提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题清单都已经整理在小程序->首页->JQUERY->day01里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词-> JQUERY ->day01里

3. 今天的重点视频、扩展高频鄙视题视频已经整理在小程序->在线-> JQUERY ->day01里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，功能不会做，立刻问！

4. 刷笔试题时遇到不会的笔试题，立刻问！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

每个案例三遍:

1. 连注释+代码抄一遍

2. 保留注释，删掉代码，自己试着填回来

3. 删除注释和代码，自己使用用人话把注释写回来！——才算自己会！

正课:

1. 什么是jQuery

2. 如何使用jQuery

3. jQuery的原理

4. 查找元素

一. 什么是jQuery:

1. 什么是: jQuery是第三方开发的执行DOM操作的极简化的函数库

(1). 第三方开发: 既不是自己写的，也不是浏览器原生自带的，必须先下载才能使用的东西。比如: bootstrap

(2). 执行DOM操作: jQuery还是DOM，学习jQuery还是在学习DOM，而且jQuery还是做DOM五件事: 增删改查+事件绑定

(3). 极简化:

a. 优点: jQuery对DOM操作的每一步都进行了终极的简化！

1). 比如:

i. DOM中: 按id查找一个元素:

var 一个元素=document.getElementById("id名")

ii. jq中: var 一个元素=$("#id名")

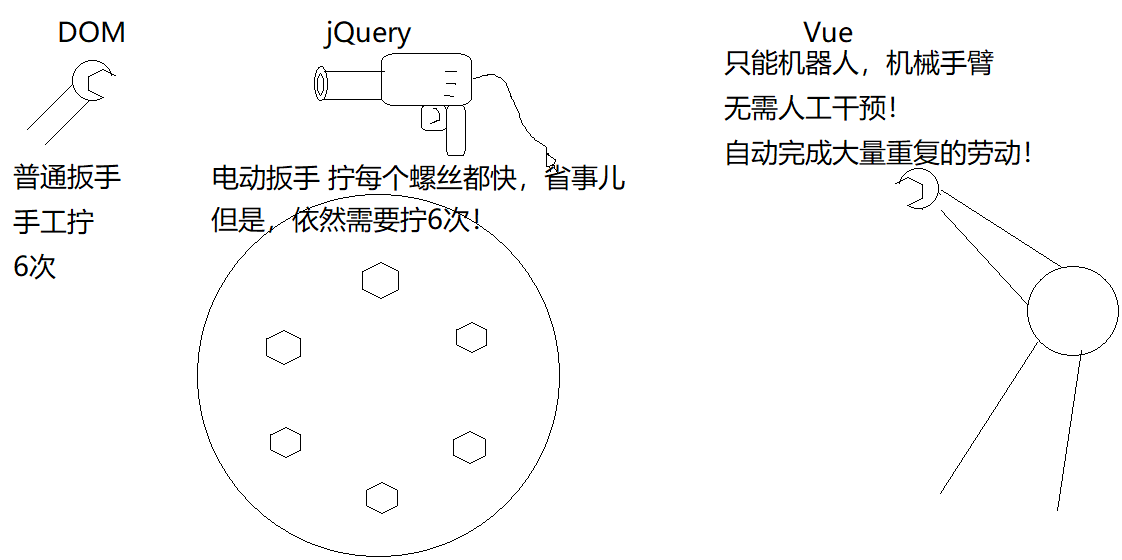
2). 比如:

i. DOM中: 获取一个元素的下一个兄弟元素: 元素.nextElementSibling()

ii. jq中: 元素.next()

b. 问题: 仅仅对每一步进行了简化，但是没有从根本上减少开发步骤！

c. 解决: 框架才是从根本上彻底减少了开发步骤！



(4). 函数库: jQuery中是用函数解决一切问题。没有属性了

比如: 获取元素的内容:

DOM中: 元素.innerHTML 属性

Jq中: 元素.html() 方法

2. 何时: 在框架出现之前，几乎所有旧的项目和旧的库都是基于jQuery开发的.

但是，因为框架的出现和移动端的大量需求，到时jQuery逐渐失宠！

3. 为什么:

(1). 简单！

(2). 解决了大部分浏览器兼容性问题: 只要jQuery让用的，都没有兼容性问题

4. 缺点:

(1). 依然包含大量重复的编码

(2). 不支持移动端！

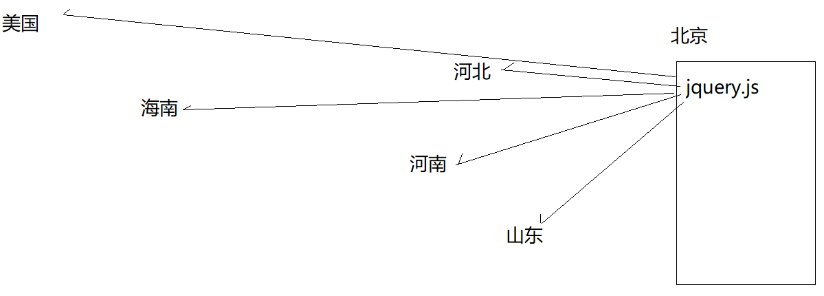
二. 如何使用jQuery:

1. 下载: jquery.com 2种方式:

(1). 将jquery.js文件下载到项目本地，在网页中使用相对路径引用js文件

a. 在HTML页面中: <script src="js/jquery-1.11.3.js">

b. 缺点: 因为我的服务器很可能只部署在一个地点，导致不同地区的用户，下载jquery.js文件的速度可能不相同！

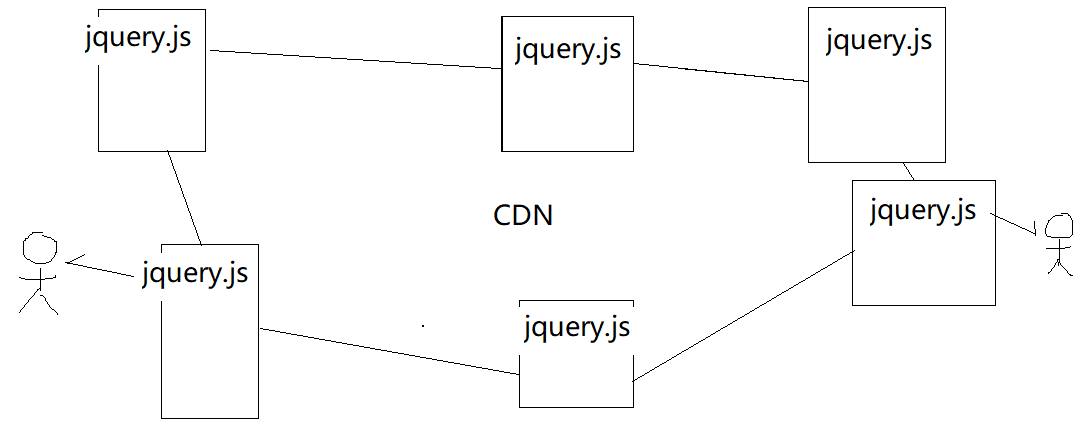


(2). 使用CDN网络中共享的jquery.js文件

a. CDN网络: 基于全球互联网部署的智能内容分发网络。

b. 原理: 在全球各地都有服务器，所有服务器的内容实时保持同步。而且CDN网络可根据用户所处的位置距离哪台服务器网络状况最好！来选择最优的服务器为用户下载资源。

c. 结果: 全球各地的用户，访问同一种文件，下载速度差别不大的！



d. 如何: <script src="https://code.jquery.com/jquery-1.12.4.js"></script>

2. 版本:

1.x: 只有1.x支持旧浏览器(IE8)

未压缩版: 包含完备的注释、代码格式和见名知义的变量名，可读性好，但是体积大，下载慢，所以适合于学习和开发时使用。不适合于生产环境发布上线后使用。

压缩版: 去掉了所有注释、代码格式，极简化了变量名，可读性差，但是体积小，下载快，所以适合于生产环境发布上线后使用。不适合于学习和开发时使用。

上课用的是 jquery-1.11.3.js

2.x: 不再支持旧浏览器(IE8)

3.x: 不再支持旧浏览器(IE8)，还添加了ES5和ES6的支持.

3. 我的第一个jQuery程序:

示例: 分别使用DOM和jQuery实现记录一个按钮的点击次数

1\_DOM.html

|  |
| --- |
| <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css">  <script  src="https://code.jquery.com/jquery-1.12.4.js"></script>  </head>  <body>  <h1>jQuery/$工厂函数</h1>  <button id="btn1" class="btn btn-success">click me(0)</button>  <!-- <script src="js/jquery-1.11.3.js"></script> -->  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 查找id为btn1的按钮  var btn1=document.getElementById("btn1");  //2. 绑定事件处理函数  btn1.onclick=function(){    //3. 查找要修改的元素    //本例中: 就是要改自己的内容(this)    //4. 修改元素    //4.1 先取出按钮的内容中的数字，转为整数    var n=parseInt(//parseInt天生不但可以转为整数，而且还可自动去掉数字后非数字字符！      this.innerHTML //"click me(0)"                     // 01234567890          .slice(9)  //          ||    )    //想要0    //4.2 将数字+1    n++    //4.3 将新数字放回按钮中    this.innerHTML=`click me(${n})`;  }  </script>  </body> |

1\_jQuery.html

|  |
| --- |
| <head lang="en">  <meta charset="UTF-8">  <title></title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css">  <script  src="https://code.jquery.com/jquery-1.12.4.js"></script>  </head>  <body>  <h1>jQuery/$工厂函数</h1>  <button id="btn1" class="btn btn-success">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: 查找id为btn1的按钮  //var btn1=document.getElementById("btn1");  var $btn1=$("#btn1");  //2. 绑定事件处理函数  //btn1.onclick=function(){  $btn1.click(function(){    //3. 查找要修改的元素    //本例中: 就是要改自己的内容(this)    var $btn=$(this);    //4. 修改元素    //4.1 先取出按钮的内容中的数字，转为整数    var n=parseInt(//parseInt天生不但可以转为整数，而且还可自动去掉数字后非数字字符！      //this.innerHTML //"click me(0)"      $btn.html()      // 01234567890          .slice(9)    //          ||    )    //想要0    //4.2 将数字+1    n++    //4.3 将新数字放回按钮中    //this.innerHTML=`click me(${n})`;    $btn.html(`click me(${n})`)  })  </script>  </body> |

三. jQuery原理:

1. 引入<script src="js/jquery.js">时，内存中其实添加了一种新的类型，包括:

(1). 构造函数: 负责反复创建该类型的子对象

function jQuery(){

}

(2). 原型对象: 负责存储该类型的子对象共用的方法

jQuery.prototype:{

.click() .html() ... ...

}

2. 问题: 因为DOM家的元素对象，方法和属性都很繁琐不好用！所以，DOM家的元素对象想使用jQuery家的简化版函数来简化DOM操作。但是，不是一家人不进一家门！原生的DOM元素对象，默认无法使用jQuery家的简化版函数的。

3. 解决:要想调用jQuery家简化版函数，必须先创建jQuery家孩子才行！

4. 问题: 但是，因为jQuery家的简化版函数都是执行DOM操作的，最终效果都要作用到一个DOM家的孩子上！

5. 解决: 所以，今后只要创建jQuery家子对象时，都要提供一个选择器或一个DOM元素对象，来查找并将找到的DOM元素对象包装进jQuery家的孩子中。

6. 结果: jQuery对象肚子里就装了一个找到的DOM元素。将来对整个jQuery对象调用简化版的函数，其实都将作用于jQuery对象肚子里的这个DOM元素上。且所有的简化版函数，最终还是要被悄悄自动翻译为原生DOM函数或属性。

7. 问题: 每次想做DOM操作，都要创建jQuery对象，每次创建对象都要反复写new jQuery()！太长了！！！不愿意写！

8. 解决:

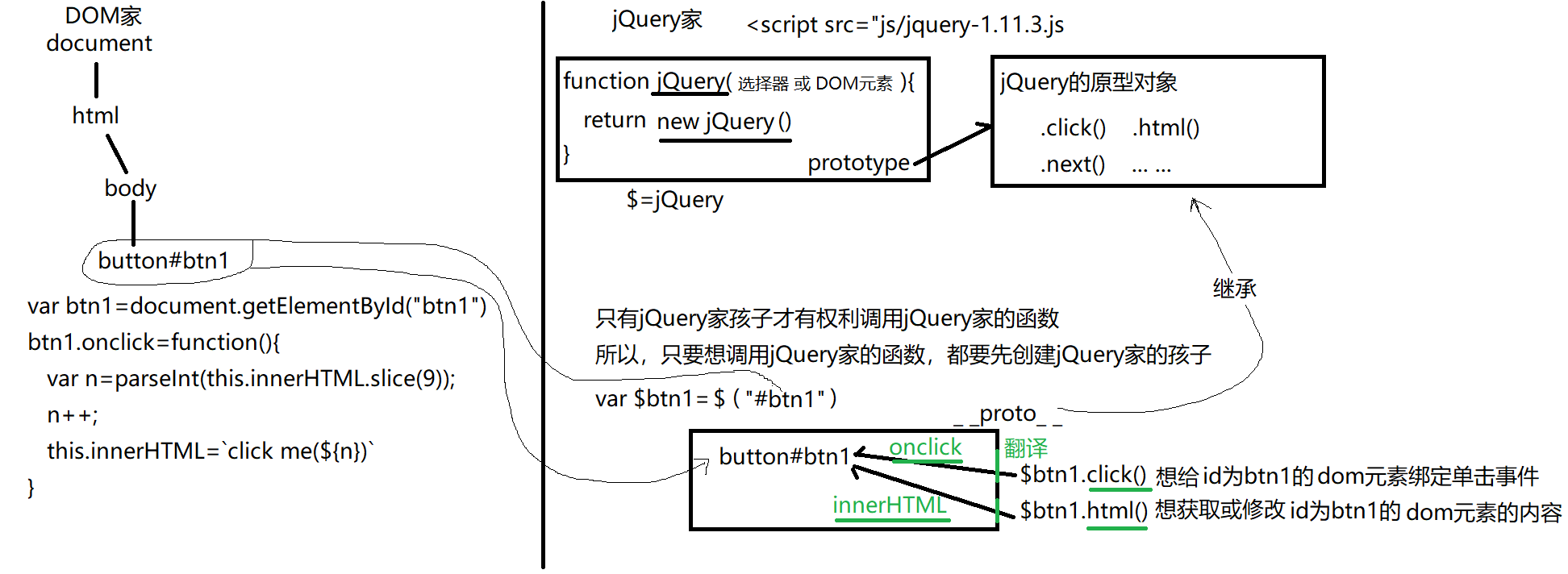
(1). 将new jQuery()放在了jQuery()函数内，从此再创建jQuery家子对象时无需再写new，只写jQuery()即可。内部执行的其实还是new jQuery()。

(2). 又因为连jQuery都懒得写！干脆！给jQuery起了个外号: $=jQuery。从此创建jQuery家孩子，只需要写$()即可！但是$()低层相当于new jQuery()。

强调: 今后应该尽量减少$()的使用次数。因为每$()一次都是重新创建一个jQuery对象，浪费内存。如果一个$()可能被反复使用，则可以先保存在变量中，再反复使用变量。

9. 说明: 保存jQuery家子对象的变量名，习惯上都以$开头

比如: var $btn1=$("#btn1")



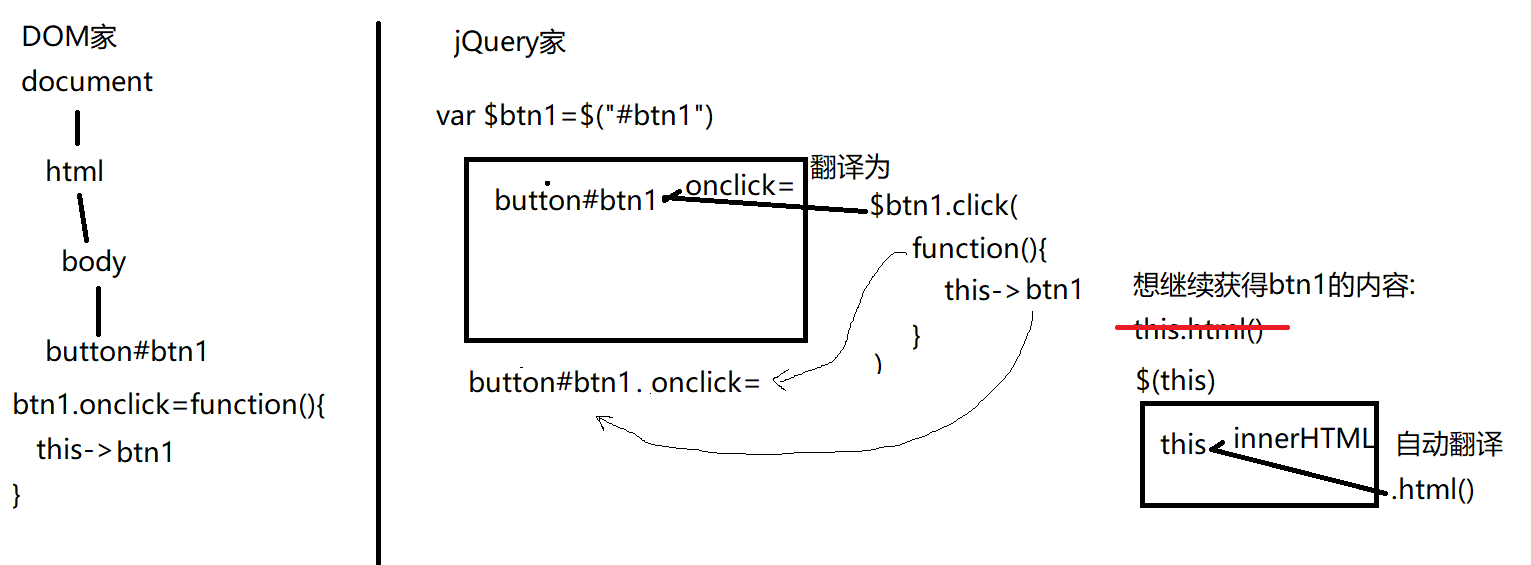
10. 问题: $btn1.click(function(){ ... this-> ? ...})

因为.$btn1.click(function(){ ... })会被自动翻译为btn1.onclick=function(){ ... }

所以: btn1.onclick=function(){ ... this->btn1 ...}

11. 问题: $btn1.click(function(){ ... this-> btn1 无法继续调用jQuery家简化版函数 ...})

解决: $(this)再包装成一个新的jQuery子对象。就可继续使用jQuery家简化版函数了。



12. 到底什么是jQuery子对象: 其实就是保存找到的多个DOM元素对象的类数组对象

13. 示例: 三个按钮，记录每个按钮的点击次数：

|  |
| --- |
| <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>  //为三个按钮绑定单击事件，记录单击次数  //查找到三个button  var $btns=$("button");  console.log($btns);  //为每个按钮绑定单击事件  //DOM中:  // for(var btn of $btns){  //   btn.onclick=function(){  //     //... ...  //   }  // }  //jq中:  $btns.click(function(){    //alert("疼!")    //3. 查找要修改的元素    //本例中: 就是要改自己的内容(this)    var $btn=$(this);    //4. 修改元素    //4.1 先取出按钮的内容中的数字，转为整数    var n=parseInt(    //获取按钮的内容      $btn.html().slice(9)    )    //4.2 将数字+1    n++    //4.3 将新数字放回按钮中    //this.innerHTML=`click me(${n})`;    //修改按钮的内容    $btn.html(`click me(${n})`)  })  </script>  </body> |

|  |
| --- |
| 总结: jQuery简化版函数的三大特点:  1. 自带for循环: 如果希望对找到的所有元素执行相同的DOM操作时，只需要对整个jQuery对象整体调用一次简化版函数。翻译时，会被自动应用到jQuery对象中每个DOM元素上    2. 凡是跟获取或修改有关的函数，都是一个函数两用: 用重载实现的  (1). 如果调用这种函数时，没有传入新值，则默认执行读取旧值的操作  (2). 如果调用这种函数时，给了新值，则自动切换为将新值修改进元素中。  3. (敬请期待) |

四. 查找元素:

1. jQuery绝大多数情况只用按选择器查找:

2. jQuery的$()中支持所有CSS3的选择器

3. jQuery还扩展了一些jQuery独有的选择器(CSS3中没有):

(1). 基本过滤选择器:

a. 回顾: CSS3子元素过滤选择器: :first-child :last-child :nth-child(i) :only-child

1). 什么是子元素过滤选择器: 根据元素在其父元素内的相对位置查找元素

2). :nth-child(i), i从1开始！

3). 示例: 使用子元素过滤选择器:

|  |
| --- |
| <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>  //查找每个ul中第一个li  $("ul>li:first-child").css("background-color","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("padding",10)  //查找作为ul下唯一子元素的li  $("ul>li:only-child").css("text-align","center")  </script>  </body> |

b. 基本过滤选择器:

1). 什么是: 先将所有符合条件的元素放在一个集合中，再统一编号！按元素在集合中的大排名选择元素。与元素在其父元素下的相对位置无关！

2). 下标从0开始

3). 包括:

i. :first 选择所有符合条件的元素中第一个元素

ii. :last 选择所有符合条件的元素中最后一个元素

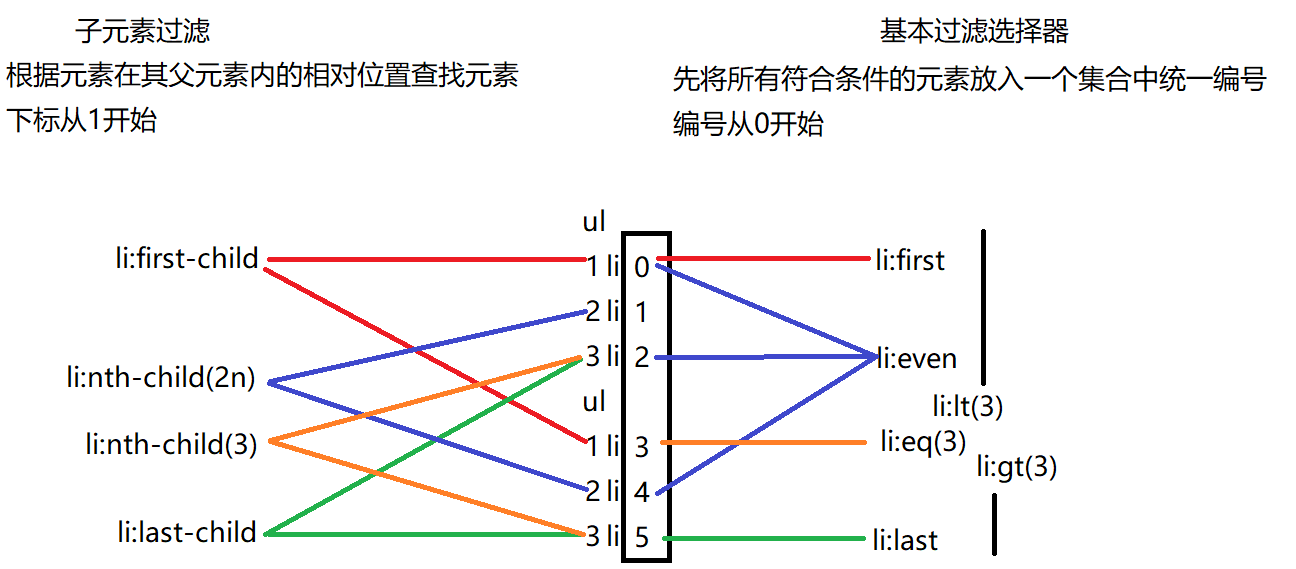
iii. :eq(i) 选择所有符合条件的元素中等于i位置的元素

iv. :gt(i) 选择所有符合条件的元素中>i位置的元素

v. :lt(i) 选择所有符合条件的元素中<i位置的元素

vi. :even 选择所有符合条件的元素中下标为偶数的位置的元素

vii. :odd 选择所有符合条件的元素中下标为奇数的位置的元素



4). 示例: 使用基本过滤选择器:

|  |
| --- |
| <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("background-color","red");  //查找最后一个li  $("ul>li:last").css("color","green")  //查找处于偶数位置的li  $("ul>li:odd").css("box-shadow","0 0 5px blue")  //查找第2个li  $("ul>li:eq(1)").css("padding",10)  </script>  </body> |

提前学:

修改元素的样式:

DOM中: 元素.style.css属性="新值"

Jq中: $(...).css("css属性","新值")

总结：

1. 查找元素: 4大类

(1). 不需要查找就可直接获得的元素或节点对象: 5个

a. document

b. document.documentElement <html>

c. document.head <head>

d. document.body <body>

e. document.forms[i] <form>

(2). 按节点间关系查找: 2大类关系 6个属性

何时: 如果已经获得一个元素，想找周围附件的元素时

a. 父子关系: 4个属性

1). 获得一个元素的父元素: 元素.parentNode

2). 获得一个元素下的所有直接子元素: 元素.children

3). 获得一个元素下的第一个直接子元素: 元素.firstElementChild

4). 获得一个元素下的最后一个直接子元素: 元素.lastElementChild

b. 兄弟关系: 2个属性

1). 获得一个元素的前一个兄弟元素: 元素.previousElementSibling

2). 获得一个元素的后一个兄弟元素: 元素.nextElementSibling

(3). 按HTML特征查找: 4个方法

何时: 只按一个条件就可找到想要的元素时

a. 按id查找一个元素: var 一个元素=document.getElementById("id名")

b. 按标签名查找多个元素:

var 类数组对象=任意父元素.getElementsByTagName("标签名")

c. 按class名查找多个元素:

var 类数组对象=任意父元素.getElementsByClassName("class名")

d. 按name名查找多个表单元素

var 类数组对象=document.getElementsByName("name名")

(4). 按选择器查找: 2个方法

何时: 如果查找条件复杂时

a. 只查找一个符合条件的元素:

var 一个元素=任意父元素.querySelector("选择器")

b. 查找多个符合条件的元素:

var 类数组对象=任意父元素.querySelectorAll("选择器")

|  |
| --- |
| 总结: 返回值:  1. 如果原函数返回下标位置i，则找不到，返回-1  2. 如果原函数返回一个对象或数组，则找不到，返回null  3. 如果原函数返回一个类数组对象，则找不到，返回空类数组对象{ length:0 } |

2. 修改元素: 3样:

(1). 内容: 3种:

a. 获取或修改元素的原始HTML内容: 元素.innerHTML

b. 获取或修改元素的纯文本内容: 元素.textContent

c. 获取或修改表单元素的值: 元素.value

(2). 属性: 3种:

a. 字符串类型的HTML标准属性: 2种

1). 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

2). HTML DOM: 元素.属性名

b. bool类型的HTML标准属性:

1). 只能用HTML DOM: 元素.属性名 值必须是bool类型

~~2). 不能用核心DOM4个方法~~

c. 自定义扩展属性: 2种:

1). 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

2). 如果自定义属性名以"data-"为前缀，还可用: 元素.dataset.属性名

3). ~~不能用HTML DOM~~

(3). 样式:

a. 只修改一个元素的一个css属性: 元素.style.css属性

b. 获取一个元素的css属性: var style=getComputedStyle(元素); style.css属性

c. 批量修改一个元素的多个css属性: 元素.className

3. 添加元素:

(1). 只添加一个元素: 3步

a. 创建新的空元素对象: var新元素=document.createElement("元素名")

b. 为新元素添加关键属性: 新元素.属性名=值

c. 将新元素添加到DOM树: 3种:

1). 在父元素下所有子元素末尾追加一个新元素:

父元素.appendChild(新元素)

2). 在父元素下的一个现有子元素前插入:

父元素.insertBefore(新元素, 现有子元素)

3). 替换父元素下的一个现有子元素:

父元素.replaceChild(新元素, 现有子元素)

(2). 添加多个元素时: 2种:

a. 同时添加父元素和多个子元素: 先将所有子元素加入到父元素中。最后再将父元素一次性添加到DOM树

b. 父元素已经在页面上，需要添加多个平级子元: 先将所有子元素加入到文档片段对象中，最后再一次性将文档片段对象加入到DOM树: 3步

1). 创建一个文档片段对象:

var frag=document.createDocumentFragment();

2). 将子元素加入到文档片段对象中:

frag.appendChild(新子元素)

3). 将文档片段对象整体加入DOM树

父元素.appendChild(frag)

4. 删除元素: 父元素.removeChild(子元素)

|  |
| --- |
| HTML DOM简写:  1. <img>: var img=new Image();  2. <table>: 逐级管理  (1). table管着行分组: 3件事  a. table可以添加行分组:  var thead=table.createTHead();  var tbody=table.createTBody();  var tfoot=table.createTFoot();  b. table可以删除行分组: table.deleteTHead()/deleteTFoot()  c. table可以获取行分组:  table.tHead table.tFoot table.tBodies[i]  (2). 行分组管着行: 3件事  a. 行分组可以添加一个新行:  1). 行分组末尾追加一个新行: var tr=行分组.insertRow()  2). 行分组开头插入一个新行: var tr=行分组.insertRow(0)  3). 行分组中任意位置插入一个新行: var tr=行分组.insertRow(i)  b. 删除行:  1). 今后删除行尽量都用table作为.前的主语: table.deleteRow(tr.rowIndex);  ~~2). 通常不用行分组删除: 行分组.deleteRow(i)~~  c. 获取行: 行分组.rows[i]  (3). 行管着格: 3件事  a. 行可以添加一格: 行末尾追加一格: var td=tr.insertCell()  b. 删除一格: tr.deleteCell(i)  c. 获取一格: tr.cells[i]  3. <form>  (1). 获得<form>元素下的表单元素:  a. 标准做法: form.elements[i或name或id]  b. 如果表单元素上有name属性可简写为: form.name名  (2). 让表单元素自动获得焦点: 表单元素.focus() |

5. 事件:

(1). 如何绑定事件处理函数: 2种:

a. 如果一个事件属性上只绑定一个处理函数: 赋值方式:

元素.on事件名=function(){ ... }

|  |
| --- |
| 常用事件名:  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 |

b. 如果一个事件属性上想同时绑定多个处理函数: 添加事件监听对象:

元素.addEventListener("事件名", 事件处理函数)

移除事件监听: 元素.removeEventListener("事件名", 原事件处理函数对象)

(2). 事件模型: 3个阶段: 捕获、目标触发、冒泡执行

(3). 事件对象:

a. 如何获得事件对象: 元素.on事件名=function(e){ ... }

b. 取消冒泡: e.stopPropagation();

c. 利用冒泡/事件委托:

1). 优化: 如果多个平级子元素都要绑定相同的事件处理函数时，应该只在父元素上绑定一次处理函数即可！——结果: 点击任何一个子元素，都会冒泡到父元素上执行

2). 问题: 如果事件绑定在父元素上，this->父元素，不再指实际出发事件的子元素。

解决: 只要使用事件委托，都要用e.target代替this，获得目标元素

3). 问题: 如果只希望子元素中部分想要的元素执行事件处理函数，而不希望所有子元素都执行事件处理函数.

解决: 只要使用事件委托，都要先判断e.target是否想要的。

判断依据可以是任何条件: nodeName, className, innerHTML...

d. 取消默认行为:

1). 问题: 有的元素身上带有默认的行为，但是这个行为我们不想要！

2). 比如: <a href="#">xxx</a>，当点击这个a时，会自动在地址栏加#。

3). 解决: e.preventDefault()

阻止 默认

4). 示例: 阻止a自动在地址栏结尾添加#

|  |
| --- |
| <body>  <a id="a1" href="#">click me</a>  <script>  var a1=document.getElementById("a1");  a1.onclick=function(e){    e.preventDefault();  }  </script>  </body> |

e. 获得鼠标位置坐标: 3组坐标:

1). 鼠标所在位置相对于显示屏左上角的距离: e.screenX e.screenY

显示屏 显示屏

2). 鼠标所在位置相对于浏览器内文档显示区左上角的距离:

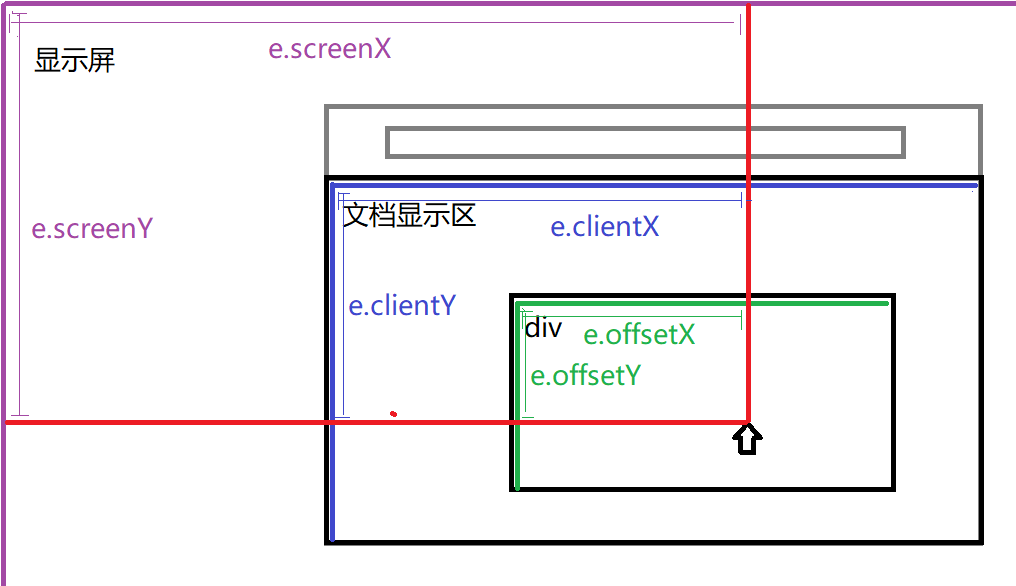
e.clientX e.clientY

客户端 客户端

3). 鼠标所在位置相对于事件绑定的元素左上角的距离:

e.offsetX e.offsetY

偏移 偏移



4). 示例: 获取鼠标的三组坐标系

|  |
| --- |
| <head>  <meta charset="UTF-8">  <title>在当前显示区范围内实现点不到的小方块</title>  <style>  div{    position:fixed;    top:150px;    left:150px;    width:200px;    height:200px;    background-image:url(images/xiaoxin.gif);    background-size:100%;  }  </style>  </head>  <body>  <div id="pop"></div>  <script>  var pop=document.getElementById("pop");  pop.onclick=function(e){    alert(`    距离屏幕左上角: e.screenX:${e.screenX}, e.screenY:${e.screenY}    距离文档显示区左上角: e.clientX:${e.clientX}, e.clientY:${e.clientY}    距离当前元素左上角: e.offsetX:${e.offsetX},e.offsetY:${e.offsetY}    `)  }  </script>  </body> |

f. 页面滚动事件:

1). 在页面滚动时，自动执行:

// 当滚动时自动执行

window.onscroll=function(){

//希望获得页面滚动过的距离: body顶部超出文档显示区顶部的距离

var scrollTop=document.documentElement.scrollTop//能用第一就用第一个

||document.body.scrollTop//如果不能用第一个就退一步用第二个

}

2). 用程序主动控制让网页滚动到哪个位置:

window.scrollTo(水平方向滚动到的位置, 垂直方向滚动到的位置)

// 滚动 0

3). 示例: 当页面滚动到500时显示回到顶部按钮，点回到顶部可以跳转回顶部

|  |
| --- |
| <head>  <meta charset="UTF-8">  <title>根据页面滚动位置显示浮动框</title>  <style>    body{height:2000px;}    #toTop{      position:fixed;      bottom:100px;      right:0;      display:none;    }  </style>  </head>  <body>  <div id="toTop">    <a href="#">返回顶部</a>  </div>  <script>  window.onscroll=function(){    var scrollTop=document.documentElement.scrollTop                  ||document.body.scrollTop;    console.log(scrollTop);    var div=document.getElementById("toTop");  //当页面滚动过的距离>=500时就让div#toTop显示出来    if(scrollTop>=500){      div.style.display="block";    }else{//否则如果页面滚动过的距离<500时就让div#toTop隐藏起来      div.style.display="none";    }  }  //单击a返回顶部  var a=document.querySelector("#toTop>a");  a.onclick=function(e){    e.preventDefault();    window.scrollTo(0,0);  }  </script>  </body> |

作业:

1. (必须)复习CSS3中所有选择器

2. (必须)看小程序视频学习jQuery新增选择器

小程序->在线->jQuery->day01 作业:分别使用css和jquery实现按钮组效果和表格隔行变色效果。

3. (扩展)看小程序视频学习拖拽效果

小程序->在线->DOM->day04 5. 鼠标坐标 x y 原生DOM拖拽效果

4. (扩展)看小程序视频学习页面滚动动画效果

小程序->在线->DOM->day04 作业: 原生DOM页面滚动动画效果