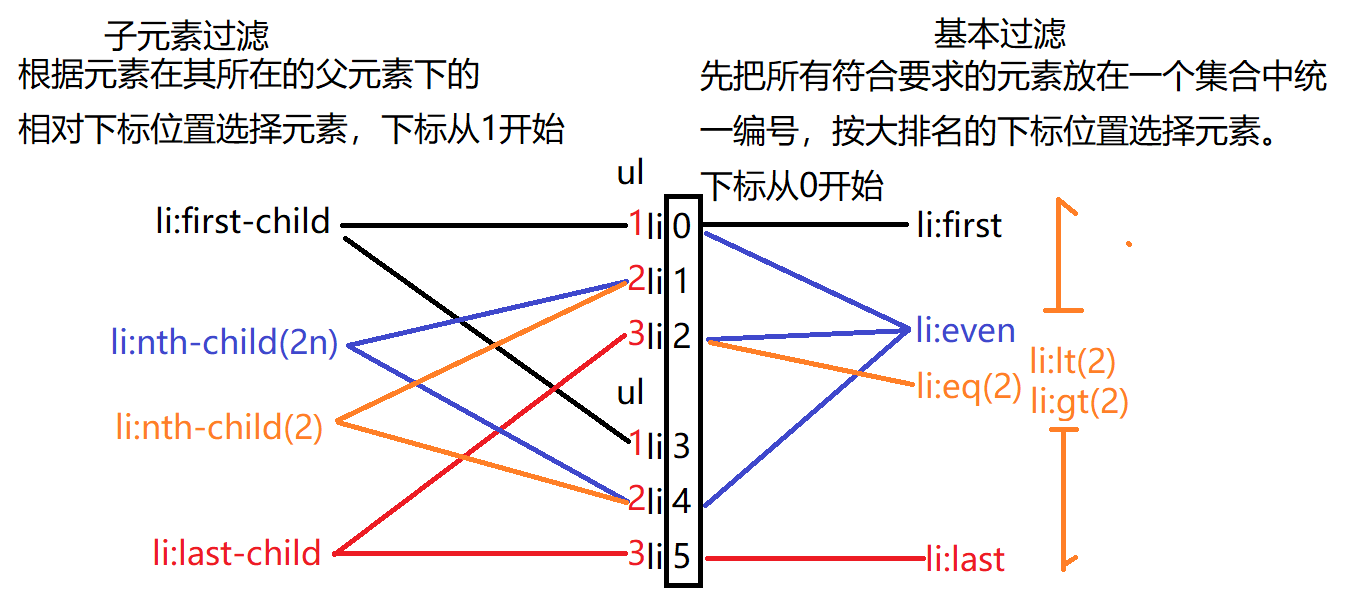
6). :gt(i) gt是greater than(大于)的简写。选择所有符合要求的元素中下标>i位置的所有元素

7). :lt(i) lt是less than(小于)的缩写。选择所有符合要求的元素中下标<i位置的所有元素

c. 强调:

1). 因为jq中新增的选择器，底层是用js程序模拟的。所以，jq新增的选择器的下标都从0开始！

2). :even虽然表示偶数位置的意思，但是，因为下标从0开始，实际选择的是正常人眼里的奇数位置的元素; :odd虽然表示奇数位置的意思，但是因为下标从0开始，实际选择的是正常人眼里的偶数位置的元素



d. 示例:使用基本过滤选择并修改元素

6\_basic filter.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <meta charset="utf-8" />    <title></title>  </head>  <body>    <h3>基本过滤选择器.</h3>    <ul>      <li>child1-basic0</li>      <li>child2-basic1</li>      <li>child3-basic2</li>    </ul>    <ul>      <li>child1-basic3</li>      <li>child2-basic4</li>      <li>child3-basic5</li>    </ul>    <ul>      <li>child1-basic6</li>    </ul>    <script src="js/jquery-1.11.3.js"></script>    <script>      //查找第一个li      $("ul>li:first").css("border", "2px solid red")      //查找最后一个li      $("ul>li:last").css("color", "green");      //查找处于偶数(用户视角)位置的li      $("ul>li:odd").css("box-shadow", "0 0 5px blue");      //查找第2个(用户视角)li      $("ul>li:eq(1)").css("background-color", "yellow");    </script>  </body>  </html>  运行结果: |

(3). 示例: 按钮组效果:

7\_basic filter2.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <meta charset="utf-8" />    <title>...</title>    <style>      ul{        list-style:none      }      ul>li{        float:left;        border:1px solid #aaa;        padding:5px 10px;      }      /\*用纯css实现按钮组效果\*/      /\*第一个li左上，左下变圆角\*/      /\* ul>li:first-child{        border-radius:5px 0 0 5px      } \*/      /\*最后一个li右上, 右下变圆角\*/      /\* ul>li:last-child{        border-radius:0 5px 5px 0      } \*/      /\*将第一li之后所有兄弟去掉左边框\*/      /\* ul>li~li{        border-left:0      } \*/    </style>    <script src="js/jquery-1.11.3.js"></script>  </head>  <body>    <h1>实现按钮组效果</h1>    <ul>      <li>1</li>      <li>2</li>      <li>3</li>      <li>4</li>    </ul>    <script>      /\*用js实现按钮组效果\*/      /\*第一个li左上，左下变圆角\*/      $("ul>li:first").css("border-radius","5px 0 0 5px");      /\*最后一个li右上, 右下变圆角\*/      $("ul>li:last").css("border-radius","0 5px 5px 0");      /\*将第一li之后所有兄弟去掉左边框\*/      $("ul>li:gt(0)").css("border-left",0)    </script>  </body>  </html>  运行结果: |

(4). 示例: 实现表格隔行变色效果：

7\_basic filter3.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <meta charset="utf-8" />    <title>...</title>    <style>      table{        width:400px;        border-collapse: collapse;        text-align:center;      }      table>thead{        border-bottom:3px solid #aaa      }      /\*表格隔行变色效果\*/      /\*tbody中第一行之后的所有兄弟行加上边框\*/      /\* tbody>tr~tr{        border-top:1px solid gray      } \*/      /\*偶数行变色\*/      /\* tbody>tr:nth-child(2n){        background-color:lightBlue;      } \*/    </style>    <script src="js/jquery-1.11.3.js"></script>  </head>  <body>    <h1>实现表格隔行变色效果</h1>    <!--table>(thead>tr>th\*4)+(tbody>tr\*4>td\*4)-->    <table>      <thead>        <tr>          <th>#</th>          <th>First Name</th>          <th>Last Name</th>          <th>User Name</th>        </tr>      </thead>      <tbody>        <tr>          <td>1</td>          <td>亮</td>          <td>成</td>          <td>@liang</td>        </tr>        <tr>          <td>2</td>          <td>然</td>          <td>李</td>          <td>@ran</td>        </tr>        <tr>          <td>3</td>          <td>东</td>          <td>张</td>          <td>@dong</td>        </tr>        <tr>          <td>4</td>          <td>华</td>          <td>吴</td>          <td>@hua</td>        </tr>      </tbody>    </table>    <script>      /\*表格隔行变色效果\*/      /\*tbody中第一行之后的所有兄弟行加上边框\*/      $("tbody>tr:gt(0)").css("border-top","1px solid gray")      /\*偶数行变色\*/      $("tbody>tr:odd").css("background-color","lightBlue")    </script>  </body>  </html>  运行结果: |

(5). 总结: 如果今后一个效果既可以用css来实现，又可以用js来实现，首选css。

a. css的效率远比js高的多！

b. css的一致性很好！现在写好的css，将来到框架中，复制粘贴过去，直接可用！几乎不用修改！而现在写的js代码，将来到了框架中，几乎都要重新编写！

对应小程序视频: 小程序->在线->JQUERY->day01 4. jQuery其它选择器

3. 内容过滤选择器: (css中没有，jq新增的)

(1). 什么是: 根据元素的内容中的关键词或子元素特征来选择元素。

(2). 包括:

a. :contains(文本) 选择元素内容中包含指定文本的元素

b. :has(选择器) 选择包含符合要求的子元素的父元素

c. :parent 选择内容非空的元素

d. :empty 选择内容为空的元素

(3). 示例: 使用内容过滤选择器选择元素并修改元素

10\_content filter.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <link rel="stylesheet" href="css/bootstrap.css"/>    <style>    </style>  </head>  <body>  <div class="container">  <h1>jQuery中的选择器——内容过滤选择器</h1>    <button>提交订单</button>    <button>Submit注册信息</button>    <button>马上提交</button>    <button>清空重填</button>    <hr/>    <div class="alert" id="alert1"></div>    <div class="alert" id="alert2">      <span class="close">×</span>    </div>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>    //选择包含"提交"二字的按钮，变为绿色按钮    $("button:contains(提交)")    .css("background-color","lightGreen")    //选中包含.close按钮的.alert元素，让它们变为红色的警告框    $(".alert:has(.close)")    .css("background-color","pink")    //选中不包含.close按钮的.alert元素，让它们变为绿色的警告框    $(".alert:not(:has(.close))")    .css("background-color","lightGreen")    //选择内容为空的.alert元素，添加红色阴影    $(".alert:empty")    .css("box-shadow","0 0 5px red")    //选择内容不为空的.alert元素，添加绿色阴影    $(".alert:parent")    .css("box-shadow","0 0 5px green")  </script>  </body>  </html>  运行结果: |

4. 可见性过滤选择器:

(1). 什么是: 根据元素是否隐藏来选择隐藏的元素

(2). 包括:

a. :visible 专门选择可见的元素

b. :hidden 专门选择不可见的元素

(3). 坑: HTML+CSS中要隐藏一个元素有四种办法:

a. display:none

b. opacity:0

c. visibility:hidden

d. <input type="hidden">

但是，:hidden只能选择其中2种: display:none和<input type="hidden">

5. 表单元素过滤选择器:

(1). :input 专门选择一个表单中的所有表单元素(input, select, textarea, button)

vs input 是css中的普通的元素选择器，只能选择input元素

(2). 问题: 仅仅是input元素就有很多种！如何精确的选择某一类input元素呢？

(3). 解决: jq中为每种input的type属性值都提供了一个专门的选择器

a. :text 专门选择普通文本框<input type="text">

b. :radio 专门选择所有单选按钮<input type="radio">

c. :button 专门选择所有input按钮<input type="button">

... HTML中<input>元素有几种type值，jq就新增了几种对应选择器 ...

(4). 示例: 使用表单元素过滤选择器实现点同意才启用元素效果

13\_form state selector.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title>.....</title>    <meta charset="utf-8"/>    <style>    </style>   </head>   <body>    <form>      用户名:<input disabled></br><span>\*</span>      密码:<input type="password" disabled></br><span>\*</span>      <input type="checkbox">我同意本站的使用条款<br>      <input type="submit" value="提交注册信息" disabled/>    </form>    <script src="js/jquery-1.11.3.js"></script>    <script>      //DOM 4步      //1. 查找触发事件的元素      //本例中：用户点type=checkbox的复选框触发变化      var $chb=$(":checkbox")      //2. 绑定事件处理函数      $chb.click(function(){        //3. 查找要修改的元素        //本例中: 选择除了:checkbox之外的其余表单元素        var $others=$(":input:not(:checkbox)")        //4. 修改元素        //如果当前checkbox被选中时        if(this.checked==true){          //其它元素启用: disabled=false          //jq中修改标准属性，都可用.prop()代替DOM中的.操作！          $others.prop("disabled",false);        }else{//否则如果当前checkbox未选中          //其它元素禁用: disabled=true          $others.prop("disabled",true);        }      })    </script>   </body>  </html>  运行结果: |

五. 修改元素: 3种:

强调: 所有和修改相关的函数，一律一个函数两用

对应小程序视频: 小程序->在线->JQUERY->day01 5. jquery修改元素的内容 html() ...

1. 内容: 3种:

(1). 开始标签到结束标签之间的原始的HTML内容

a. DOM中: 元素.innerHTML

b. jq中: $元素.html("新HTML内容")

(2). 开始标签到结束标签之间的纯文本内容

a. DOM中: 元素.textContent

b. jq中: $元素.text("新文本内容")

(3). 表单元素的值:

a. DOM中: 元素.value

b. jq中: $元素.val("新值")

(4). 示例: 尝试获得元素的三种内容:

14\_html\_val.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title> new document </title>    <meta charset="utf-8">   </head>   <body>    <p>来自<a href="javascript:;">&lt;&lt;新华社&gt;&gt;</a>的消息</p>    <input><button>百度一下</button>    <script src="js/jquery-1.11.3.js"></script>    <script>      //获得p元素的原始html内容      console.log($("p").html());      //获得p元素的纯文本内容      console.log($("p").text());      //想点按钮获得文本框中用户输入的搜索关键词      $("button").click(function(){        //获得文本框的内容        var value=$("input").val();        console.log(`搜索 ${value} 相关的内容`)      })    </script>   </body>  </html>  运行结果: |

(5). 示例: 表单验证

14\_html\_val2.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title> new document </title>    <meta charset="utf-8">   </head>   <body>    <h1>操作元素的内容和值</h1>    <form action="">      用户名:<input type="text" name="uname">            <span></span><br>      密码:<input type="password" name="upwd">            <span></span><br>      <input type="button" value="提交注册信息">    </form>    <script src="js/jquery-1.11.3.js"></script>    <script>      //正确时，使用图片:"<img src='img/ok.png'>"      //姓名错误时: "<img src='img/err.png'>用户名必须介于3~9位之间!"      //密码错误时: "<img src='img/err.png'>密码必须介于6~8位之间!"      //DOM 4步:      //1. 查找触发事件的元素      //本例中:姓名文本框失去焦点时触发验证      var $inputName=$(":text");      //2. 绑定事件处理函数      $inputName.blur(function(){        var $this=$(this);        //3. 查找要修改的元素        //本例中：无论验证是否通过，都要修改当前input文本框旁边的下一个兄弟span        var $span=$this.next();                    // jq家的简化版函数！                //this.nextElementSibling;        //4. 修改元素        //先获得当前文本框中用户输入的姓名        var uname=$this.val();        //如果验证当前文本框的内容通过        if(uname.length>=3&&uname.length<=9){          //修改span的内容为验证通过的内容          $span.html(`<img src='img/ok.png'>`);        }else{//否则如果验证当前文本框的内容不通过          //修改span的内容为验证失败的内容          $span.html(`<img src='img/err.png'>用户名必须介于3~9位之间!`);        }      })    </script>   </body>  </html>  运行结果: |

2. 属性

3. 样式

总结:

DOM5件事: 增删改查+事件绑定

DOM 4步:

1. 查找触发事件的元素

2. 绑定事件处理函数

3. 查找要修改的元素

4. 修改元素:3步:

4.1 取出元素上的旧值(可省略)

4.2 计算出新值

4.3 将新值再修改回元素上！

总结: 事件:

1. 绑定事件:

(1). 在html中:

a. html中: <元素 on事件名="事件处理函数()">

b. js中: function 事件处理函数(){ ... }

(2). 在js中使用赋值方式: 一个事件只绑定一个处理函数

元素.on事件名=function(){ ... }

(3). 在js中用添加事件监听: 一个事件绑定多个处理函数

a. 元素.addEventListener("事件名", 事件处理函数)

b. 元素.removeEventListener("事件名", 原事件处理函数对象)

2. 事件模型: 捕获，目标触发，冒泡

3. 事件对象:

(1). 获得事件对象:

元素.on事件名=function(e){ ... }

(2). 阻止冒泡: e.stopPropagation()

(3). 当多个子元素都要绑定相同事件时，利用冒泡/事件委托3步:

a. 事件只在父元素上绑定一次

b. e.target代替this

c. 判断e.target的任意特征是否是我们想要的元素

(4). 阻止元素默认行为:

e.preventDefault()

(5). 获取鼠标位置:

a. 相对于屏幕左上角的x，y坐标:

e.screenX, e.screenY

屏幕 屏幕

b. 相对于文档显示区左上角的x，y用坐标:

e.clientX, e.clientY

客户端 客户端

c. 相对于事件所在元素左上角的x，y坐标:

e.offsetX e.offsetY

偏移 偏移

|  |
| --- |
| 总结: this 5种: 判断this，~~一定不要看定义在哪儿！~~只看调用时！  1. obj.fun() this->obj  2. fun() 或 (function(){ ... })() 或 多数回调函数 this->window  3. new Fun() this->new正在创建的新对象  4. 类型名.prototype.共有方法=function(){ ... } this->将来谁调用指谁，同第一种情况  5. 事件处理函数中的this->当前正在触发事件的DOM元素对象 |

总结: jQuery

1. 引入jQuery.js: 2种

(1). 引入项目本地jquery.js文件:

<script src="js/jquery-1.11.3.js">

(2). 引入CDN网络中共享的jquery.js文件:

<script src="官方提供的CDN上jquery.js文件地址">  
2. 创建jQuery类型子对象: 2种  
(1). 只有jQuery类型子对象才能使用jQuery家简化版函数。

DOM家元素对象无权直接使用jQuery家简化版函数。

所以只要使用jQuery家简化版函数，都要先创建jQuery家子对象，其中保存一个要操作的DOM元素对象。  
(2). 如何:2种:

a. 查找DOM元素对象，并保存进新创建的jQuery对象中:

var $jq子对象=$("选择器")

b. 不查找，直接将DOM元素对象保存进新创建的jQuery对象中:

var $jq子对象=$(DOM元素对象)

3. 原理:

(1). $=jQuery=new jQuery  
(2). jq子对象其实是一个类数组对象，可保存找到的多个DOM元素对象

(3). 对jq子对象调用简化版函数，会被自动翻译为对应的原生DOM的方法和属性。

所以jQuery中的所有简化版函数效果和DOM中原生方法和属性效果一样。

jQuery中的this、e、e.target等，和DOM中的完全一样！

4. jQuery简化版函数3大特点:

(1). 自带for循环: 对整个jquery子对象调用一次简化版函数，等效于对jQuery子对象中保存的每个DOM元素对象分别调用一次对等的DOM原生方法或属性——不用自己写for循环

(2). 一个函数两用: 调用函数时:

a. 没给新值作为参数，默认执行获取旧值的操作

b. 给了新值作为参数，自动切换为执行修改操作

(3). （待续...）

5. 查找元素:

(1). jQuery支持用所有CSS3选择器查找

(2). jQuery新增选择器: (css中不能用)

a. 基本过滤: (下标从0开始)

:first :last :eq(i) :lt(i) :gt(i) :even :odd

b. 内容过滤:

:contains(文本) :has(选择器) :parent :empty

c. 可见性过滤:

:visible :hidden(只能选择display:none和input type="hidden")

d. 表单元素过滤:   
:input :text :password :radio :checkbox ... ...  
(3).用节点间关系查找: 2大类关系, 6个属性

a. 父子关系：（待续...）

b. 兄弟关系：2个

1). 前一个兄弟: $元素.prev()

被自动翻译->元素.previousElementSibling

2). 下一个兄弟: $元素.next()

被自动翻译->元素.nextElementSibling

6. 修改元素: 3种:

修改中的所有函数，都是一个函数两用！

(1). 内容: 3种:

a. 原始HTML内容:

$元素.html("新HTML内容")

被翻译为-> .innerHTML

b. 纯文本内容:

$元素.text("纯文本内容")

被翻译为-> .textContent

c. 表单元素的值:

$元素.val("新值")

被翻译为-> .value

(2). 属性: 3种:

a. 字符串类型的HTML标准属性:（待续...）  
b. bool类型的HTML标准属性:

$元素.prop("属性名", bool值)

被翻译为->元素.属性名=bool值

c. 自定义扩展属性:（待续...）

(3). 样式:

a. 获取或修改内联样式:

$元素.css("css属性名", "属性值")

获取属性值时被翻译为getComputedStyle(元素)

修改属性值时被翻译为.style.css属性=属性值

b. 使用class批量修改样式: （待续...）

8. 添加删除替换克隆元素: （待续...）

9. 事件绑定: $jq对象.事件名(function(){ ... })

今日对应小程序视频列表:

小程序->在线->JQUERY->day01 1. 什么是jQuery, 如何使用jQuery

2. 第一个jQuery程序 jQuery原理

3. 子元素过滤 child filter 基本过滤

4. jQuery其它选择器

5. jquery修改元素的内容 html() ...

作业:

1. 复习今日小程序中问题清单: 小程序->首页->JQUERY->day01

2. 复习第二阶段所有选择器！

3. 预习小程序中明日课程内容: 小程序->在线->JQUERY->day02

4.（凡是觉得DOM知识点串不起来）看小程序视频学习DOM版学子商城项目(持续到下周一之前)

小程序->在线->DOM->day04 扩展: DOM版学子商城...

家里环境xampp和nodejs能用，且第一阶段学的好的同学:

从小程序中网盘地址中下载xzserver\_start.zip，看视频中day01和day02，跟着做

家里环境xampp和nodejs不能用，或者第一阶段学的不好的同学：

从小程序中网盘地址中只下载public.zip，看视频中day01和day02，跟着做

然后，所有网页用live server运行

视频中所有ajax请求的服务器端接口地址，都改为我的新浪云服务器地址，就不需要你再运行服务器端了。也不需要你了解任何服务器端知识了:

/index 改为 http://xzserver.applinzi.com/index

/details 改为 http://xzserver.applinzi.com/details?lid=xxx